In this chapter you will learn about:

- Computer languages or programming languages
- Three broad categories of programming languages – machine, assembly, and high-level languages
- Commonly used programming language tools such as assembler, compiler, linker, and interpreter
- Concepts of object-oriented programming languages
- Some popular programming languages such as FORTRAN, COBOL, BASIC, Pascal, C, C++, C#, Java, RPG, LISP and SNOBOL
- Related concepts such as Subprogram, Characteristics of a good programming language, and factors to consider while selecting a language for coding an application
Broad Classification of Computer Languages

- Machine language
- Assembly language
- High-level language
Machine Language

- Only language of a computer understood by it without using a translation program
- Normally written as strings of binary 1s and 0s
- Written using decimal digits if the circuitry of the computer being used permits this
A Typical Machine Language Instruction Format

<table>
<thead>
<tr>
<th>OPCODE</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>(operation code)</td>
<td>(Address/Location)</td>
</tr>
</tbody>
</table>

- OPCODE tells the computer which operation to perform from the instruction set of the computer.
- OPERAND tells the address of the data on which the operation is to be performed.
A Sample Machine Language Program

In Binary
(Difficult to read and understand)

In Decimal
(Easier to read and understand)

001000000000001100111001 10001471
00110000000010000100001 14002041
0110000000011100101110 30003456
10100011111101100101110 50773456
000000000000000000000000 00000000
Advantages & Limitations of Machine Language

**Advantage**
- Can be executed very fast

**Limitations**
- Machine Dependent
- Difficult to program
- Error prone
- Difficult to modify
Assembly/Symbolic Language

Programming language that overcomes the limitations of machine language programming by:

- Using alphanumeric mnemonic codes instead of numeric codes for the instructions in the instruction set. 
  e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add.

- Allowing storage locations to be represented in form of alphanumeric addresses instead of numeric addresses. 
  e.g. representing memory locations 1000, 1001, and 1002 as FRST, SCND, and ANSR respectively.

- Providing pseudo-instructions that are used for instructing the system how we want the program to be assembled inside the computer’s memory. 
  e.g. START PROGRAM AT 0000; SET ASIDE AN ADDRESS FOR FRST.
Assembler

Software that translates as assembly language program into an equivalent machine language program of a computer.

Assembly language program → Assembler → Machine language program

(Source Program) → One-to-one correspondence → (Object Program)
### An Example of Assembly Language Program

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLT</td>
<td>00</td>
<td>Halt, used at the end of program to stop</td>
</tr>
<tr>
<td>CLA</td>
<td>10</td>
<td>Clear and add into A register</td>
</tr>
<tr>
<td>ADD</td>
<td>14</td>
<td>Add to the contents of A register</td>
</tr>
<tr>
<td>SUB</td>
<td>15</td>
<td>Subtract from the contents of A register</td>
</tr>
<tr>
<td>STA</td>
<td>30</td>
<td>Store A register</td>
</tr>
</tbody>
</table>

A subset of the set of instructions supported by a computer.
An Example of Assembly Language Program

START PROGRAM AT 0000
START DATA AT 1000
SET ASIDE AN ADDRESS FOR FRST
SET ASIDE AN ADDRESS FOR SCND
SET ASIDE AN ADDRESS FOR ANSR
CLA FRST
ADD SCND
STA ANSR
HLT

Sample assembly language program for adding two numbers and storing the result
### An Example of Assembly Language Program

<table>
<thead>
<tr>
<th>Symbolic name</th>
<th>Memory location</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRST</td>
<td>1000</td>
</tr>
<tr>
<td>SCND</td>
<td>1001</td>
</tr>
<tr>
<td>ANSR</td>
<td>1002</td>
</tr>
</tbody>
</table>

Mapping table set up by the assembler for the data items of the assembly language program.
### An Example of Assembly Language Program

<table>
<thead>
<tr>
<th>Memory location</th>
<th>Contents</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Address</td>
<td>Clear and add the number stored at FRST to A register</td>
</tr>
<tr>
<td>0000</td>
<td>10</td>
<td>1000</td>
</tr>
<tr>
<td>0001</td>
<td>14</td>
<td>1001</td>
</tr>
<tr>
<td>0002</td>
<td>30</td>
<td>1002</td>
</tr>
<tr>
<td>0003</td>
<td>00</td>
<td>Halt</td>
</tr>
</tbody>
</table>

- Reserved for FRST
- Reserved for SCND
- Reserved for ANSR

Equivalent machine language program for the assembly language program
Advantages of Assembly Language Over Machine Language

- Easier to understand and use
- Easier to locate and correct errors
- Easier to modify
- No worry about addresses
- Easily relocatable
- Efficiency of machine language
Limitations of Assembly Language

- Machine dependent
- Knowledge of hardware required
- Machine level coding

Ref Page 215  Chapter 12: Computer Languages  Slide 15/59
Typical Uses of Assembly Language

- Mainly used today to fine-tune important parts of programs written in a high-level language to improve the program’s execution efficiency
Any assembly language instruction that gets translated into several machine language instructions is called a **macro instruction**.

Several assembly languages support such macro instructions to speed up the coding process.

Assemblers of such assembly languages are designed to produce multiple machine language instructions for each macro instruction of the assembly language.
High-Level Languages

- Machine independent
- Do not require programmers to know anything about the internal structure of computer on which high-level language programs will be executed
- Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions
Compiler

- Translator program (software) that translates a high-level language program into its equivalent machine language program
- Compiles a set of machine language instructions for every program instruction in a high-level language
One-to-many correspondence

High-level language program \(\rightarrow\) Input \(\rightarrow\) Compiler \(\rightarrow\) Output \(\rightarrow\) Machine language program

(Source Program) \(\rightarrow\) Input \(\rightarrow\) Compiler \(\rightarrow\) Output \(\rightarrow\) (Object Program)

One-to-many correspondence
A computer supporting languages L1 and L2

Illustrating the requirement of a separate compiler for each high-level language supported by a computer
Illustrating the machine independence characteristic of a high-level language. Separate compilers are required for the same language on different computers.
Syntax Errors

In addition to doing translation job, compilers also automatically detect and indicate syntax errors.

Syntax errors are typically of following types:

- Illegal characters
- Illegal combination of characters
- Improper sequencing of instructions in a program
- Use of undefined variable names

*Note*: A compiler cannot detect logic errors in a program.
The Process of Removing Syntax Errors From A Source Program

1. START
2. Edit source program
3. Compile source program
4. Syntax errors detected?
   - Yes: Generate list of coded error messages
   - No: Generate object program
5. Object program
6. STOP
For a large software, storing all the lines of program code in a single source file will be:

- Difficult to work with
- Difficult to deploy multiple programmers to concurrently work towards its development
- Any change in the source program would require the entire source program to be recompiled

Hence, a modular approach is generally adapted to develop large software where the software consists of multiple source program files

No need to write programs for some modules as it might be available in library offering the same functionality

(Continued on next slide)
Each source program file can be independently modified and compiled to create a corresponding object program file.

Linker program (software) is used to properly combine all the object program files (modules).

Creates the final executable program (load module).
Interpreter

- Interpreter is a high-level language translator
- Takes one statement of a high-level language program, translates it into machine language instructions
- Immediately executes the resulting machine language instructions
- Compiler simply translates the entire source program into an object program and is not involved in its execution
Role of an Interpreter

High-level language program (Source Program) → Interpreter (translates and executes statement-by-statement) → Output → Result of program execution
Intermediate Language Compiler & Interpreter

- New type of compiler and interpreter combines the speed, ease, and control of both compiler and interpreter.

- Compiler first compiles the source program to an *intermediate* object program.

- Intermediate object program is not a machine language code but written in an intermediate language that is virtually machine independent.

- Interpreter takes intermediate object program, converts it into machine language program and executes it.
Benefits of Intermediate Language
Compiler & Interpreter

- Intermediate object program is in compiled form and thus is not original source code, so safer and easier to share
- Intermediate object program is based on a standard Intermediate Definition Language (IDL)
- Interpreter can be written for any computer architecture and operating system providing virtual machine environment to the executing program
- Newer Interpreter compiles intermediate program, in memory, into final host machine language program and executes it
- This technique is called *Just-In-Time (JIT) Compilation*
Advantages of High-Level Languages

- Machine independent
- Easier to learn and use
- Fewer errors during program development
- Lower program preparation cost
- Better documentation
- Easier to maintain
Limitations of High-Level Languages

- Lower execution efficiency
- Less flexibility to control the computer’s CPU, memory and registers
Object-Oriented Programming Languages

- Programming languages are used for simulating real-world problems on computers
- Much of the real world is made up of objects
- Essence of OOP is to solve a problem by:
  - Identifying the real-world objects of the problem
  - Identifying processing required of them
  - Creating simulations of objects, processes, and their communications
FORTRAN

- Stands for **FORmula TRANslation**
- Originally developed by John Backus and his team at IBM followed by several revisions
- Standardized by ANSI as FORTRAN-77 and FORTRAN-90
- Designed for solving scientific & engineering problems
- Oriented towards solving problems of a mathematical nature
- Popular language amongst scientists and engineers
A Sample FORTRAN Program

C FORTRAN PROGRAM TO COMPUTE
C THE SUM OF 10 NUMBERS
    SUM = 0
    DO 50 I = 1, 10
    READ (5, 10) N
    SUM = SUM + N
50 CONTINUE
    WRITE (6, 20) SUM
10 FORMAT (F6.2)
20 FORMAT (1X, 'THE SUM OF GIVEN NUMBERS = ', F10.2)
STOP
END
COBOL

- Stands for **CO**mmom **B**usiness **O**riented **L**anguage
- Originally developed started under Grace Hopper followed by COnference on DAta SYstems Languages (CODASYL)
- Standardized by ANSI as COBOL-74, COBOL-85, and COBOL-2002
- Designed for programming business data processing applications
- Designed to have the appearance and structure of a business report written in English, hence often referred to as a self-documenting language
IDENTIFICATION DIVISION.
PROGRAM_ID. SUMUP.
AUTHOR. P K SINHA.
* THIS PROGRAM COMPUTES AND PRINTS
* THE SUM OF GIVEN NUMBERS.

ENVIROMENT DIVISION.
CONFIGURATION SECTION.
SOURCE_COMPUTER. BURROUGHS_6700.
OBJECT_COMPUTER. BURROUGHS_6700.
INPUT_OUTPUT SECTION.
FILE_CONTROL.
   SELECT DATA_FILE ASSIGN TO DISK.
   SELECT OUTPUT_FILE ASSIGN TO PRINTER.

DATA DIVISION.
FILE SECTION.

(Continued on next slide)
FD DATA_FILE
    RECORD CONTAINS 80 CHARACTERS
    LABEL RECORD IS OMITTED
    DATA RECORD IS INPUT_DATA_RECORD.

01 INPUT_DATA_RECORD.
   05 NPICTURE 9(6)V99.
   05 FILLER PICTURE X(72).

FD OUTPUT_FILE
    RECORD CONTAINS 132 CHARACTERS
    LABEL RECORD IS OMITTED
    DATA RECORD IS OUTPUT_RECORD.

(Continued on next slide)
A Sample COBOL Program

(Continued from previous slide..)

```
01 OUTPUT_RECORD.
   05 FILLER PICTURE X.
   05 TITLE PICTURE X(25).
   05 SUM PICTURE 9(10)V99.

05 FILLER PICTURE X(94).

WORKING_STORAGE SECTION.
77 MESSAGE PICTURE X(25)
VALUE IS "THE SUM OF GIVEN NUMBERS=".

PROCEDURE DIVISION.
OPEN_FILES.
   OPEN INPUT DATA_FILE.
   OPEN OUTPUT OUTPUT_FILE.

INITIALIZATION.
   MOVE SPACES TO OUTPUT_RECORD.
   MOVE ZERO TO SUM.
```

(Continued on next slide)
A Sample COBOL Program

(Continued from previous slide..)

```
PROCESS_LOOP.
    READ DATA_FILE AT END GO TO PRINT_PARA.
    ADD N TO SUM.
    GO TO PROCESS_LOOP.

PRINT_PARA.
    MOVE MESSAGE TO TITLE.
    WRITE OUTPUT_RECORD.

END_OF_JOB.
    CLOSE DATA_FILE.
    CLOSE OUTPUT_FILE.
    STOP RUN.
```
BASIC

- Stands for Beginners All-purpose Symbolic Instruction Code
- Developed by Professor John Kemeny and Thomas Kurtz at Darmouth College in the United States
- Standardized by ANSI as BASIC-78
- Designed to be an interactive language and to use an interpreter instead of a compiler
- Simple to implement, learn and use language. Hence, it is a widely used language on personal computers
- Flexible and reasonably powerful language and can be used for both business and scientific applications
REM PROGRAM TO COMPUTE
REM THE SUM OF 10 NUMBERS
LET S = 0
FOR I = 1 TO 10
READ N
LET S = S + N
NEXT I
PRINT "THE SUM OF GIVEN NUMBERS = "; S
DATA 4, 20, 15, 32, 48
DATA 12, 3, 9, 14, 44
END;
Pascal

- Named after the famous seventeenth-century French mathematician Blaise Pascal
- Developed by Professor Nicklaus Wirth of Federal Institute of Technology in Zurich
- Encourages programmers to write well-structured, modular programs, instills good program practices
- Recognized as an educational language and is used to teach programming to beginners
- Suitable for both scientific & business applications
- Has features to manipulate numbers, vectors, matrices, strings, sets, records, files, and lists
PROGRAM SUMNUMS (INPUT, OUTPUT);
(* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS *)

(* DECLARATION OF VARIABLES *)
VAR SUM, N : REAL;
VAR I : INTEGER;

(* MAIN PROGRAM LOGIC STARTS HERE *)
BEGIN
  SUM := 0;
  FOR I := 1 TO 10 DO begin
    READ (N);
    SUM := SUM + N;
    END;
  WRITELN (‘THE SUM OF GIVEN NUMBERS=’, SUM);
END;
Developed in 1972 at AT&T’s Bell laboratories, USA by Dennis Ritchie and Brian Kernighan

Standardized by ANSI and ISO as C89, C90, C99

High-level programming languages (mainly machine independence) with the efficiency of an assembly language

Language of choice of programmers for portable systems software and commercial software packages like OS, compiler, spreadsheet, word processor, and database management systems
/* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS */
/* Directives to include standard library and header */
#include <stdlib.h>
#include <stdio.h>
/* Main function starts here */
void main ( )
{
    /* Declaration of variables */
    float Sum = 0.0, N = 0.0;
    int Count = 0;
    for (Count = 0; Count < 10; Count++)
    {
        printf(“
Give a number:”);
        scanf(“%f”, N);
        Sum += N;
    }
    printf(“THE SUM OF GIVEN NUMBERS = %f”, &Sum);
}
C++

- Named C++ as ++ is increment operator and C language is incremented to its next level with C++
- Developed by Bjarne Stroustrup at Bell Labs in the early 1980s
- Contains all elements of the basic C language
- Expanded to include numerous object-oriented programming features
- Provides a collection of predefined classes, along with the capability of user-defined classes
Java

- Development started at Sun Microsystems in 1991 by a team led by James Gosling
- Developed to be similar to C++ with fewer features to keep it simple and easy to use
- Compiled code is machine-independent and developed programs are simple to implement and use
- Uses *just-in-time* compilation
- Used in embedded systems such as hand-held devices, telephones and VCRs
- Comes in two variants – Java Runtime Engine (JRE) and Java Software Development Kit (SDK)
C# (C Sharp)

- Object-oriented programming language developed by Anders Hejlsberg and released by Microsoft as part of Microsoft’s .NET technology initiative.
- Standardized by ECMA and ISO.
- Syntactically and semantically very close to C++ and adopts various object-oriented features from both C++ and Java.
- Compilers target the Common Language Infrastructure (CLI) implemented by Common Language Runtime (CLR) of .NET Framework.
- CLR provides important services such as memory management, exception handling, and security.
RPG

- Stands for Report Program Generator
- Developed by IBM to meet customer requests for an easy and economic mechanism for producing reports
- Designed to generate the output reports resulting from the processing of common business applications
- Easier to learn and use as compared to COBOL
- Programmers use very detailed coding sheets to write specifications about input, calculations, and output
LISP

- Stands for **LIS**t **P**rocessing
- Developed in 1959 by John McCarthy of MIT
- Designed to have features for manipulating non-numeric data, such as symbols and strings of text
- Due to its powerful list processing capability, it is extensively used in the areas of pattern recognition, artificial intelligence, and for simulation of games
- Functional programming language in which all computation is accomplished by applying functions to arguments
SNOBOL

- Stands for String Oriented Symbolic Language
- Used for non-numeric applications
- Powerful string manipulation features
- Widely used for applications in the area of text processing
Characteristics of a Good Programming Language

- Simplicity
- Naturalness
- Abstraction
- Efficiency
- Structured Programming Support
- Compactness
- Locality
- Extensibility
- Suitability to its environment
Factors for Selecting a Language for Coding an Application

- Nature of the application
- Familiarity with the language
- Ease of learning the language
- Availability of program development tools
- Execution efficiency
- Features of a good programming language
Subprogram

- Program written in a manner that it can be brought into use in other programs and used whenever needed without rewriting
- Also referred to as subroutine, sub-procedure, or function
- Subprogram call statement contains the name of the subprogram followed by a list of parameters enclosed within a pair of parentheses
- Intrinsic subprograms (also called built-in-functions) are those provided with the programming language
- Programmer-written subprograms are written and used as and when they are needed
Structure of a Subprogram

Subprogram name: `sqrt`  Parameter: `x`

- Subprogram header
- Set of instructions that perform the intended task
- ...
A subprogram that calls the subprogram twice
## Key Words/Phrases

- Assembler
- Assembly language
- BASIC
- Built-in function
- C
- C++
- C#
- COBOL
- Coding
- Compiler
- Computer language
- FORTRAN
- Function
- High-level language
- HotJava Interpreter
- Intrinsic subprogram
- Intermediate compiler and Interpreter
- Java
- Just-in-time compilation
- Language processor
- Linker
- LISP
- Load module
- Logic error
- Low-level language
- Machine language
- Macro instructions
- Object program
- Object-oriented programming
- Opcode
- Operand
- Pascal
- Programmer
- Programming
- Programming language
- Pseudo instruction
- RPG
- Self-documenting language

*(Continued on next slide)*
Key Words/Phrases

(Continued from previous slide..)

- SNOBOL
- Source program
- Sub-procedure
- Subprogram
- Subroutine
- Symbolic language
- Syntax error
- Syntax rules
- Written subprograms
Chapter 12: Computer Languages

In this chapter you will learn about:

- Computer languages or programming languages
- Three broad categories of programming languages – machine, assembly, and high-level languages
- Commonly used programming language tools such as assembler, compiler, linker, and interpreter
- Concepts of object-oriented programming languages
- Some popular programming languages such as FORTRAN, COBOL, BASIC, Pascal, C, C++, C#, Java, RPG, LISP and SNOBOL
- Related concepts such as Subprogram, Characteristics of a good programming language, and factors to consider while selecting a language for coding an application
Broad Classification of Computer Languages

- Machine language
- Assembly language
- High-level language

Machine Language

- Only language of a computer understood by it without using a translation program
- Normally written as strings of binary 1s and 0s
- Written using decimal digits if the circuitry of the computer being used permits this
A Typical Machine Language Instruction Format

<table>
<thead>
<tr>
<th>OPCODE</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>(operation code)</td>
<td>(Address/Location)</td>
</tr>
</tbody>
</table>

- OPCODE tells the computer which operation to perform from the instruction set of the computer.
- OPERAND tells the address of the data on which the operation is to be performed.

A Sample Machine Language Program

<table>
<thead>
<tr>
<th>In Binary</th>
<th>In Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>001000000000001100111001</td>
<td>10001471</td>
</tr>
<tr>
<td>0011000000000100010001</td>
<td>14002041</td>
</tr>
<tr>
<td>011000000000011100101110</td>
<td>30003456</td>
</tr>
<tr>
<td>101000111111011100101110</td>
<td>50773456</td>
</tr>
<tr>
<td>000000000000000000000000</td>
<td>00000000</td>
</tr>
</tbody>
</table>

(Difficult to read and understand) (Easier to read and understand)
Advantages & Limitations of Machine Language

**Advantage**
- Can be executed very fast

**Limitations**
- Machine Dependent
- Difficult to program
- Error prone
- Difficult to modify

Assembly/Symbolic Language

Programming language that overcomes the limitations of machine language programming by:

- Using alphanumeric mnemonic codes instead of numeric codes for the instructions in the instruction set
  e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add

- Allowing storage locations to be represented in form of alphanumeric addresses instead of numeric addresses
  e.g. representing memory locations 1000, 1001, and 1002 as FRST, SCND, and ANSR respectively

- Providing pseudo-instructions that are used for instructing the system how we want the program to be assembled inside the computer’s memory
  e.g. START PROGRAM AT 0000; SET ASIDE AN ADDRESS FOR FRST
Assembler

Software that translates as assembly language program into an equivalent machine language program of a computer.

A subset of the set of instructions supported by a computer.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLT</td>
<td>00</td>
<td>Halt, used at the end of program to stop</td>
</tr>
<tr>
<td>CLA</td>
<td>10</td>
<td>Clear and add into A register</td>
</tr>
<tr>
<td>ADD</td>
<td>14</td>
<td>Add to the contents of A register</td>
</tr>
<tr>
<td>SUB</td>
<td>15</td>
<td>Subtract from the contents of A register</td>
</tr>
<tr>
<td>STA</td>
<td>30</td>
<td>Store A register</td>
</tr>
</tbody>
</table>

An Example of Assembly Language Program
Sample assembly language program for adding two numbers and storing the result:

```
START PROGRAM AT 0000
START DATA AT 1000
SET ASIDE AN ADDRESS FOR FRST
SET ASIDE AN ADDRESS FOR SCND
SET ASIDE AN ADDRESS FOR ANSR
CLA FRST
ADD SCND
STA ANSR
HLT
```

Mapping table set up by the assembler for the data items of the assembly language program:

<table>
<thead>
<tr>
<th>Symbolic name</th>
<th>Memory location</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRST</td>
<td>1000</td>
</tr>
<tr>
<td>SCND</td>
<td>1001</td>
</tr>
<tr>
<td>ANSR</td>
<td>1002</td>
</tr>
</tbody>
</table>
An Example of Assembly Language Program

<table>
<thead>
<tr>
<th>Memory location</th>
<th>Contents</th>
<th>Address</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Address</td>
<td>Clear and add the number stored at FRST to A register</td>
<td>Add the number stored at SCND to the contents of A register</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>14</td>
<td>1001</td>
<td>Add the number stored at SCND to the contents of A register</td>
</tr>
<tr>
<td>0002</td>
<td>30</td>
<td>1002</td>
<td>Store the contents of A register into ANSR</td>
</tr>
<tr>
<td>0003</td>
<td>00</td>
<td></td>
<td>Halt</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td>Reserved for FRST</td>
</tr>
<tr>
<td>1001</td>
<td></td>
<td></td>
<td>Reserved for SCND</td>
</tr>
<tr>
<td>1002</td>
<td></td>
<td></td>
<td>Reserved for ANSR</td>
</tr>
</tbody>
</table>

Equivalent machine language program for the assembly language program

Advantages of Assembly Language Over Machine Language

- Easier to understand and use
- Easier to locate and correct errors
- Easier to modify
- No worry about addresses
- Easily relocatable
- Efficiency of machine language
Limitations of Assembly Language

- Machine dependent
- Knowledge of hardware required
- Machine level coding

Typical Uses of Assembly Language

- Mainly used today to fine-tune important parts of programs written in a high-level language to improve the program’s execution efficiency
Assembly Languages with Macro Instructions

- Any assembly language instruction that gets translated into several machine language instructions is called a **macro instruction**
- Several assembly languages support such macro instructions to speed up the coding process
- Assemblers of such assembly languages are designed to produce multiple machine language instructions for each macro instruction of the assembly language

High-Level Languages

- Machine independent
- Do not require programmers to know anything about the internal structure of computer on which high-level language programs will be executed
- Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions
Compiler

- Translator program (software) that translates a high-level language program into its equivalent machine language program.
- Compiles a set of machine language instructions for every program instruction in a high-level language.

Diagram:

```
High-level language program → Input → Compiler → Output → Machine language program

(Source Program) → One-to-many correspondence → (Object Program)
```
Illustrating the requirement of a separate compiler for each high-level language supported by a computer.

Illustrating the machine independence characteristic of a high-level language. Separate compilers are required for the same language on different computers.
Syntax Errors

In addition to doing translation job, compilers also automatically detect and indicate syntax errors.

Syntax errors are typically of following types:

- Illegal characters
- Illegal combination of characters
- Improper sequencing of instructions in a program
- Use of undefined variable names

*Note*: A compiler cannot detect logic errors in a program.

The Process of Removing Syntax Errors From A Source Program

1. Edit source program
2. Compile source program
3. Check for syntax errors
   - If yes, generate list of coded error messages
   - If no, generate object program
4. STOP
For a large software, storing all the lines of program code in a single source file will be:
- Difficult to work with
- Difficult to deploy multiple programmers to concurrently work towards its development
- Any change in the source program would require the entire source program to be recompiled

Hence, a modular approach is generally adapted to develop large software where the software consists of multiple source program files

No need to write programs for some modules as it might be available in library offering the same functionality

Each source program file can be independently modified and compiled to create a corresponding object program file

Linker program (software) is used to properly combine all the object program files (modules)

Creates the final executable program (load module)
**Interpreter**

- Interpreter is a high-level language translator
- Takes one statement of a high-level language program, translates it into machine language instructions
- Immediately executes the resulting machine language instructions
- Compiler simply translates the entire source program into an object program and is not involved in its execution

**Role of an Interpreter**

- High-level language program (Source Program) → Input
- Interpreter (translates and executes statement-by-statement) → Output
- Result of program execution
Intermediate Language Compiler & Interpreter

- New type of compiler and interpreter combines the speed, ease, and control of both compiler and interpreter.
- Compiler first compiles the source program to an intermediate object program.
- Intermediate object program is not a machine language code but written in an intermediate language that is virtually machine independent.
- Interpreter takes intermediate object program, converts it into machine language program and executes it.

Benefits of Intermediate Language Compiler & Interpreter

- Intermediate object program is in compiled form and thus is not original source code, so safer and easier to share.
- Intermediate object program is based on a standard Intermediate Definition Language (IDL).
- Interpreter can be written for any computer architecture and operating system providing virtual machine environment to the executing program.
- Newer Interpreter compiles intermediate program, in memory, into final host machine language program and executes it.
- This technique is called Just-In-Time (JIT) Compilation.
Advantages of High-Level Languages

- Machine independent
- Easier to learn and use
- Fewer errors during program development
- Lower program preparation cost
- Better documentation
- Easier to maintain

Limitations of High-Level Languages

- Lower execution efficiency
- Less flexibility to control the computer’s CPU, memory and registers
Object-Oriented Programming Languages

Programming languages are used for simulating real-world problems on computers.

1. Much of the real world is made up of objects.
2. Essence of OOP is to solve a problem by:
   a. Identifying the real-world objects of the problem.
   b. Identifying processing required of them.
   c. Creating simulations of objects, processes, and their communications.

FORTRAN

1. Stands for FORMULA TRANslation.
2. Originally developed by John Backus and his team at IBM followed by several revisions.
3. Standardized by ANSI as FORTRAN-77 and FORTRAN-90.
4. Designed for solving scientific & engineering problems.
5. Oriented towards solving problems of a mathematical nature.
6. Popular language amongst scientists and engineers.

Ref Page 223  Chapter 12: Computer Languages  Slide 33/59
A Sample FORTRAN Program

```fortran
C FORTRAN PROGRAM TO COMPUTE
    C THE SUM OF 10 NUMBERS
        SUM = 0
        DO 50 I = 1, 10
            READ (5, 10) N
            SUM = SUM + N
        50 CONTINUE
        WRITE (6, 20) SUM
    10 FORMAT (F6.2)
    20 FORMAT (1X, 'THE SUM OF GIVEN NUMBERS = ', F10.2)
STOP
END
```

COBOL

- Stands for **CO**mmonly **B**usiness **O**riented **L**anguage
- Originally developed started under Grace Hopper followed by CONference on DAta SYstems Languages (CODASYL)
- Standardized by ANSI as COBOL-74, COBOL-85, and COBOL-2002
- Designed for programming business data processing applications
- Designed to have the appearance and structure of a business report written in English, hence often referred to as a self-documenting language
IDENTIFICATION DIVISION.
PROGRAM_ID. SUMUP.
AUTHOR. P K SINHA.
* THIS PROGRAM COMPUTES AND PRINTS
* THE SUM OF GIVEN NUMBERS.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE_COMPUTER. BURROUGHS_6700.
OBJECT_COMPUTER. BURROUGHS_6700.
INPUT_OUTPUT SECTION.
FILE_CONTROL.
   SELECT DATA_FILE ASSIGN TO DISK.
   SELECT OUTPUT_FILE ASSIGN TO PRINTER.

DATA DIVISION.
FILE SECTION.

FD DATA_FILE
   RECORD CONTAINS 80 CHARACTERS
   LABEL RECORD IS OMITTED
   DATA RECORD IS INPUT_DATA_RECORD.

   01 INPUT_DATA_RECORD.
      05 NPICTURE 9(6)V99.
      05 FILLER PICTURE X(72).

FD OUTPUT_FILE
   RECORD CONTAINS 132 CHARACTERS
   LABEL RECORD IS OMITTED
   DATA RECORD IS OUTPUT_RECORD.
A Sample COBOL Program

(Continued from previous slide.)

```cobol
01 OUTPUT_RECORD.
  05 FILLER PICTURE X.
  05 TITLE PICTURE X(25).
  05 SUM PICTURE 9(10)V99.
  05 FILLER PICTURE X(94).

WORKING_STORAGE SECTION.
  77 MESSAGE PICTURE X(25)
  VALUE IS "THE SUM OF GIVEN NUMBERS=".

PROCEDURE DIVISION.
OPEN_FILES:
  OPEN INPUT DATA_FILE.
  OPEN OUTPUT OUTPUT_FILE.

INITIALIZATION.
  MOVE SPACES TO OUTPUT_RECORD.
  MOVE ZERO TO SUM.

PROCESS_LOOP.
  READ DATA_FILE AT END GO TO PRINT_PARA.
  ADD N TO SUM.
  GO TO PROCESS_LOOP.

PRINT_PARA.
  MOVE MESSAGE TO TITLE.
  WRITE OUTPUT_RECORD.

END_OF_JOB.
  CLOSE DATA_FILE.
  CLOSE OUTPUT_FILE.
  STOP RUN.
```

(Continued on next slide)
BASIC

- Stands for Beginners All-purpose Symbolic Instruction Code
- Developed by Professor John Kemeny and Thomas Kurtz at Darmouth College in the United States
- Standardized by ANSI as BASIC-78
- Designed to be an interactive language and to use an interpreter instead of a compiler
- Simple to implement, learn and use language. Hence, it is a widely used language on personal computers
- Flexible and reasonably powerful language and can be used for both business and scientific applications

A Sample BASIC Program

```
5 REM PROGRAM TO COMPUTE
6 REM THE SUM OF 10 NUMBERS
10 LET S = 0
20 FOR I = 1 TO 10
30 READ N
40 LET S = S + N
50 NEXT I
60 PRINT "THE SUM OF GIVEN NUMBERS = "; S
70 DATA 4, 20, 15, 32, 48
80 DATA 12, 3, 9, 14, 44
90 END;
```
**Pascal**

- Named after the famous seventeenth-century French mathematician Blaise Pascal
- Developed by Professor Nicklaus Wirth of Federal Institute of Technology in Zurich
- Encourages programmers to write well-structured, modular programs, instills good program practices
- Recognized as an educational language and is used to teach programming to beginners
- Suitable for both scientific & business applications
- Has features to manipulate numbers, vectors, matrices, strings, sets, records, files, and lists

---

**A Sample Pascal Program**

```pascal
PROGRAM SUMNUMS (INPUT, OUTPUT);
(* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS *)

(* DECLARATION OF VARIABLES *)
VAR SUM, N : REAL;
VAR I : INTEGER;

(* MAIN PROGRAM LOGIC STARTS HERE *)
BEGIN
  SUM := 0;
  FOR I := 1 TO 10 DO
    BEGIN
      READ (N);
      SUM := SUM + N;
    END;
  WRITELN ('THE SUM OF GIVEN NUMBERS=', SUM);
END;
```
Developed in 1972 at AT&T’s Bell laboratories, USA by Dennis Ritchie and Brian Kernighan
- Standardized by ANSI and ISO as C89, C90, C99
- High-level programming languages (mainly machine independence) with the efficiency of an assembly language
- Language of choice of programmers for portable systems software and commercial software packages like OS, compiler, spreadsheet, word processor, and database management systems

A Sample C Program

```c
/* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS */
/* Directives to include standard library and header */
#include <stdlib.h>
#include <stdio.h>
/* Main function starts here */
void main ( )
{
    /* Declaration of variables */
    float Sum = 0.0, N = 0.0;
    int Count = 0;
    for (Count = 0; Count < 10; Count++)
    {
        printf("Give a number:");
        scanf("%f", N);
        Sum += N;
    }
    printf("THE SUM OF GIVEN NUMBERS = %f", &Sum);
}
```
C++

- Named C++ as ++ is increment operator and C language is incremented to its next level with C++
- Developed by Bjarne Stroustrup at Bell Labs in the early 1980s
- Contains all elements of the basic C language
- Expanded to include numerous object-oriented programming features
- Provides a collection of predefined classes, along with the capability of user-defined classes

Java

- Development started at Sun Microsystems in 1991 by a team led by James Gosling
- Developed to be similar to C++ with fewer features to keep it simple and easy to use
- Compiled code is machine-independent and developed programs are simple to implement and use
- Uses just-in-time compilation
- Used in embedded systems such as hand-held devices, telephones and VCRs
- Comes in two variants – Java Runtime Engine (JRE) and Java Software Development Kit (SDK)
C# (C Sharp)

- Object-oriented programming language developed by Anders Hejlsberg and released by Microsoft as part of Microsoft’s .NET technology initiative
- Standardized by ECMA and ISO
- Syntactically and semantically very close to C++ and adopts various object-oriented features from both C++ and Java
- Compilers target the Common Language Infrastructure (CLI) implemented by Common Language Runtime (CLR) of .NET Framework
- CLR provides important services such as, memory management, exception handling, and security

RPG

- Stands for Report Program Generator
- Developed by IBM to meet customer requests for an easy and economic mechanism for producing reports
- Designed to generate the output reports resulting from the processing of common business applications
- Easier to learn and use as compared to COBOL
- Programmers use very detailed coding sheets to write specifications about input, calculations, and output
LISP

- Stands for **LIST** Processing
- Developed in 1959 by John McCarthy of MIT
- Designed to have features for manipulating non-numeric data, such as symbols and strings of text
- Due to its powerful list processing capability, it is extensively used in the areas of pattern recognition, artificial intelligence, and for simulation of games
- Functional programming language in which all computation is accomplished by applying functions to arguments

SNOBOL

- Stands for **String** Oriented sym**B**olic **L**anguage
- Used for non-numeric applications
- Powerful string manipulation features
- Widely used for applications in the area of text processing
Characteristics of a Good Programming Language

- Simplicity
- Naturalness
- Abstraction
- Efficiency
- Structured Programming Support
- Compactness
- Locality
- Extensibility
- Suitability to its environment

Factors for Selecting a Language for Coding an Application

- Nature of the application
- Familiarity with the language
- Ease of learning the language
- Availability of program development tools
- Execution efficiency
- Features of a good programming language
Subprogram

- Program written in a manner that it can be brought into use in other programs and used whenever needed without rewriting
- Also referred to as subroutine, sub-procedure, or function
- Subprogram call statement contains the name of the subprogram followed by a list of parameters enclosed within a pair of parentheses
- Intrinsic subprograms (also called built-in-functions) are those provided with the programming language
- Programmer-written subprograms are written and used as and when they are needed

Structure of a Subprogram

```
Subprogram name Parameter

sqrt (x)
```

Subprogram header

- Set of instructions that perform the intended task

Subprogram body
Flow of Control in Case of Subprogram Calls

1. Subprogram call statement
2. Subprogram header
3. Subprogram body
4. Next statement
5. Subprogram call statement
6. Next statement
7. Next statement
8. A subprogram
9. Flow of control
10. A program that calls the subprogram twice

Key Words/Phrases

- Assembler
- Assembly language
- BASIC
- Built-in function
- C
- C++
- C#
- COBOL
- Coding
- Compiler
- Computer language
- FORTRAN
- Function
- High-level language
- HotJava Interpreter
- Intrinsic subprogram
- Intermediate compiler and Interpreter
- Java
- Just-in-time compilation
- Language processor
- Linker
- LISP
- Load module
- Logic error
- Low-level language
- Machine language
- Macro instructions
- Object program
- Object-oriented programming
- Opcode
-Operand
- Pascal
- Programmer
- Programming
- Programming language
- Pseudo instruction
- RPG
- Self-documenting language

(Continued on next slide)
Key Words/Phrases

(Continued from previous slide...)

- SNOBOL
- Source program
- Sub-procedure
- Subprogram
- Subroutine
- Symbolic language
- Syntax error
- Syntax rules
- Written subprograms
Chapter 12
Computer Languages

Learning Objectives

In this chapter you will learn about:
- Computer languages or programming languages
- Three broad categories of programming languages – machine, assembly, and high-level languages
- Commonly used programming language tools such as assembler, compiler, linker, and interpreter
- Concepts of object-oriented programming languages
- Some popular programming languages such as FORTRAN, COBOL, BASIC, Pascal, C, C++, C#, Java, RPG, LISP and SNOBOL
- Related concepts such as Subprogram, Characteristics of a good programming language, and factors to consider while selecting a language for coding an application

Broad Classification of Computer Languages

- Machine language
- Assembly language
- High-level language
Machine Language

- Only language of a computer understood by it without using a translation program.
- Normally written as strings of binary 1s and 0s.
- Written using decimal digits if the circuitry of the computer being used permits this.

A Typical Machine Language Instruction Format

<table>
<thead>
<tr>
<th>OPCODE (operation code)</th>
<th>OPERAND (Address/Location)</th>
</tr>
</thead>
</table>

- OPCODE tells the computer which operation to perform from the instruction set of the computer.
- OPERAND tells the address of the data on which the operation is to be performed.

A Sample Machine Language Program

| 00100000000000000110111001 | 10001471 |
| 00110000000000100000110001 | 14002041 |
| 01100000000111101011001100 | 30003456 |
| 10100011111011001011100110 | 50773456 |
| 00000000000000000000000000 | 00000000 |

In Binary (Difficult to read and understand) In Decimal (Easier to read and understand)
### Advantages & Limitations of Machine Language

<table>
<thead>
<tr>
<th>Advantage</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be executed very fast</td>
<td>Machine Dependent</td>
</tr>
<tr>
<td></td>
<td>Difficult to program</td>
</tr>
<tr>
<td></td>
<td>Error prone</td>
</tr>
<tr>
<td></td>
<td>Difficult to modify</td>
</tr>
</tbody>
</table>

### Assembly/Symbolic Language

Programming language that overcomes the limitations of machine language programming by:

- Using alphanumeric mnemonic codes instead of numeric codes for the instructions in the instruction set e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add
- Allowing storage locations to be represented in form of alphanumeric addresses instead of numeric addresses e.g. representing memory locations 1000, 1001, and 1002 as FRST, SCND, and ANSR respectively
- Providing pseudo-instructions that are used for instructing the system how we want the program to be assembled inside the computer’s memory e.g. START PROGRAM AT 0000; SET ASIDE AN ADDRESS FOR FRST

### Assembler

Software that translates as assembly language program into an equivalent machine language program of a computer.

```
<table>
<thead>
<tr>
<th>Assembly language program</th>
<th>Input</th>
<th>Output</th>
<th>Machine language program</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Source Program)</td>
<td></td>
<td></td>
<td>(Object Program)</td>
</tr>
</tbody>
</table>
```
A subset of the set of instructions supported by a computer

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLT</td>
<td>00</td>
<td>Halt, used at the end of program to stop</td>
</tr>
<tr>
<td>CLA</td>
<td>10</td>
<td>Clear and add into A register</td>
</tr>
<tr>
<td>ADD</td>
<td>14</td>
<td>Add to the contents of A register</td>
</tr>
<tr>
<td>SUB</td>
<td>15</td>
<td>Subtract from the contents of A register</td>
</tr>
<tr>
<td>STA</td>
<td>30</td>
<td>Store A register</td>
</tr>
</tbody>
</table>

A subset of the set of instructions supported by a computer

Sample assembly language program for adding two numbers and storing the result

<table>
<thead>
<tr>
<th>Symbolic name</th>
<th>Memory location</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRST</td>
<td>1000</td>
</tr>
<tr>
<td>SCND</td>
<td>1001</td>
</tr>
<tr>
<td>ANSR</td>
<td>1002</td>
</tr>
</tbody>
</table>

Mapping table set up by the assembler for the data items of the assembly language program
An Example of Assembly Language Program

<table>
<thead>
<tr>
<th>Memory Location</th>
<th>Contents</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>10</td>
<td>1000</td>
</tr>
<tr>
<td>0001</td>
<td>14</td>
<td>1001</td>
</tr>
<tr>
<td>0002</td>
<td>30</td>
<td>1002</td>
</tr>
<tr>
<td>0003</td>
<td>00</td>
<td>Reserved for ANSR</td>
</tr>
<tr>
<td>1000</td>
<td>Reserved for FRST</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>Reserved for SCND</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>Reserved for ANSR</td>
<td></td>
</tr>
</tbody>
</table>

Equivalent machine language program for the assembly language program

Advantages of Assembly Language Over Machine Language

- Easier to understand and use
- Easier to locate and correct errors
- Easier to modify
- No worry about addresses
- Easily relocatable
- Efficiency of machine language

Limitations of Assembly Language

- Machine dependent
- Knowledge of hardware required
- Machine level coding
Typical Uses of Assembly Language

- Mainly used today to fine-tune important parts of programs written in a high-level language to improve the program’s execution efficiency.

Assembly Languages with Macro Instructions

- Any assembly language instruction that gets translated into several machine language instructions is called a macro instruction.
- Several assembly languages support such macro instructions to speed up the coding process.
- Assemblers of such assembly languages are designed to produce multiple machine language instructions for each macro instruction of the assembly language.

High-Level Languages

- Machine independent.
- Do not require programmers to know anything about the internal structure of the computer on which high-level language programs will be executed.
- Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions.
A translator program (software) that translates a high-level language program into its equivalent machine language program.

Compiles a set of machine language instructions for every program instruction in a high-level language.

The one-to-many correspondence between high-level language programs and machine language programs demonstrates the requirement of a separate compiler for each high-level language supported by a computer.
Compiler

Illustrating the machine independence characteristic of a high-level language. Separate compilers are required for the same language on different computers.

Syntax Errors

In addition to doing translation job, compilers also automatically detect and indicate syntax errors. Syntax errors are typically of following types:

- Illegal characters
- Illegal combination of characters
- Improper sequencing of instructions in a program
- Use of undefined variable names

Note: A compiler cannot detect logic errors in a program.

The Process of Removing Syntax Errors From A Source Program

START

Edit source program

Compile source program

Syntax errors detected?

Yes

No

Generate object program

Stop

Yes

Generate list of coded error messages

Object program

Stop
### Linker

6. For a large software, storing all the lines of program code in a single source file will be:
   - Difficult to work with
   - Difficult to deploy multiple programmers to concurrently work towards its development
   - Any change in the source program would require the entire source program to be recompiled

6. Hence, a modular approach is generally adapted to develop large software where the software consists of multiple source program files
6. No need to write programs for some modules as it might be available in library offering the same functionality

---

### Linker

6. Each source program file can be independently modified and compiled to create a corresponding object program file
6. Linker program (software) is used to properly combine all the object program files (modules)
6. Creates the final executable program (load module)

---

### Interpreter

6. Interpreter is a high-level language translator
6. Takes one statement of a high-level language program, translates it into machine language instructions
6. Immediately executes the resulting machine language instructions
6. Compiler simply translates the entire source program into an object program and is not involved in its execution
**Role of an Interpreter**

- High-level language program (Source Program) \(\xrightarrow{\text{Input}}\) Interpreter (translates and executes statement-by-statement) \(\xrightarrow{\text{Output}}\) Result of program execution

**Intermediate Language Compiler & Interpreter**

- New type of compiler and interpreter combines the speed, ease, and control of both compiler and interpreter.
- Compiler first compiles the source program to an intermediate object program.
- Intermediate object program is not a machine language code but written in an intermediate language that is virtually machine independent.
- Interpreter takes intermediate object program, converts it into machine language program and executes it.

**Benefits of Intermediate Language Compiler & Interpreter**

- Intermediate object program is in compiled form and thus is not original source code, so safer and easier to share.
- Intermediate object program is based on a standard Intermediate Definition Language (IDL).
- Interpreter can be written for any computer architecture and operating system providing virtual machine environment to the executing program.
- Newer Interpreter compiles intermediate program, in memory, into final host machine language program and executes it.
- This technique is called Just-In-Time (JIT) Compilation.
Advantages of High-Level Languages

- Machine independent
- Easier to learn and use
- Fewer errors during program development
- Lower program preparation cost
- Better documentation
- Easier to maintain

Limitations of High-Level Languages

- Lower execution efficiency
- Less flexibility to control the computer’s CPU, memory and registers

Object-Oriented Programming Languages

- Programming languages are used for simulating real-world problems on computers
- Much of the real world is made up of objects
- Essence of OOP is to solve a problem by:
  - Identifying the real-world objects of the problem
  - Identifying processing required of them
  - Creating simulations of objects, processes, and their communications
FORTRAN

- Stands for FORmula TRANslation
- Originally developed by John Backus and his team at IBM followed by several revisions
- Standardized by ANSI as FORTRAN-77 and FORTRAN-90
- Designed for solving scientific & engineering problems
- Oriented towards solving problems of a mathematical nature
- Popular language amongst scientists and engineers

A Sample FORTRAN Program

```
C FORTRAN PROGRAM TO COMPUTE
C THE SUM OF 10 NUMBERS
SUM = 0
DO 50 I = 1, 10
READ (5, 10) N
SUM = SUM + N
50 CONTINUE
WRITE (6, 20) SUM
10 FORMAT (F6.2)
20 FORMAT (1X, 'THE SUM OF GIVEN NUMBERS = ', F10.2)
STOP
END
```

COBOL

- Stands for COmmom Business Oriented Language
- Originally developed started under Grace Hopper followed by COnference on DAta Systems Languages (CODASYL)
- Standardized by ANSI as COBOL-74, COBOL-85, and COBOL-2002
- Designed for programming business data processing applications
- Designed to have the appearance and structure of a business report written in English, hence often referred to as a self-documenting language
**A Sample COBOL Program**

```cobol
IDENTIFICATION DIVISION.
PROGRAM-ID. SUMUP.
AUTHOR. P K SINHA.
* THIS PROGRAM COMPUTES AND PRINTS
  * THE SUM OF GIVEN NUMBERS.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE_COMPUTER. BURROUGHS_6700.
OBJECT_COMPUTER. BURROUGHS_6700.
INPUT_OUTPUT SECTION.
FILE_CONTROL.
  SELECT DATA_FILE ASSIGN TO DISK.
  SELECT OUTPUT_FILE ASSIGN TO PRINTER.
DATA DIVISION.
FILE SECTION.
FD DATA_FILE
  RECORD CONTAINS 80 CHARACTERS
  LABEL RECORD IS OMITTED
  DATA RECORD IS INPUT_DATA_RECORD.
01 INPUT_DATA_RECORD.
  05 N PICTURE 9(6)V99.
  05 FILLER PICTURE X(72).
FD OUTPUT_FILE
  RECORD CONTAINS 132 CHARACTERS
  LABEL RECORD IS OMITTED
  DATA RECORD IS OUTPUT_RECORD.
01 OUTPUT_RECORD.
    05 FILLER PICTURE X.
    05 TITLE PICTURE X(25).
    05 SUM PICTURE 9(10)V99.
    05 FILLER PICTURE X(94).
WORKING STORAGE SECTION.
77 MESSAGE PICTURE X(25).
VALUE IS "THE SUM OF GIVEN NUMBERS=".
PROCEDURE DIVISION.
OPEN_FILES.
  OPEN INPUT DATA_FILE.
  OPEN OUTPUT OUTPUT_FILE.
INITIALIZATION.
  MOVE SPACES TO OUTPUT_RECORD.
  MOVE ZERO TO SUM.
```
A Sample COBOL Program

PROCESS_LOOP.
READ DATA_FILE AT END GO TO
PRINT_PARA.
   ADD N TO SUM.
   GO TO PROCESS_LOOP.
PRINT_PARA.
   MOVE MESSAGE TO TITLE.
   WRITE OUTPUT_RECORD.
END_OFJOB.
   CLOSE DATA_FILE.
   CLOSE OUTPUT_FILE.
   STOP RUN.

A Sample BASIC Program

REM PROGRAM TO COMPUTE
REM THE SUM OF 10 NUMBERS
10 LET S = 0
20 FOR I = 1 TO 10
30 READ N
40 LET S = S + N
50 NEXT I
60 PRINT "THE SUM OF GIVEN NUMBERS = "; S
70 DATA 4, 20, 15, 32, 48
80 DATA 12, 3, 9, 14, 44
90 END;

BASIC

§ Stands for Beginners All-purpose Symbolic Instruction Code
§ Developed by Professor John Kemeny and Thomas Kurtz at Darmouth College in the United States
§ Standardized by ANSI as BASIC-78
§ Designed to be an interactive language and to use an interpreter instead of a compiler
§ Simple to implement, learn and use language. Hence, it is a widely used language on personal computers
§ Flexible and reasonably powerful language and can be used for both business and scientific applications
### Pascal

- Named after the famous seventeenth-century French mathematician Blaise Pascal
- Developed by Professor Nicklaus Wirth of Federal Institute of Technology in Zurich
- Encourages programmers to write well-structured, modular programs, instills good program practices
- Recognized as an educational language and is used to teach programming to beginners
- Suitable for both scientific & business applications
- Has features to manipulate numbers, vectors, matrices, strings, sets, records, files, and lists

#### A Sample Pascal Program

```pascal
PROGRAM SUMNUMS (INPUT, OUTPUT); (* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS *)
(* DECLARATION OF VARIABLES *)
VAR SUM, N : REAL;
VAR I : INTEGER;
(* MAIN PROGRAM LOGIC STARTS HERE *)
BEGIN
  SUM := 0;
  FOR I := 1 TO 10 DO
    BEGIN
      READ (N);
      SUM := SUM + N;
    END;
  WRITELN ('THE SUM OF GIVEN NUMBERS=', SUM);
END;
```

### C

- Developed in 1972 at AT&T’s Bell laboratories, USA by Dennis Ritchie and Brian Kernighan
- Standardized by ANSI and ISO as C89, C90, C99
- High-level programming languages (mainly machine independence) with the efficiency of an assembly language
- Language of choice of programmers for portable systems software and commercial software packages like OS, compiler, spreadsheet, word processor, and database management systems
/* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS */
/* Directives to include standard library and header */
#include <stdlib.h>
#include <stdio.h>
/* Main function starts here */
void main ( )
{
    /* Declaration of variables */
    float Sum = 0.0, N = 0.0;
    int Count = 0;
    for (Count = 0; Count < 10; Count++)
    {
        printf("Give a number:");
        scanf("%f", N);
        Sum += N;
    }
    printf("THE SUM OF GIVEN NUMBERS = %f", &Sum);
}
C++
- Named C++ as ++ is increment operator and C language is incremented to its next level with C++
- Developed by Bjarne Stroustrup at Bell Labs in the early 1980s
- Contains all elements of the basic C language
- Expanded to include numerous object-oriented programming features
- Provides a collection of predefined classes, along with the capability of user-defined classes
Java
- Development started at Sun Microsystems in 1991 by a team led by James Gosling
- Developed to be similar to C++ with fewer features to keep it simple and easy to use
- Compiled code is machine-independent and developed programs are simple to implement and use
- Uses just-in-time compilation
- Used in embedded systems such as hand-held devices, telephones and VCRs
- Comes in two variants – Java Runtime Engine (JRE) and Java Software Development Kit (SDK)
C# (C Sharp)

- Object-oriented programming language developed by Anders Hejlsberg and released by Microsoft as part of Microsoft's .NET technology initiative.
- Standardized by ECMA and ISO.
- Syntactically and semantically very close to C++ and adopts various object-oriented features from both C++ and Java.
- Compilers target the Common Language Infrastructure (CLI) implemented by Common Language Runtime (CLR) of .NET Framework.
- CLR provides important services such as, memory management, exception handling, and security.

RPG

- Stands for Report Program Generator.
- Developed by IBM to meet customer requests for an easy and economic mechanism for producing reports.
- Designed to generate the output reports resulting from the processing of common business applications.
- Easier to learn and use as compared to COBOL.
- Programmers use very detailed coding sheets to write specifications about input, calculations, and output.

LISP

- Stands for List Processing.
- Developed in 1959 by John McCarthy of MIT.
- Designed to have features for manipulating non-numeric data, such as symbols and strings of text.
- Due to its powerful list processing capability, it is extensively used in the areas of pattern recognition, artificial intelligence, and for simulation of games.
- Functional programming language in which all computation is accomplished by applying functions to arguments.
SNOBOL.

- Stands for **S**tri**n**g **O**riented **s**ym**B**ollic **L**anguage
- Used for non-numeric applications
- Powerful string manipulation features
- Widely used for applications in the area of text processing

Characteristics of a Good Programming Language

- Simplicity
- Naturalness
- Abstraction
- Efficiency
- Structured Programming Support
- Compactness
- Locality
- Extensibility
- Suitability to its environment

Factors for Selecting a Language for Coding an Application

- Nature of the application
- Familiarity with the language
- Ease of learning the language
- Availability of program development tools
- Execution efficiency
- Features of a good programming language
Subprogram

1. Program written in a manner that it can be brought into use in other programs and used whenever needed without rewriting
2. Also referred to as subroutine, sub-procedure, or function
3. Subprogram call statement contains the name of the subprogram followed by a list of parameters enclosed within a pair of parentheses
4. Intrinsic subprograms (also called built-in-functions) are those provided with the programming language
5. Programmer-written subprograms are written and used as and when they are needed

Structure of a Subprogram

- Subprogram name
- Parameter

Subprogram header

- Set of instructions that perform the intended task

Subprogram body

Flow of Control in Case of Subprogram Calls

- A subprogram
- Flow of control
- A program that calls the subprogram twice
<table>
<thead>
<tr>
<th>Key Words/Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>§ Assembler</td>
</tr>
<tr>
<td>§ Assembly language</td>
</tr>
<tr>
<td>§ BASIC</td>
</tr>
<tr>
<td>§ Built-in function</td>
</tr>
<tr>
<td>§ C</td>
</tr>
<tr>
<td>§ C++</td>
</tr>
<tr>
<td>§ COBOL</td>
</tr>
<tr>
<td>§ Coding</td>
</tr>
<tr>
<td>§ Compiler</td>
</tr>
<tr>
<td>§ Computer language</td>
</tr>
<tr>
<td>§ FORTRAN</td>
</tr>
<tr>
<td>§ Function</td>
</tr>
<tr>
<td>§ High-level language</td>
</tr>
<tr>
<td>§ HotJava Interpreter</td>
</tr>
<tr>
<td>§ Intermediate compiler and Interpreter</td>
</tr>
<tr>
<td>§ Java</td>
</tr>
<tr>
<td>§ Task-in-time compilation</td>
</tr>
<tr>
<td>§ Language processor</td>
</tr>
<tr>
<td>§ LiSsP</td>
</tr>
<tr>
<td>§ Load module</td>
</tr>
<tr>
<td>§ Logic error</td>
</tr>
<tr>
<td>§ Low-level language</td>
</tr>
<tr>
<td>§ Machine language</td>
</tr>
<tr>
<td>§ Macro instructions</td>
</tr>
<tr>
<td>§ Object program</td>
</tr>
<tr>
<td>§ Object-oriented programming</td>
</tr>
<tr>
<td>§ Opcode</td>
</tr>
<tr>
<td>§ Operator</td>
</tr>
<tr>
<td>§ Programmer</td>
</tr>
<tr>
<td>§ Programming</td>
</tr>
<tr>
<td>§ Pseudo instruction</td>
</tr>
<tr>
<td>§ RFG</td>
</tr>
<tr>
<td>§ Self-documenting language</td>
</tr>
</tbody>
</table>

(Continued from previous slide.)
Chapter 13
System Implementation and Operation

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
In this chapter you will learn about:

- Main activities of implementation and operation phase
- Testing and debugging of programs
- Complete documentation of the system
- Change over to the new system
- System evaluation and
- System maintenance
Testing and Debugging

- Program errors are known as *bugs*.
- Process of detecting and correcting these errors is called *debugging*.
- *Testing* is the process of making sure that the program performs the intended task.
- *Debugging* is the process of locating and eliminating program errors.
Types of Program Errors

- **Syntax errors**
  - Occurs when the rules or syntax of the programming language are not followed
  - For example, incorrect punctuation, incorrect word sequence, undefined terms, and misuse of terms
  - Syntax errors are detected by a language processor

- **Logic errors**
  - Occurs due to errors in planning a program’s logic
  - Such errors cause the program to produce incorrect output.
  - These errors cannot be detected by a language processor
Testing of a Program

- Testing procedure involves running program to process input test data, and comparing obtained results with correct results.

- Test data must test each logical function of the program, and should include all types of possible valid and invalid data.

- Program internally released for testing is known as *alpha version* and the test conducted on it is called *alpha testing*.

- Program released for additional testing to a selected set of external users is *beta version* and test conducted on it called is *beta testing*.
Debugging a Program for Syntax Errors

- Relatively easier to detect and correct syntax errors than logic errors in a program
- Language processors are designed to automatically detect syntax errors
- Single syntax error often causes multiple error messages to be generated by the language processor
- Removal of the syntax error will result in the removal of all associated error messages
Logic errors are more difficult to detect than syntax errors as the computer does not produce any error message for such errors.

One or more of the following methods are commonly used for locating logic errors:

1. Doing hand simulation of the program code
2. Putting print statements in the program code
3. Using a debugger (a software tool that assists a programmer in following the program’s execution step-by-step)
4. Using memory dump (printout of the contents of main memory and registers)
### Difference Between Testing and Debugging

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Testing</th>
<th>Debugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Testing is the process of validating the correctness of a program. Its objective is to demonstrate that the program meets its design specifications.</td>
<td>Debugging is the process of eliminating errors in a program. Its objective is to detect the exact cause and remove known errors in the program.</td>
</tr>
<tr>
<td>2</td>
<td>Testing is complete when all desired verifications against specifications have been performed.</td>
<td>Debugging is complete when all known errors in the program have been fixed. Not that debugging process ends only temporarily as it must be restarted whenever a new error is found in the program.</td>
</tr>
</tbody>
</table>

(Continued on next slide)
### Difference Between Testing and Debugging

(Continued from previous slide..)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Testing</th>
<th>Debugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>➥ Testing is a definable process which can and should be planned and scheduled properly.</td>
<td>➥ Debugging being a reactive process cannot be planned ahead of time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>➥ It must be carried out as and when errors are found in a program.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>➥ Testing can begin in the early stages of software development.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>➥ Although the test runs of a program can be done only after the program is coded, but the decision of what to test, how to test, and with what kind of data to test, can and should be done before the coding is started.</td>
</tr>
</tbody>
</table>
Process of collecting, organizing, storing, and otherwise maintaining a complete historical record of programs and other documents used or prepared during the different phases of the life cycle of a software.

Three commonly used forms of documentation are:

- Program comments
- System manual
- User manual
Changeover to the New System

- When a software is ready for use, it is deployed at site for use by the intended users.
- At this stage, a changeover from the old system of operation to the new system takes place.
- Three normally followed methods to carry out the changeover process are:
  - Immediate changeover
  - Parallel run
  - Phased conversion
Changeover to the New System

(a) Immediate changeover

Old system

Cut-off date

New system

Time
Changeover to the New System

Old system in operation

New system in operation

Overlapping period of complete operation of both the old and the new systems

(b) Parallel run
Changeover to the New System

Old system completely operational → Old and new systems in operation in parts → New system completely operational

(c) Phased conversion

Ref Page 247  Chapter 13: System Implementation and Operation  Slide 14/17
System Evaluation

- Process of evaluating a system (after it is put in operation) to verify whether or not it is meeting its objectives

- Points normally considered for evaluating a system are:
  - Performance evaluation
  - Cost analysis
  - Time analysis
  - User satisfaction
  - Ease of modification
  - Failure rate
System Maintenance

- Process of incorporating changes in an existing system to enhance, update, or upgrade its features
- On an average, maintenance cost of a computerized system is two to four times more than the initial development cost
<table>
<thead>
<tr>
<th>Key Words/Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beta testing</td>
</tr>
<tr>
<td>Bugs</td>
</tr>
<tr>
<td>Comments</td>
</tr>
<tr>
<td>Debugger</td>
</tr>
<tr>
<td>Debugging</td>
</tr>
<tr>
<td>Documentation</td>
</tr>
<tr>
<td>Immediate changeover</td>
</tr>
<tr>
<td>Logic errors</td>
</tr>
<tr>
<td>Memory dump</td>
</tr>
<tr>
<td>Parallel run</td>
</tr>
<tr>
<td>Phased conversion</td>
</tr>
<tr>
<td>Syntax errors</td>
</tr>
<tr>
<td>System evaluation</td>
</tr>
<tr>
<td>System maintenance</td>
</tr>
<tr>
<td>System manual</td>
</tr>
<tr>
<td>Testing</td>
</tr>
<tr>
<td>User manual</td>
</tr>
</tbody>
</table>
In this chapter you will learn about:

- Main activities of implementation and operation phase
- Testing and debugging of programs
- Complete documentation of the system
- Change over to the new system
- System evaluation and
- System maintenance
Testing and Debugging

- Program errors are known as *bugs*.
- Process of detecting and correcting these errors is called *debugging*.
- *Testing* is the process of making sure that the program performs the intended task.
- *Debugging* is the process of locating and eliminating program errors.

Types of Program Errors

- **Syntax errors**
  - Occurs when the rules or syntax of the programming language are not followed.
  - For example, incorrect punctuation, incorrect word sequence, undefined terms, and misuse of terms.
  - Syntax errors are detected by a language processor.

- **Logic errors**
  - Occurs due to errors in planning a program’s logic.
  - Such errors cause the program to produce incorrect output.
  - These errors cannot be detected by a language processor.
Testing of a Program

- Testing procedure involves running program to process input test data, and comparing obtained results with correct results.
- Test data must test each logical function of the program, and should include all types of possible valid and invalid data.
- Program internally released for testing is known as *alpha version* and the test conducted on it is called *alpha testing*.
- Program released for additional testing to a selected set of external users is *beta version* and test conducted on it called is *beta testing*.

Debugging a Program for Syntax Errors

- Relatively easier to detect and correct syntax errors than logic errors in a program.
- Language processors are designed to automatically detect syntax errors.
- Single syntax error often causes multiple error messages to be generated by the language processor.
- Removal of the syntax error will result in the removal of all associated error messages.
Debugging a Program for Logic Errors

- Logic errors are more difficult to detect than syntax errors as computer does not produce any error message for such errors.
- One or more of following methods are commonly used for locating logic errors:
  - Doing hand simulation of the program code
  - Putting print statements in the program code
  - Using a debugger (a software tool that assists a programmer in following the program’s execution step-by-step)
  - Using memory dump (printout of the contents of main memory and registers)

Difference Between Testing and Debugging

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Testing</th>
<th>Debugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Testing is the process of validating the correctness of a program. Its objective is to demonstrate that the program meets its design specifications.</td>
<td>Debugging is the process of eliminating errors in a program. Its objective is to detect the exact cause and remove known errors in the program.</td>
</tr>
<tr>
<td>2</td>
<td>Testing is complete when all desired verifications against specifications have been performed.</td>
<td>Debugging is complete when all known errors in the program have been fixed. Note that debugging process ends only temporarily as it must be restarted whenever a new error is found in the program.</td>
</tr>
</tbody>
</table>
### Difference Between Testing and Debugging

(Continued from previous slide.)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Testing</th>
<th>Debugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Testing is a definable process which can and should be planned and scheduled properly</td>
<td>Debugging being a reactive process cannot be planned ahead of time. It must be carried out as and when errors are found in a program</td>
</tr>
<tr>
<td>4</td>
<td>Testing can begin in the early stages of software development. Although the test runs of a program can be done only after the program is coded, but the decision of what to test, how to test, and with what kind of data to test, can and should be done before the coding is started.</td>
<td>Debugging can begin only after the program is coded. The approach used for debugging largely depends on the personal choice of the programmer and the type of problem in the program.</td>
</tr>
</tbody>
</table>

### Documentation

- Process of collecting, organizing, storing, and otherwise maintaining a complete historical record of programs and other documents used or prepared during the different phases of the life cycle of a software.
- Three commonly used forms of documentation are:
  - Program comments
  - System manual
  - User manual

---

Ref Page 242  Chapter 13: System Implementation and Operation  Slide 9/17

Ref Page 243  Chapter 13: System Implementation and Operation  Slide 10/17
When a software is ready for use, it is deployed at site for use by the intended users. At this stage, a changeover from the old system of operation to the new system takes place. Three normally followed methods to carry out the changeover process are:

- Immediate changeover
- Parallel run
- Phased conversion
**Changeover to the New System**

(b) Parallel run

Overlapping period of complete operation of both the old and the new systems

(c) Phased conversion

Old system completely operational

Old and new systems in operation in parts

New system completely operational

Old system

New system

Time
System Evaluation

- Process of evaluating a system (after it is put in operation) to verify whether or not it is meeting its objectives
- Points normally considered for evaluating a system are:
  - Performance evaluation
  - Cost analysis
  - Time analysis
  - User satisfaction
  - Ease of modification
  - Failure rate

System Maintenance

- Process of incorporating changes in an existing system to enhance, update, or upgrade its features
- On an average, maintenance cost of a computerized system is two to four times more than the initial development cost
## Key Words/Phrases

- Beta testing
- Bugs
- Comments
- Debugger
- Debugging
- Documentation
- Immediate changeover
- Logic errors
- Memory dump
- Parallel run
- Phased conversion
- Syntax errors
- System evaluation
- System maintenance
- System manual
- Testing
- User manual
In this chapter you will learn about:

- Main activities of the implementation and operation phase
- Testing and debugging of programs
- Complete documentation of the system
- Change over to the new system
- System evaluation and
- System maintenance

Testing and Debugging

- Program errors are known as bugs
- Process of detecting and correcting these errors is called debugging
- Testing is the process of making sure that the program performs the intended task
- Debugging is the process of locating and eliminating program errors
Types of Program Errors

- **Syntax errors**
  - Occurs when the rules or syntax of the programming language are not followed
  - For example, incorrect punctuation, incorrect word sequence, undefined terms, and misuse of terms
  - Syntax errors are detected by a language processor

- **Logic errors**
  - Occurs due to errors in planning a program's logic
  - Such errors cause the program to produce incorrect output.
  - These errors cannot be detected by a language processor

Testing of a Program

- Testing procedure involves running program to process input test data, and comparing obtained results with correct results
- Test data must test each logical function of the program, and should include all types of possible valid and invalid data
- Program internally released for testing is known as alpha version and the test conducted on it is called alpha testing
- Program released for additional testing to a selected set of external users is beta version and test conducted on it called is beta testing

Debugging a Program for Syntax Errors

- Relatively easier to detect and correct syntax errors than logic errors in a program
- Language processors are designed to automatically detect syntax errors
- Single syntax error often causes multiple error messages to be generated by the language processor
- Removal of the syntax error will result in the removal of all associated error messages
### Debugging a Program for Logic Errors

- Logic errors are more difficult to detect than syntax errors as computer does not produce any error message for such errors.

### Methods for Locating Logic Errors:
- Doing hand simulation of the program code
- Putting print statements in the program code
- Using a debugger (a software tool that assists a programmer in following the program’s execution step-by-step)
- Using memory dump (printout of the contents of main memory and registers)

### Difference Between Testing and Debugging

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Testing</th>
<th>Debugging</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Testing is the process of validating the correctness of a program. Its objective is to demonstrate that the program meets its design specifications.</td>
<td>Debugging is the process of eliminating errors in a program. Its objective is to detect the exact cause and remove known errors in the program.</td>
</tr>
<tr>
<td>2</td>
<td>Testing is complete when all desired verifications against specifications have been performed.</td>
<td>Debugging is complete when all known errors in the program have been fixed. Note that debugging process ends only temporarily as it must be restarted whenever a new error is found in the program.</td>
</tr>
</tbody>
</table>

### Additional Points:
- Debugging can begin only after the program is coded.
- Testing can begin in the early stage of software development.
- Although the test runs of a program can be done even after the program is coded, the decision of what to test, how to test, and with what kind of data to test, can and should be done before the coding is started.
Documentation

- Process of collecting, organizing, storing, and otherwise maintaining a complete historical record of programs and other documents used or prepared during the different phases of the life cycle of a software.
- Three commonly used forms of documentation are:
  - Program comments
  - System manual
  - User manual

Changeover to the New System

- When a software is ready for use, it is deployed at site for use by the intended users.
- At this stage, a changeover from the old system of operation to the new system takes place.
- Three normally followed methods to carry out the changeover process are:
  - Immediate changeover
  - Parallel run
  - Phased conversion

Changeover to the New System

- Old system in operation
- New system in operation

Old system
New system
Cut-off date

(a) Immediate changeover
Slide 13/17

**Chapter 13: System Implementation and Operation**

**Changeover to the New System**

1. **Old system** in operation
2. **New system** in operation
3. Overlapping period of complete operation of both the old and the new systems

**Parallel run**

- **Old system** in operation
- **New system** in operation

**Old system** completely operational

**New system** completely operational

**Phased conversion**

- **Old system** in operation in parts
- **New system** in operation in parts

**System Evaluation**

- Process of evaluating a system (after it is put in operation) to verify whether or not it is meeting its objectives
- Points normally considered for evaluating a system are:
  - Performance evaluation
  - Cost analysis
  - Time analysis
  - User satisfaction
  - Ease of modification
  - Failure rate
System Maintenance

§ Process of incorporating changes in an existing system to enhance, update, or upgrade its features

§ On an average, maintenance cost of a computerized system is two to four times more than the initial development cost

Key Words/Phrases

§ Beta testing
§ Bugs
§ Comments
§ Debugger
§ Debugging
§ Documentation
§ Immediate changeover
§ Logic errors
§ Memory dump
§ Parallel run
§ Phased conversion
§ Syntax errors
§ System evaluation
§ System maintenance
§ System manual
§ Testing
§ User manual
Chapter 14

Operating Systems

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
In this chapter you will learn about:

- Definition and need for operating system
- Main functions of an operating system
- Commonly used mechanisms for:
  - Process management
  - Memory management
  - File management
  - Security
  - Command interpretation module
- Some commonly used OS capability enhancement software
- Some popular operating systems
Definition and Need for OS

- Integrated set of programs that controls the resources (the CPU, memory, I/O devices, etc.) of a computer system
- Provides its users with an interface or virtual machine that is more convenient to use than the bare machine
- Two primary objectives of an OS are:
  - Making a computer system convenient to use
  - Managing the resources of a computer system
The operating system layer hides the details of the hardware from the programmer and provides the programmer with convenient interface for using the system.
Main Functions of an OS

- Process management
- Memory management
- File management
- Security
- Command interpretation
Parameters for Measuring System Performance

- **Throughput**: Amount of work that the system is able to do per unit time
- **Turnaround time**: Interval from the time of submission of a job to the system for processing to the time of completion of the job
- **Response time**: Interval from the time of submission of a job to the system for processing to the time the first response for the job is produced by the system
A process (also called job) is a program in execution.

Process management manages the processes submitted to a system in a manner to minimize idle time of processors (CPUs, I/O processors, etc.) of the system.
Process Management Mechanisms in Early Systems

- **Manual loading mechanism:** Jobs were manually loaded one after another in a computer by the computer operator.

- **Batch processing mechanism:** Batch of jobs was submitted together to the computer and job-to-job transition was done automatically by the operating system.

- **Job Control Language (JCL):** Control statements were used to facilitate job loading and unloading.
Use of Job Control Statements in Batch Processing (An Example)

- $JOB, ONGC05839, USER=SINHA
- $COBOL
- $LOAD
- $RUN
- $END

Data for program

COBOL program
Multiprogramming

- **Uniprogramming**: Only one job is processed at a time and all system resources are available exclusively for the job until its completion.

- **Multiprogramming**: Interleaved execution of two or more different and independent programs by a computer.

- **Types of Multiprogramming**:
  - **Multiprogramming with fixed tasks (MFT)**: Fixed number of jobs can be processed concurrently.
  - **Multiprogramming with variable tasks (MVT)**: Number of jobs can vary.

- **Area occupied by each job residing simultaneously in the main memory is known as a memory partition**.
- **CPU bound**: Mostly perform computations with little I/O operations. Scientific and engineering computations usually fall in this category.

- **I/O bound**: Mostly perform I/O operations with little computation. Commercial data processing applications usually fall in this category.
Uniprogramming System

Only one job is processed by the system at a time and all the system resources are exclusively available for the job until it completes.
Multiprogramming System

Secondary disk storage

Writing output data

Execution in progress

Main memory

Operating system

Job A

Job B

Job C (Waiting for CPU)

CPU
Process States in Multiprogramming

- **New job** → **Ready**
  - Job is allocated the CPU for execution
- **Ready** → **Running**
  - Job processing completed
- **Running** → **Blocked**
  - Job must wait for I/O completion
- **Blocked** → **Ready**
  - I/O completed

Ref Page 257  Chapter 14: Operating Systems  Slide 14/54
Requirements of Multiprogramming Systems

- Large memory
- Memory protection
- Job status preservation
- Proper job mix (CPU and I/O bound jobs)
- CPU scheduling
Process Control Block (PCB)

- process identifier
- process state
- program counter
- values of various CPU registers
- accounting and scheduling information
- I/O status information

PCB is used to preserve the job status of each loaded process in a multiprogramming system.
Multitasking

- Interleaved execution of multiple jobs (often referred to as *tasks* of same user) in a single-user system
- Computer systems used for multitasking are uniprocessor systems (having only one CPU)
- Treated differently from multiprogramming that refers to interleaved execution of multiple jobs in a multi-user system
Multithreading

- Thread is basic unit of CPU utilization. Threads share a CPU in the same way as processes do.
- All threads of a process also share the same set of operating system resources.
- All threads of a process inherit parent’s address space and security parameters.
- Each thread of a process has its own program counter, its own register states, and its own stack.
- Referred as mini-process or lightweight process.
(a) Single-threaded and (b) multithreaded processes. A single-threaded process corresponds to a process of a traditional operating system. [Reproduced with permission, from the book titled Distributed Operating Systems: Concepts and Design by Pradeep K. Sinha. © 1997 IEEE, USA].
Multiprocessing

- System with two or more CPUs having ability to execute multiple processes concurrently
- Multiple CPUs are used to process either instructions from different and independent programs or different instructions from the same program simultaneously
- Types of multiprocessing:
  - *Tightly-coupled*: Single system-wide primary memory shared by all processors
  - *Loosely-coupled*: Each processor has its own local memory
CPU, Memory, and I/O Processors of a Computer System

- I/O Units
- I/O Processors
- Main memory
- CPU
Simultaneous interactive use of a computer system by many users in such a way that each one feels that he/she is the sole user of the system

User terminals connected to the same computer simultaneously

Uses multiprogramming with a special CPU scheduling algorithm

Short period during which a user process gets to use CPU is known as time slice, time slot, or quantum

CPU is taken away from a running process when the allotted time slice expires
Process State Diagram for a Time-Sharing System

- **New Job** → **Ready**
  - Job is allocated to CPU for execution
- **Ready** → **Running**
  - Allotted time slice is over
- **Running** → **Job processing completed**
  - Job must wait for I/O completion
- **Blocked** → **Ready**
  - I/O completed
Advantages of Time-sharing Systems

- Reduces CPU idle time
- Provides advantages of quick response time
- Offers good computing facility to small users
Memory is an important resource of a computer system that must be properly managed for the overall system performance.

Memory management module:
- Keeps track of parts of memory in use and parts not in use.
- Allocates memory to processes as needed and deallocates when no longer needed.
Uniprogramming Memory Model

- Used in systems that process one job only at a time, and all system resources are available exclusively for the job until it completes
- Simple and easy to implement
- Does not lead to proper utilization of the main memory as unoccupied memory space by the currently active user process remains unused
- Used only on very small or dedicated computer systems
Uniprogramming Memory Model

Operating system area

Operating system

User process

User area

Unused
Two memory management schemes used to facilitate this are:

- **Multiprogramming with fixed number of memory partitions**: User area of the memory is divided into a number of fixed-sized partitions.

- **Multiprogramming with variable number of memory partitions**: Number, size and location of the partitions vary dynamically as processes come and go.
Multiprogramming with Fixed Number of Memory Partition

- Operating system area
- User area divided into $n$ equal-sized partitions

<table>
<thead>
<tr>
<th></th>
<th>Operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Partition 1</td>
</tr>
<tr>
<td></td>
<td>Partition 2</td>
</tr>
<tr>
<td></td>
<td>Partition 3</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Partition n</td>
</tr>
</tbody>
</table>
The number, size, and location of the partitions vary dynamically as processes come and go. (contd...)
The number, size, and location of the partitions vary dynamically as processes come and go.
Virtual Memory

Memory management scheme that allows execution of processes that might not be completely loaded in the main memory.

It does not require the entire process to be in memory before the process can execute.
Three basic concepts used for its realization are:

- **On-line secondary storage:** Used to keep a process’s address space ready to be loaded into the memory.

- **Swapping:** Process of transferring a block of data from the on-line secondary storage to main memory (swapping in) or vice-versa (swapping out).

- **Demand paging:** Scheme of swapping in of pages of a process as and when needed during execution of the process, rather than loading all the pages before starting the process’s execution.
Advantages of Virtual Memory

- Provides a large virtual memory to programmers on a system having smaller physical memory.
- Enables execution of a process on a system whose main memory size is less than the total memory required by the process.
- Enables a process’s execution to be started even when sufficient free memory for loading the entire process is not available.
- Makes programming easier there no longer need to worry about the memory size limitations.
- Often leads to less I/O activity resulting in better throughput, turnaround time, and response time.
Disadvantages of Virtual Memory

- Difficult to implement because it requires algorithms to support demand paging
- If used carelessly, it may substantially decrease performance due to high page fault rate
File Management

- A **file** is a collection of related information.
- Every file has a name, its data and attributes.
- File’s name uniquely identifies it in the system and is used by its users to access it.
- File’s data is its contents.
- File’s attributes contain information such as date & time of its creation, date & time of last access, date & time of last update, its current size, its protection features, etc.
- File management module of an operating system takes care of file-related activities such as structuring, accessing, naming, sharing, and protection of files.
Two commonly supported file access methods are:

- **Sequential access**: Information stored in a file can be accessed sequentially (in the order in which they are stored, starting at the beginning)
- **Random access**: Information stored in a file can be accessed randomly irrespective of the order in which the bytes or records are stored
File Operations

- Set of commands provided by an operating system to deal with files and their contents
- Typical file operations include create, delete, open, close, read, write, seek, get attributes, set attributes, rename, and copy
File naming deals with the rules for naming files in an operating system. This may include such rules as:

- Maximum number of characters that a file name may have
- Special characters allowed in a file name
- Distinction between upper case and lower case letters
- Multi-part file names allow file extensions to be part of a file name. File extensions indicate something about the file and its content
- Used by applications to check for the intended type of file before operating on it
### File Extensions (Example)

<table>
<thead>
<tr>
<th>File extension</th>
<th>Its meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bas</td>
<td>Basic source program file</td>
</tr>
<tr>
<td>.c</td>
<td>C source program file</td>
</tr>
<tr>
<td>.ftn</td>
<td>Fortran source program file</td>
</tr>
<tr>
<td>.pas</td>
<td>Pascal source program file</td>
</tr>
<tr>
<td>.obj</td>
<td>Object file (compiler output, not yet linked)</td>
</tr>
<tr>
<td>.bin</td>
<td>Executable binary program file</td>
</tr>
<tr>
<td>.lib</td>
<td>Library of .obj files used by the linker</td>
</tr>
<tr>
<td>.dat</td>
<td>Data file</td>
</tr>
<tr>
<td>.hlp</td>
<td>Text file for HELP command</td>
</tr>
<tr>
<td>.man</td>
<td>Online manual page file</td>
</tr>
</tbody>
</table>

(Continued on next slide)
### File Extensions (Example)

<table>
<thead>
<tr>
<th>File extension</th>
<th>Its meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.man</td>
<td>Online manual page file</td>
</tr>
<tr>
<td>.txt</td>
<td>General text file</td>
</tr>
<tr>
<td>.bak</td>
<td>Backup file</td>
</tr>
<tr>
<td>.doc</td>
<td>Microsoft word document file</td>
</tr>
<tr>
<td>.wav</td>
<td>Microsoft windows sound file</td>
</tr>
<tr>
<td>.wk4</td>
<td>Lotus 1-2-3 spreadsheet file</td>
</tr>
<tr>
<td>.xls</td>
<td>Microsoft Excel spreadsheet file</td>
</tr>
<tr>
<td>.jpg</td>
<td>JPEG graphics file</td>
</tr>
<tr>
<td>.gif</td>
<td>GIF graphics file</td>
</tr>
</tbody>
</table>
Security

- Deals with protecting the various resources and information of a computer system against destruction and unauthorized access.

- **External security:** Deals with securing computer against external factors such as fires, floods, earthquakes, stolen disks/tapes, etc. by maintaining adequate backup, using security guards, allowing access to sensitive information to only trusted employees/users, etc.

- **Internal security:** Deals with user authentication, access control, and cryptography mechanisms.
Security

- **User authentication**: Deals with the problem of verifying the identity of a user (person or program) before permitting access to the requested resource.

- **Access Control**: Once authenticated, access control mechanisms prohibit a user/process from accessing those resources/information that he/she/it is not authorized to access.

- **Cryptography**: Means of encrypting private information so that unauthorized access cannot use information.
Command Interpretation

- Provides a set of commands using which the user can give instructions to the computer for getting some job done by it.
- Commands supported by the command interpretation module are known as **system calls**.

(Continued on next slide)
Two types of user interfaces supported by various operating systems are:

- **Command-line interface**: User gives instructions to the computer by typing the commands.

- **Graphical User Interface (GUI)**: User gives commands to the system by selecting icon or menu item displayed on the screen with the use of a point-and-draw device.
Perform several tasks of routine nature, frequently needed by users but are not provided as part of the OS

They are primarily grouped into three categories:

Translating programs: Translate a source program into an object program

Library programs: Consist of frequently used functions and operations

Utility programs: Assist users with system maintenance tasks such as disk formatting, data compression, data backups, antivirus utilities
UNIX OS

- Developed in the early 1970s at Bell Laboratories by Ken Thompson and Dennis Ritchie
- Written in C high-level language, hence, highly portable
- Multi-user, time-sharing OS
- Used on a wide variety of computers ranging from notebook computers to super computers
- Especially prevalent on RISC workstations such as those from Sun Microsystems, Hewlett-Packard, IBM, and Silicon Graphics
- Structured in three layers – kernel, shell, and utilities
MS-DOS

- Stands for Microsoft Disk Operating System.
- Single-user OS for IBM and IBM-compatible personal computers (PC)
- Structured in three layers – BIOS (Basic Input Output System), kernel, and shell
- Very popular in the 1980s, now not in much use and development with the launch of Microsoft Windows OS in 1990s
Microsoft Windows

- Developed by Microsoft to overcome limitations of MS-DOS operating system
- Single-user, multitasking OS
- Native interface is a GUI
- Designed to be not just an OS but also a complete operating environment
- OS of choice for most PCs after 1990
Microsoft Windows NT

- Multi-user, time-sharing OS developed by Microsoft
- Designed to have UNIX-like features so that it can be used for powerful workstations, network, and database servers
- Supports multiprogramming and is designed to take advantage of multiprocessing on systems having multiple processors
- Native interface is a GUI
- Built-in networking and communications features
- Provides strict system security
- Rich set of tools for software development
**Linux**

- Open-source OS enhanced and backed by thousands of programmers world-wide
- Multi-tasking, multiprocessing OS, originally designed to be used in PCs
- Name “Linux” is derived from its inventor Linus Torvalds
- Several Linux distributions available (Red Hat, SuSE). Difference in distribution is mostly set of tools, number and quality of applications, documentation, support, and service
Keywords/Phrases

- Access control
- Batch processing
- Command interpretation
- Command-line interface (CLI)
- CPU-bound jobs
- Cryptography
- Demand paging
- External security
- File
- File attributes
- File extensions
- File management
- Graphical User Interface (GUI)
- I/O-bound jobs
- Internal security
- Job control language (JCL)
- Library programs
- Linux
- Loosely coupled system

- Memory management
- Memory partition
- Microsoft Windows
- Microsoft Windows NT
- MS-DOS
- Multiprocessing
- Multiprogramming
- Multiprogramming with fixed tasks (MFT)
- Multiprogramming with variable tasks (MVT)
- Operating systems
- Multithreading
- Process
- Process Control Block (PCB) Multitasking
- Process management
- Random access files
- Response time
- Security
- Sequential access files
- Swapping

(Continued on next slide)
Keywords/Phrases

- Throughput
- Tightly coupled system
- Time-sharing
- Time slice
- Time slot
- Translating programs
- Turnaround time
- Unix
- User authentication
- Utility programs
- Virtual machine
- Virtual memory
Chapter 14
Operating Systems

Learning Objectives

In this chapter you will learn about:

- Definition and need for operating system
- Main functions of an operating system
- Commonly used mechanisms for:
  - Process management
  - Memory management
  - File management
  - Security
  - Command interpretation module
- Some commonly used OS capability enhancement software
- Some popular operating systems
Definition and Need for OS

- Integrated set of programs that controls the resources (the CPU, memory, I/O devices, etc.) of a computer system
- Provides its users with an interface or virtual machine that is more convenient to use than the bare machine
- Two primary objectives of an OS are:
  - Making a computer system convenient to use
  - Managing the resources of a computer system

Logical Architecture of a Computer System

The operating system layer hides the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system.
Main Functions of an OS

- Process management
- Memory management
- File management
- Security
- Command interpretation

Parameters for Measuring System Performance

- **Throughput:** Amount of work that the system is able to do per unit time
- **Turnaround time:** Interval from the time of submission of a job to the system for processing to the time of completion of the job
- **Response time:** Interval from the time of submission of a job to the system for processing to the time the first response for the job is produced by the system
A **process** (also called **job**) is a program in execution

**Process management** manages the processes submitted to a system in a manner to minimize *idle time* of processors (CPUs, I/O processors, etc.) of the system.

**Process Management Mechanisms in Early Systems**

- **Manual loading mechanism**: Jobs were manually loaded one after another in a computer by the computer operator.

- **Batch processing mechanism**: Batch of jobs was submitted together to the computer and job-to-job transition was done automatically by the operating system.

- **Job Control Language (JCL)**: Control statements were used to facilitate job loading and unloading.
Use of Job Control Statements in Batch Processing (An Example)

$JOB, ONGC05839, USER=SINHA

$LOAD

$RUN

$COBOL

$END

Data for program

COBOL program

Multiprogramming

- Uniprogramming: Only one job is processed at a time and all system resources are available exclusively for the job until its completion.
- Multiprogramming: Interleaved execution of two or more different and independent programs by a computer.
- Types of Multiprogramming:
  - Multiprogramming with fixed tasks (MFT): Fixed number of jobs can be processed concurrently.
  - Multiprogramming with variable tasks (MVT): Number of jobs can vary.
- Area occupied by each job residing simultaneously in the main memory is known as a memory partition.
**Job**

- **CPU bound:** Mostly perform computations with little I/O operations. Scientific and engineering computations usually fall in this category.
- **I/O bound:** Mostly perform I/O operations with little computation. Commercial data processing applications usually fall in this category.

**Uniprogramming System**

Only one job is processed by the system at a time and all the system resources are exclusively available for the job until it completes.
Multiprogramming System

Secondary disk storage

Main memory

- Operating system
  - Job A
  - Job B
  - Job C (Waiting for CPU)

CPU

Execution in progress

Writing output data

Process States in Multiprogramming

- Ready
- Running
- Blocked

New job → Ready

Job is allocated the CPU for execution → Running

Job processing completed

I/O completed → Blocked

Job must wait for I/O completion
Requirements of Multiprogramming Systems

- Large memory
- Memory protection
- Job status preservation
- Proper job mix (CPU and I/O bound jobs)
- CPU scheduling

Process Control Block (PCB)

<table>
<thead>
<tr>
<th>process identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>process state</td>
</tr>
<tr>
<td>program counter</td>
</tr>
<tr>
<td>values of various CPU registers</td>
</tr>
<tr>
<td>accounting and scheduling information</td>
</tr>
<tr>
<td>I/O status information</td>
</tr>
</tbody>
</table>

PCB is used to preserve the job status of each loaded process in a multiprogramming system.
Multitasking

- Interleaved execution of multiple jobs (often referred to as tasks of same user) in a single-user system
- Computer systems used for multitasking are uniprocessor systems (having only one CPU)
- Treated differently from multiprogramming that refers to interleaved execution of multiple jobs in a multi-user system

Multithreading

- Thread is basic unit of CPU utilization. Threads share a CPU in the same way as processes do
- All threads of a process also share the same set of operating system resources
- All threads of a process inherit parent’s address space and security parameters
- Each thread of a process has its own program counter, its own register states, and its own stack
- Referred as mini-process or lightweight process
(a) Single-threaded and (b) multithreaded processes. A single-threaded process corresponds to a process of a traditional operating system. [Reproduced with permission, from the book titled Distributed Operating Systems: Concepts and Design by Pradeep K. Sinha. © 1997 IEEE, USA].

**Multithreading System**

- Single-threaded process
- Multithreaded processes

**Multiprocessing**

- System with two or more CPUs having ability to execute multiple processes concurrently
- Multiple CPUs are used to process either instructions from different and independent programs or different instructions from the same program simultaneously
- Types of multiprocessing:
  - *Tightly-coupled*: Single system-wide primary memory shared by all processors
  - *Loosely-coupled*: Each processor has its own local memory
CPU, Memory, and I/O Processors of a Computer System

Multiprocessing System
Time-sharing

- Simultaneous interactive use of a computer system by many users in such a way that each one feels that he/she is the sole user of the system
- User terminals connected to the same computer simultaneously
- Uses multiprogramming with a special CPU scheduling algorithm
- Short period during which a user process gets to use CPU is known as time slice, time slot, or quantum
- CPU is taken away from a running process when the allotted time slice expires

Process State Diagram for a Time-Sharing System

- Ready: Job is allocated to CPU for execution
- Running: Allocated time slice is over
- Blocked: Job must wait for I/O completion
- Job is processed and completed

New Job → Ready

I/O completed → Blocked

Job processing completed → Running → Allotted time slice is over → Running

Advantages of Time-sharing Systems

- Reduces CPU idle time
- Provides advantages of quick response time
- Offers good computing facility to small users

Memory Management

- Memory is important resource of a computer system that must be properly managed for the overall system performance
- Memory management module:
  - Keeps track of parts of memory in use and parts not in use
  - Allocates memory to processes as needed and deallocated when no longer needed
Uniprogramming Memory Model

- Used in systems that process one job only at a time, and all system resources are available exclusively for the job until it completes
- Simple and easy to implement
- Does not lead to proper utilization of the main memory as unoccupied memory space by the currently active user process remains unused
- Used only on very small or dedicated computer systems
Two memory management schemes used to facilitate this are:

- **Multiprogramming with fixed number of memory partitions**: User area of the memory is divided into a number of fixed-sized partitions

- **Multiprogramming with variable number of memory partitions**: Number, size and location of the partitions vary dynamically as processes come and go
The number, size, and location of the partitions vary dynamically as processes come and go. (contd...)

(a) (b) (c) (d)

The number, size, and location of the partitions vary dynamically as processes come and go.
Virtual Memory

Memory management scheme that allows execution of processes that might not be completely loaded in the main memory.

It does not require the entire process to be in memory before the process can execute.

Virtual Memory Realization

Three basic concepts used for its realization are:

- **On-line secondary storage**: Used to keep a process’s address space ready to be loaded into the memory.

- **Swapping**: Process of transferring a block of data from the on-line secondary storage to main memory (swapping in) or vice-versa (swapping out).

- **Demand paging**: Scheme of swapping in of pages of a process as and when needed during execution of the process, rather than loading all the pages before starting the process’s execution.
Advantages of Virtual Memory

- Provides a large virtual memory to programmers on a system having smaller physical memory
- Enables execution of a process on a system whose main memory size is less than the total memory required by the process
- Enables a process’s execution to be started even when sufficient free memory for loading the entire process is not available
- Makes programming easier there no longer need to worry about the memory size limitations
- Often leads to less I/O activity resulting in better throughput, turnaround time, and response time

Disadvantages of Virtual Memory

- Difficult to implement because it requires algorithms to support demand paging
- If used carelessly, it may substantially decrease performance due to high page fault rate
File Management

- A file is a collection of related information
- Every file has a name, its data and attributes
- File’s name uniquely identifies it in the system and is used by its users to access it
- File’s data is its contents
- File’s attributes contain information such as date & time of its creation, date & time of last access, date & time of last update, its current size, its protection features, etc.
- File management module of an operating system takes care of file-related activities such as structuring, accessing, naming, sharing, and protection of files.

File Access Methods

Two commonly supported file access methods are:

- **Sequential access**: Information stored in a file can be accessed sequentially (in the order in which they are stored, starting at the beginning)
- **Random access**: Information stored in a file can be accessed randomly irrespective of the order in which the bytes or records are stored.
File Operations

- Set of commands provided by an operating system to deal with files and their contents
- Typical file operations include create, delete, open, close, read, write, seek, get attributes, set attributes, rename, and copy

File Naming

File naming deals with the rules for naming files in an operating system. This may include such rules as:

- Maximum number of characters that a file name may have
- Special characters allowed in a file name
- Distinction between upper case and lower case letters
- Multi-part file names allow file extensions to be part of a file name. File extensions indicate something about the file and its content
- Used by applications to check for the intended type of file before operating on it
### File Extensions (Example)

<table>
<thead>
<tr>
<th>File extension</th>
<th>Its meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bas</td>
<td>Basic source program file</td>
</tr>
<tr>
<td>.c</td>
<td>C source program file</td>
</tr>
<tr>
<td>.ftn</td>
<td>Fortran source program file</td>
</tr>
<tr>
<td>.pas</td>
<td>Pascal source program file</td>
</tr>
<tr>
<td>.obj</td>
<td>Object file (compiler output, not yet linked)</td>
</tr>
<tr>
<td>.bin</td>
<td>Executable binary program file</td>
</tr>
<tr>
<td>.lib</td>
<td>Library of .obj files used by the linker</td>
</tr>
<tr>
<td>.dat</td>
<td>Data file</td>
</tr>
<tr>
<td>.hlp</td>
<td>Text file for HELP command</td>
</tr>
<tr>
<td>.man</td>
<td>Online manual page file</td>
</tr>
</tbody>
</table>

(Continued on previous slide)

<table>
<thead>
<tr>
<th>File extension</th>
<th>Its meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.man</td>
<td>Online manual page file</td>
</tr>
<tr>
<td>.txt</td>
<td>General text file</td>
</tr>
<tr>
<td>.bak</td>
<td>Backup file</td>
</tr>
<tr>
<td>.doc</td>
<td>Microsoft word document file</td>
</tr>
<tr>
<td>.wav</td>
<td>Microsoft Windows sound file</td>
</tr>
<tr>
<td>.wk4</td>
<td>Lotus 1-2-3 spreadsheet file</td>
</tr>
<tr>
<td>.xls</td>
<td>Microsoft Excel spreadsheet file</td>
</tr>
<tr>
<td>.jpg</td>
<td>JPEG graphics file</td>
</tr>
<tr>
<td>.gif</td>
<td>GIF graphics file</td>
</tr>
</tbody>
</table>

(Continued from previous slide)
Security

- Deals with protecting the various resources and information of a computer system against destruction and unauthorized access

- **External security:** Deals with securing computer against external factors such as fires, floods, earthquakes, stolen disks/tapes, etc. by maintaining adequate backup, using security guards, allowing access to sensitive information to only trusted employees/users, etc.

- **Internal security:** Deals with user authentication, access control, and cryptography mechanisms

---

Security

- **User authentication:** Deals with the problem of verifying the identity of a user (person or program) before permitting access to the requested resource

- **Access Control:** Once authenticated, access control mechanisms prohibit a user/process from accessing those resources/information that he/she/it is not authorized to access

- **Cryptography:** Means of encrypting private information so that unauthorized access cannot use information
Command Interpretation

- Provides a set of commands using which the user can give instructions to the computer for getting some job done by it
- Commands supported by the command interpretation module are known as system calls

(Continued on next slide)

Command Interpretation

Two types of user interfaces supported by various operating systems are:

- **Command-line interface**: User gives instructions to the computer by typing the commands
- **Graphical User Interface (GUI)**: User gives commands to the system by selecting icon or menu item displayed on the screen with the use of a point-and-draw device

(Continued from previous slide)
**OS Capability Enhancement Software**

- Perform several tasks of routine nature, frequently needed by users but are not provided as part of the OS
- They are primarily grouped into three categories:
  - **Translating programs**: Translate a source program into an object program
  - **Library programs**: Consist of frequently used functions and operations
  - **Utility programs**: Assist users with system maintenance tasks such as disk formatting, data compression, data backups, antivirus utilities

**UNIX OS**

- Developed in the early 1970s at Bell Laboratories by Ken Thompson and Dennis Ritchie
- Written in C high-level language, hence, highly portable
- Multi-user, time-sharing OS
- Used on a wide variety of computers ranging from notebook computers to super computers
- Especially prevalent on RISC workstations such as those from Sun Microsystems, Hewlett-Packard, IBM, and Silicon Graphics
- Structured in three layers – kernel, shell, and utilities
MS-DOS

- Stands for Microsoft Disk Operating System.
- Single-user OS for IBM and IBM-compatible personal computers (PC)
- Structured in three layers – BIOS (Basic Input Output System), kernel, and shell
- Very popular in the 1980s, now not in much use and development with the launch of Microsoft Windows OS in 1990s

Microsoft Windows

- Developed by Microsoft to overcome limitations of MS-DOS operating system
- Single-user, multitasking OS
- Native interface is a GUI
- Designed to be not just an OS but also a complete operating environment
- OS of choice for most PCs after 1990
Microsoft Windows NT

- Multi-user, time-sharing OS developed by Microsoft
- Designed to have UNIX-like features so that it can be used for powerful workstations, network, and database servers
- Supports multiprogramming and is designed to take advantage of multiprocessing on systems having multiple processors
- Native interface is a GUI
- Built-in networking and communications features
- Provides strict system security
- Rich set of tools for software development

Linux

- Open-source OS enhanced and backed by thousands of programmers world-wide
- Multi-tasking, multiprocessing OS, originally designed to be used in PCs
- Name “Linux” is derived from its inventor Linus Torvalds
- Several Linux distributions available (Red Hat, SuSE). Difference in distribution is mostly set of tools, number and quality of applications, documentation, support, and service
Keywords/Phrases

- Access control
- Batch processing
- Command interpretation
- Command-line interface (CLI)
- CPU-bound jobs
- Cryptography
- Demand paging
- External security
- File
- File attributes
- File management
- Graphical User Interface (GUI)
- I/O-bound jobs
- Internal security
- Job control language (JCL)
- Library programs
- Linux
- Loosely coupled system
- Memory management
- Memory partition
- Microsoft Windows
- Microsoft Windows NT
- MS-DOS
- Multiprocessing
- Multiprogramming
- Multiprogramming with fixed tasks (MFT)
- Multiprogramming with variable tasks (MVT)
- Operating systems
- Multithreading
- Process
- Process Control Block (PCB) Multitasking
- Process management
- Random access files
- Response time
- Security
- Sequential access files
- Swapping

(Continued from previous slide)

- Throughput
- Tightly coupled system
- Time-sharing
- Time slice
- Time slot
- Translating programs
- Turnaround time
- Unix
- User authentication
- Utility programs
- Virtual machine
- Virtual memory
Chapter 14
Operating Systems

Learning Objectives

In this chapter you will learn about:

- Definition and need for operating system
- Main functions of an operating system
- Commonly used mechanisms for:
  - Process management
  - Memory management
  - File management
  - Security
  - Command interpretation module
- Some commonly used OS capability enhancement software
- Some popular operating systems

Definition and Need for OS

- Integrated set of programs that controls the resources (the CPU, memory, I/O devices, etc.) of a computer system
- Provides its users with an interface or virtual machine that is more convenient to use than the bare machine
- Two primary objectives of an OS are:
  - Making a computer system convenient to use
  - Managing the resources of a computer system
The operating system layer hides the details of the hardware from the programmer and provides the programmer with convenient interface for using the system.

**Main Functions of an OS**

- Process management
- Memory management
- File management
- Security
- Command interpretation

**Parameters for Measuring System Performance**

- **Throughput**: Amount of work that the system is able to do per unit time
- **Turnaround time**: Interval from the time of submission of a job to the system for processing to the time of completion of the job
- **Response time**: Interval from the time of submission of a job to the system for processing to the time the first response for the job is produced by the system
Process Management

A process (also called job) is a program in execution. Process management manages the processes submitted to a system in a manner to minimize idle time of processors (CPUs, I/O processors, etc.) of the system.

Process Management Mechanisms in Early Systems

1. Manual loading mechanism: Jobs were manually loaded one after another in a computer by the computer operator.
2. Batch processing mechanism: Batch of jobs was submitted together to the computer and job-to-job transition was done automatically by the operating system.
3. Job Control Language (JCL): Control statements were used to facilitate job loading and unloading.

Use of Job Control Statements in Batch Processing (An Example)

```
$JOB, ONGC05839, USER=SINHA
$LOAD $COBOL
$RUN data for program
$END
```

```
Multiprogramming

- **Uniprogramming**: Only one job is processed at a time and all system resources are available exclusively for the job until its completion.
- **Multiprogramming**: Interleaved execution of two or more different and independent programs by a computer.
- Types of Multiprogramming:
  - Multiprogramming with fixed tasks (MFT): Fixed number of jobs can be processed concurrently.
  - Multiprogramming with variable tasks (MVT): Number of jobs can vary.
- Area occupied by each job residing simultaneously in the main memory is known as a **memory partition**.

Job

- **CPU bound**: Mostly perform computations with little I/O operations. Scientific and engineering computations usually fall in this category.
- **I/O bound**: Mostly perform I/O operations with little computation. Commercial data processing applications usually fall in this category.

Uniprogramming System

Only one job is processed by the system at a time and all the system resources are exclusively available for the job until it completes.
### Multiprogramming System

- **Main memory**
- **Operating system**
  - Job A
  - Job B
  - Job C (Waiting for CPU)

- **Secondary disk storage**
- **Execution in progress**
- **Writing output data**

### Process States in Multiprogramming

- **New job**
- **Ready**
  - Job is allocated the CPU for execution
- **Running**
- **Blocked**
  - Job must wait for I/O completion
- **I/O completed**

### Requirements of Multiprogramming Systems

- Large memory
- Memory protection
- Job status preservation
- Proper job mix (CPU and I/O bound jobs)
- CPU scheduling
Process Control Block (PCB)

- process identifier
- process state
- program counter
- values of various CPU registers
- accounting and scheduling information
- I/O status information

PCB is used to preserve the job status of each loaded process in a multiprogramming system.

Multitasking

- Interleaved execution of multiple jobs (often referred to as tasks of same user) in a single-user system
- Computer systems used for multitasking are uniprocessor systems (having only one CPU)
- Treated differently from multiprogramming that refers to interleaved execution of multiple jobs in a multi-user system

Multithreading

- Thread is basic unit of CPU utilization. Threads share a CPU in the same way as processes do
- All threads of a process also share the same set of operating system resources
- All threads of a process inherit parent’s address space and security parameters
- Each thread of a process has its own program counter, its own register states, and its own stack
- Referred as mini-process or lightweight process
Chapter 14: Operating Systems

(a) Single-threaded and (b) multithreaded processes. A single-threaded process corresponds to a process of a traditional operating system. [Reproduced with permission, from the book titled Distributed Operating Systems: Concepts and Design by Pradeep K. Sinha. © 1997 IEEE, USA].

Multiprocessing

- System with two or more CPUs having ability to execute multiple processes concurrently
- Multiple CPUs are used to process either instructions from different and independent programs or different instructions from the same program simultaneously
- Types of multiprocessing:
  - Tightly-coupled: Single system-wide primary memory shared by all processors
  - Loosely-coupled: Each processor has its own local memory

CPU, Memory, and I/O Processors of a Computer System

- Main memory
- I/O Units
- I/O Processors
- CPU
Simultaneous interactive use of a computer system by many users in such a way that each one feels that he/she is the sole user of the system.

User terminals connected to the same computer simultaneously.

Uses multiprogramming with a special CPU scheduling algorithm.

Short period during which a user process gets to use CPU is known as time slice, time slot, or quantum.

CPU is taken away from a running process when the allotted time slice expires.
Advantages of Time-sharing Systems

- Reduces CPU idle time
- Provides advantages of quick response time
- Offers good computing facility to small users

Memory Management

- Memory is an important resource of a computer system that must be properly managed for the overall system performance
- Memory management module:
  - Keeps track of parts of memory in use and parts not in use
  - Allocates memory to processes as needed and deallocates when no longer needed

Uniprogramming Memory Model

- Used in systems that process one job only at a time, and all system resources are available exclusively for the job until it completes
- Simple and easy to implement
- Does not lead to proper utilization of the main memory as unoccupied memory space by the currently active user process remains unused
- Used only on very small or dedicated computer systems
Two memory management schemes used to facilitate this are:

- Multiprogramming with fixed number of memory partitions: User area of the memory is divided into a number of fixed-sized partitions.
- Multiprogramming with variable number of memory partitions: Number, size and location of the partitions vary dynamically as processes come and go.
The number, size, and location of the partitions vary dynamically as processes come and go.

Virtual Memory

Memory management scheme that allows execution of processes that might not be completely loaded in the main memory.

It does not require the entire process to be in memory before the process can execute.
Three basic concepts used for its realization are:

- **On-line secondary storage**: Used to keep a process’s address space ready to be loaded into the memory.
- **Swapping**: Process of transferring a block of data from the on-line secondary storage to main memory (swapping in) or vice-versa (swapping out).
- **Demand paging**: Scheme of swapping in of pages of a process as and when needed during execution of the process, rather than loading all the pages before starting the process’s execution.

**Advantages of Virtual Memory**

- Provides a large virtual memory to programmers on a system having smaller physical memory.
- Enables execution of a process on a system whose main memory size is less than the total memory required by the process.
- Enables a process’s execution to be started even when sufficient free memory for loading the entire process is not available.
- Makes programming easier as there no longer need to worry about the memory size limitations.
- Often leads to less I/O activity resulting in better throughput, turnaround time, and response time.

**Disadvantages of Virtual Memory**

- Difficult to implement because it requires algorithms to support demand paging.
- If used carelessly, it may substantially decrease performance due to high page fault rate.
File Management

- A file is a collection of related information
- Every file has a name, its data and attributes
- File’s name uniquely identifies it in the system and is used by its users to access it
- File’s data is its contents
- File’s attributes contain information such as date & time of its creation, date & time of last access, date & time of last update, its current size, its protection features, etc.
- File management module of an operating system takes care of file-related activities such as structuring, accessing, naming, sharing, and protection of files

File Access Methods

Two commonly supported file access methods are:
- **Sequential access:** Information stored in a file can be accessed sequentially (in the order in which they are stored, starting at the beginning)
- **Random access:** Information stored in a file can be accessed randomly irrespective of the order in which the bytes or records are stored

File Operations

- Set of commands provided by an operating system to deal with files and their contents
- Typical file operations include create, delete, open, close, read, write, seek, get attributes, set attributes, rename, and copy
File naming deals with the rules for naming files in an operating system. This may include such rules as:

1. Maximum number of characters that a file name may have
2. Special characters allowed in a file name
3. Distinction between upper case and lower case letters
4. Multi-part file names allow file extensions to be part of a file name. File extensions indicate something about the file and its content
5. Used by applications to check for the intended type of file before operating on it

File Extensions (Example)

<table>
<thead>
<tr>
<th>File extension</th>
<th>Its meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bas</td>
<td>Basic source program file</td>
</tr>
<tr>
<td>.c</td>
<td>C source program file</td>
</tr>
<tr>
<td>.f90</td>
<td>Fortran source program file</td>
</tr>
<tr>
<td>.pas</td>
<td>Pascal source program file</td>
</tr>
<tr>
<td>.obj</td>
<td>Object file (compiler output, not yet linked)</td>
</tr>
<tr>
<td>.bin</td>
<td>Executable binary program file</td>
</tr>
<tr>
<td>.lib</td>
<td>Library of .obj files used by the linker</td>
</tr>
<tr>
<td>.dat</td>
<td>Data file</td>
</tr>
<tr>
<td>.hlp</td>
<td>Text file for HELP command</td>
</tr>
<tr>
<td>.man</td>
<td>Online manual page file</td>
</tr>
</tbody>
</table>

(Continued from previous slide)

<table>
<thead>
<tr>
<th>File extension</th>
<th>Its meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>.txt</td>
<td>General text file</td>
</tr>
<tr>
<td>.bak</td>
<td>Backup file</td>
</tr>
<tr>
<td>.doc</td>
<td>Microsoft word document file</td>
</tr>
<tr>
<td>.wav</td>
<td>Microsoft windows sound file</td>
</tr>
<tr>
<td>.wk4</td>
<td>Lotus 1-2-3 spreadsheet file</td>
</tr>
<tr>
<td>.xls</td>
<td>Microsoft Excel spreadsheet file</td>
</tr>
<tr>
<td>.jpg</td>
<td>JPEG graphics file</td>
</tr>
<tr>
<td>.gif</td>
<td>GIF graphics file</td>
</tr>
</tbody>
</table>
Security

6 Deals with protecting the various resources and information of a computer system against destruction and unauthorized access

6 External security: Deals with securing computer against external factors such as fires, floods, earthquakes, stolen disks/tapes, etc. by maintaining adequate backup, using security guards, allowing access to sensitive information to only trusted employees/users, etc.

6 Internal security: Deals with user authentication, access control, and cryptography mechanisms

User authentication: Deals with the problem of verifying the identity of a user (person or program) before permitting access to the requested resource

Access Control: Once authenticated, access control mechanisms prohibit a user/process from accessing those resources/information that he/she/it is not authorized to access

Cryptography: Means of encrypting private information so that unauthorized access cannot use information

Command Interpretation

6 Provides a set of commands using which the user can give instructions to the computer for getting some job done by it

6 Commands supported by the command interpretation module are known as system calls

(Continued on next slide)
Two types of user interfaces supported by various operating systems are:

- **Command-line interface**: User gives instructions to the computer by typing the commands
- **Graphical User Interface (GUI)**: User gives commands to the system by selecting icon or menu item displayed on the screen with the use of a point-and-draw device

They are primarily grouped into three categories:

- **Translating programs**: Translate a source program into an object program
- **Library programs**: Consist of frequently used functions and operations
- **Utility programs**: Assist users with system maintenance tasks such as disk formatting, data compression, data backups, antivirus utilities

- Developed in the early 1970s at Bell Laboratories by Ken Thompson and Dennis Ritchie
- Written in C high-level language, hence, highly portable
- Multi-user, time-sharing OS
- Used on a wide variety of computers ranging from notebook computers to super computers
- Especially prevalent on RISC workstations such as those from Sun Microsystems, Hewlett-Packard, IBM, and Silicon Graphics
- Structured in three layers – kernel, shell, and utilities
### MS-DOS

- Stands for Microsoft Disk Operating System.
- Single-user OS for IBM and IBM-compatible personal computers (PC).
- Structured in three layers – BIOS (Basic Input Output System), kernel, and shell.
- Very popular in the 1980s, not in much use and development with the launch of Microsoft Windows OS in the 1990s.

### Microsoft Windows

- Developed by Microsoft to overcome limitations of MS-DOS operating system.
- Single-user, multitasking OS.
- Native interface is a GUI.
- Designed to be not just an OS but also a complete operating environment.
- OS of choice for most PCs after 1990.

### Microsoft Windows NT

- Multi-user, time-sharing OS developed by Microsoft.
- Designed to have UNIX-like features so that it can be used for powerful workstations, network, and database servers.
- Supports multiprogramming and is designed to take advantage of multiprocessing on systems having multiple processors.
- Native interface is a GUI.
- Built-in networking and communications features.
- Provides strict system security.
- Rich set of tools for software development.
Linux

8. Open-source OS enhanced and backed by thousands of programmers worldwide
8. Multi-tasking, multiprocessing OS, originally designed to be used in PCs
8. Name “Linux” is derived from its inventor Linus Torvalds
8. Several Linux distributions available (Red Hat, SuSE). Difference in distribution is mostly set of tools, number and quality of applications, documentation, support, and service

Keywords/Phrases

- Access control
- Batch processing
- Command interpretation
- Command-line interface (CLI)
- CPU-bound jobs
- Cryptography
- Demand paging
- External security
- File
- File attributes
- File extensions
- File management
- Graphical User Interface (GUI)
- I/O-bound jobs
- Internal security
- Job control language (JCL)
- Library programs
- Linux
- Loosely coupled system
- Memory management
- Memory partition
- Microsoft Windows
- Microsoft Windows NT
- MS-DOS
- Multiprocessing
- Multi-programming
- Multi-programming with fixed tasks (MFT)
- Multi-programming with variable tasks (MVT)
- Operating systems
- Multithreading
- Process
- Process Control Block (PCB)
- Multitasking
- Random access files
- Response time
- Security
- Sequential access files
- Swapping
- Throughput
- Tightly coupled system
- Time-sharing
- Time slice
- Time slot
- Translating programs
- Turnaround time
- Unix
- User authentication
- Utility programs
- Virtual machines
- Virtual memory
Chapter 15

Application Software Packages

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
In this chapter you will learn about:

- Word-processing package
- Spreadsheet package
- Graphics package
- Personal assistance package
**Word-Processing Package**

- **Word-processing** describes use of hardware and software to create, edit, view, format, store, retrieve, and print documents (written material such as letters, reports, books, etc.)

- **Word-processing package** enables us to do all these on a computer system
Commonly Supported Features in a Word-Processing Package

- Entering text
- Editing text
- Formatting page style
- Formatting text
- Entering mathematical symbols
- Displaying documents
- Saving, retrieving and deleting documents
- Printing documents
- Importing text, graphics and images
- Searching and replacing text string
- Checking spelling
- Checking grammar and style
Word-Processing (Few Terminologies)

- **Style sheet**: Pre-stored page format that can be used while creating a new document or can be applied to an existing document.

- **Font**: Complete set of characters with the same style and size. A word-processing package comes with several standard fonts.

- **Points**: A point is 1/72 of an inch, and the size refers to the distance from the top of the tallest character to the bottom of the character that extends the lowest. Font size is measured in points.

(Continued on next slide)
Word-Processing (Few Terminologies)

- Three commonly used font styles are *italic*, **bold** and underline.

- **Justification**: Alignment of text on the left or the right margin, or on both margins. Four types of justification are:
  - Left-justification
  - Right-justification
  - Center-justification
  - Full-justification
Different Font Types

This sentence is written in Times New Roman font.

This sentence is written in Helvetica font.

This sentence is written in Palatino font.

This sentence is written in Courier New font.

This sentence is written in Antique Olive font.
This sentence is written in 10 point Times New Roman font.

This sentence is written in 12 point Times New Roman font.

This sentence is written in 16 point Times New Roman font.

This sentence is written in 24 point Times New Roman font.

This sentence is written in 36 point Times New Roman font.
Different Font Styles

This sentence is written in italic style.

This sentence is written in bold style.

This sentence is written in underline style.

You can even make individual words italic, bold, or underline.
The term *hardware* refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

**(a) Left Justified text**

The term *hardware* refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

**(b) Right Justified text**

The term *hardware* refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

**(c) Centered text**

The term *hardware* refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.
\[
\left\{ t \ \left| R(t) \land \left[ \exists u \ (u) \right] (S(u) \land \neg u[1] = u[2]) \right. \right. \\
\left. \left. \left\{ <a, b, c> \left| \exists <a, b> ( <a, b> \in r \land <a, c> \in s) \right. \right. \right. \right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. \\
\right. 
\]
Spreadsheet Package

- **Spreadsheet package** is a numeric data analysis tool that allows us to create a computerized ledger.
- Useful for any numerical analysis problem whose data can be organized as rows and columns.
Uses of Spreadsheet Package

- Maintaining and analyzing inventory, payroll, and other accounting records by accountants
- Preparing budgets and bid comparisons by business analysts
- Recording grades of students and carrying out various types of analysis of the grades by educators
- Analyzing experimental results by scientists and researchers
- Tracking stocks and keeping records of investor accounts by stockbrokers
- Creating and tracking personal budgets, loan payments, etc. by individuals
Support for a large number of cells

Support for addressing a range of cells by the addresses of the endpoint cells

Support for different types of cell data (such as label, numeric value, formula, and date & time)

Support for use of relative and absolute cell addresses in formula

Support for a wide range of commands

Support for displaying numeric data in the form of graphs and charts
### Sample Spreadsheet

The table below represents a sample spreadsheet with the following columns:

- **A**: Final Exam Marks Sheet (CLASS-X: 2001)
- **B**: NAME
- **C**: PHYS
- **D**: CHEM
- **E**: MATHS
- **F**: TOTAL
- **G**: PERCENT

<table>
<thead>
<tr>
<th>Row</th>
<th>Column</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>FINAL EXAM MARKS SHEET(CLASS-X: 2001)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>P. Davis</td>
<td>92</td>
<td>95</td>
<td>88</td>
<td>275</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>A. Raje</td>
<td>86</td>
<td>82</td>
<td>94</td>
<td>262</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>D. Rana</td>
<td>75</td>
<td>83</td>
<td>85</td>
<td>243</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>M. Ray</td>
<td>77</td>
<td>75</td>
<td>72</td>
<td>224</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>J. Smith</td>
<td>94</td>
<td>92</td>
<td>96</td>
<td>282</td>
</tr>
</tbody>
</table>

- **Alphabetic Value in a Cell**: P. Davis, A. Raje, D. Rana, M. Ray, J. Smith
- **Numeric Value in a Cell**: 92, 95, 88, 275
- **Result of the function @SUM(B9..D9)**: 110
- **Result of the formula + E9/3**: 30.67
- **A label running across multiple columns**: NAME, PHYS, CHEM, MATHS, TOTAL, PERCENT
- **A label**: FINAL EXAM MARKS SHEET(CLASS-X: 2001)
- **Cell F4**: P. Davis 92 95 88 275 91.66
- **Cell C11**: Result of the function @SUM(B9..D9)
Examples of a Line Graph, a Bar Chart and a Pie Chart

(a) A line graph

(b) A bar chart

(c) A pie chart
**Graphics package** enables us to use a computer system for creating, editing, viewing, storing, retrieving and printing designs, drawings, pictures, graphs and anything else that can be drawn in the traditional manner.
Common Features of Graphics Package

- Drawing designs
- Painting drawings and pictures
- Presenting graphs and charts
- Dragging-and-dropping graphic objects
- Importing graphic objects
- Capturing screen snapshots
Computer Graphics (Few Terminologies)

- **Computer-aided-design (CAD):** Integration of computers and graphics design packages for the purpose of automating the design and drafting process.

- **Vector graphics:** Graphic object composed of patterns of lines, points, circles, arcs and other geometric shapes that can be easily represented by few geometric parameters.

- **Raster graphics:** Graphic object composed of patterns of dots called **pixels**.
Personal-assistance package allows individuals to:

- Use personal computers for storing and retrieving their personal information
- Planning and managing their schedules, contacts, finances and inventory of important items
Common Features of Personal Assistance Package

- Calendar
- To-do list
- Address book
- Investments book
- Inventory book
### Key Words/Phrases

- Bit-mapped image
- Bold
- Cell
- Center justification
- Clip art library
- Computer Aided Design (CAD)
- Font
- Full justification
- Graphics package
- Italic
- Justification
- Landscape mode
- Left justification
- Personal assistance package
- Portrait mode
- Raster graphics
- Right justification
- Spreadsheet package
- Style sheet
- Underline
- Vector graphics
- What You See Is What you Get (WYSIWYG)
- Word-processing
- Word-processing package
Chapter 15
Application Software Packages

Learning Objectives

In this chapter you will learn about:

- Word-processing package
- Spreadsheet package
- Graphics package
- Personal assistance package
Word-processing describes use of hardware and software to create, edit, view, format, store, retrieve, and print documents (written material such as letters, reports, books, etc.)

Word-processing package enables us to do all these on a computer system.

<table>
<thead>
<tr>
<th>Commonly Supported Features in a Word-Processing Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entering text</td>
</tr>
<tr>
<td>Editing text</td>
</tr>
<tr>
<td>Formatting page style</td>
</tr>
<tr>
<td>Formatting text</td>
</tr>
<tr>
<td>Entering mathematical symbols</td>
</tr>
<tr>
<td>Displaying documents</td>
</tr>
<tr>
<td>Saving, retrieving and deleting documents</td>
</tr>
<tr>
<td>Printing documents</td>
</tr>
<tr>
<td>Importing text, graphics and images</td>
</tr>
<tr>
<td>Searching and replacing text string</td>
</tr>
<tr>
<td>Checking spelling</td>
</tr>
<tr>
<td>Checking grammar and style</td>
</tr>
</tbody>
</table>
**Word-Processing (Few Terminologies)**

- **Style sheet**: Pre-stored page format that can be used while creating a new document or can be applied to an existing document.

- **Font**: Complete set of characters with the same style and size. A word-processing package comes with several standard fonts.

- **Points**: A point is 1/72 of an inch, and the size refers to the distance from the top of the tallest character to the bottom of the character that extends the lowest. Font size is measured in points.

(Continued on next slide)

- Three commonly used font styles are *italic*, **bold** and **underline**.

- **Justification**: Alignment of text on the left or the right margin, or on both margins. Four types of justification are:
  - Left-justification
  - Right-justification
  - Center-justification
  - Full-justification
Different Font Types

This sentence is written in Times New Roman font.

This sentence is written in Helvetica font.

This sentence is written in Palatino font.

This sentence is written in Courier New font.

This sentence is written in Antique Olive font.

Different Font Sizes

This sentence is written in 10 point Times New Roman font.

This sentence is written in 12 point Times New Roman font.

This sentence is written in 16 point Times New Roman font.

This sentence is written in 24 point Times New Roman font.

This sentence is written in 36 point Times New Roman font.
Different Font Styles

This sentence is written in italic style.

This sentence is written in bold style.

This sentence is written in underline style.

You can even make individual words italic, bold, or underline.

Different Justification Styles

The term hardware refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

(a) Left Justified text

The term hardware refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

(b) Right Justified text

The term hardware refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

(c) Centered text
Mathematical Symbols

\[
\left\{ t^{(2)} \mid R(t) \land \exists u^{(w)} (S(u) \land \neg u[1] = u[2]) \right\}
\]

\[
\left\{ <a,b,c> \exists <a,b> (<a,b>\in r \land <a,c>\in s) \right\}
\]

Spreadsheet Package

- **Spreadsheet package** is a numeric data analysis tool that allows us to create a computerized ledger
- Useful for any numerical analysis problem whose data can be organized as rows and columns
Uses of Spreadsheet Package

- Maintaining and analyzing inventory, payroll, and other accounting records by accountants
- Preparing budgets and bid comparisons by business analysts
- Recording grades of students and carrying out various types of analysis of the grades by educators
- Analyzing experimental results by scientists and researchers
- Tracking stocks and keeping records of investor accounts by stockbrokers
- Creating and tracking personal budgets, loan payments, etc. by individuals

Common Features of Spreadsheet Package

- Support for a large number of cells
- Support for addressing a range of cells by the addresses of the endpoint cells
- Support for different types of cell data (such as label, numeric value, formula, and date & time)
- Support for use of relative and absolute cell addresses in formula
- Support for a wide range of commands
- Support for displaying numeric data in the form of graphs and charts
Sample Spreadsheet

<table>
<thead>
<tr>
<th>Column letters</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row numbers</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>FINAL EXAM MARKS SHEET(CLASS-X: 2001)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>NAME</td>
<td>PHYS</td>
<td>CHEM</td>
<td>MATHS</td>
<td>TOTAL</td>
<td>PERCE</td>
</tr>
<tr>
<td>C</td>
<td>P. Davis</td>
<td>92</td>
<td>95</td>
<td>88</td>
<td>275</td>
<td>91.66</td>
</tr>
<tr>
<td>D</td>
<td>A. Raje</td>
<td>86</td>
<td>82</td>
<td>94</td>
<td>262</td>
<td>87.33</td>
</tr>
<tr>
<td>E</td>
<td>D. Rana</td>
<td>75</td>
<td>83</td>
<td>85</td>
<td>243</td>
<td>81.00</td>
</tr>
<tr>
<td>F</td>
<td>M. Ray</td>
<td>77</td>
<td>75</td>
<td>72</td>
<td>224</td>
<td>74.66</td>
</tr>
<tr>
<td>G</td>
<td>J. Smith</td>
<td>94</td>
<td>92</td>
<td>96</td>
<td>282</td>
<td>94.00</td>
</tr>
<tr>
<td>H</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

Examples of a Line Graph, a Bar Chart and a Pie Chart

(a) A line graph
(b) A bar chart
(c) A pie chart
Graphics package enables us to use a computer system for creating, editing, viewing, storing, retrieving and printing designs, drawings, pictures, graphs and anything else that can be drawn in the traditional manner.

- Drawing designs
- Painting drawings and pictures
- Presenting graphs and charts
- Dragging-and-dropping graphic objects
- Importing graphic objects
- Capturing screen snapshots
Computer Graphics (Few Terminologies)

- **Computer-aided-design (CAD):** Integration of computers and graphics design packages for the purpose of automating the design and drafting process
- **Vector graphics:** Graphic object composed of patterns of lines, points, circles, arcs and other geometric shapes that can be easily represented by few geometric parameters
- **Raster graphics:** Graphic object composed of patterns of dots called **pixels**

Personal-assistance Package

**Personal-assistance package** allows individuals to:

- Use personal computers for storing and retrieving their personal information
- Planning and managing their schedules, contacts, finances and inventory of important items
Common Features of Personal Assistance Package

- Calendar
- To-do list
- Address book
- Investments book
- Inventory book

Key Words/Phrases

- Bit-mapped image
- Bold
- Cell
- Center justification
- Clip art library
- Computer Aided Design (CAD)
- Font
- Full justification
- Graphics package
- Italic
- Justification
- Landscape mode
- Left justification
- Personal assistance package
- Portrait mode
- Raster graphics
- Right justification
- Spread sheet package
- Style sheet
- Underline
- Vector graphics
- What You See Is What you Get (WYSIWYG)
- Word-processing
- Word-processing package
Chapter 15
Application Software Packages

In this chapter you will learn about:

- Word-processing package
- Spreadsheet package
- Graphics package
- Personal assistance package

Learning Objectives

Word-Processing Package

- **Word-processing** describes use of hardware and software to create, edit, view, format, store, retrieve, and print documents (written material such as letters, reports, books, etc.)
- **Word-processing package** enables us to do all these on a computer system
Commonly Supported Features in a Word-Processing Package

- Entering text
- Editing text
- Formatting page style
- Formatting text
- Entering mathematical symbols
- Displaying documents
- Saving, retrieving and deleting documents
- Printing documents
- Importing text, graphics and images
- Searching and replacing text string
- Checking spelling
- Checking grammar and style

Word-Processing (Few Terminologies)

- **Style sheet**: Pre-stored page format that can be used while creating a new document or can be applied to an existing document
- **Font**: Complete set of characters with the same style and size. A word-processing package comes with several standard fonts
- **Points**: A point is 1/72 of an inch, and the size refers to the distance from the top of the tallest character to the bottom of the character that extends the lowest. Font size is measured in points

(Continued on next slide)

Word-Processing (Few Terminologies)

- Three commonly used font styles are italic, bold and underline.
- **Justification**: Alignment of text on the left or the right margin, or on both margins. Four types of justification are:
  - Left-justification
  - Right-justification
  - Center-justification
  - Full-justification
Different Font Types

This sentence is written in Times New Roman font.

This sentence is written in Helvetica font.

This sentence is written in Palatino font.

This sentence is written in Courier New font.

This sentence is written in Antique Olive font.

Different Font Sizes

This sentence is written in 10 point Times New Roman font.

This sentence is written in 12 point Times New Roman font.

This sentence is written in 16 point Times New Roman font.

This sentence is written in 24 point Times New Roman font.

This sentence is written in 36 point Times New Roman font.

Different Font Styles

This sentence is written in italic style.

This sentence is written in bold style.

This sentence is written in underline style.

You can even make individual words italic, bold, or underline.
The term hardware refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

(a) Left Justified text
The term hardware refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

(b) Right Justified text
The term hardware refers to the physical devices of a computer system. Thus, the input, storage, processing, control, and output devices are hardware.

(c) Centered text

Mathematical Symbols
\[
\{ t \in R \land \exists u \in S (u) \land \neg u[1] = u[2] \} = \{ a, b, c \mid a < b \land b < c \leq a \land c < s \}
\]

Spreadsheet Package

- **Spreadsheet package** is a numeric data analysis tool that allows us to create a computerized ledger.
- Useful for any numerical analysis problem whose data can be organized as rows and columns.
Uses of Spreadsheet Package

- Maintaining and analyzing inventory, payroll, and other accounting records by accountants
- Preparing budgets and bid comparisons by business analysts
- Recording grades of students and carrying out various types of analysis of the grades by educators
- Analyzing experimental results by scientists and researchers
- Tracking stocks and keeping records of investor accounts by stockbrokers
- Creating and tracking personal budgets, loan payments, etc. by individuals

Common Features of Spreadsheet Package

- Support for a large number of cells
- Support for addressing a range of cells by the addresses of the endpoint cells
- Support for different types of cell data (such as label, numeric value, formula, and date & time)
- Support for use of relative and absolute cell addresses in formula
- Support for a wide range of commands
- Support for displaying numeric data in the form of graphs and charts

Sample Spreadsheet

<table>
<thead>
<tr>
<th>Column letters</th>
<th>Row numbers</th>
<th>A label running across multiple columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>NAME EXAMS MARKS CLASS X JUMP</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>PHYS CHEM MATHS TOTAL PERCENTAGE</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>D. Rane 75 65 85 242 81.05</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>P. Davis 77 70 75 227 74.76</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>A. Raje 86 82 74 262 87.33</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>J. Smith 90 90 90 270 90.00</td>
</tr>
</tbody>
</table>

Cell C11: Result of the function: SUM(B9..D9)

Cell C13: Result of the formula: E9/3
Examples of a Line Graph, a Bar Chart, and a Pie Chart

(a) A line graph
(b) A bar chart
(c) A pie chart

Graphics Package

Graphics package enables us to use a computer system for creating, editing, viewing, storing, retrieving and printing designs, drawings, pictures, graphs and anything else that can be drawn in the traditional manner.

Common Features of Graphics Package

- Drawing designs
- Painting drawings and pictures
- Presenting graphs and charts
- Dragging-and-dropping graphic objects
- Importing graphic objects
- Capturing screen snapshots
Computer Graphics (Few Terminologies)

- **Computer-aided-design (CAD):** Integration of computers and graphics design packages for the purpose of automating the design and drafting process
- **Vector graphics:** Graphic object composed of patterns of lines, points, circles, arcs and other geometric shapes that can be easily represented by few geometric parameters
- **Raster graphics:** Graphic object composed of patterns of dots called pixels

Personal-assistance Package

**Personal-assistance package** allows individuals to:

- Use personal computers for storing and retrieving their personal information
- Planning and managing their schedules, contacts, finances and inventory of important items

Common Features of Personal Assistance Package

- Calendar
- To-do list
- Address book
- Investments book
- Inventory book
<table>
<thead>
<tr>
<th>Key Words/Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-mapped image</td>
</tr>
<tr>
<td>Bold</td>
</tr>
<tr>
<td>Cell</td>
</tr>
<tr>
<td>Center justification</td>
</tr>
<tr>
<td>Clip art library</td>
</tr>
<tr>
<td>Computer Aided Design (CAD)</td>
</tr>
<tr>
<td>Font</td>
</tr>
<tr>
<td>Full justification</td>
</tr>
<tr>
<td>Graphics package</td>
</tr>
<tr>
<td>Italic</td>
</tr>
<tr>
<td>Justification</td>
</tr>
<tr>
<td>Landscape mode</td>
</tr>
<tr>
<td>Left justification</td>
</tr>
<tr>
<td>Personal assistance package</td>
</tr>
<tr>
<td>Portrait mode</td>
</tr>
<tr>
<td>Raster graphics</td>
</tr>
<tr>
<td>Right justification</td>
</tr>
<tr>
<td>Spread sheet</td>
</tr>
<tr>
<td>Style sheet</td>
</tr>
<tr>
<td>Underline</td>
</tr>
<tr>
<td>Vector graphics</td>
</tr>
<tr>
<td>What You See Is What you Get</td>
</tr>
<tr>
<td>WYSIWYG</td>
</tr>
<tr>
<td>Word-processing</td>
</tr>
<tr>
<td>Word-processing package</td>
</tr>
</tbody>
</table>
Chapter 16

Business Data Processing

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
In this chapter you will learn about:

- Difference between data and information
- Data processing converts raw data into useful information
- Data storage hierarchy commonly used to facilitate data processing
- Standard methods of organizing data
- Basic concepts of database systems
Data is a collection of facts – unorganized but able to be organized into useful information.

Information is data arranged in an order and form that is useful to the people who receive it.

Data processing is a series of actions or operations that converts data into useful information.

A data processing system includes resources such as people, procedures, and devices used to process input data for producing desirable output.
Data Storage Hierarchy

- **Level 0**: Bit - A single binary digit (0 or 1)
- **Level 1**: Character - Multiple related bits are combined to form a character (byte)
- **Level 2**: Field - Multiple related characters are combined to form a field
- **Level 3**: Record - Multiple related fields are combined to form a record
- **Level 4**: File - Multiple related records are combined to form a file
- **Level 5**: Database - Multiple related files are integrated to form a database
Relationship Among Character, Field, Record, and File

A field having 4 characters

Records of a file

A record

Fields

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>First Name</th>
<th>Last Name</th>
<th>Hours worked</th>
<th>Hourly rate</th>
<th>Tax rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>Pradeep</td>
<td>Sinha</td>
<td>45</td>
<td>12.00</td>
<td>0.08</td>
</tr>
<tr>
<td>0002</td>
<td>Ravi</td>
<td>Patel</td>
<td>42</td>
<td>10.00</td>
<td>0.07</td>
</tr>
<tr>
<td>0003</td>
<td>Pratap</td>
<td>Singh</td>
<td>43</td>
<td>15.00</td>
<td>0.10</td>
</tr>
<tr>
<td>0004</td>
<td>Kumar</td>
<td>Rana</td>
<td>40</td>
<td>14.00</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Ref. Page 297  Chapter 16: Business Data Processing  Slide 5/32
Standard Methods of Organizing Data

- **File-oriented approach**: Application’s data is organized into one or more files and application program processes them to generate the desired output.

- **Database-oriented approach**: Data from multiple related files are integrated together to form a database:
  - Provides greater query flexibility
  - Reduces data redundancy
  - Solves data integrity (inconsistency) problem
  - Makes data independent of the application programs
  - Includes data security features at database level, record level, and field level

(Continued on next slide)
In *file-oriented approach* of organizing data, an application’s data is organized into one or more files.

Application program processes the data stored in these files to generate the desired output.

Set of programs is provided to facilitate the users in organizing, creating, deleting, updating, and manipulating their files.

All these programs together form a File Management System (FMS).
A file management system supports following file types:

- **Transaction file**: Stores input data until it can be processed
- **Master file**: Contains all current data relevant to an application
- **Output file**: Stores output produced by one program that is used as input to another program
- **Report file**: Holds a copy of a report generated by an application
- **Backup file**: Copy of a file, created as a safety precaution against loss of data
File Organizations

- File organization is the physical organization of the records of a file for convenience of storage and retrieval of data records.
- Three commonly used file organizations are:
  - **Sequential**: Records are stored one after another in ascending or descending order determined by the value of the key field of the records.
  - **Direct/random**: Desired record pertaining to current transaction can be directly located by its key field value without having to navigate through sequence of other records.

(Continued on next slide)
**Indexed sequential:** There are two files for every data file – the data file which contains the records stored in the file, and the smaller index file which contains the key and disk address of each record stored in the data file.
### Organization of An Indexed Sequential File

<table>
<thead>
<tr>
<th>Employee Code (key)</th>
<th>Address Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>1003</td>
</tr>
<tr>
<td>0002</td>
<td>1001</td>
</tr>
<tr>
<td>0003</td>
<td>1004</td>
</tr>
<tr>
<td>0004</td>
<td>1002</td>
</tr>
</tbody>
</table>

#### Index file

<table>
<thead>
<tr>
<th>Address Location</th>
<th>Employee Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>0002 R. S. Patel...</td>
</tr>
<tr>
<td>1002</td>
<td>0004 R. K. Rana...</td>
</tr>
<tr>
<td>1003</td>
<td>0001 K. P. Sinha...</td>
</tr>
<tr>
<td>1004</td>
<td>0003 N. P. Singh...</td>
</tr>
</tbody>
</table>

#### Data file
File Utilities

- Routines to perform a variety of generalized operations on data files
- Operations performed by some commonly used file utilities are Sorting, Searching, Merging, Copying, Printing, and Maintenance
### Sorting On One Key

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>Department Code</th>
<th>Other fields (Name, Address, Qualification, Basic Salary, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>123</td>
<td>3</td>
<td>---</td>
</tr>
<tr>
<td>124</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>176</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>178</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>202</td>
<td>3</td>
<td>---</td>
</tr>
<tr>
<td>213</td>
<td>1</td>
<td>---</td>
</tr>
</tbody>
</table>

Sorting on ascending employee code sequence
## Sorting On Two Key

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>Department Code</th>
<th>Other fields (Name, Address, Qualification, Basic Salary, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>178</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>213</td>
<td>1</td>
<td>---</td>
</tr>
<tr>
<td>101</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>176</td>
<td>2</td>
<td>---</td>
</tr>
<tr>
<td>123</td>
<td>3</td>
<td>---</td>
</tr>
<tr>
<td>202</td>
<td>3</td>
<td>---</td>
</tr>
</tbody>
</table>

Sorting on a ascending employee code (secondary key) within ascending department code (primary key)
### Merging of Two Files

#### Input file

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

#### Output file

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
</tr>
<tr>
<td>139</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
</tr>
<tr>
<td>152</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

#### Input file

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td></td>
</tr>
<tr>
<td>139</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
</tr>
<tr>
<td>152</td>
<td></td>
</tr>
</tbody>
</table>

#### Merging of files A and B to produce file C

File A:

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

File B:

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td></td>
</tr>
<tr>
<td>139</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
</tr>
<tr>
<td>152</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

File C:

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

Ref. Page 304  Chapter 16: Business Data Processing  Slide 15/32
In *database-oriented approach* of organizing data, a set of programs is provided to facilitate users in organizing, creating, deleting, updating, and manipulating data in a database.

All these programs together form a *Database Management System (DBMS)*.
Database Models

- *Database model* defines the manner in which the various files of a database are linked together.

- Four commonly used database models are:
  - Hierarchical
  - Network
  - Relational
  - Object-oriented
Network Database

A child element can have more than one parent element

This child element has no parent element
### Members data table.

<table>
<thead>
<tr>
<th>Membership No.</th>
<th>Member’s name</th>
<th>Member’s Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>83569</td>
<td>K. N. Raina</td>
<td>C-15, Sarita Vihar, Pune-7</td>
</tr>
<tr>
<td>62853</td>
<td>D. P. Singh</td>
<td>A-22, Anand Park, Pune-5</td>
</tr>
<tr>
<td>12859</td>
<td>R. Pandey</td>
<td>D-18, Vrindavan, Pune-7</td>
</tr>
<tr>
<td>32228</td>
<td>R. S. Gupta</td>
<td>A-12, Nandanvan, Pune-2</td>
</tr>
<tr>
<td>23466</td>
<td>S. K. Ray</td>
<td>B-05, Royal Villa, Pune-3</td>
</tr>
<tr>
<td>11348</td>
<td>P. K. Sen</td>
<td>B-16, Anand Park, Pune-5</td>
</tr>
<tr>
<td>16185</td>
<td>T. N. Murli</td>
<td>A-11, Vrindavan, Pune-7</td>
</tr>
</tbody>
</table>

### Borrowed books data table

<table>
<thead>
<tr>
<th>Borrower (Membership No.)</th>
<th>Book No. (ISBN)</th>
<th>Due Date (DD-MM-YYYY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12859</td>
<td>27-21675-2</td>
<td>10-12-2007</td>
</tr>
<tr>
<td>11348</td>
<td>89303-530-0</td>
<td>08-11-2007</td>
</tr>
<tr>
<td>32228</td>
<td>13-201702-5</td>
<td>10-11-2007</td>
</tr>
<tr>
<td>16185</td>
<td>22-68111-7</td>
<td>05-12-2007</td>
</tr>
<tr>
<td>12859</td>
<td>71606-214-0</td>
<td>06-11-2007</td>
</tr>
<tr>
<td>62853</td>
<td>13-48049-8</td>
<td>15-11-2007</td>
</tr>
<tr>
<td>11348</td>
<td>18-23614-1</td>
<td>12-11-2007</td>
</tr>
</tbody>
</table>

### Books data table

<table>
<thead>
<tr>
<th>Book No. (ISBN)</th>
<th>Book Title</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-201702-5</td>
<td>Concepts of Physics</td>
<td>H. C. Verma</td>
</tr>
<tr>
<td>13-48049-8</td>
<td>Concepts of Chemistry</td>
<td>S. S. Dubey</td>
</tr>
<tr>
<td>18-23614-1</td>
<td>Astrology for You</td>
<td>N. K. Sharma</td>
</tr>
<tr>
<td>22-68111-7</td>
<td>Fundamentals of Computers</td>
<td>K. Ramesh</td>
</tr>
<tr>
<td>27-21675-2</td>
<td>C++ Programming</td>
<td>R. P. Rajan</td>
</tr>
<tr>
<td>71606-214-0</td>
<td>Computer Networks</td>
<td>A. N. Rai</td>
</tr>
<tr>
<td>89303-530-0</td>
<td>Database Systems</td>
<td>P. N. Dixit</td>
</tr>
</tbody>
</table>
List of overdue books as on 10-11-2007

<table>
<thead>
<tr>
<th>Membership No.</th>
<th>Member’s Name</th>
<th>Member’s Address</th>
<th>Due Date</th>
<th>Book No.</th>
<th>Book Title</th>
<th>Book Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>11348</td>
<td>P. K. Sen</td>
<td>B-16, Anand Park, Pune-5</td>
<td>08-11</td>
<td>89303-530-0</td>
<td>Database Systems</td>
<td>P. N. Dixit</td>
</tr>
<tr>
<td>12859</td>
<td>R. Pandey</td>
<td>D-18, Vrindavan, Pune-7</td>
<td>06-11</td>
<td>71606-214-0</td>
<td>Computer Networks</td>
<td>A. N. Rai</td>
</tr>
</tbody>
</table>

A report of overdue books as of 10-11-2007 from the sample database of previous slide
Object-Oriented Database

Vehicle
- Id
- Color
- Specifications
- Manufacturer

TwoWheeler
- Other details of the vehicle like with/without gear, seating capacity, etc.

FourWheeler
- Other details of the vehicle like no. of doors, seating capacity, etc.

VehicleSpecs
- Length
- Width
- Height
- Engine Type
- Fuel Type
- Fuel Tank Capacity
- No. of Wheels

Company
- Name
- Location
- President

DomesticCompany
- Other details of the company

ForeignCompany
- Other details of the company

Employee
- Id
- Name
- Age

Class/subclass link
- Attribute/domain link

Ref. Page 309  Chapter 16: Business Data Processing  Slide 22/32
DBMS allows users to organize, process and retrieve selected data from a database without knowing about the underlying database structure.

Four major components of a DBMS that enable this are:

- **Data Definition Language (DDL):** Used to define the structure (schema) of a database.
- **Data Manipulation Language (DML):** Provides commands to enable the users to enter and manipulate the data.

(Continued on next slide)
Main Components of a DBMS

(Continued from previous slide)

- **Query Language**: Enables users to define their requirements for extracting the desired information from the database in the form of queries.

- **Report generator**: Enables the users of a database to design the layout of a report so that it can be presented in the desired format.
Creating a Database

Creation of a database is a three step process:

- Defining its structure (schema)
- Designing forms (custom screens) for displaying and entering data
- Entering the data into it
## EMPLOYEE DATABASE DATA ENTRY FORM

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE ID:</td>
<td>856392</td>
</tr>
<tr>
<td>SEX:</td>
<td>M</td>
</tr>
<tr>
<td>AGE:</td>
<td>42</td>
</tr>
<tr>
<td>EMPLOYEE NAME:</td>
<td>SINHA</td>
</tr>
<tr>
<td>LAST NAME:</td>
<td>SINHA</td>
</tr>
<tr>
<td>FIRST NAME:</td>
<td>PRADIP</td>
</tr>
<tr>
<td>MIDDLE NAME:</td>
<td>KUMAR</td>
</tr>
<tr>
<td>CONTACT ADDRESS:</td>
<td></td>
</tr>
<tr>
<td>ADDRESS 1:</td>
<td>F/8, ANAND PARK</td>
</tr>
<tr>
<td>ADDRESS 2:</td>
<td>SOCIETY, AUNDH</td>
</tr>
<tr>
<td>CITY:</td>
<td>PUNE</td>
</tr>
<tr>
<td>STATE:</td>
<td>MH</td>
</tr>
<tr>
<td>POSTAL CODE:</td>
<td>411007</td>
</tr>
<tr>
<td>TELEPHONE NO.:</td>
<td>(020) 5680-489</td>
</tr>
<tr>
<td>ANY OTHER INFORMATION:</td>
<td>IS FLUENT IN JAPANESE</td>
</tr>
</tbody>
</table>
All database systems provide commands to view, modify, delete, or add records of an already established database.

Many database systems also provide a facility to set up a filter allowing users to browse through and view only those records that meet some criterion.
Commonly supported features for enabling a user to search for desired information in a database are:

- **Find command**: Used for simple database queries
- **Query language**: Used for more complex database queries
- **Query By Example (QBE)**: Provides a simple user interface for specifying search criteria
Creating Reports

- Reports are generated by using report generator of a database system to assemble the output of a database query in desired format.
- Report generator enables user to specify layout of the report, titles & subtitles for the report, column headings for various fields, and other elements to make the report appear more presentable.
### LIST OF EMPLOYEES WHO BELONG TO PUNE

**DATE: DECEMBER 15, 2007**

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>ADDRESS-1</th>
<th>ADDRESS-2</th>
<th>TELEPHONE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gupta</td>
<td>Rajiv</td>
<td>A-12, Nandanvan</td>
<td>M. G. Road</td>
<td>4623-4892</td>
</tr>
<tr>
<td>Murli</td>
<td>Tapan</td>
<td>A-11, Vrindavan</td>
<td>Pashan Road</td>
<td>5863-4905</td>
</tr>
<tr>
<td>Pandey</td>
<td>Rupa</td>
<td>D-18, Vrindana</td>
<td>Pashan Road</td>
<td>5865-3236</td>
</tr>
<tr>
<td>Raina</td>
<td>Pushpa</td>
<td>C-15, Sarita Vihar</td>
<td>Aundh Road</td>
<td>5755-8328</td>
</tr>
<tr>
<td>Ray</td>
<td>Suhas</td>
<td>B-05, Royal Villa</td>
<td>M. G. Road</td>
<td>4685-6356</td>
</tr>
<tr>
<td>Sen</td>
<td>Prakash</td>
<td>B-16, Anand Park</td>
<td>Aundh Road</td>
<td>5762-3333</td>
</tr>
<tr>
<td>Singh</td>
<td>Deepak</td>
<td>A-22, Anand Park</td>
<td>Aundh Road</td>
<td>5728-6287</td>
</tr>
</tbody>
</table>

The report is sorted to present the list in alphabetical order of their last name.
### Key Words/Phrases

- Activity ratio
- Backup file
- Collision
- Copying
- Data
- Data Definition Language (DDL)
- Data dependence
- Data dictionary
- Data file
- Data integrity
- Data Manipulation Language (DML)
- Data processing
- Data redundancy
- Data storage hierarchy
- Database
- Database administrator
- Database Management System (DBMS)
- Database model
- Direct file
- Field
- File
- File Management System (FMS)
- File utilities
- Filter
- Hashing
- Hashing algorithm
- Hierarchical database
- Index file
- Indexed sequential file
- Information
- Master file
- Merging
- Network database
- Output file
- Peripheral Interchange Program
- Primary key

(Continued on next slide)
Key Words/Phrases

- Query By Example
- Query language
- Record
- Relational database
- Report file
- Report Generator
- Schema
- Searching
- Secondary key
- Secondary key
- Sequential file
- Sorting
- Transaction file
- Tuple

(Continued from previous slide)
Chapter 16
Business Data Processing

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

Learning Objectives

In this chapter you will learn about:
- Difference between data and information
- Data processing converts raw data into useful information
- Data storage hierarchy commonly used to facilitate data processing
- Standard methods of organizing data
- Basic concepts of database systems
Data Processing

- Data is a collection of facts – unorganized but able to be organized into useful information
- Information is data arranged in an order and form that is useful to the people who receive it
- Data processing is a series of actions or operations that converts data into useful information
- A data processing system includes resources such as people, procedures, and devices used to process input data for producing desirable output

Data Storage Hierarchy

- Level 0: Bit – A single binary digit (0 or 1)
- Level 1: Character – Multiple related bits are combined to form a character (byte)
- Level 2: Field – Multiple related characters are combined to form a field
- Level 3: Record – Multiple related fields are combined to form a record
- Level 4: File – Multiple related records are combined to form a file
- Level 5: Database – Multiple related files are integrated to form a database
**Relationship Among Character, Field, Record, and File**

A field having 4 characters

A record

Records of a file

---

**Standard Methods of Organizing Data**

- **File-oriented approach**: Application’s data is organized into one or more files and application program processes them to generate the desired output.

- **Database-oriented approach**: Data from multiple related files are integrated together to form a database:
  - Provides greater query flexibility
  - Reduces data redundancy
  - Solves data integrity (inconsistency) problem
  - Makes data independent of the application programs
  - Includes data security features at database level, record level, and field level

(Continued on next slide)
In file-oriented approach of organizing data, an application's data is organized into one or more files. Application program processes the data stored in these files to generate the desired output. Set of programs is provided to facilitate the users in organizing, creating, deleting, updating, and manipulating their files. All these programs together form a File Management System (FMS).

A file management system supports following file types:

- **Transaction file**: Stores input data until it can be processed.
- **Master file**: Contains all current data relevant to an application.
- **Output file**: Stores output produced by one program that is used as input to another program.
- **Report file**: Holds a copy of a report generated by an application.
- **Backup file**: Copy of a file, created as a safety precaution against loss of data.
File Organizations

File organization is the physical organization of the records of a file for convenience of storage and retrieval of data records.

Three commonly used file organizations are:

- **Sequential**: Records are stored one after another in ascending or descending order determined by the value of the key field of the records.

- **Direct/random**: Desired record pertaining to current transaction can be directly located by its key field value without having to navigate through sequence of other records.

(Continued on next slide)

Indexed sequential: There are two files for every data file – the data file which contains the records stored in the file, and the smaller index file which contains the key and disk address of each record stored in the data file.
### Organization of An Indexed Sequential File

<table>
<thead>
<tr>
<th>Employee Code (key)</th>
<th>Address Location</th>
<th>Address Location</th>
<th>Employee Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>1003</td>
<td>1001</td>
<td>0002 R. S. Patel ...</td>
</tr>
<tr>
<td>0002</td>
<td>1001</td>
<td>1002</td>
<td>0004 R. K. Rana ...</td>
</tr>
<tr>
<td>0003</td>
<td>1004</td>
<td>1003</td>
<td>0001 K. P. Sinha ...</td>
</tr>
<tr>
<td>0004</td>
<td>1002</td>
<td>1004</td>
<td>0003 N. P. Singh ...</td>
</tr>
</tbody>
</table>

Index file | Data file

---

### File Utilities

- Routines to perform a variety of generalized operations on data files
- Operations performed by some commonly used file utilities are Sorting, Searching, Merging, Copying, Printing, and Maintenance
### Sorting On One Key

**Employee Code** | **Department Code** | **Other fields (Name, Address, Qualification, Basic Salary, etc.)**
---|---|---
101 | 2 | ---
123 | 3 | ---
124 | 1 | ---
176 | 2 | ---
178 | 1 | ---
202 | 3 | ---
213 | 1 | ---

Sorting on ascending employee code sequence

### Sorting On Two Key

**Employee Code** | **Department Code** | **Other fields (Name, Address, Qualification, Basic Salary, etc.)**
---|---|---
124 | 1 | ---
178 | 1 | ---
213 | 1 | ---
101 | 2 | ---
176 | 2 | ---
123 | 3 | ---
202 | 3 | ---

Sorting on a ascending employee code (secondary key) within ascending department code (primary key)
Merging of Two Files

File A

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
</tr>
<tr>
<td>137</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

File B

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>139</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
</tr>
<tr>
<td>152</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

File C

<table>
<thead>
<tr>
<th>Employee code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td></td>
</tr>
</tbody>
</table>

Merging of files A and B to produce file C

Database Management System

- In *database-oriented approach* of organizing data, a set of programs is provided to facilitate users in organizing, creating, deleting, updating, and manipulating data in a database.

- All these programs together form a *Database Management System (DBMS)*.
Database Models

- A database model defines the manner in which the various files of a database are linked together.
- Four commonly used database models are:
  - Hierarchical
  - Network
  - Relational
  - Object-oriented

Hierarchical Database

![Diagram of Hierarchical Database](image)
Network Database

- A child element can have more than one parent element
- This child element has no parent element

Relational Database

- (a) Members data table.
- (b) Borrowed books data table
- (c) Books data table

### Members data table

<table>
<thead>
<tr>
<th>Membership No.</th>
<th>Member's name</th>
<th>Member's Address</th>
<th>Due Date (DD-MM-YYYY)</th>
<th>Borrower (Membership No.)</th>
<th>Book No. (ISBN)</th>
<th>Book Title</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>62853</td>
<td>D. P. Singh</td>
<td>A-22, Anand Park, Pune-5</td>
<td>08-11-2007</td>
<td>11348</td>
<td>89303-530-0</td>
<td>Computer Networks</td>
<td>S. S. Dubey</td>
</tr>
<tr>
<td>32228</td>
<td>R. S. Gupta</td>
<td>A-12, Nandanvan, Pune-2</td>
<td>05-12-2007</td>
<td>16185</td>
<td>22-68111-7</td>
<td>Concepts of Physics</td>
<td>R. P. Rajan</td>
</tr>
<tr>
<td>16185</td>
<td>T. N. Murli</td>
<td>A-11, Vrindavan, Pune-7</td>
<td>12-11-2007</td>
<td>11348</td>
<td>18-23614-1</td>
<td>Database Systems</td>
<td>P. N. Dvvt</td>
</tr>
</tbody>
</table>

### Borrowed books data table

<table>
<thead>
<tr>
<th>Membership No.</th>
<th>Member's name</th>
<th>Due Date (DD-MM-YYYY)</th>
<th>Borrower (Membership No.)</th>
<th>Book No. (ISBN)</th>
<th>Book Title</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>32228</td>
<td>R. S. Gupta</td>
<td>05-12-2007</td>
<td>16185</td>
<td>22-68111-7</td>
<td>Concepts of Physics</td>
<td>R. P. Rajan</td>
</tr>
<tr>
<td>23466</td>
<td>S. K. Ray</td>
<td>06-11-2007</td>
<td>12859</td>
<td>71606-214-0</td>
<td>Computer Networks</td>
<td>A. N. Rai</td>
</tr>
<tr>
<td>62853</td>
<td>D. P. Singh</td>
<td>08-11-2007</td>
<td>11348</td>
<td>89303-530-0</td>
<td>Concepts of Chemistry</td>
<td>N. K. Sharma</td>
</tr>
</tbody>
</table>

### Books data table

<table>
<thead>
<tr>
<th>Book No. (ISBN)</th>
<th>Book Title</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-201702-5</td>
<td>Concepts of Physics</td>
<td>H. C. Verma</td>
</tr>
<tr>
<td>13-48049-8</td>
<td>Concepts of Chemistry</td>
<td>S. S. Dubey</td>
</tr>
<tr>
<td>18-23614-1</td>
<td>Astrology for You</td>
<td>N. K. Sharma</td>
</tr>
<tr>
<td>22-68111-7</td>
<td>Fundamentals of Computers</td>
<td>K. Ramesh</td>
</tr>
<tr>
<td>27-21675-2</td>
<td>C++ Programming</td>
<td>R. P. Rajan</td>
</tr>
<tr>
<td>71606-214-0</td>
<td>Computer Networks</td>
<td>A. N. Rai</td>
</tr>
<tr>
<td>89303-530-0</td>
<td>Database Systems</td>
<td>P. N. Dvvt</td>
</tr>
</tbody>
</table>
List of overdue books as on 10-11-2007

<table>
<thead>
<tr>
<th>Membership No.</th>
<th>Member's Name</th>
<th>Member's Address</th>
<th>Due Date</th>
<th>Book No.</th>
<th>Book Title</th>
<th>Book Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>11348</td>
<td>P. K. Sen</td>
<td>B-16, Anand Park, Pune-5</td>
<td>08-11</td>
<td>89303-530-0</td>
<td>Database Systems</td>
<td>P. N. Dixit</td>
</tr>
<tr>
<td>12859</td>
<td>R. Pandey</td>
<td>D-18, Vrindavan, Pune-7</td>
<td>06-11</td>
<td>71606-214-0</td>
<td>Computer Networks</td>
<td>A. N. Rai</td>
</tr>
</tbody>
</table>

A report of overdue books as of 10-11-2007 from the sample database of previous slide

Object-Oriented Database

- Vehicle
  - Id
  - Color
  - Specifications
  - Manufacturer
  - TwoWheeler
    - Other details of the vehicle like with/without gear, seating capacity, etc.
  - FourWheeler
    - Other details of the vehicle like no. of doors, seating capacity, etc.
- VehicleSpecs
  - Length
  - Width
  - Height
  - Engine Type
  - Fuel Type
  - Fuel Tank Capacity
  - No. of Wheels
- Company
  - Name
  - Location
  - President
  - DomesticCompany
    - Other details of the company
  - ForeignCompany
    - Other details of the company
- Employee
  - Id
  - Name
  - Age

Ref. Page 308  Chapter 16: Business Data Processing  Slide 21/32
Ref. Page 309  Chapter 16: Business Data Processing  Slide 22/32
DBMS allows users to organize, process and retrieve selected data from a database without knowing about the underlying database structure.

Four major components of a DBMS that enable this are:

- **Data Definition Language (DDL):** Used to define the structure (schema) of a database.
- **Data Manipulation Language (DML):** Provides commands to enable the users to enter and manipulate the data.

(Continued from previous slide)

**Query Language:** Enables users to define their requirements for extracting the desired information from the database in the form of queries.

**Report generator:** Enables the users of a database to design the layout of a report so that it can be presented in the desired format.
Creating a Database

Creation of a database is a three step process:

- Defining its structure (schema)
- Designing forms (custom screens) for displaying and entering data
- Entering the data into it

Sample Database Form

EMPLOYEE DATABASE DATA ENTRY FORM

<table>
<thead>
<tr>
<th>EMPLOYEE ID:</th>
<th>SEX:</th>
<th>AGE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>856392</td>
<td>M</td>
<td>42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE NAME:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST NAME:</td>
</tr>
<tr>
<td>MIDDLE NAME:</td>
</tr>
<tr>
<td>LAST NAME:</td>
</tr>
<tr>
<td>PRADEEP KUMAR</td>
</tr>
<tr>
<td>SINHA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONTACT ADDRESS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS 1:</td>
</tr>
<tr>
<td>ADDRESS 2:</td>
</tr>
<tr>
<td>CITY:</td>
</tr>
<tr>
<td>STATE:</td>
</tr>
<tr>
<td>POSTAL CODE:</td>
</tr>
<tr>
<td>7/8, ANAND PARK</td>
</tr>
<tr>
<td>SOCIETY, AUNDH</td>
</tr>
<tr>
<td>PUNE</td>
</tr>
<tr>
<td>MH</td>
</tr>
<tr>
<td>411007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TELEPHONE NO.:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(020) 5680-489</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ANY OTHER INFORMATION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS FLUENT IN JAPANESE LANGUAGE</td>
</tr>
</tbody>
</table>
Viewing, Modifying, Deleting, and Adding Records

- All database systems provide commands to view, modify, delete, or add records of an already established database.
- Many database systems also provide a facility to set up a filter allowing the user to browse through and view only those records that meet some criterion.

Searching a Database

Commonly supported features for enabling a user to search for desired information in a database are:

- *Find command*: Used for simple database queries.
- *Query language*: Used for more complex database queries.
- *Query By Example (QBE)*: Provides a simple user interface for specifying search criteria.
Creating Reports

Reports are generated by using report generator of a database system to assemble the output of a database query in desired format.

- Report generator enables user to specify layout of the report, titles & subtitles for the report, column headings for various fields, and other elements to make the report appear more presentable.

Sample Output of Report

LIST OF EMPLOYEES WHO BELONG TO PUNE
DATE: DECEMBER 15, 2007

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>ADDRESS-1</th>
<th>ADDRESS-2</th>
<th>TELEPHONE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gupta</td>
<td>Rajiv</td>
<td>A-12, Nandanvan</td>
<td>M. G. Road</td>
<td>4623-4892</td>
</tr>
<tr>
<td>Murli</td>
<td>Tapan</td>
<td>A-11, Vrindavan</td>
<td>Pashan Road</td>
<td>5863-4905</td>
</tr>
<tr>
<td>Pandey</td>
<td>Rupa</td>
<td>D-18, Vrindana</td>
<td>Pashan Road</td>
<td>5865-3236</td>
</tr>
<tr>
<td>Raina</td>
<td>Pushpa</td>
<td>C-15, Sarita Vihar</td>
<td>Aundh Road</td>
<td>5755-8328</td>
</tr>
<tr>
<td>Ray</td>
<td>Suhas</td>
<td>B-05, Royal Villa</td>
<td>M. G. Road</td>
<td>4685-6356</td>
</tr>
<tr>
<td>Sen</td>
<td>Prakash</td>
<td>B-16, Anand Park</td>
<td>Aundh Road</td>
<td>5762-3333</td>
</tr>
<tr>
<td>Singh</td>
<td>Deepak</td>
<td>A-22, Anand Park</td>
<td>Aundh Road</td>
<td>5728-6287</td>
</tr>
</tbody>
</table>

The report is sorted to present the list in alphabetical order of their last name.
## Chapter 16: Business Data Processing

(Continued from previous slide)

### Key Words/Phrases

- Activity ratio
- Backup file
- Collision
- Copying
- Data
- Data Definition Language (DDL)
- Data dictionary
- Data file
- Data integrity
- Data Manipulation Language (DML)
- Data processing
- Data redundancy
- Data storage hierarchy
- Database
- Database administrator
- Database Management System (DBMS)
- Database model
- Direct file
- Field
- File
- File Management System (FMS)
- File utilities
- Filter
- Hashing
- Hashing algorithm
- Hierarchical database
- Index file
- Indexed sequential file
- Information
- Master file
- Merging
- Network database
- Output file
- Peripheral Interchange Program
- Primary key
- Query By Example
- Query language
- Record
- Relational database
- Report file
- Report Generator
- Schema
- Searching
- Secondary key
- Secondary key
- Sequential file
- Sorting
- Transaction file
- Tuple

(Continued on next slide)
Chapter 16
Business Data Processing

Learning Objectives

In this chapter you will learn about:
1. Difference between data and information
2. Data processing converts raw data into useful information
3. Data storage hierarchy commonly used to facilitate data processing
4. Standard methods of organizing data
5. Basic concepts of database systems

Data Processing

6. Data is a collection of facts – unorganized but able to be organized into useful information
7. Information is data arranged in an order and form that is useful to the people who receive it
8. Data processing is a series of actions or operations that converts data into useful information
9. A data processing system includes resources such as people, procedures, and devices used to process input data for producing desirable output
**Data Storage Hierarchy**

A single binary digit (0 or 1)

Multiple related bits are combined to form a character (byte)

Multiple related characters are combined to form a field

Multiple related fields are combined to form a record

Multiple related records are combined to form a file

Multiple related files are integrated to form a database

---

**Relationship Among Character, Field, Record, and File**

A field having 4 characters

A record

Records of a file

---

**Standard Methods of Organizing Data**

1. **File-oriented approach**: Application’s data is organized into one or more files and application program processes them to generate the desired output.

2. **Database-oriented approach**: Data from multiple related files are integrated together to form a database:
   - Provides greater query flexibility
   - Reduces data redundancy
   - Solves data integrity (inconsistency) problem
   - Makes data independent of the application programs
   - Includes data security features at database level, record level, and field level
In file-oriented approach of organizing data, an application’s data is organized into one or more files. Application program processes the data stored in these files to generate the desired output. A set of programs is provided to facilitate the users in organizing, creating, deleting, updating, and manipulating their files. All these programs together form a File Management System (FMS).

A file management system supports following file types:

- **Transaction file**: Stores input data until it can be processed.
- **Master file**: Contains all current data relevant to an application.
- **Output file**: Stores output produced by one program that is used as input to another program.
- **Report file**: Holds a copy of a report generated by an application.
- **Backup file**: Copy of a file, created as a safety precaution against loss of data.

File organization is the physical organization of the records of a file for convenience of storage and retrieval of data records. Three commonly used file organizations are:

- **Sequential**: Records are stored one after another in ascending or descending order determined by the value of the key field of the records.
- **Direct/random**: Desired record pertaining to current transaction can be directly located by its key field value without having to navigate through sequence of other records.
Indexed sequential: There are two files for every data file – the data file which contains the records stored in the file, and the smaller index file which contains the key and disk address of each record stored in the data file.

Organization of An Indexed Sequential File

Index file

Data file

Employee Code (key) | Address Location | Employee Record
--- | --- | ---
0001 | 1003 | 0002 R. S. Patel...
0002 | 1001 | 0004 R. K. Rana...
0003 | 1004 | 0001 K. P. Sinha...
0004 | 1002 | 0003 N. P. Singh...

File Utilities

Routines to perform a variety of generalized operations on data files
Operations performed by some commonly used file utilities are Sorting, Searching, Merging, Copying, Printing, and Maintenance
### Sorting On One Key

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>Department Code</th>
<th>Other fields (Name, Address, Qualification, Basic Salary, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>213</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Sorting on ascending employee code sequence

### Sorting On Two Key

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>Department Code</th>
<th>Other fields (Name, Address, Qualification, Basic Salary, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>213</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Sorting on a ascending employee code (secondary key) within ascending department code (primary key)

### Merging of Two Files

Merging of files A and B to produce file C

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td></td>
</tr>
<tr>
<td>146</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee Code</th>
<th>Other fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>139</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
</tr>
<tr>
<td>152</td>
<td></td>
</tr>
</tbody>
</table>

Merging of files A and B to produce file C
In database-oriented approach of organizing data, a set of programs is provided to facilitate users in organizing, creating, deleting, updating, and manipulating data in a database.

All these programs together form a Database Management System (DBMS).

Database Model defines the manner in which the various files of a database are linked together.

Four commonly used database models are:

- Hierarchical
- Network
- Relational
- Object-oriented

Hierarchical Database

- Organization
  - Personnel Department
    - Managers
    - Support Staff
  - Finance Department
    - Managers
    - Support Staff
  - Technical Department
    - Managers
    - Engineers
    - Technicians
    - Support Staff

A parent element

A child element
Network Database

Relational Database

Sample Report
**Object-Oriented Database**

- **Vehicle**
  - Color
  - Specifications
  - Manufacturer

- **Two Wheeler**
  - Other details of the vehicle
  - No. of wheels

- **Four Wheeler**
  - Other details of the vehicle
  - Fuel capacity, etc.

- **Company**
  - Location

- **Employee**
  - Name

**Main Components of a DBMS**

- DBMS allows users to organize, process and retrieve selected data from a database without knowing about the underlying database structure.

- Four major components of a DBMS that enable this are:
  - **Data Definition Language (DDL):** Used to define the structure (schema) of a database.
  - **Data Manipulation Language (DML):** Provides commands to enable the users to enter and manipulate the data.

- **Query Language:** Enables users to define their requirements for extracting the desired information from the database in the form of queries.

- **Report generator:** Enables the users of a database to design the layout of a report so that it can be presented in the desired format.
Creating a Database

Creation of a database is a three step process:
6 Defining its structure (schema)
6 Designing forms (custom screens) for displaying and entering data
6 Entering the data into it

Sample Database Form

EMPLOYEE DATABASE DATA ENTRY FORM

EMPLOYEE  ID: 856392
SEX: M
AGE: 42
LAST NAME: SINHA
FIRST NAME: PRADEEP
MIDDLE NAME: KUMAR
ADDRESS 1: F/8, ANAND PARK
ADDRESS 2: SOCIETY, AUNDH
CITY: PUNE
STATE: MH
POSTAL CODE: 411007

TELEPHONE NO.: (020) 5680-489
ANY OTHER INFORMATION: IS FLUENT IN JAPANESE LANGUAGE

Viewing, Modifying, Deleting, and Adding Records

6 All database systems provide commands to view, modify, delete, or add records of an already established database
6 Many database systems also provide a facility to set up a filter allowing user to browse through and view only those records that meet some criterion
### Searching a Database

Commonly supported features for enabling a user to search for desired information in a database are:

- **Find command**: Used for simple database queries
- **Query language**: Used for more complex database queries
- **Query By Example (QBE)**: Provides a simple user interface for specifying search criteria

### Creating Reports

- Reports are generated by using report generator of a database system to assemble the output of a database query in desired format
- Report generator enables user to specify layout of the report, titles & subtitles for the report, column headings for various fields, and other elements to make the report appear more presentable

### Sample Output of Report

#### LIST OF EMPLOYEES WHO BELONG TO PUNE

**DATE: DECEMBER 15, 2007**

<table>
<thead>
<tr>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>ADDRESS-1</th>
<th>ADDRESS-2</th>
<th>TELEPHONE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gupta</td>
<td>Rajiv</td>
<td>A-12, Nandanvan</td>
<td>M. G. Road</td>
<td>4623-4892</td>
</tr>
<tr>
<td>Murlti</td>
<td>Tapan</td>
<td>A-11, Vrindavan</td>
<td>Pashan Road</td>
<td>5863-4905</td>
</tr>
<tr>
<td>Panday</td>
<td>Rupa</td>
<td>D-18, Vridana</td>
<td>Pashan Road</td>
<td>5865-3256</td>
</tr>
<tr>
<td>Raina</td>
<td>Pushpa</td>
<td>C-15, Sarasita Vihar</td>
<td>Aundh Road</td>
<td>5755-8328</td>
</tr>
<tr>
<td>Ray</td>
<td>Suhas</td>
<td>B-05, Royal Villa</td>
<td>M. G. Road</td>
<td>4685-6356</td>
</tr>
<tr>
<td>Sen</td>
<td>Prakash</td>
<td>B-16, Anand Park</td>
<td>Aundh Road</td>
<td>5762-3333</td>
</tr>
<tr>
<td>Singh</td>
<td>Deepak</td>
<td>A-32, Anand Park</td>
<td>Aundh Road</td>
<td>5728-6287</td>
</tr>
</tbody>
</table>

The report is sorted to present the list in alphabetical order of their last name.
### Key Words/Phrases

<table>
<thead>
<tr>
<th>Activity ratio</th>
<th>Database model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup file</td>
<td>Direct file</td>
</tr>
<tr>
<td>Collision</td>
<td>File</td>
</tr>
<tr>
<td>Copying</td>
<td>File utilities</td>
</tr>
<tr>
<td>Data</td>
<td>Flushing</td>
</tr>
<tr>
<td>Data Definition Language (DDL)</td>
<td>Hashing algorithm</td>
</tr>
<tr>
<td>Data dependence</td>
<td>Hierarchy database</td>
</tr>
<tr>
<td>Data dictionary</td>
<td>Index file</td>
</tr>
<tr>
<td>Data integrity</td>
<td>Inverted file</td>
</tr>
<tr>
<td>Data Manipulation Language (DML)</td>
<td>Information</td>
</tr>
<tr>
<td>Data processing</td>
<td>Master file</td>
</tr>
<tr>
<td>Data redundancy</td>
<td>Merging</td>
</tr>
<tr>
<td>Data storage hierarchy</td>
<td>Network database</td>
</tr>
<tr>
<td>Database</td>
<td>Output file</td>
</tr>
<tr>
<td>Database administrator</td>
<td>Peripheral Interchange Program</td>
</tr>
<tr>
<td>Database Management System (DBMS)</td>
<td>Primary key</td>
</tr>
</tbody>
</table>

(Continued on next slide)

### Key Words/Phrases

(Continued from previous slide)

<table>
<thead>
<tr>
<th>Query By Example</th>
<th>Report Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query language</td>
<td>Schema</td>
</tr>
<tr>
<td>Relational database</td>
<td>Searching</td>
</tr>
<tr>
<td>Report file</td>
<td>Secondary key</td>
</tr>
<tr>
<td>Report Generator</td>
<td>Secondary key</td>
</tr>
<tr>
<td>Schema</td>
<td>Sequential file</td>
</tr>
<tr>
<td>Searching</td>
<td>Sorting</td>
</tr>
<tr>
<td>Secondary key</td>
<td>Transaction file</td>
</tr>
<tr>
<td>Secondary key</td>
<td>Tuple</td>
</tr>
</tbody>
</table>
Chapter 17

Data Communications and Computer Networks

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
In this chapter you will learn about:

- Basic elements of a communication system
- Techniques, channels, and devices used to transmit data between distant locations
- Types of computer networks
- Communication protocols and their use in computer networks
- Internetworking tools and their use in building large computer networks
- Characteristics and advantages of distributed data processing
Basic Elements of a Communication System

Sender (source)  
Creates and sends a message

Medium  
Carries the message

Receiver (sink)  
Receives the message
Data Transmission Modes

- **(a) Simplex**: Sender (or Receiver) → Receiver (or Sender)
- **(b) Half-duplex**: Sender (or Receiver) OR Receiver (or Sender)
- **(c) Full-duplex**: Sender (and Receiver) AND Receiver (and Sender)
**Bandwidth**: Range of frequencies available for data transmission. It refers to data transmission rate. Higher the bandwidth, the more data it can transmit.

**Baud**: Unit of measurement of data transfer rate. Measured in bits per second (bps).
**Narrowband**: Sub-voice grade channels in range from 45 to 300 baud. Mainly used for telegraph lines and low-speed terminals

**Voiceband**: Voice grade channels with speed up to 9600 baud. Mainly used for ordinary telephone voice communication and slow I/O devices

**Broadband**: High speed channels with speed up to 1 million baud or more. Mainly used for high-speed computer-to-computer communication or for simultaneous transmission of data
The most commonly used ones are:

- Twisted-pair wire (UTP cable)
- Coaxial cable
- Microwave system
- Communications satellite
- Optical fibers
Unshielded Twisted-Pair (UTP) Cable
Microwave Communication System

Transmitting station —> Transmitting antennas —> Receiving antennas —> Receiving station

In between repeaters

Line of sight

Ref. Page 324
Satellite Communication System

Satellite in space

Transmitting station on earth

Receiving station on earth

6 GHz Uplink

4 GHz Downlink

Ref. Page 325 Chapter 17: Data Communications and Computer Networks Slide 11/57
Optical Fiber Communication System

Sender → Electrical signal → Electrical to light wave converter → Optical fiber → Light to electrical wave converter → Amplifier → Electrical signal → Receiver

- Electrical signal
- Optical fiber
- Light to electrical wave converter
- Optical fiber
- Amplifier
- Electrical signal
Digital and Analog Data Transmission

- **Analog signal**: Transmitted power varies over a continuous range. Example: sound, light, and radio waves.
- **Digital signal**: Sequence of voltage pulses represented in binary form.
- Computer generated data signal is digital, whereas telephone lines carry analog signals.

(Continued on next slide)
When digital data is to be sent over an analog facility, digital signals must be converted to analog form.

Conversion of digital signal to analog form is known as modulation.

Conversion of analog signal to digital form is known as demodulation.

Digital transmission of data is preferred over analog transmission of data due to lower cost, higher transmission speeds, and lower error rate.
Analog and Digital Signals

(a) Analog signal

(b) Digital signal
Modulation Techniques

- **Amplitude Modulation (AM):** Two binary values (0 and 1) of digital data are represented by two different amplitudes of the carrier signal, keeping frequency and phase constant.

- **Frequency Modulation (FM):** Two binary values of digital data are represented by two different frequencies, while amplitude and phase are kept constant.

- **Phase Modulation (PM):** Two binary values of digital data are represented by shift in phase of carrier signal.
Modems

- Modem is short for **MOdulator/DEModulator**
- Special device used for conversion of digital data to analog form (modulation) and vice-versa (demodulation)
- Essential piece of hardware where two digital devices (say two computers) want to communicate over an analog transmission channel (say a telephone line)
Use of Modems in Data Communications

Digital signals

Analog signals on telephone line

Digital signals

Sender Computer

Modulator

Demodulator

Receiver Computer

modem at sender computer end

modem at receiver computer end

0110

0100

0100

0110

0110
Factors for Modem Selection

- Transmission speed
- Internal versus external
- Facsimile facility
Data Transmission Services

- Data transmission service providers are popularly known as *common carriers*

- Various types of services offered by common carriers are:
  - **Dial-up line**: Operates in a manner similar to a telephone line
  - **Leased line**: Special conditioned telephone line that directly and permanently connects two computers
  - **Integrated Services Digital Network (ISDN)**: Telephone system that provides digital (not analog) telephone and data services

(Continued on next slide)
Value Added Network (VAN): Provides value-added data transmission service. Value added over and above the standard services of common carriers may include e-mail, data encryption/decryption, access to commercial databases, and code conversion for communication between computers.
Multiplexing

- Method of dividing physical channel into many logical channels so that a number of independent signals may be simultaneously transmitted.
- Electronic device that performs multiplexing is known as a *multiplexer*.
- Multiplexing enables a single transmission medium to concurrently transmit data between several transmitters and receivers.
Two Basic Methods of Multiplexing

- **Frequency-Division Multiplexing (FDM):** Available bandwidth of a physical medium is divided into several smaller, disjoint logical bandwidths. Each component bandwidth is used as a separate communication line.

- **Time-Division Multiplexing (TDM):** Total time available in a channel is divided among several users, and each user of the channel is allotted a time slice during which he/she may transmit a message.
Frequency-Division Multiplexing

Signal-1 → 40 KHz → Modulator → Sending end → Channel → Receiving end → Demodulator → Signal-1
Signal-2 → 50 KHz
Signal-3 → 60 KHz
Signal-4 → 70 KHz
Signal-5 → 80 KHz
Time-Division Multiplexing

Signal A

Signal B

Signal C

Time sliced signals

Sending end

Receiving end

Channel

Demultiplexer

Reassembled signals

Sending end

Receiving end

Demultiplexer

Reassembled signals

A3 A2 A1

B3 B2 B1

C3 C2 C1

A3 A2 A1

B3 B2 B1

C3 C2 C1
Asynchronous and Synchronous Transmission

- Two modes of data transmission on a communication line are asynchronous and synchronous.

- Asynchronous transmission
  - Sender can send data at any convenient time and the receiver will accept it.
  - Data is transmitted character by character at irregular intervals.
  - Well suited to many keyboard type terminals.

(Continued on next slide)
Asynchronous and Synchronous Transmission

- Synchronous transmission
  - Sender and receiver must synchronize with each other to get ready for data transmission before it takes place
  - Entire blocks of characters are framed and transmitted together
  - Well suited to remote communication between a computer and such devices as buffered terminals and printers

(Continued from previous slide)
Data Transmission

(a) Asynchronous transmission

Irregular time intervals between two characters
Each character framed by start and stop bits

(b) Synchronous transmission

Indefinite time interval between two blocks of data
A block of characters may consist of hundreds of characters

Trailer containing end of block indication
Header containing synchronizing and other information
Switching Techniques

- Data is often transmitted from source to destination through a network of intermediate nodes.
- Switching techniques deal with the methods of establishing communication links between the sender and receiver in a communication network.
- Three commonly used switching techniques are:
  - **Circuit switching:** Dedicated physical path is established between sending and receiving stations through nodes of the network for the duration of communication.
Message switching: Sender appends receiver’s destination address to the message and it is transmitted from source to destination either by store-and-forward method or broadcast method.

Packet switching: Message is split up into fixed size packets and each packet is transmitted independently from source to destination node. Either store-and-forward or broadcast method is used for transmitting the packets. All the packets of a message are re-assembled into original message at the destination node.
Circuit Switching Method

Dotted line indicates establishment of physical path

Switching nodes

Source node

Destination node
Either path 1-2-3-4 or 1-5-4 may be used to transmit a message from A to B.
Broadcast Method of Message Switching

Nodes \[\rightarrow\] 1 \[\rightarrow\] 2 \[\rightarrow\] 3 \[\rightarrow\] \ldots \ldots \[\rightarrow\] n

Message \[\rightarrow\] Broadcast Channel
Routing Techniques

- In a WAN, when multiple paths exist between the source and destination nodes of a packet, any one of the paths may be used to transfer the packet.

- Selection of path to be used for transmitting a packet is determined by the routing technique used.

- Two popularly used routing algorithms are:
  - **Source routing**: Source node selects the entire path before sending the packet.
  - **Hop-by-hop routing**: Each node along the path decides only the next node for the path.
Network Topologies

Term *network topology* refers to the way in which the nodes of a network are linked together.

Although number network topologies are possible, four major ones are:

- Star network
- Ring network
- Completely connected network
- Multi-access bus network
Star Network

Host
Node

Star Network
Ring Network
Completely Connected Network

![Diagram of a completely connected network](image)
Multi-Access Bus Network

Computers (nodes)

Single communication line shared by all nodes
Hybrid Network

Ring

Star

Completely connected
Network Types

- Networks are broadly classified into two types: Local Area Network (LAN) and Wide Area Network (WAN).

- Local Area Network (LAN) as compared to WAN:
  - Limited to a small geographic coverage
  - Has much higher data transmission rate
  - Experiences fewer data transmission errors
  - Has lower data communication cost
  - Typically owned by a single organization

- Networks that share some of the characteristics of both LANs and WANs are referred to as Metropolitan Area Network (MAN).
Protocol is a set of formal operating rules, procedures, or conventions that govern a given process.

Communication protocol describes rules that govern transmission of data over communication networks.

Roles of communication protocol:
- Data sequencing
- Data routing
- Data formatting
- Flow control
- Error control

(Continued on next slide)
Communication Protocols

(Continued from previous slide)

- Precedence and order of transmission
- Connection establishment and termination
- Data security
- Log information.

Communication protocols are normally split up into a series of modules logically composed of a succession of layers.

Terms *protocol suite*, *protocol family*, or *protocol stack* are used to refer to the collection of protocols (of all layers) of a network system.
Network Interface Card (NIC)

- Hardware device that allows a computer to be connected to a network, both functionally and physically
- Printed circuit board installed on to one of the expansion slots of computer
- Provides a port on the back to which network cable is attached
The OSI Model

- The Open System Interconnection (OSI) model is a framework for defining standards for linking heterogeneous computers in a packet switched network.

- Standardized OSI protocol makes it possible for any two heterogeneous computer systems, located anywhere in the world, to easily communicate with each other.

- Separate set of protocols is defined for each layer in its seven-layer architecture. Each layer has an independent function.
Layers, Interfaces, and Protocols in the OSI Model

Node 1
- Process A
  - Layer 7 (application)
  - Interface
  - Layer 6 (presentation)
  - Interface
  - Layer 5 (session)
  - Interface
  - Layer 4 (transport)
  - Interface
  - Layer 3 (network)
  - Interface
  - Layer 2 (data link)
  - Interface
  - Layer 1 (physical)

Node 2
- Process B
  - Application protocol
  - Interface
  - Layer 7 (application)
  - Interface
  - Layer 6 (presentation)
  - Interface
  - Layer 5 (session)
  - Interface
  - Layer 4 (transport)
  - Interface
  - Layer 3 (network)
  - Interface
  - Layer 2 (data link)
  - Interface
  - Layer 1 (physical)
An example illustrating transfer of message M from sending node to the receiving node in the OSI model: \( H_n \), header added by layer n; \( T_n \), trailer added by layer n.
Internetworking

- Interconnecting two or more networks to form a single network is called *internetworking*, and the resulting network is called an *internetwork*.
- Goal of internetworking is to hide details of different physical networks, so that resulting internetwork functions as a single coordinated unit.
- Tools such as bridges, routers, brouters, and gateways are used for internetworking.
- The Internet is the best example of an internetwork.
Bridges

- Operate at bottom two layers of the OSI model
- Connect networks that use the same communication protocols above data-link layer but may use different protocols at physical and data-link layers
Routers

- Operates at network layer of the OSI model
- Used to interconnect those networks that use the same high-level protocols above network layer
- Smarter than bridges as they not only copy data from one network segment to another, but also choose the best route for the data by using routing table
Gateways

- Operates at the top three layers of the OSI model (session, presentation and application)
- Used for interconnecting dissimilar networks that use different communication protocols
- Since gateways interconnect dissimilar networks, protocol conversion is the major job performed by them
Wireless Computing Systems

- Wireless computing system uses wireless communication technologies for interconnecting computer systems
- Enhances functionality of computing equipment by freeing communication from location constraints of wired computing systems
- Wireless computing systems are of two types:
  - **Fixed wireless systems**: Support little or no mobility of the computing equipment associated with the wireless network
  - **Mobile wireless systems**: Support mobility of the computing equipment to access resources associated with the wireless network
Wireless Technologies

- 2G and 3G
- Wireless LAN
- WiMAX
- Wireless Local Loop (WLL)
- Radio-router
- Multihop Wireless Network
- Wireless Application Protocol (WAP)
B Distributed Computing Systems

- Configuration where many independent computer systems are connected, and messages, processing task, programs, data, and other resources are transmitted between cooperating computer systems.

- Such an arrangement enables sharing of many hardware and software resources as well as information among several users who may be sitting far away from each other.
Main Advantages of Distributed Computing Systems

- Inherently distributed applications
- Information sharing among distributed users
- Resource sharing
- Shorter response times and higher throughput
- Higher reliability
- Extensibility and incremental growth
- Better flexibility in meeting users’ needs
Keywords/Phrases

- Amplifier
- Amplitude Modulation (AM)
- Application layer
- ARPANET
- Asynchronous transmission
- Bandwidth
- Baud
- Bridge
- Broadband
- Broadcast
- C-band transmission
- Circuit switching
- Coaxial cable
- Common Carriers
- Communication protocol
- Communications satellite
- Completely connected network
- Computer network
- Concentrators
- Data-link layer
- Demodulation
- Dial-up line
- Distributed Computing System
- Ethernet
- Fax modem
- File Transfer Protocol (FTP)
- Font-End Processors (FEP)
- Frequency Modulation (FM)
- Frequency-Division Multiplexing (FDM)
- Full duplex
- Gateway
- Half duplex
- Hop-by-hop routing
- Hybrid network
- Internet Protocol (IP)
- Internetworking
- ISDN (Integrated Services Digital Network)
- Ku-band transmission
- Leased line
- Local Area Network (LAN)
- Message switching

(Continued on next slide)
Keywords/Phrases

- Metropolitan Area Network (MAN)
- Microwave system
- Mobile computing
- Modem
- Modulation
- Multi-access Bus network
- Multiplexer
- Narrowband
- Network Interface Card (NIC)
- Network layer
- Network topology
- Nomadic computing
- Optical fibers
- OSI Model
- Packet switching
- Phase Modulation (PM)
- Physical layer
- POTS (Plain Old Telephone Service)
- Presentation layer
- Protocol family
- Protocol stack

- Protocol suite
- Repeater
- Ring network
- Router
- Session layer
- Simplex
- Source routing
- Star network
- Store-and-forward
- Synchronous transmission
- Time-Division Multiplexing (TDM)
- Transport Control Protocol (TCP)
- Transport layer
- Twisted-pair
- Unshielded twisted-pair (UTP)
- User Datagram Protocol (UDP)
- Value Added Network (VAN)
- Voiceband
- VSAT (Very Small Aperture Terminals)
- Wide Area Network (WAN)
- Wireless network

(Continued from previous slide)
Chapter 17
Data Communications
and Computer Networks

Learning Objectives

In this chapter you will learn about:

- Basic elements of a communication system
- Techniques, channels, and devices used to transmit data between distant locations
- Types of computer networks
- Communication protocols and their use in computer networks
- Internetworking tools and their use in building large computer networks
- Characteristics and advantages of distributed data processing
Basic Elements of a Communication System

- **Sender** (source) creates and sends a message through the **Medium**.
- **Receiver** (sink) receives the message.

Data Transmission Modes

- (a) **Simplex** transmission: Sender to Receiver
- (b) **Half-duplex** transmission: Sender (or Receiver) to Receiver (or Sender)
- (c) **Full-duplex** transmission: Sender (and Receiver) to Receiver (and Sender)
Data Transmission Speed

β Bandwidth: Range of frequencies available for data transmission. It refers to data transmission rate. Higher the bandwidth, the more data it can transmit.

β Baud: Unit of measurement of data transfer rate. Measured in bits per second (bps)

Data Transmission Speed Category

β Narrowband: Sub-voice grade channels in range from 45 to 300 baud. Mainly used for telegraph lines and low-speed terminals.

β Voiceband: Voice grade channels with speed up to 9600 baud. Mainly used for ordinary telephone voice communication and slow I/O devices.

β Broadband: High speed channels with speed up to 1 million baud or more. Mainly used for high-speed computer-to-computer communication or for simultaneous transmission of data.
The most commonly used ones are:

- Twisted-pair wire (UTP cable)
- Coaxial cable
- Microwave system
- Communications satellite
- Optical fibers

Unshielded Twisted-Pair (UTP) Cable
Coaxial Cable

- Outer PVC shield
- Copper mesh
- PVC insulation
- Central copper wire

Microwave Communication System

- Transmitting antennas
- Receiving antennas
- In between repeaters
- Line of sight
- Transmitting station
- Receiving station

Ref. Page 323  Chapter 17: Data Communications and Computer Networks  Slide 9/57

Ref. Page 324  Chapter 17: Data Communications and Computer Networks  Slide 10/57
Satellite Communication System

Satellite in space

6 GHz
Uplink

4 GHz
Downlink

Transmitting
station on earth

Receiving
station on earth

Optical Fiber Communication System

Sender
Electrical signal

Electrical to light wave converter

Optical fiber

Light waves

Light to electrical wave converter

Amplifier

Electrical signal

Receiver
Digital and Analog Data Transmission

- Analog signal: Transmitted power varies over a continuous range. Example: sound, light, and radio waves
- Digital signal: Sequence of voltage pulses represented in binary form
- Computer generated data signal is digital, whereas telephone lines carry analog signals

(Continued from previous slide)

When digital data is to be sent over an analog facility, digital signals must be converted to analog form
- Conversion of digital signal to analog form is known as modulation
- Conversion of analog signal to digital form is known as demodulation
- Digital transmission of data is preferred over analog transmission of data due to lower cost, higher transmission speeds, and lower error rate
Analog and Digital Signals

(a) Analog signal

(b) Digital signal

Modulation Techniques

- **Amplitude Modulation (AM):** Two binary values (0 and 1) of digital data are represented by two different amplitudes of the carrier signal, keeping frequency and phase constant.

- **Frequency Modulation (FM):** Two binary values of digital data are represented by two different frequencies, while amplitude and phase are kept constant.

- **Phase Modulation (PM):** Two binary values of digital data are represented by shift in phase of carrier signal.
Modems

- Modem is short for MODulator/DEModulator
- Special device used for conversion of digital data to analog form (modulation) and vice-versa (demodulation)
- Essential piece of hardware where two digital devices (say two computers) want to communicate over an analog transmission channel (say a telephone line)

Use of Modems in Data Communications

Diagram showing the process of converting digital signals to analog signals and vice versa using modems at both the sender and receiver ends.
Factors for Modem Selection

- Transmission speed
- Internal versus external
- Facsimile facility

Data Transmission Services

- Data transmission service providers are popularly known as common carriers
- Various types of services offered by common carriers are:
  - **Dial-up line**: Operates in a manner similar to a telephone line
  - **Leased line**: Special conditioned telephone line that directly and permanently connects two computers
  - **Integrated Services Digital Network (ISDN)**: Telephone system that provides digital (not analog) telephone and data services
**Data Transmission Services**

(Continued from previous slide)

- **Value Added Network (VAN):** Provides value-added data transmission service. Value added over and above the standard services of common carriers may include e-mail, data encryption/decryption, access to commercial databases, and code conversion for communication between computers.

**Multiplexing**

- Method of dividing physical channel into many logical channels so that a number of independent signals may be simultaneously transmitted.
- Electronic device that performs multiplexing is known as a *multiplexer*.
- Multiplexing enables a single transmission medium to concurrently transmit data between several transmitters and receivers.
Two Basic Methods of Multiplexing

- **Frequency-Division Multiplexing (FDM):** Available bandwidth of a physical medium is divided into several smaller, disjoint logical bandwidths. Each component bandwidth is used as a separate communication line.

- **Time-Division Multiplexing (TDM):** Total time available in a channel is divided among several users, and each user of the channel is allotted a time slice during which he/she may transmit a message.

Frequency-Division Multiplexing

<table>
<thead>
<tr>
<th>Signal</th>
<th>Frequency (KHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>

Modulator

Demodulator

50 KHz

60 KHz

70 KHz

80 KHz

40 KHz

50 KHz

60 KHz

70 KHz

80 KHz

Signal-1

Signal-2

Signal-3

Signal-4

Signal-5

Sending end

Receiving end

Channel

Frequency-Division Multiplexing
**Time-Division Multiplexing**

![Diagram of Time-Division Multiplexing]

**Asynchronous and Synchronous Transmission**

- Two modes of data transmission on a communication line are asynchronous and synchronous.

- Asynchronous transmission
  - Sender can send data at any convenient time and the receiver will accept it.
  - Data is transmitted character by character at irregular intervals.
  - Well suited to many keyboard type terminals.

(Continued on next slide)
Asynchronous and Synchronous Transmission

(Continued from previous slide)

- Synchronous transmission
  - Sender and receiver must synchronize with each other to get ready for data transmission before it takes place
  - Entire blocks of characters are framed and transmitted together
  - Well suited to remote communication between a computer and such devices as buffered terminals and printers

Data Transmission

(a) Asynchronous transmission

- Irregular time intervals between two characters
- Each character framed by start and stop bits

(b) Synchronous transmission

- Indefinite time interval between two blocks of data
- A block of characters may consist of hundreds of characters
- Trailer containing end of block indication
- Header containing synchronizing and other information
Switching Techniques

- Data is often transmitted from source to destination through a network of intermediate nodes.
- Switching techniques deal with the methods of establishing communication links between the sender and receiver in a communication network.
- Three commonly used switching techniques are:
  - **Circuit switching**: Dedicated physical path is established between sending and receiving stations through nodes of the network for the duration of communication.
  - **Message switching**: Sender appends receiver’s destination address to the message and it is transmitted from source to destination either by store-and-forward method or broadcast method.
  - **Packet switching**: Message is split up into fixed size packets and each packet is transmitted independently from source to destination node. Either store-and-forward or broadcast method is used for transmitting the packets. All the packets of a message are re-assembled into original message at the destination node.
Circuit Switching Method

Switching nodes

Dotted line indicates establishment of physical path

Source node

Destination node

-store-and-forward method of message switching

Store-and-Forward Method of Message Switching

Either path 1-2-3-4 or 1-5-4 may be used to transmit a message from A to B.
Broadcast Method of Message Switching

Nodes → 1 → 2 → 3 → ... → n

Broadcast Channel

Routing Techniques

- In a WAN, when multiple paths exist between the source and destination nodes of a packet, any one of the paths may be used to transfer the packet.
- Selection of path to be used for transmitting a packet is determined by the routing technique used.
- Two popularly used routing algorithms are:
  - **Source routing**: Source node selects the entire path before sending the packet.
  - **Hop-by-hop routing**: Each node along the path decides only the next node for the path.
Network Topologies

- Term network topology refers to the way in which the nodes of a network are linked together.
- Although numerous network topologies are possible, four major ones are:
  - Star network
  - Ring network
  - Completely connected network
  - Multi-access bus network
**Ring Network**

![Ring Network Diagram]

**Completely Connected Network**

![Completely Connected Network Diagram]
Multi-Access Bus Network

Computers (nodes)

Single communication line shared by all nodes

Hybrid Network

Ring

Star

Completely connected
Network Types

Networks are broadly classified into two types: Local Area Network (LAN) and Wide Area Network (WAN).

- Local Area Network (LAN) as compared to WAN:
  - Limited to a small geographic coverage
  - Has much higher data transmission rate
  - Experiences fewer data transmission errors
  - Has lower data communication cost
  - Typically owned by a single organization

Networks that share some of the characteristics of both LANs and WANs are referred to as Metropolitan Area Network (MAN).

Communication Protocols

- Protocol is a set of formal operating rules, procedures, or conventions that govern a given process
- Communication protocol describes rules that govern transmission of data over communication networks
- Roles of communication protocol:
  - Data sequencing
  - Data routing
  - Data formatting
  - Flow control
  - Error control

(Continued on next slide)
Communication Protocols

(Continued from previous slide)

- Precedence and order of transmission
- Connection establishment and termination
- Data security
- Log information.

Communication protocols are normally split up into a series of modules logically composed of a succession of layers.

Terms *protocol suite*, *protocol family*, or *protocol stack* are used to refer to the collection of protocols (of all layers) of a network system.

Network Interface Card (NIC)

- Hardware device that allows a computer to be connected to a network, both functionally and physically
- Printed circuit board installed on to one of the expansion slots of computer
- Provides a port on the back to which network cable is attached
The OSI Model

- The Open System Interconnection (OSI) model is a framework for defining standards for linking heterogeneous computers in a packet-switched network.
- Standardized OSI protocol makes it possible for any two heterogeneous computer systems, located anywhere in the world, to easily communicate with each other.
- Separate set of protocols is defined for each layer in its seven-layer architecture. Each layer has an independent function.

Layers, Interfaces, and Protocols in the OSI Model

![Diagram of the OSI Model with layers, interfaces, and protocols labeled.](image)
Internetworking

- Interconnecting two or more networks to form a single network is called **internetworking**, and the resulting network is called an **internetwork**

- Goal of internetworking is to hide details of different physical networks, so that resulting internetwork functions as a single coordinated unit

- Tools such as bridges, routers, brouters, and gateways are used for internetworking

- The Internet is the best example of an internetwork
Bridges

- Operate at bottom two layers of the OSI model
- Connect networks that use the same communication protocols above data-link layer but may use different protocols at physical and data-link layers

Routers

- Operates at network layer of the OSI model
- Used to interconnect those networks that use the same high-level protocols above network layer
- Smarter than bridges as they not only copy data from one network segment to another, but also choose the best route for the data by using routing table
Gateways

- Operates at the top three layers of the OSI model (session, presentation and application)
- Used for interconnecting dissimilar networks that use different communication protocols
- Since gateways interconnect dissimilar networks, protocol conversion is the major job performed by them.

Wireless Computing Systems

- Wireless computing system uses wireless communication technologies for interconnecting computer systems
- Enhances functionality of computing equipment by freeing communication from location constraints of wired computing systems
- Wireless computing systems are of two types:
  - **Fixed wireless systems**: Support little or no mobility of the computing equipment associated with the wireless network
  - **Mobile wireless systems**: Support mobility of the computing equipment to access resources associated with the wireless network
Wireless Technologies

- 2G and 3G
- Wireless LAN
- WiMAX
- Wireless Local Loop (WLL)
- Radio-router
- Multihop Wireless Network
- Wireless Application Protocol (WAP)

Distributed Computing Systems

- Configuration where many independent computer systems are connected, and messages, processing task, programs, data, and other resources are transmitted between cooperating computer systems.
- Such an arrangement enables sharing of many hardware and software resources as well as information among several users who may be sitting far away from each other.
Main Advantages of Distributed Computing Systems

- Inherently distributed applications
- Information sharing among distributed users
- Resource sharing
- Shorter response times and higher throughput
- Higher reliability
- Extensibility and incremental growth
- Better flexibility in meeting users’ needs

Keywords/Phrases

- Amplifier
- Amplitude Modulation (AM)
- Application layer
- ARPANET
- Asynchronous transmission
- Bandwidth
- Baud
- Bridge
- Broadband
- Broadcast
- C-band transmission
- Circuit switching
- Coaxial cable
- Common Carriers
- Communication protocol
- Communications satellite
- Completely connected network
- Computer network
- Concentrators
- Data-link layer
- Demodulation
- Dial-up line
- Distributed Computing System
- Fax modem
- File Transfer Protocol (FTP)
- Font-End Processors (FEP)
- Frequency Modulation (FM)
- Frequency-Division Multiplexing (FDM)
- Full duplex
- Gateway
- Half duplex
- Hop-by-hop routing
- Hybrid network
- Internet Protocol (IP)
- Internetworking
- ISDN (Integrated Services Digital Network)
- Ku-band transmission
- Leased line
- Local Area Network (LAN)
- Message switching

(Continued on next slide)
<table>
<thead>
<tr>
<th>Keywords/Phrases</th>
<th>Keywords/Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metropolitan Area Network (MAN)</td>
<td>Protocol suite</td>
</tr>
<tr>
<td>Microwave system</td>
<td>Repeater</td>
</tr>
<tr>
<td>Mobile computing</td>
<td>Ring network</td>
</tr>
<tr>
<td>Modem</td>
<td>Router</td>
</tr>
<tr>
<td>Modulation</td>
<td>Session layer</td>
</tr>
<tr>
<td>Multi-access Bus network</td>
<td>Simplex</td>
</tr>
<tr>
<td>Multiplexer</td>
<td>Source routing</td>
</tr>
<tr>
<td>Narrowband</td>
<td>Star network</td>
</tr>
<tr>
<td>Network Interface Card (NIC)</td>
<td>Store-and-forward</td>
</tr>
<tr>
<td>Network layer</td>
<td>Synchronous transmission</td>
</tr>
<tr>
<td>Network topology</td>
<td>Time-Division Multiplexing (TDM)</td>
</tr>
<tr>
<td>Nomadic computing</td>
<td>Transport Control Protocol (TCP)</td>
</tr>
<tr>
<td>Optical fibers</td>
<td>Transport layer</td>
</tr>
<tr>
<td>OSI Model</td>
<td>Twisted-pair</td>
</tr>
<tr>
<td>Packet switching</td>
<td>Unshielded twisted-pair (UTP)</td>
</tr>
<tr>
<td>Phase Modulation (PM)</td>
<td>User Datagram Protocol (UDP)</td>
</tr>
<tr>
<td>Physical layer</td>
<td>Value Added Network (VAN)</td>
</tr>
<tr>
<td>POTS (Plain Old Telephone Service)</td>
<td>Voiceband</td>
</tr>
<tr>
<td>Presentation layer</td>
<td>VSAT (Very Small Aperture Terminals)</td>
</tr>
<tr>
<td>Protocol family</td>
<td>Wide Area Network (WAN)</td>
</tr>
<tr>
<td>Protocol stack</td>
<td>Wireless network</td>
</tr>
</tbody>
</table>

(Continued from previous slide)
In this chapter you will learn about:

- Basic elements of a communication system
- Techniques, channels, and devices used to transmit data between distant locations
- Types of computer networks
- Communication protocols and their use in computer networks
- Internetworking tools and their use in building large computer networks
- Characteristics and advantages of distributed data processing
Data Transmission Modes

(a) Simplex

Sender

Receiver

OR

(b) Half-duplex

Sender (or Receiver)

OR

Receiver (or Sender)

AND

(c) Full-duplex

Sender (and Receiver)

AND

Receiver (and Sender)

Data Transmission Speed

- **Bandwidth**: Range of frequencies available for data transmission. It refers to data transmission rate. Higher the bandwidth, the more data it can transmit.

- **Baud**: Unit of measurement of data transfer rate. Measured in bits per second (bps).

Data Transmission Speed Category

- **Narrowband**: Sub-voice grade channels in range from 45 to 300 baud. Mainly used for telegraph lines and low-speed terminals.

- **Voiceband**: Voice grade channels with speed up to 9600 baud. Mainly used for ordinary telephone voice communication and slow I/O devices.

- **Broadband**: High speed channels with speed up to 1 million baud or more. Mainly used for high-speed computer-to-computer communication or for simultaneous transmission of data.
The most commonly used ones are:

- Twisted-pair wire (UTP cable)
- Coaxial cable
- Microwave system
- Communications satellite
- Optical fibers
Microwave Communication System

- Transmitting station
- Receiving station
- In between repeaters
- Transmitting antennas
- Receiving antennas
- Line of sight

Satellite Communication System

- Satellite in space
- 6 GHz Uplink
- 4 GHz Downlink
- Transmitting station on earth
- Receiving station on earth

Optical Fiber Communication System

- Electrical signal
- Optical fiber
- Light to electrical wave converter
- Electrical signal
- Light waves
Digital and Analog Data Transmission

- Analog signal: Transmitted power varies over a continuous range. Example: sound, light, and radio waves
- Digital signal: Sequence of voltage pulses represented in binary form
- Computer generated data signal is digital, whereas telephone lines carry analog signals

(Continued from previous slide)

Digital and Analog Data Transmission

- When digital data is to be sent over an analog facility, digital signals must be converted to analog form
- Conversion of digital signal to analog form is known as modulation
- Conversion of analog signal to digital form is known as demodulation
- Digital transmission of data is preferred over analog transmission of data due to lower cost, higher transmission speeds, and lower error rate

(Continued on next slide)

Analog and Digital Signals

(a) Analog signal

(b) Digital signal

Voltage

0 1

0 1 0 1 0 0

t

t
**Modulation Techniques**

- **Amplitude Modulation (AM):** Two binary values (0 and 1) of digital data are represented by two different amplitudes of the carrier signal, keeping frequency and phase constant.

- **Frequency Modulation (FM):** Two binary values of digital data are represented by two different frequencies, while amplitude and phase are kept constant.

- **Phase Modulation (PM):** Two binary values of digital data are represented by shift in phase of carrier signal.

**Modems**

- Modem is short for **Modulator/DE**modulator.
- Special device used for conversion of digital data to analog form (modulation) and vice-versa (demodulation).
- Essential piece of hardware where two digital devices (say two computers) want to communicate over an analog transmission channel (say a telephone line).

**Use of Modems in Data Communications**

[Diagram showing the use of modems in data communications with labels for digital signals, modulators, and demodulators at both sender and receiver computer ends.]
Factors for Modem Selection

- Transmission speed
- Internal versus external
- Facsimile facility

Data Transmission Services

Data transmission service providers are popularly known as common carriers.

Various types of services offered by common carriers are:

- **Dial-up line**: Operates in a manner similar to a telephone line
- **Leased line**: Special conditioned telephone line that directly and permanently connects two computers
- **Integrated Services Digital Network (ISDN)**: Telephone system that provides digital (not analog) telephone and data services

(Continued from previous slide)

**Value Added Network (VAN)**: Provides value-added data transmission service. Value added over and above the standard services of common carriers may include e-mail, data encryption/decryption, access to commercial databases, and code conversion for communication between computers

(Continued from previous slide)
Multiplexing

- Method of dividing physical channel into many logical channels so that a number of independent signals may be simultaneously transmitted
- Electronic device that performs multiplexing is known as a multiplexer
- Multiplexing enables a single transmission medium to concurrently transmit data between several transmitters and receivers

Two Basic Methods of Multiplexing

- Frequency-Division Multiplexing (FDM): Available bandwidth of a physical medium is divided into several smaller, disjoint logical bandwidths. Each component bandwidth is used as a separate communication line
- Time-Division Multiplexing (TDM): Total time available in a channel is divided among several users, and each user of the channel is allotted a time slice during which he/she may transmit a message
Two modes of data transmission on a communication line are asynchronous and synchronous.

**Asynchronous transmission**
- Sender can send data at any convenient time and the receiver will accept it.
- Data is transmitted character by character at irregular intervals.
- Well suited to many keyboard type terminals.

**Synchronous transmission**
- Sender and receiver must synchronize with each other to get ready for data transmission before it takes place.
- Entire blocks of characters are framed and transmitted together.
- Well suited to remote communication between a computer and such devices as buffered terminals and printers.
Irregular time intervals between two characters
Each character framed by start and stop bits

(a) Asynchronous transmission

Indefinite time interval between two blocks of data
A block of characters may consist of hundreds of characters
Trailer containing end of block indication
Header containing synchronizing and other information

(b) Synchronous transmission

Switching Techniques

- Data is often transmitted from source to destination through a network of intermediate nodes
- Switching techniques deal with the methods of establishing communication links between the sender and receiver in a communication network
- Three commonly used switching techniques are:
  - **Circuit switching**: Dedicated physical path is established between sending and receiving stations through nodes of the network for the duration of communication
  (Continued from previous slide)

(Continued on next slide)
Slide 31/57

Chapter 17: Data Communications and Computer Networks

Ref. Page

Source node

Dotted line indicates establishment of physical path

Switching nodes

Destination node

Slide 32/57

Circuit Switching Method

Store-and-Forward Method of Message Switching

Either path 1-2-3-4 or 1-5-4 may be used to transmit a message from A to B.

Slide 33/57

Broadcast Method of Message Switching

Nodes

Message

Broadcast Channel
Routing Techniques

- In a WAN, when multiple paths exist between the source and destination nodes of a packet, any one of the paths may be used to transfer the packet.
- Selection of path to be used for transmitting a packet is determined by the routing technique used.
- Two popularly used routing algorithms are:
  - **Source routing**: Source node selects the entire path before sending the packet.
  - **Hop-by-hop routing**: Each node along the path decides only the next node for the path.

Network Topologies

- Term network topology refers to the way in which the nodes of a network are linked together.
- Although numerous network topologies are possible, four major ones are:
  - Star network.
  - Ring network.
  - Completely connected network.
  - Multi-access bus network.

Star Network

- Diagram showing a star network with a central node connected to multiple host nodes.
Ring Network

Completely Connected Network

Multi-Access Bus Network
Network Types

- Networks are broadly classified into two types: Local Area Network (LAN) and Wide Area Network (WAN)
- Local Area Network (LAN) as compared to WAN:
  - Limited to a small geographic coverage
  - Has much higher data transmission rate
  - Experiences fewer data transmission errors
  - Has lower data communication cost
  - Typically owned by a single organization
- Networks that share some of the characteristics of both LANs and WANs are referred to as Metropolitan Area Network (MAN)

Communication Protocols

- Protocol is a set of formal operating rules, procedures, or conventions that govern a given process
- Communication protocol describes rules that govern transmission of data over communication networks
- Roles of communication protocol:
  - Data sequencing
  - Data routing
  - Data formatting
  - Flow control
  - Error control

(Continued on next slide)
Communication Protocols

(Continued from previous slide)

6. Precedence and order of transmission
7. Connection establishment and termination
8. Data security

Communication protocols are normally split up into a series of modules logically composed of a succession of layers.

Terms protocol suite, protocol family, or protocol stack are used to refer to the collection of protocols (of all layers) of a network system.

Network Interface Card (NIC)

6. Hardware device that allows a computer to be connected to a network, both functionally and physically
7. Printed circuit board installed on to one of the expansion slots of computer
8. Provides a port on the back to which network cable is attached

The OSI Model

6. The Open System Interconnection (OSI) model is framework for defining standards for linking heterogeneous computers in a packet switched network
7. Standardized OSI protocol makes it possible for any two heterogeneous computer systems, located anywhere in the world, to easily communicate with each other
8. Separate set of protocols is defined for each layer in its seven-layer architecture. Each layer has an independent function
Internetworking

- Interconnecting two or more networks to form a single network is called internetworking, and the resulting network is called an internetwork.
- Goal of internetworking is to hide details of different physical networks, so that resulting internetwork functions as a single coordinated unit.
- Tools such as bridges, routers, brouters, and gateways are used for internetworking.
- The Internet is the best example of an internetwork.
Bridges

- Operate at bottom two layers of the OSI model
- Connect networks that use the same communication protocols above data-link layer but may use different protocols at physical and data-link layers

Routers

- Operates at network layer of the OSI model
- Used to interconnect those networks that use the same high-level protocols above network layer
- Smarter than bridges as they not only copy data from one network segment to another, but also choose the best route for the data by using routing table

Gateways

- Operates at the top three layers of the OSI model (session, presentation and application)
- Used for interconnecting dissimilar networks that use different communication protocols
- Since gateways interconnect dissimilar networks, protocol conversion is the major job performed by them
Wireless Computing Systems

- Wireless computing system uses wireless communication technologies for interconnecting computer systems.
- Enhances functionality of computing equipment by freeing communication from location constraints of wired computing systems.
- Wireless computing systems are of two types:
  - **Fixed wireless systems**: Support little or no mobility of the computing equipment associated with the wireless network.
  - **Mobile wireless systems**: Support mobility of the computing equipment to access resources associated with the wireless network.

Wireless Technologies

- 2G and 3G
- Wireless LAN
- WiMAX
- Wireless Local Loop (WLL)
- Radio-router
- Multihop Wireless Network
- Wireless Application Protocol (WAP)

Distributed Computing Systems

- Configuration where many independent computer systems are connected, and messages, processing task, programs, data, and other resources are transmitted between cooperating computer systems.
- Such an arrangement enables sharing of many hardware and software resources as well as information among several users who may be sitting far away from each other.
Main Advantages of Distributed Computing Systems

- Inherently distributed applications
- Information sharing among distributed users
- Resource sharing
- Shorter response times and higher throughput
- Higher reliability
- Extensibility and incremental growth
- Better flexibility in meeting users’ needs

Keywords/Phrases

- Amplifier
- Amplitude Modulation (AM)
- Application layer
- AS/MET
- Asynchronous transmission
- Bandwidth
- Baud
- Bridge
- Broadband
- Broadcast
- C-band transmission
- Circuit switching
- Cloud computing
- Common Carriers
- Communication protocol
- Communications satellite
- Completely connected network
- Computer network
- Concentrators
- Data-link layer
- Demodulation
- Dial-up line
- Distributed Computing System
- Ethernet
- Fax modem
- File Transfer Protocol (FTP)
- Front End Processors (FEP)
- Frequency Modulation (FM)
- Frequency-Division Multiplexing (FDM)
- Full duplex
- Gateway
- Half duplex
- High-bye-hop routing
- Hybrid network
- Internet Protocol (IP)
- Internetworking
- ISDN (Integrated Services Digital Network)
- Ku-band transmission
- Leased line
- Local Area Network (LAN)
- Metropolitan Area Network (MAN)
- Microwave system
- Mobile computing
- Modem
- Modulation
- Multi-access Bus network
- Multiplexer
- Narrowband
- Network Interface Card (NIC)
- Network layer
- Network topology
- Nomadic computing
- Optical fiber
- Optical network
- Packet switching
- Phase Modulation (PM)
- Physical layer
- POTS (Plain Old Telephone Service)
- Presentation layer
- Protocol family
- Protocol stack
- Protocol suite
- Repeaters
- Repeater network
- Router
- Session layer
- Simplex
- Source routing
- Star network
- Store-and-forget
- Synchronous transmission
- Time-Division Multiplexing (TDM)
- Transport Control Protocol (TCP)
- Transport layer
- Twisted-pair
- Unshielded twisted-pair (UTP)
- User Datagram Protocol (UDP)
- Value Added Network (VAN)
- Voiceband
- VSAT (Very Small Aperture Terminals)
- Wide Area Network (WAN)
- Wireless network
Learning Objectives

In this chapter you will learn about:

- The Internet
- Evolution and basic services on Internet
- World Wide Web (WWW)
- WWW browsers
- Uses of the Internet
The Internet

- The Internet is a network of computers that links many different types of computers all over the world.
- Network of networks sharing a common mechanism for addressing (identifying) computers, and a common set of communication protocols.
- Evolved from the basic ideas of ARPANET (the first WAN that had only four sites in 1969) for interconnecting computers.
- Initially used only by research organizations and universities to share and exchange information.

(Continued on next slide)
In 1989, the US Government lifted restrictions on the use of the Internet and allowed it to be used for commercial purposes as well.

Internet has rapidly grown and continues to grow at a rapid pace.

Interconnects more than 30,000 networks, allowing more than 10 million computers and more than 50 million computer users in more than 150 countries to communicate with each other.
Basic Services of the Internet

- **Electronic Mail (e-mail):** Allows user to send a mail (message) to another Internet user in any part of the world in a near-real-time manner.

- **File Transfer Protocol (FTP):** Allows user to move a file from one computer to another on the Internet.

- **Telnet:** Allows user to log in to another computer somewhere on the Internet.

- **Usenet News:** Allows group of users to exchange their views/ideas/information.
Electronic Mail

- E-mail is a rapid and productive communication tool because:
  - Faster than paper mail
  - Unlike telephone, the persons communicating with each other need not be available at the same time
  - Unlike fax documents, e-mail documents can be stored in a computer and be easily edited using editing programs
File Transfer Protocol

- Moving a file from a remote computer to one's own computer is known as downloading.
- Moving a file from one's own computer to a remote computer is known as uploading.
- Anonymous FTP site is a computer allowing a user to log in with a username of anonymous and a password that is the user's email address.
- Anonymous FTP sites are called publicly accessible sites because they can be accessed by any user on the Internet.
Telnet

Some common uses of telnet service are:

β Using the computing power of the remote computer
β Using a software on the remote computer
β Accessing remote computer’s database or archive
β Logging in to one’s own computer from another computer
Several usenet news groups exist on the Internet and are called newsgroups.

In a moderated newsgroup only selected members have the right to directly post (write) a message to the virtual notice board. Other members can only read the posted messages.

In a nonmoderated newsgroup any member can directly post a message to the virtual notice board.

Netiquette (network etiquette) deals with rules of framing messages that will not hurt others.
World Wide Web (WWW or W3)

- Hypertext documents on the Internet are known as web pages
- Web pages are created by using a special language called *HyperText Markup Language (HTML)*
- WWW uses the client-server model and an Internet Protocol called *HyperText Transport Protocol (HTTP)* for interaction among the computers on the Internet
- Any computer on the Internet that uses the HTTP protocol is called a web server and any computer that can access that server is called a web client

(Continued on next slide)
World Wide Web (WWW or W3)

(Continued from previous slide..)

- It uses the concept of hypertext for information storage and retrieval on the Internet
- Hypertext documents enable this by using a series of links
- Link is a special type of item in a hypertext document that connects the document to another document providing more information about the linked item
Pradeep K. Sinha has been involved in the research and development of distributed systems for almost a decade. At present Dr. Sinha is working at the Centre for Development of Advanced Computing (C-DAC), Pune, India. Before joining C-DAC, Dr. Sinha worked with the Multimedia Systems Research Laboratory (MSRL) of Panasonic in Tokyo, Japan.
WWW Browsers

WWW browser is a special software loaded on a web client computer that normally provides the following navigation facilities to users:

- Does not require a user to remotely log in to a web server computer or to log out again when done
- Allows user to visit the server computer’s web site and to access information stored on it by specifying its **URL (Uniform Resource Locator)** address

(Continued on next slide)
WWW Browsers

(Continued from previous slide..)

- Allows user to create and maintain a personal hotlist of favorite URL addresses of server computers that user is likely to frequently visit in future

- Allows user to download information in various formats from server computers to user’s own computer
Some important current strategic uses of the Internet are:

- On-line communication
- Software sharing
- Exchange of views on topics of common interest
- Posting of information of general interest
- Product promotion
- Feedback about products
- Customer support service
- On-line journals and magazines
- On-line shopping
- World-wide video conferencing
Keywords/Phrases

- Anonymous ftp site
- Browser
- Download
- Electronic mail (e-mail)
- File Transfer Protocol (FTP)
- Hypertext
- Hypertext Transport Protocol (HTTP)
- Internet
- Newsgroup
- Publicly accessible sites
- Standard Generalized Markup Language (SGML)
- Telnet
- Uniform Resource Locator (URL)
- Upload
- Usenet
- Web client
- Web Server
- World Wide Web (WWW)
Chapter 18
The Internet

In this chapter you will learn about:
- The Internet
- Evolution and basic services on Internet
- World Wide Web (WWW)
- WWW browsers
- Uses of the Internet
The Internet

- The Internet is a network of computers that links many different types of computers all over the world
- Network of networks sharing a common mechanism for addressing (identifying) computers, and a common set of communication protocols
- Evolved from the basic ideas of ARPANET (the first WAN that had only four sites in 1969) for interconnecting computers
- Initially used only by research organizations and universities to share and exchange information

(Continued from previous slide.)

- In 1989, the US Government lifted restrictions on the use of the Internet and allowed it to be used for commercial purposes as well
- Internet has rapidly grown and continues to grow at a rapid pace
- Interconnects more than 30,000 networks, allowing more than 10 million computers and more than 50 million computer users in more than 150 countries to communicate with each other
Basic Services of the Internet

- **Electronic Mail (e-mail):** Allows user to send a mail (message) to another Internet user in any part of the world in a near-real-time manner
- **File Transfer Protocol (FTP):** Allows user to move a file from one computer to another on the Internet
- **Telnet:** Allows user to log in to another computer somewhere on the Internet
- **Usenet News:** Allows group of users to exchange their views/ideas/information

Electronic Mail

- E-mail is a rapid and productive communication tool because:
  - Faster than paper mail
  - Unlike telephone, the persons communicating with each other need not be available at the same time
  - Unlike fax documents, e-mail documents can be stored in a computer and be easily edited using editing programs
File Transfer Protocol

- Moving a file from a remote computer to one's own computer is known as downloading.
- Moving a file from one's own computer to a remote computer is known as uploading.
- Anonymous FTP site is a computer allowing a user to log in with a username of anonymous and a password that is user's e-mail address.
- Anonymous FTP sites are called publicly accessible sites because they can be accessed by any user on the Internet.

Telnet

Some common uses of telnet service are:
- Using the computing power of the remote computer.
- Using a software on the remote computer.
- Accessing remote computer's database or archive.
- Logging in to one's own computer from another computer.
Usenet News

- Several usenet news groups exist on the Internet and are called newsgroups.
- In a *moderated newsgroup* only selected members have the right to directly post (write) a message to the virtual notice board. Other members can only read the posted messages.
- In a *nonmoderated newsgroup* any member can directly post a message to the virtual notice board.
- *Netiquette* (network etiquette) deals with rules of framing messages that will not hurt others.

World Wide Web (WWW or W3)

- Hypertext documents on the Internet are known as web pages.
- Web pages are created by using a special language called *HyperText Markup Language (HTML)*.
- WWW uses the client-server model and an Internet Protocol called *HyperText Transport Protocol (HTTP)* for interaction among the computers on the Internet.
- Any computer on the Internet that uses the HTTP protocol is called a web server and any computer that can access that server is called a web client.
World Wide Web (WWW or W3)

It uses the concept of hypertext for information storage and retrieval on the Internet.

Hypertext documents enable this by using a series of links.

Link is a special type of item in a hypertext document that connects the document to another document providing more information about the linked item.

Example of Hypertext Document

Pradeep K. Sinha has been involved in the research and development of distributed systems for almost a decade. At present Dr. Sinha is working at the Centre for Development of Advanced Computing (C-DAC), Pune, India. Before joining C-DAC, Dr. Sinha worked with the Multimedia Systems Research Laboratory (MSRL) of Panasonic in Tokyo, Japan.
WWW Browsers

WWW browser is a special software loaded on a web client computer that normally provides following navigation facilities to users:

- Does not require a user to remotely log in to a web server computer or to log out again when done
- Allows user to visit the server computer’s web site and to access information stored on it by specifying its **URL (Uniform Resource Locator)** address

(Continued on next slide)

(Continued from previous slide...)

- Allows user to create and maintain a personal **hotlist** of favorite URL addresses of server computers that user is likely to frequently visit in future
- Allows user to download information in various formats from server computers to user’s own computer
Some important current strategic uses of the Internet are:
- On-line communication
- Software sharing
- Exchange of views on topics of common interest
- Posting of information of general interest
- Product promotion
- Feedback about products
- Customer support service
- On-line journals and magazines
- On-line shopping
- World-wide video conferencing

Keywords/Phrases:
- Anonymous ftp site
- Browser
- Download
- Electronic mail (e-mail)
- File Transfer Protocol (FTP)
- Hypertext
- Hypertext Transport Protocol (HTTP)
- Internet
- Newsgroup
- Publicly accessible sites
- Standard Generalized Markup Language (SGML)
- Telnet
- Uniform Resource Locator (URL)
- Upload
- Usenet
- Web client
- Web Server
- World Wide Web (WWW)
Learning Objectives

In this chapter you will learn about:

- The Internet
- Evolution and basic services on Internet
- World Wide Web (WWW)
- WWW browsers
- Uses of the Internet

The Internet

- The Internet is a network of computers that links many different types of computers all over the world
- Network of networks sharing a common mechanism for addressing (identifying) computers, and a common set of communication protocols
- Evolved from the basic ideas of ARPANET (the first WAN that had only four sites in 1969) for interconnecting computers
- Initially used only by research organizations and universities to share and exchange information

(Continued on next slide)
The Internet

- In 1989, the US Government lifted restrictions on the use of the Internet and allowed it to be used for commercial purposes as well.
- Internet has rapidly grown and continues to grow at a rapid pace.
- Interconnects more than 30,000 networks, allowing more than 10 million computers and more than 50 million computer users in more than 150 countries to communicate with each other.

Basic Services of the Internet

- Electronic Mail (e-mail): Allows user to send a mail (message) to another Internet user in any part of the world in a near-real-time manner.
- File Transfer Protocol (FTP): Allows user to move a file from one computer to another on the Internet.
- Telnet: Allows user to log in to another computer somewhere on the Internet.
- Usenet News: Allows group of users to exchange their views/ideas/information.

Electronic Mail

- E-mail is a rapid and productive communication tool because:
  - Faster than paper mail
  - Unlike telephone, the persons communicating with each other need not be available at the same time.
  - Unlike fax documents, e-mail documents can be stored in a computer and be easily edited using editing programs.
File Transfer Protocol

- Moving a file from a remote computer to one's own computer is known as downloading.
- Moving a file from one's own computer to a remote computer is known as uploading.
- Anonymous ftp site is a computer allowing a user to log in with a username of anonymous and password that is user's e-mail address.
- Anonymous ftp sites are called publicly accessible sites because they can be accessed by any user on the Internet.

Telnet

Some common uses of telnet service are:
- Using the computing power of the remote computer.
- Using a software on the remote computer.
- Accessing remote computer's database or archive.
- Logging in to one's own computer from another computer.

Usenet News

- Several usenet news groups exist on the Internet and are called newsgroups.
- In a moderated newsgroup only selected members have the right to directly post (write) a message to the virtual notice board. Other members can only read the posted messages.
- In a nonmoderated newsgroup any member can directly post a message to the virtual notice board.
- Netiquette (network etiquette) deals with rules of framing messages that will not hurt others.
World Wide Web (WWW or W3)

1. Hypertext documents on the Internet are known as web pages.
2. Web pages are created by using a special language called HyperText Markup Language (HTML).
3. WWW uses the client-server model and an Internet Protocol called HyperText Transport Protocol (HTTP) for interaction among the computers on the Internet.
4. Any computer on the Internet that uses the HTTP protocol is called a web server, and any computer that can access that server is called a web client.

(Continued from previous slide...)

World Wide Web (WWW or W3)

5. It uses the concept of hypertext for information storage and retrieval on the Internet.
6. Hypertext documents enable this by using a series of links.
7. Link is a special type of item in a hypertext document that connects the document to another document providing more information about the linked item.

Example of Hypertext Document

Pradeep K. Sinha has been involved in the research and development of distributed systems for almost a decade. At present, Dr. Sinha is working at the Centre for Development of Advanced Computing (C-DAC), Pune, India. Before joining C-DAC, Dr. Sinha worked with the Multimedia Systems Research Laboratory (MSRL) of Panasonic in Tokyo, Japan.
WWW Browsers

WWW browser is a special software loaded on a web client computer that normally provides following navigation facilities to users:

- Does not require a user to remotely log in to a web server computer or to log out again when done
- Allows user to visit the server computer’s web site and to access information stored on it by specifying its URL (Uniform Resource Locator) address

(...Continued on next slide...)

WWW Browsers

- Allows user to create and maintain a personal hotlist of favorite URL addresses of server computers that user is likely to frequently visit in future
- Allows user to download information in various formats from server computers to user’s own computer

(...Continued from previous slide...)

Uses of the Internet

Some important current strategic uses of the Internet are:

- On-line communication
- Software sharing
- Exchange of views on topics of common interest
- Posting of information of general interest
- Product promotion
- Feedback about products
- Customer support service
- On-line journals and magazines
- On-line shopping
- World-wide video conferencing
### Keywords/Phrases

- Anonymous ftp site
- Browser
- Download
- Electronic mail (e-mail)
- File Transfer Protocol (FTP)
- Hyper text
- Hypertext Transport Protocol (HTTP)
- Internet
- News group
- Publicly accessible sites
- Standard Generalized Markup Language (SGML)
- Telnet
- Uniform Resource Locator (URL)
- Upload
- Usenet
- Web client
- Web Server
- World Wide Web (WWW)
Chapter 19
Multimedia

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
Learning Objectives

In this chapter you will learn about:

- Multimedia
- Multimedia computer system
- Main components of multimedia and their associated technologies
- Common multimedia applications
Multimedia

- Media is something that can be used for presentation of information.

- Two basic ways to present some information are:
  - **Unimedia presentation**: Single media is used to present information
  - **Multimedia presentation**: More than one media is used to present information

- Multimedia presentation of any information greatly enhances the comprehension capability of the user as it involves use of more of our senses
Common Media

- Common media for storage, access, and transmission of information are:
  - Text (alphanumerical characters)
  - Graphics (line drawings and images)
  - Animation (moving images)
  - Audio (sound)
  - Video (Videographed real-life events)

- Multimedia in information technology refers to use of more than one of these media for information presentation to users.
Multimedia Computer System

- Multimedia computer system is a computer having capability to integrate two or more types of media (text, graphics, animation, audio, and video)

- In general, size for multimedia information is much larger than plain text information

- Multimedia computer systems require:
  - Faster CPU
  - Larger storage devices (for storing large data files)
  - Larger main memory (for large data size)
  - Good graphics terminals
  - I/O devices to play any multimedia

(Continued on next slide)
Alphanumeric characters are used to present information in text form. Computers are widely used for text processing.

Keyboards, OCRs, computer screens, and printers are some commonly used hardware devices for processing text media.

Text editing, text searching, hypertext, and text importing/exporting are some highly desirable features of a multimedia computer system for better presentation and use of text information.
Graphics Media

- Computer graphics deals with generation, representation, manipulation, and display of pictures (line drawings and images) with a computer

- Locating devices (such as a mouse, a joystick, or a stylus), digitizers, scanners, digital cameras, computer screens with graphics display capability, laser printers, and plotters are some common hardware devices for processing graphics media

- Some desirable features of a multimedia computer system are painting or drawing software, screen capture software, clip art, graphics importing, and software support for high resolution

Ref. Page 369  Chapter 19: Multimedia  Slide 7/16
Animation Media

- **Computer animation** deals with generation, sequencing, and display (at a specified rate) of a set of images (called frames) to create an effect of visual change or motion, similar to a movie film (video).

- Animation is commonly used in those instances where videography is not possible or animation can better illustrate the concept than video.

- Animation deals with displaying a sequence of images at a reasonable speed to create an impression of movement. For a jerk-free full motion animation, 25 to 30 frames per second is required.

(Continued on next slide)
Scanners, digital cameras, video capture board interfaced to a video camera or VCR, computer monitors with image display capability, and graphics accelerator board are some common hardware devices for processing animation media.

Some desirable features of a multimedia computer system with animation facility are animation creation software, screen capture software, animation clips, animation file importing, software support for high resolution, recording and playback capabilities, and transition effects.
Virtual reality is a relatively new technology using which the user can put a pair of goggles and a glove and tour a three-dimensional world that exists only in the computer, but appears realistic to the user.
Audio Media

β *Computer audio* deals with synthesizing, recording, and playback of audio or sound with a computer.

β Sound board, microphone, speaker, MIDI devices, sound synthesizer, sound editor and audio mixer are some commonly used hardware devices for processing audio media.

β Some desirable features of a multimedia computer system are audio clips, audio file importing, software support for high quality sound, recording and playback capabilities, text-to-speech conversion software, speech-to-text conversion software, and voice recognition software.
**Video Media**

- Computer video deals with recording and display of a sequence of images at a reasonable speed to create an impression of movement. Each individual image of such a sequence is called a frame.

- Video camera, video monitor, video board, and video editor are some of the commonly used hardware devices for processing video media.

- Some desirable features of a multimedia computer system with video facility are video clips and recording and playback capabilities.
Multimedia Applications

- Multimedia presentation
- Foreign language learning
- Video games
- Special effects in films
- Multimedia kiosks as help desks
- Animated advertisements
- Multimedia conferencing
Media Center Computer

- There is a growing trend of owning a personal computer (PC) at home like other electronic equipment
- New terminologies like “infotainment” and “edutainment” have evolved to refer to computers as versatile tools
- Media center PC provides following functionalities:
  - Server as PC, TV, radio, and music system
  - Serve as digital photo album and digital library
  - Server as Game station and DVD/CD Player
  - Allows play, pause, and record of TV programs
  - Provides Electronic Programming Guide (EPG)
Keywords/Phrases

- Animation
- Audio
- Clip art
- Cognitive graphics
- Computer Aided Design (CAD)
- Computer Aided Manufacturing (CAM)
- Frames
- Generative graphics
- Graphics
- Multimedia
- Media Center Computer
- Pixel
- Refresh rate
- Text
- Transducer
- Transition effects
- Video
- Virtual reality
Chapter 19
Multimedia

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha

Learning Objectives

In this chapter you will learn about:

- Multimedia
- Multimedia computer system
- Main components of multimedia and their associated technologies
- Common multimedia applications
Multimedia

Media is something that can be used for presentation of information.

Two basic ways to present some information are:

- **Unimedia presentation:** Single media is used to present information
- **Multimedia presentation:** More than one media is used to present information

Multimedia presentation of any information greatly enhances the comprehension capability of the user as it involves use of more of our senses.

Common Media

Common media for storage, access, and transmission of information are:

- Text (alphanumeric characters)
- Graphics (line drawings and images)
- Animation (moving images)
- Audio (sound)
- Video (Videographed real-life events)

Multimedia in information technology refers to use of more than one of these media for information presentation to users.
**Multimedia Computer System**

- Multimedia computer system is a computer having capability to integrate two or more types of media (text, graphics, animation, audio, and video)
- In general, size for multimedia information is much larger than plain text information
- Multimedia computer systems require:
  - Faster CPU
  - Larger storage devices (for storing large data files)
  - Larger main memory (for large data size)
  - Good graphics terminals
  - I/O devices to play any multimedia

(Continued from previous slide.)

**Text Media**

- Alphanumeric characters are used to present information in text form. Computers are widely used for text processing
- Keyboards, OCRs, computer screens, and printers are some commonly used hardware devices for processing text media
- Text editing, text searching, hypertext, and text importing/exporting are some highly desirable features of a multimedia computer system for better presentation and use of text information

(Continued from previous slide.)
Graphics Media

- **Computer graphics** deals with generation, representation, manipulation, and display of pictures (line drawings and images) with a computer.
- Locating devices (such as a mouse, a joystick, or a stylus), digitizers, scanners, digital cameras, computer screens with graphics display capability, laser printers, and plotters are some common hardware devices for processing graphics media.
- Some desirable features of a multimedia computer system are painting or drawing software, screen capture software, clip art, graphics importing, and software support for high resolution.

Animation Media

- **Computer animation** deals with generation, sequencing, and display (at a specified rate) of a set of images (called frames) to create an effect of visual change or motion, similar to a movie film (video).
- Animation is commonly used in those instances where videography is not possible or animation can better illustrate the concept than video.
- Animation deals with displaying a sequence of images at a reasonable speed to create an impression of movement. For a jerk-free full motion animation, 25 to 30 frames per second is required.
Animation Media

(Continued from previous slide...)

- Scanners, digital cameras, video capture board interfaced to a video camera or VCR, computer monitors with image display capability, and graphics accelerator board are some common hardware devices for processing animation media.

- Some desirable features of a multimedia computer system with animation facility are animation creation software, screen capture software, animation clips, animation file importing, software support for high resolution, recording and playback capabilities, and transition effects.

Virtual Reality

- Virtual reality is a relatively new technology using which the user can put a pair of goggles and a glove and tour a three-dimensional world that exists only in the computer, but appears realistic to the user.
Audio Media

- Computer audio deals with synthesizing, recording, and playback of audio or sound with a computer.
- Sound board, microphone, speaker, MIDI devices, sound synthesizer, sound editor and audio mixer are some commonly used hardware devices for processing audio media.
- Some desirable features of a multimedia computer system are audio clips, audio file importing, software support for high quality sound, recording and playback capabilities, text-to-speech conversion software, speech-to-text conversion software, and voice recognition software.

Video Media

- Computer video deals with recording and display of a sequence of images at a reasonable speed to create an impression of movement. Each individual image of such a sequence is called a frame.
- Video camera, video monitor, video board, and video editor are some of the commonly used hardware devices for processing video media.
- Some desirable features of a multimedia computer system with video facility are video clips and recording and playback capabilities.
Multimedia Applications

- Multimedia presentation
- Foreign language learning
- Video games
- Special effects in films
- Multimedia kiosks as help desks
- Animated advertisements
- Multimedia conferencing

Media Center Computer

- There is a growing trend of owning a personal computer (PC) at home like other electronic equipment
- New terminologies like “infotainment” and “edutainment” have evolved to refer to computers as versatile tools
- Media center PC provides following functionalities:
  - Server as PC, TV, radio, and music system
  - Serve as digital photo album and digital library
  - Server as Game station and DVD/CD Player
  - Allows play, pause, and record of TV programs
  - Provides Electronic Programming Guide (EPG)
Media Center Computer

- System Unit
- High-resolution display screen
- Remote Control
- Mouse
- Keyboard

Keywords/Phrases

- Animation
- Audio
- Clip art
- Cognitive graphics
- Computer Aided Design (CAD)
- Computer Aided Manufacturing (CAM)
- Frames
- Generative graphics
- Graphics
- Multimedia
- Media Center Computer
- Pixel
- Refresh rate
- Text
- Transducer
- Transition effects
- Video
- Virtual reality
Chapter 19
Multimedia

Learning Objectives

In this chapter you will learn about:

- Multimedia
- Multimedia computer system
- Main components of multimedia and their associated technologies
- Common multimedia applications

Multimedia

1. Media is something that can be used for presentation of information.

2. Two basic ways to present some information are:
   - Unimedia presentation: Single media is used to present information
   - Multimedia presentation: More than one media is used to present information

3. Multimedia presentation of any information greatly enhances the comprehension capability of the user as it involves use of more of our senses
### Common Media

- Text (alphanumeric characters)
- Graphics (line drawings and images)
- Animation (moving images)
- Audio (sound)
- Video (videographed real-life events)

Multimedia in information technology refers to use of more than one of these media for information presentation to users.

### Multimedia Computer System

- Multimedia computer system is a computer having capability to integrate two or more types of media (text, graphics, animation, audio, and video).
- In general, size for multimedia information is much larger than plain text information.

Multimedia computer systems require:
- Faster CPU
- Larger storage devices (for storing large data files)
- Larger main memory (for large data size)
- Good graphics terminals
- I/O devices to play any multimedia

### Text Media

- Alphanumeric characters are used to present information in text form. Computers are widely used for text processing.
- Keyboards, OCRs, computer screens, and printers are some commonly used hardware devices for processing text media.
- Text editing, text searching, hypertext, and text importing/exporting are some highly desirable features of a multimedia computer system for better presentation and use of text information.
Graphics Media

- Computer graphics deals with generation, representation, manipulation, and display of pictures (line drawings and images) with a computer.
- Locating devices (such as a mouse, a joystick, or a stylus), digitizers, scanners, digital cameras, computer screens with graphics display capability, laser printers, and plotters are some common hardware devices for processing graphics media.
- Some desirable features of a multimedia computer system are painting or drawing software, screen capture software, clip art, graphics importing, and software support for high resolution.

Animation Media

- Computer animation deals with generation, sequencing, and display (at a specified rate) of a set of images (called frames) to create an effect of visual change or motion, similar to a movie film (video).
- Animation is commonly used in those instances where videography is not possible or animation can better illustrate the concept than video.
- Animation deals with displaying a sequence of images at a reasonable speed to create an impression of movement. For a jerk-free full motion animation, 25 to 30 frames per second is required.

(Continued from previous slide..)

- Scanners, digital cameras, video capture board interfaced to a video camera or VCR, computer monitors with image display capability, and graphics accelerator board are some common hardware devices for processing animation media.
- Some desirable features of a multimedia computer system with animation facility are animation creation software, screen capture software, animation clips, animation file importing, software support for high resolution, recording and playback capabilities, and transition effects.

(Continued on next slide)
**Virtual Reality**

Virtual reality is a relatively new technology using which the user can put a pair of goggles and a glove and tour a three-dimensional world that exists only in the computer, but appears realistic to the user.

**Audio Media**

- Computer audio deals with synthesizing, recording, and playback of audio or sound with a computer.
- Sound board, microphone, speaker, MIDI devices, sound synthesizer, sound editor and audio mixer are some commonly used hardware devices for processing audio media.
- Some desirable features of a multimedia computer system are audio clips, audio file importing, software support for high quality sound, recording and playback capabilities, text-to-speech conversion software, speech-to-text conversion software, and voice recognition software.

**Video Media**

- Computer video deals with recording and display of a sequence of images at a reasonable speed to create an impression of movement. Each individual image of such a sequence is called a frame.
- Video camera, video monitor, video board, and video editor are some of the commonly used hardware devices for processing video media.
- Some desirable features of a multimedia computer system with video facility are video clips and recording and playback capabilities.
Multimedia Applications

- Multimedia presentation
- Foreign language learning
- Video games
- Special effects in films
- Multimedia kiosks as help desks
- Animated advertisements
- Multimedia conferencing

Media Center Computer

- There is a growing trend of owning a personal computer (PC) at home like other electronic equipment
- New terminologies like "infotainment" and "edutainment" have evolved to refer to computers as versatile tools
- Media center PC provides following functionalities:
  - Serve as PC, TV, radio, and music system
  - Serve as digital photo album and digital library
  - Serve as Game station and DVD/CD Player
  - Allows play, pause, and record of TV programs
  - Provides Electronic Programming Guide (EPG)
Keywords/Phrases

- Animation
- Audio
- Clip art
- Cognitive graphics
- Computer Aided Design (CAD)
- Computer Aided Manufacturing (CAM)
- Frames
- Generative graphics
- Graphics
- Multimedia
- Media Center Computer
- Pixel
- Refresh rate
- Text
- Transducer
- Transition effects
- Video
- Virtual reality
Chapter 20

Classification of Computers

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
Learning Objectives

In this chapter you will learn about:

- Classifications of computers
- Common types of computers today
- Characteristic features of various types of computers in use today
Computer Classification

- Traditionally, computers were classified by their size, processing speed, and cost.
- Based on these factors, computers were classified as microcomputers, minicomputers, mainframes, and supercomputers.
- However, with rapidly changing technology, this classification is no more relevant.
- Today, computers are classified based on their mode of use.
Based on their mode of use, computers are classified as:

- Notebook computers
- Personal computers
- Workstations
- Mainframe systems
- Supercomputers
- Clients and servers
- Handheld computers
Notebook Computers

- Portable computers mainly meant for use by people who need computing resource wherever they go.
- Approximately of the size of an 8½ x 11 inch notebook and can easily fit inside a briefcase.
- Weigh around 2 kg only.
- Comfortably placed on ones lap while being used. Hence, they are also called laptop PC.
- Lid with display screen is foldable in a manner that when not in use it can be folded to flush with keyboard to convert the system into notebook form.

(Continued on next slide)
Notebook Computers

(Continued from previous slide..)

- Designed to operate with chargeable batteries
- Mostly used for word processing, spreadsheet computing, data entry, and power point presentations
- Normally run MS-DOS or MS WINDOWS operating system
- Some manufacturers are also offering models with GNU/Linux or its distributions
- Each device of laptop is designed to use little power and remain suspended if not used
Notebook Computers

Foldable flat screen

Keyboard, trackball, hard disk, floppy disk drive, etc. are in this unit
Personal Computers (PCs)

- Non-portable, general-purpose computer that fits on a normal size office table
- Designed to meet personal computing needs of individuals
- Often used by children and adults for education and entertainment also
- Generally used by one person at a time, supports multitasking
- Two common models of PCs are desktop model and tower model
- Popular OS are MS-DOS, MS-Windows, Windows-NT, Linux, and UNIX
Common PC Models

(a) Desktop model

(b) Tower model

- Monitor
- System Unit
- Keyboard
- Mouse
Workstations

- Powerful desktop computer designed to meet the computing needs of engineers, architects, and other professionals
- Provides greater processing power, larger storage, and better graphics display facility than PCs
- Commonly used for computer-aided design, multimedia applications, simulation of complex scientific and engineering problems, and visualization
- Generally run the UNIX operating system or a variation of it
- Operating system is generally designed to support multiuser environment
Mainframe Systems

- Mainly used by large organizations as banks, insurance companies, hospitals, railways, etc.
- Used for data handling and information processing requirements
- Used in such environments where a large number of users need to share a common computing facility
- Oriented to input/output-bound applications

(Continued on next slide)
Mainframe Systems

(Continued from previous slide..)

- Typically consist of a host computer, front-end computer, back-end computer, console terminals, magnetic disk drives, tape drives, magnetic tape library, user terminals, printers, and plotters.
- Typical mainframe system looks like a row of large file cabinets and needs a large room.
- Smaller configuration (slower host and subordinate computers, lesser storage space, and fewer user terminals) is often referred to as a minicomputer system.
Mainframe Computer Systems

SYstem ROOM (Entry restricted to system administrators and maintenance staff)

Magnetic Disk Drives

Magnetic Tape Library

Back-end Processor

Host Processor

Front-end Processor

Console

Printer

Plotter

Magnetic Tape Drive

User Terminal

User Terminal

User Terminal

USERS ROOM (Entry restricted to authorized users)
Supercomputers

- Most powerful and most expensive computers available at a given time.
- Primarily used for processing complex scientific applications that require enormous processing power.
- Well known supercomputing applications include:
  - Analysis of large volumes of seismic data
  - Simulation of airflow around an aircraft
  - Crash simulation of the design of an automobile
  - Solving complex structure engineering problems
  - Weather forecasting

(Continued on next slide)
Supercomputers also support multiprogramming

Supercomputers primarily address processor-bound applications
Use multiprocessing and parallel processing technologies to solve complex problems faster

Also known as *parallel computers* or *parallel processing systems*

Modern supercomputers employ hundreds of processors and are also known as *massively parallel processors*
C-DAC’s PARAM 10000 Supercomputer
Client and Server Computers

- Client-server computing environment has multiple clients, one/more servers, and a network
- **Client** is a PC/workstation with user-friendly interface running client processes that send service requests to the server
- **Server** is generally a relatively large computer that manages a shared resource and provides a set of shared user services to the clients
- Server runs the server process that services client requests for use of managed resources
- **Network** may be a single LAN or WAN or an internet work
Client-Server Computing

- Involves splitting an application into tasks and putting each task on a computer where it can be handled most efficiently.
- Computers and operating systems of a client and a server may be different.
- Common for one server to use the services of another server, and hence act both as client and server.
- Concept of client and server computers is purely role-based and may change dynamically as the role of a computer changes.
Client-Server Computing Environment

- File Server
- Database Server
- LAN or WAN or an Internet of Networks
- PC (Client)
- Workstation (Client)
- Workstation (Client)
- PC (Client)
Handheld Computers

- Small computing device that can be used by holding in hand, also known as *palmtop*

- Size, weight, and design are such that it can be used comfortably by holding in hand

- Types of Handheld are:
  - **Tablet PC**: Miniaturized laptop with light weight, screen flip, handwriting and voice recognition
  - **PDA/Pocket PC**: Acts as PIM device with LCD touch screen, pen for handwriting recognition, PC based synchronization, and optionally mobile phone services
  - **Smartphone**: Fully functional mobile phone with computing power, voice centric, do not have a touch screen and are smaller than PDA

Ref. Page 389
Handheld Computers

(a) Table PC  
(b) PDA/Pocket PC  
(c) Smartphone

Ref. Page 391  Chapter 20: Classification of Computers  Slide 22/26
## Comparison of Different Types of Computers

<table>
<thead>
<tr>
<th>Types of Computers</th>
<th>Note book</th>
<th>PC</th>
<th>Work station</th>
<th>Mainframe system</th>
<th>Super computer</th>
<th>Client</th>
<th>Server</th>
<th>Handheld</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key features</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>Very small (can be placed on ones lap)</td>
<td>Small (slightly larger than PC)</td>
<td>Medium (needs a large room)</td>
<td>Large (needs a large room)</td>
<td>Generally small (may be large if it is also play the role of a server)</td>
<td>Generally large</td>
<td>Generally large</td>
<td>Very small (can be placed on ones palm)</td>
</tr>
<tr>
<td>Processing power</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Higher</td>
<td>Highest</td>
<td>Generally low</td>
<td>Generally high</td>
<td>Low</td>
</tr>
<tr>
<td>Main memory capacity</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Higher</td>
<td>Highest</td>
<td>Generally low</td>
<td>Generally high</td>
<td>Low</td>
</tr>
<tr>
<td>Hard disk storage capacity</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Highest</td>
<td>Higher</td>
<td>Generally low</td>
<td>Generally high</td>
<td>Low</td>
</tr>
<tr>
<td>Has its own monitor, keyboard, and mouse for user interface</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Generally no</td>
<td>Generally no</td>
<td>Yes</td>
<td>Generally no</td>
<td>No</td>
</tr>
</tbody>
</table>

(Continued on next slide)
### Comparison of Different Types of Computers

(Continued from previous slide..)

<table>
<thead>
<tr>
<th>Types of Computers</th>
<th>Notebook</th>
<th>PC</th>
<th>Work station</th>
<th>Mainframe system</th>
<th>Super computer</th>
<th>Client</th>
<th>Server</th>
<th>Handheld</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key features</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display facility</td>
<td>Foldable flat screen small display</td>
<td>Medium size display screen</td>
<td>Large-screen color monitor which can display high resolution graphics</td>
<td>Generally not available</td>
<td>Generally not available</td>
<td>Medium to large screen monitor</td>
<td>Generally not available</td>
<td>Small display</td>
</tr>
<tr>
<td>Single/multiple processors</td>
<td>Single</td>
<td>Generally single</td>
<td>Generally multiple</td>
<td>Multiple</td>
<td>Multiple</td>
<td>Generally single</td>
<td>Generally multiple</td>
<td>Single</td>
</tr>
<tr>
<td>Single/multiple – User oriented</td>
<td>Single</td>
<td>Single</td>
<td>Generally single</td>
<td>Multiple</td>
<td>Multiple</td>
<td>Single</td>
<td>Multiple</td>
<td>Single</td>
</tr>
<tr>
<td>Popular operating systems</td>
<td>MS-DOS, MS-Windows</td>
<td>MS-DOS, MS-Windows, Windows-NT, Linux, Unix</td>
<td>Unix or a variation of it</td>
<td>A variation of Unix, or proprietary</td>
<td>A variation of Unix, or proprietary</td>
<td>MS-DOS, MS-Windows, Windows-NT, Linux, Unix</td>
<td>Windows -NT, Unix or its variation, or proprietary</td>
<td>MS-Windows Mobile, Palm OS, Symbian OS, Linux, Blackbery OS</td>
</tr>
</tbody>
</table>

(Continued on next slide)
## Comparison of Different Types of Computers

(Continued from previous slide..)

<table>
<thead>
<tr>
<th>Types of Computers</th>
<th>Notebook Key features</th>
<th>Personal Computer Key features</th>
<th>Workstation Key features</th>
<th>Mainframe System Key features</th>
<th>Super Computer Key features</th>
<th>Client Key features</th>
<th>Server Key features</th>
<th>Handheld Key features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popular usage</td>
<td>Word processing; Spreadsheet; Data Entry; Preparing presentation materials; and Making presentations</td>
<td>Personal computing needs of individuals either in their working places or at their homes; and Education and entertainment of children and adults</td>
<td>Computing needs of engineers, architects, designers; Simulation of complex scientific and engineering problems and visualizing the results of simulation; and Multimedia applications</td>
<td>Data and information processing of I/O-bound applications</td>
<td>Large processor-bound application like complex scientific simulation</td>
<td>Provide highly user-friendly interface in a client-server computing environment</td>
<td>Manage a shared resource and provide a set of shared user services in a client-server computing environment</td>
<td>Computing, Personal Information Management (PIM), cell phone, digital camera</td>
</tr>
<tr>
<td>Major vendors</td>
<td>IBM, Compaq, Siemens, Toshiba</td>
<td>IBM, Apple, Compaq, Dell, Zenith, Siemens, Toshiba, Hewlett-Packard</td>
<td>Sun Microsystems, IBM, DEC, Hewlett-Packard, Silicon Graphics</td>
<td>IBM, DEC</td>
<td>Cray, IBM, Silicon Graphics, Fujitsu, Intel, C-DAC</td>
<td>Same as PC and Workstation vendors</td>
<td>Same as Workstation, Mainframe System, &amp; Super-computer vendors</td>
<td>Nokia, Sony, Motorola, Dell, Hewlett-Packard</td>
</tr>
</tbody>
</table>

Ref. Page 392  Chapter 20: Classification of Computers  Slide 25/26
Key Words/Phrases

- Back-end computer
- Client computer
- Client process
- Front-end computer
- Host computer
- Handheld
- I/O-bound application
- Laptop PC
- Mainframe system
- Massively parallel processors
- Minicomputer
- Notebook computer
- Parallel computers
- Parallel processing system
- Personal Computer (PC)
- Processor-bound application
- Server computer
- Server process
- Supercomputer
- System board
- Workstation
Learning Objectives

In this chapter you will learn about:

- Classifications of computers
- Common types of computers today
- Characteristic features of various types of computers in use today
Computer Classification

- Traditionally, computers were classified by their size, processing speed, and cost.
- Based on these factors, computers were classified as microcomputers, minicomputers, mainframes, and supercomputers.
- However, with rapidly changing technology, this classification is no more relevant.
- Today, computers are classified based on their mode of use.

Types of Computers

Based on their mode of use, computers are classified as:
- Notebook computers
- Personal computers
- Workstations
- Mainframe systems
- Supercomputers
- Clients and servers
- Handheld computers
Notebook Computers

- Portable computers mainly meant for use by people who need computing resource wherever they go
- Approximately of the size of an 8½ x 11 inch notebook and can easily fit inside a briefcase
- Weigh around 2 kg only.
- Comfortably placed on one’s lap while being used. Hence, they are also called laptop PC
- Lid with display screen is foldable in a manner that when not in use it can be folded to flush with keyboard to convert the system into notebook form

(Continued from previous slide...)

- Designed to operate with chargeable batteries
- Mostly used for word processing, spreadsheet computing, data entry, and power point presentations
- Normally run MS-DOS or MS WINDOWS operating system
- Some manufacturers are also offering models with GNU/Linux or its distributions
- Each device of laptop is designed to use little power and remain suspended if not used
Notebook Computers

- Foldable flat screen
- Keyboard, trackball, hard disk, floppy disk drive, etc. are in this unit

Personal Computers (PCs)

- Non-portable, general-purpose computer that fits on a normal size office table
- Designed to meet personal computing needs of individuals
- Often used by children and adults for education and entertainment also
- Generally used by one person at a time, supports multitasking
- Two common models of PCs are desktop model and tower model
- Popular OS are MS-DOS, MS-Windows, Windows-NT, Linux, and UNIX
Common PC Models

(a) Desktop model

(b) Tower model

Workstations

- Powerful desktop computer designed to meet the computing needs of engineers, architects, and other professionals
- Provides greater processing power, larger storage, and better graphics display facility than PCs
- Commonly used for computer-aided design, multimedia applications, simulation of complex scientific and engineering problems, and visualization
- Generally run the UNIX operating system or a variation of it
- Operating system is generally designed to support multiuser environment
Mainframe Systems

- Mainly used by large organizations as banks, insurance companies, hospitals, railways, etc.
- Used for data handling and information processing requirements
- Used in such environments where a large number of users need to share a common computing facility
- Oriented to input/output-bound applications

(Continued on next slide)

Mainframe Systems

- Typically consist of a host computer, front-end computer, back-end computer, console terminals, magnetic disk drives, tape drives, magnetic tape library, user terminals, printers, and plotters
- Typical mainframe system looks like a row of large file cabinets and needs a large room
- Smaller configuration (slower host and subordinate computers, lesser storage space, and fewer user terminals) is often referred to as a minicomputer system

(Continued from previous slide.)
Mainframe Computer Systems

Supercomputers

- Most powerful and most expensive computers available at a given time.
- Primarily used for processing complex scientific applications that require enormous processing power.
- Well known supercomputing applications include:
  - Analysis of large volumes of seismic data
  - Simulation of airflow around an aircraft
  - Crash simulation of the design of an automobile
  - Solving complex structure engineering problems
  - Weather forecasting

(Continued on next slide)
Supercomputers

(Continued from previous slide...)

- Supercomputers also support multiprogramming
- Supercomputers primarily address processor-bound applications

Parallel Processing Systems

- Use multiprocessing and parallel processing technologies to solve complex problems faster
- Also known as parallel computers or parallel processing systems
- Modern supercomputers employ hundreds of processors and are also known as massively parallel processors
C-DAC’s PARAM 10000 Supercomputer

Client and Server Computers

- Client-server computing environment has multiple clients, one/more servers, and a network
- **Client** is a PC/workstation with user-friendly interface running client processes that send service requests to the server
- **Server** is generally a relatively large computer that manages a shared resource and provides a set of shared user services to the clients
- Server runs the server process that services client requests for use of managed resources
- **Network** may be a single LAN or WAN or an internet work
Client-Server Computing

- Involves splitting an application into tasks and putting each task on computer where it can be handled most efficiently.
- Computers and operating systems of a client and a server may be different.
- Common for one server to use the services of another server, and hence act both as client and server.
- Concept of client and server computers is purely role-based and may change dynamically as the role of a computer changes.

Client-Server Computing Environment

- File Server
- Database Server
- LAN or WAN or an Internet of Networks
- Workstation (Client)
- PC (Client)
Handheld Computers

- Small computing device that can be used by holding in hand, also known as **palmtop**
- Size, weight, and design are such that it can be used comfortably by holding in hand
- Types of Handheld are:
  - **Tablet PC**: Miniaturized laptop with light weight, screen flip, handwriting and voice recognition
  - **PDA/Pocket PC**: Acts as PIM device with LCD touch screen, pen for handwriting recognition, PC based synchronization, and optionally mobile phone services
  - **Smartphone**: Fully functional mobile phone with computing power, voice centric, do not have a touch screen and are smaller than PDA

(a) Table PC  (b) PDA/Pocket PC  (c) Smartphone
## Comparison of Different Types of Computers

<table>
<thead>
<tr>
<th>Types of Computers</th>
<th>Notebook</th>
<th>PC</th>
<th>Workstation</th>
<th>Mainframe System</th>
<th>Super Computer</th>
<th>Client</th>
<th>Server</th>
<th>Handheld</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key features</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>Very small (can be placed on one's lap)</td>
<td>Small (can be placed on an office table)</td>
<td>Medium (slightly larger than PC)</td>
<td>Large (needs a large room)</td>
<td>Large (needs a large room)</td>
<td>Generally small (may be large if it is also the role of a server)</td>
<td>Generally large</td>
<td>Very small (can be placed on one's palm)</td>
</tr>
<tr>
<td>Processing power</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Higher</td>
<td>Highest</td>
<td>Generally low</td>
<td>Generally high</td>
<td>Low</td>
</tr>
<tr>
<td>Main memory capacity</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Higher</td>
<td>Highest</td>
<td>Generally low</td>
<td>Generally high</td>
<td>Low</td>
</tr>
<tr>
<td>Hard disk storage capacity</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Highest</td>
<td>Higher</td>
<td>Generally low</td>
<td>Generally high</td>
<td>Low</td>
</tr>
<tr>
<td>Has its own monitor, keyboard, and mouse for user interface</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Generally no</td>
<td>Generally no</td>
<td>Yes</td>
<td>Generally no</td>
<td>No</td>
</tr>
</tbody>
</table>

(Continued on next slide)

(Continued from previous slide...)

<table>
<thead>
<tr>
<th>Types of Computers</th>
<th>Notebook</th>
<th>PC</th>
<th>Workstation</th>
<th>Mainframe System</th>
<th>Super Computer</th>
<th>Client</th>
<th>Server</th>
<th>Handheld</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key features</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display facility</td>
<td>Portable flat screen small display</td>
<td>Medium square display screen</td>
<td>Large-screen color monitor which can display high resolution graphics</td>
<td>Generally not available</td>
<td>Generally not available</td>
<td>Medium to large screen monitor</td>
<td>Generally not available</td>
<td>Small display</td>
</tr>
<tr>
<td>Single/multiple processors</td>
<td>Single</td>
<td>Generally single</td>
<td>Generally multiple</td>
<td>Multiple</td>
<td>Multiple</td>
<td>Generally single</td>
<td>Generally multiple</td>
<td>Single</td>
</tr>
<tr>
<td>Single/multiple – User oriented</td>
<td>Single</td>
<td>Single</td>
<td>Generally single</td>
<td>Multiple</td>
<td>Multiple</td>
<td>Single</td>
<td>Multiple</td>
<td>Single</td>
</tr>
<tr>
<td>Popular operating systems</td>
<td>MS-DOS, MS-Windows</td>
<td>MS-DOS, MS-Windows, Windows-NT, Linux, Unix</td>
<td>Unix or a variation of it</td>
<td>A variation of Unix, or proprietary</td>
<td>A variation of Unix, or proprietary</td>
<td>MS-DOS, MS-Windows, Windows-NT, Linux, Unix</td>
<td>Windows-NT, Unix or a variation, or proprietary</td>
<td>MS-Windows Mobile, Palm OS, Symbian OS, Linux, BlackBerry OS</td>
</tr>
</tbody>
</table>

(Continued on next slide)
## Comparison of Different Types of Computers

(Continued from previous slide...)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Popular usage</td>
<td>Word processing, Spreadsheet, Data Entry, Preparing presentation materials, and Making presentations</td>
<td>Personal computing needs of individuals either in their working places or at their homes; and Education and entertainment of children and adults</td>
<td>Computing needs of engineers, architects, designers, simulation of complex scientific and engineering problems and visualizing the results of simulation; and Multimedia applications</td>
<td>Data and information processing of I/O-bound applications</td>
<td>Large processor-bound applications like complex scientific simulations</td>
<td>Provide highly user-friendly interface in a client-server computing environment</td>
<td>Manage a shared resource and provide a set of shared user services in a client-server computing environment</td>
<td>Computing, Personal Information Management (PIM), cell phone, digital camera</td>
</tr>
<tr>
<td>Major vendors</td>
<td>IBM, Compaq, Siemens, Toshiba</td>
<td>IBM, Apple, Compaq, Dell, Zenith, Siemens, Toshiba, Hewlett-Packard</td>
<td>Sun Microsystems, IBM, DEC, Hewlett-Packard, Silicon Graphics</td>
<td>IBM, DEC</td>
<td>Cray, IBM, Silicon Graphics, Fujitsu, Intel, C-DAC</td>
<td>Same as PC and Workstation vendors</td>
<td>Same as Workstation, Mainframe System, &amp; Supercomputer vendors</td>
<td>None, Sony, Motorola, Dell, Hewlett-Packard</td>
</tr>
</tbody>
</table>

### Key Words/Phrases

- Back-end computer
- Client computer
- Client process
- Front-end computer
- Host computer
- Handheld
- I/O-bound application
- Laptop PC
- Mainframe system
- Massively parallel processors
- Minicomputer
- Notebook computer
- Parallel computers
- Parallel processing system
- Personal Computer (PC)
- Processor-bound application
- Server computer
- Server process
- Supercomputer
- System board
- Workstation
Chapter 20
Classification of Computers

Learning Objectives

In this chapter you will learn about:

- Classifications of computers
- Common types of computers today
- Characteristic features of various types of computers in use today

Computer Classification

- Traditionally, computers were classified by their size, processing speed, and cost
- Based on these factors, computers were classified as microcomputers, minicomputers, mainframes, and supercomputers
- However, with rapidly changing technology, this classification is no more relevant
- Today, computers are classified based on their mode of use
Types of Computers

Based on their mode of use, computers are classified as:

- Notebook computers
- Personal computers
- Workstations
- Mainframe systems
- Supercomputers
- Clients and servers
- Handheld computers

Notebook Computers

- Portable computers mainly meant for use by people who need computing resource wherever they go
- Approximately of the size of an 8½ x 11 inch notebook and can easily fit inside a briefcase
- Weigh around 2 kg only.
- Comfortably placed on ones lap while being used. Hence, they are also called laptop PC
- Lid with display screen is foldable in a manner that when not in use it can be folded to flush with keyboard to convert the system into notebook form

(Continued from previous slide)

- Designed to operate with chargeable batteries
- Mostly used for word processing, spreadsheet computing, data entry, and power point presentations
- Normally run MS-DOS or MS WINDOWS operating system
- Some manufacturers are also offering models with GNU/Linux or its distributions
- Each device of laptop is designed to use little power and remain suspended if not used
Notebook Computers

- Foldable flat screen
- Keyboard, trackball, hard disk, floppy disk drive, etc. are in the unit

Personal Computers (PCs)

- Non-portable, general-purpose computer that fits on a normal size office table
- Designed to meet personal computing needs of individuals
- Often used by children and adults for education and entertainment also
- Generally used by one person at a time, supports multitasking
- Two common models of PCs are desktop model and tower model
- Popular OS are MS-DOS, MS-Windows, Windows-NT, Linux, and UNIX

Common PC Models

- (a) Desktop model
- (b) Tower model
Workstations

- Powerful desktop computer designed to meet the computing needs of engineers, architects, and other professionals
- Provides greater processing power, larger storage, and better graphics display facility than PCs
- Commonly used for computer-aided design, multimedia applications, simulation of complex scientific and engineering problems, and visualization
- Generally run the UNIX operating system or a variation of it
- Operating system is generally designed to support multiuser environment

Mainframe Systems

- Mainly used by large organizations as banks, insurance companies, hospitals, railways, etc.
- Used for data handling and information processing requirements
- Used in such environments where a large number of users need to share a common computing facility
- Oriented to input/output-bound applications
Supercomputers

- Most powerful and most expensive computers available at a given time.
- Primarily used for processing complex scientific applications that require enormous processing power
- Well known supercomputing applications include:
  - Analysis of large volumes of seismic data
  - Simulation of airflow around an aircraft
  - Crash simulation of the design of an automobile
  - Solving complex structure engineering problems
  - Weather forecasting

Supercomputers also support multiprogramming
Supercomputers primarily address processor-bound applications
Parallel Processing Systems

- Use multiprocessing and parallel processing technologies to solve complex problems faster
- Also known as parallel computers or parallel processing systems
- Modern supercomputers employ hundreds of processors and are also known as massively parallel processors

C-DAC’s PARAM 10000 Supercomputer

Client and Server Computers

- Client-server computing environment has multiple clients, one/more servers, and a network
- Client is a PC/workstation with user-friendly interface running client processes that send service requests to the server
- Server is generally a relatively large computer that manages a shared resource and provides a set of shared user services to the clients
- Server runs the server process that services client requests for use of managed resources
- Network may be a single LAN or WAN or an internet work
Client-Server Computing

- Involves splitting an application into tasks and putting each task on a computer where it can be handled most efficiently.
- Computers and operating systems of a client and a server may be different.
- Common for one server to use the services of another server, and hence act both as client and server.
- Concept of client and server computers is purely role-based and may change dynamically as the role of a computer changes.

Client-Server Computing Environment

Handheld Computers

- Small computing device that can be used by holding in hand, also known as palmtop.
- Size, weight, and design are such that it can be used comfortably by holding in hand.
- Types of Handheld are:
  1. Tablet PC: Miniaturized laptop with light weight, screen flip, handwriting and voice recognition.
  2. PDA/Pocket PC: Acts as PIM device with LCD touch screen, pen for handwriting recognition, PC based synchronization, and optionally mobile phone services.
  3. Smartphone: Fully functional mobile phone with computing power, voice centric, do not have a touch screen and are smaller than PDA.
Handheld Computers

(a) Tablet PC
(b) PDA/Pocket PC
(c) Smartphone

---

Comparison of Different Types of Computers

<table>
<thead>
<tr>
<th>Types of Computers</th>
<th>Key Features</th>
<th>Notebook</th>
<th>PC</th>
<th>Workstations</th>
<th>Supercomputer</th>
<th>Client</th>
<th>Server</th>
<th>Handheld</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Very small</td>
<td>Small</td>
<td>Medium to large</td>
<td>Large-screen color monitor which can display high resolution graphics</td>
<td>Small</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>Generally multiple processors</td>
<td>Generally single</td>
<td>Generally single</td>
<td>Generally single</td>
<td>Generally single</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows, DOS, Linux, Unix</td>
<td>Windows-NT, Unix or its variation, or proprietary</td>
<td>Windows-NT, Unix or its variation, or proprietary</td>
<td>Windows-NT, Unix or its variation, or proprietary</td>
<td>Windows-NT, Unix or its variation, or proprietary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td>Foldable flat screen</td>
<td>Medium size display screen</td>
<td>Large-screen color monitor which can display high resolution graphics</td>
<td>Large-screen color monitor which can display high resolution graphics</td>
<td>Foldable flat screen</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Continued on next slide)
Comparison of Different Types of Computers

<table>
<thead>
<tr>
<th>Type of Computer</th>
<th>Key Words/Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation</td>
<td>Back-end computer</td>
</tr>
<tr>
<td></td>
<td>Client computer</td>
</tr>
<tr>
<td></td>
<td>Client process</td>
</tr>
<tr>
<td>Personal Computer</td>
<td>Host computer</td>
</tr>
<tr>
<td></td>
<td>Laptop PC</td>
</tr>
<tr>
<td>Mainframe</td>
<td>Massively parallel processors</td>
</tr>
<tr>
<td>Parallel Computer</td>
<td>Minicomputer</td>
</tr>
<tr>
<td></td>
<td>Notebook computer</td>
</tr>
<tr>
<td></td>
<td>Parallel processing system</td>
</tr>
<tr>
<td></td>
<td>Personal Computer (PC)</td>
</tr>
<tr>
<td></td>
<td>Processor-bound application</td>
</tr>
<tr>
<td>Supercomputer</td>
<td>Server computer</td>
</tr>
<tr>
<td></td>
<td>Server process</td>
</tr>
<tr>
<td></td>
<td>Supercomputer</td>
</tr>
<tr>
<td></td>
<td>System board</td>
</tr>
<tr>
<td></td>
<td>Workstation</td>
</tr>
</tbody>
</table>

Key Words/Phrases:
- Back-end computer
- Client computer
- Client process
- Host computer
- Laptop PC
- Massively parallel processors
- Minicomputer
- Notebook computer
- Parallel processing system
- Personal Computer (PC)
- Processor-bound application
- Server computer
- Server process
- Supercomputer
- System board
- Workstation
Chapter 21

Introduction to C Programming Language

Computer Fundamentals - Pradeep K. Sinha & Priti Sinha
Learning Objectives

In this chapter you will learn about:

- Features of C
- Various constructs and their syntax
- Data types and operators in C
- Control and Loop Structures in C
- Functions in C
- Writing programs in C
Features

- Reliable, simple, and easy to use
- Has virtues of high-level programming language with efficiency of assembly language
- Supports user-defined data types
- Supports modular and structured programming concepts
- Supports a rich library of functions
- Supports pointers with pointer operations
- Supports low-level memory and device access
- Small and concise language
- Standardized by several international standards body
## C Character Set

<table>
<thead>
<tr>
<th>Category</th>
<th>Valid Characters</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uppercase alphabets</td>
<td>A, B, C, ..., Z</td>
<td>26</td>
</tr>
<tr>
<td>Lowercase alphabets</td>
<td>a, b, c, ..., z</td>
<td>26</td>
</tr>
<tr>
<td>Digits</td>
<td>0, 1, 2, ..., 9</td>
<td>10</td>
</tr>
<tr>
<td>Special characters</td>
<td>~ `! @ # % ^ &amp; * ( ) _ − + =</td>
<td>{ } [: ; &quot; ' &lt; &gt; , . ? /</td>
</tr>
<tr>
<td></td>
<td></td>
<td>93</td>
</tr>
</tbody>
</table>
Constants

- Constant is a value that never changes
- Three primitive types of constants supported in C are:
  - Integer
  - Real
  - Character
Rules for Constructing Integer Constants

- Must have at least one digit
- + or – sign is optional
- No special characters (other than + and – sign) are allowed
- Allowable range is:
  - -32768 to 32767 for integer and short integer constants (16 bits storage)
  - -2147483648 to 2147483647 for long integer constants (32 bits storage)
- Examples are: 8, +17, -6
Rules for Constructing Real Constants in Exponential Form

- Has two parts – mantissa and exponent - separated by ‘e’ or ‘E’
- Mantissa part is constructed by the rules for constructing real constants in fractional form
- Exponent part is constructed by the rules for constructing integer constants
- Allowable range is -3.4e38 to 3.4e38
- Examples are: 8.6e5, +4.3E-8, -0.1e+4
Rules for Constructing Character Constants

- Single character from C character set
- Enclosed within single inverted comma (also called single quote) punctuation mark
- Examples are: ‘A’ ‘a’ ‘8’ ‘%’
Variables

- Entity whose value may vary during program execution
- Has a name and type associated with it
- Variable name specifies programmer given name to the memory area allocated to a variable
- Variable type specifies the type of values a variable can contain
- Example: In $i = i + 5$, $i$ is a variable
Rules for Constructing Variables Names

- Can have 1 to 31 characters
- Only alphabets, digits, and underscore (as in last_name) characters are allowed
- Names are case sensitive (nNum and nNUM are different)
- First character must be an alphabet
- Underscore is the only special character allowed
- Keywords cannot be used as variable names
- Examples are: I saving_2007 ArrSum
## Data Types Used for Variable Type Declaration

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Minimum Storage Allocated</th>
<th>Used for Variables that can contain</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>2 bytes (16 bits)</td>
<td>integer constants in the range -32768 to 32767</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes (16 bits)</td>
<td>integer constants in the range -32768 to 32767</td>
</tr>
<tr>
<td>long</td>
<td>4 bytes (32 bits)</td>
<td>integer constants in the range -2147483648 to 2147483647</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes (32 bits)</td>
<td>real constants with minimum 6 decimal digits precision</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes (64 bits)</td>
<td>real constants with minimum 10 decimal digits precision</td>
</tr>
<tr>
<td>char</td>
<td>1 byte (8 bits)</td>
<td>character constants</td>
</tr>
<tr>
<td>enum</td>
<td>2 bytes (16 bits)</td>
<td>Values in the range -32768 to 32767</td>
</tr>
<tr>
<td>void</td>
<td>No storage allocated</td>
<td>No value assigned</td>
</tr>
</tbody>
</table>
### Variable Type Declaration Examples

<table>
<thead>
<tr>
<th>Type</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>count</td>
</tr>
<tr>
<td>short</td>
<td>index</td>
</tr>
<tr>
<td>long</td>
<td>principle</td>
</tr>
<tr>
<td>float</td>
<td>area</td>
</tr>
<tr>
<td>double</td>
<td>radius</td>
</tr>
<tr>
<td>char</td>
<td>c</td>
</tr>
</tbody>
</table>
## Standard Qualifiers in C

<table>
<thead>
<tr>
<th>Category</th>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifetime</td>
<td>auto</td>
<td>Temporary variable</td>
</tr>
<tr>
<td></td>
<td>register</td>
<td>Attempt to store in processor register, fast access</td>
</tr>
<tr>
<td></td>
<td>static</td>
<td>Permanent, initialized</td>
</tr>
<tr>
<td></td>
<td>extern</td>
<td>Permanent, initialized but declaration elsewhere</td>
</tr>
<tr>
<td>Modifiability</td>
<td>const</td>
<td>Cannot be modified once created</td>
</tr>
<tr>
<td></td>
<td>volatile</td>
<td>May be modified by factors outside program</td>
</tr>
<tr>
<td>Sign</td>
<td>signed</td>
<td>+ or –</td>
</tr>
<tr>
<td></td>
<td>unsigned</td>
<td>+ only</td>
</tr>
<tr>
<td>Size</td>
<td>short</td>
<td>16 bits</td>
</tr>
<tr>
<td></td>
<td>long</td>
<td>32 bits</td>
</tr>
</tbody>
</table>
Lifetime and Visibility Scopes of Variables

- Lifetime of all variables (except those declared as `static`) is same as that of function or statement block it is declared in.
- Lifetime of variables declared in global scope and static is same as that of the program.
- Variable is visible and accessible in the function or statement block it is declared in.
- Global variables are accessible from anywhere in program.
- Variable name must be unique in its visibility scope.
- Local variable has access precedence over global variable of same name.
Keywords (or reserved words) are predefined words whose meanings are known to C compiler.

- **Keywords** cannot be used as variable names.

<table>
<thead>
<tr>
<th>auto</th>
<th>double</th>
<th>int</th>
<th>struct</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>else</td>
<td>long</td>
<td>switch</td>
</tr>
<tr>
<td>case</td>
<td>enum</td>
<td>register</td>
<td>typedef</td>
</tr>
<tr>
<td>char</td>
<td>extern</td>
<td>return</td>
<td>union</td>
</tr>
<tr>
<td>const</td>
<td>float</td>
<td>short</td>
<td>unsigned</td>
</tr>
<tr>
<td>continue</td>
<td>for</td>
<td>signed</td>
<td>void</td>
</tr>
<tr>
<td>default</td>
<td>goto</td>
<td>sizeof</td>
<td>volatile</td>
</tr>
<tr>
<td>do</td>
<td>if</td>
<td>static</td>
<td>while</td>
</tr>
</tbody>
</table>
Comments

- Comments are enclosed within `/*` and `*/`
- Comments are ignored by the compiler
- Comment can also split over multiple lines
- **Example:** `/* This is a comment statement */`
Operators

- Operators in C are categorized into data access, arithmetic, logical, bitwise, and miscellaneous.
- **Associativity** defines the order of evaluation when operators of same precedence appear in an expression.
  - a = b = c = 15, '=' has R → L associativity
  - First c = 15, then b = c, then a = b is evaluated
- **Precedence** defines the order in which calculations involving two or more operators is performed.
  - x + y * z , '*' is performed before '+'

Ref. Page 401  Chapter 21: Introduction to C Programming Language
## Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition; $x + y$</td>
<td>L $→$ R</td>
<td>4</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction; $x - y$</td>
<td>L $→$ R</td>
<td>4</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication; $x * y$</td>
<td>L $→$ R</td>
<td>3</td>
</tr>
<tr>
<td>/</td>
<td>Division; $x / y$</td>
<td>L $→$ R</td>
<td>3</td>
</tr>
<tr>
<td>%</td>
<td>Remainder (or Modulus); $x % y$</td>
<td>L $→$ R</td>
<td>3</td>
</tr>
<tr>
<td>++</td>
<td>Increment; $x++$ means post-increment (increment the value of $x$ by 1 after using its value); $++x$ means pre-increment (increment the value of $x$ by 1 before using its value)</td>
<td>L $→$ R</td>
<td>1</td>
</tr>
</tbody>
</table>

Ref. Page 401  Chapter 21: Introduction to C Programming Language  Slide 18/65
### Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>-=</td>
<td>Decrement;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x-- means post-decrement (decrement the value of x by 1 after using its value);</td>
<td>L → R</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>--x means pre-decrement (decrement the value of x by 1 before using its value)</td>
<td>R → L</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>x = y means assign the value of y to x</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>+=</td>
<td>x += 5 means x = x + 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>-=</td>
<td>x -= 5 means x = x - 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>*=</td>
<td>x *= 5 means x = x * 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>/=</td>
<td>x /= 5 means x = x / 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>%=</td>
<td>x %= 5 means x = x % 5</td>
<td>R → L</td>
<td>14</td>
</tr>
</tbody>
</table>
## Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Reverse the logical value of a single variable; !x means if the value of x is non-zero, make it zero; and if it is zero, make it one</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than; x &gt; y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than; x &lt; y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to; x &gt;= y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to; x &lt;= y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>==</td>
<td>Equal to; x == y</td>
<td>L → R</td>
<td>7</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to; x != y</td>
<td>L → R</td>
<td>7</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>AND; x &amp;&amp; y means both x and y should be true (non-zero) for result to be true</td>
<td>L → R</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OR; x</td>
</tr>
<tr>
<td>z?x:y</td>
<td>If z is true (non-zero), then the value returned is x, otherwise the value returned is y</td>
<td>R → L</td>
<td>13</td>
</tr>
</tbody>
</table>
## Bitwise Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>Complement; ( \sim x ) means All 1s are changed to 0s and 0s to 1s</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>&amp;</td>
<td>AND; ( x &amp; y ) means x AND y</td>
<td>L → R</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L → R</td>
<td>10</td>
</tr>
<tr>
<td>^</td>
<td>Exclusive OR; ( x \oplus y ) means x XOR y</td>
<td>L → R</td>
<td>9</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>Left shift; ( x &lt;&lt; 4 ) means shift all bits in x four places to the left</td>
<td>L → R</td>
<td>5</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Right shift; ( x &gt;&gt; 3 ) means shift all bits in x three places to the right</td>
<td>L → R</td>
<td>5</td>
</tr>
<tr>
<td>&amp;=</td>
<td>( x &amp;= y ) means x = x &amp; y</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>=</td>
<td>( x</td>
<td>= y ) means x = x</td>
</tr>
<tr>
<td>^=</td>
<td>( x ^= y ) means x = x ^ y</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>&lt;&lt;=</td>
<td>( x &lt;&lt;= 4 ) means shift all bits in x four places to the left and assign the result to x</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>&gt;&gt;=</td>
<td>( x &gt;&gt;= 3 ) means shift all bits in x three places to the right and assign the result to x</td>
<td>R → L</td>
<td>14</td>
</tr>
</tbody>
</table>
## Data Access Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>x[y]</td>
<td>Access $y^{th}$ element of array $x$; $y$ starts from zero and increases monotonically up to one less than declared size of array</td>
<td>$L \rightarrow R$</td>
<td>1</td>
</tr>
<tr>
<td>x.y</td>
<td>Access the member variable $y$ of structure $x$</td>
<td>$L \rightarrow R$</td>
<td>1</td>
</tr>
<tr>
<td>x$\rightarrow$y</td>
<td>Access the member variable $y$ of structure $x$</td>
<td>$L \rightarrow R$</td>
<td>1</td>
</tr>
<tr>
<td>&amp;x</td>
<td>Access the address of variable $x$</td>
<td>$R \rightarrow L$</td>
<td>2</td>
</tr>
<tr>
<td>*x</td>
<td>Access the value stored in the storage location (address) pointed to by pointer variable $x$</td>
<td>$R \rightarrow L$</td>
<td>2</td>
</tr>
</tbody>
</table>
### Miscellaneous Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>x(y)</td>
<td>Evaluates function x with argument y</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>sizeof (x)</td>
<td>Evaluate the size of variable x in bytes</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>sizeof (type)</td>
<td>Evaluate the size of data type “type” in bytes</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>(type) x</td>
<td>Return the value of x after converting it from declared data type of variable x to the new data type “type”</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>x,y</td>
<td>Sequential operator (x then y)</td>
<td>L → R</td>
<td>15</td>
</tr>
</tbody>
</table>
Statements

- C program is a combination of statements written between { and } braces.
- Each statement performs a set of operations.
- Null statement, represented by ";" or empty {} braces, does not perform any operation.
- A simple statement is terminated by a semicolon ";".
- Compound statements, called statement block, perform complex operations combining null, simple, and other block statements.
Examples of Statements

ß a = (x + y) * 10;  /* simple statement */

ß if (sell > cost)  /* compound statement follows */
{   
    profit = sell - cost;
    printf ("profit is %d", profit);
}
else  /* null statement follows */
{  
}

Simple I/O Operations

- C has no keywords for I/O operations
- Provides standard library functions for performing all I/O operations
### Basic Library Functions for I/O Operations

<table>
<thead>
<tr>
<th>I/O Library Functions</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getch()</code></td>
<td>Inputs a single character (most recently typed) from standard input (usually console).</td>
</tr>
<tr>
<td><code>getche()</code></td>
<td>Inputs a single character from console and echoes (displays) it.</td>
</tr>
<tr>
<td><code>getchar()</code></td>
<td>Inputs a single character from console and echoes it, but requires <code>Enter</code> key to be typed after the character.</td>
</tr>
<tr>
<td><code>putchar()</code> or <code>putch()</code></td>
<td>Outputs a single character on console (screen).</td>
</tr>
<tr>
<td><code>scanf()</code></td>
<td>Enables input of formatted data from console (keyboard). Formatted input data means we can specify the data type expected as input. Format specifiers for different data types are given in Figure 21.6.</td>
</tr>
<tr>
<td><code>printf()</code></td>
<td>Enables obtaining an output in a form specified by programmer (formatted output). Format specifiers are given in Figure 21.6. Newline character “\n” is used in <code>printf()</code> to get the output split over separate lines.</td>
</tr>
<tr>
<td><code>gets()</code></td>
<td>Enables input of a string from keyboard. Spaces are accepted as part of the input string, and the input string is terminated when <code>Enter</code> key is hit. Note that although <code>scanf()</code> enables input of a string of characters, it does not accept multi-word strings (spaces in-between).</td>
</tr>
<tr>
<td><code>puts()</code></td>
<td>Enables output of a multi-word string</td>
</tr>
</tbody>
</table>
**Basic Format Specifiers for `scanf()` and `printf()`**

<table>
<thead>
<tr>
<th>Format Specifiers</th>
<th>Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%d</code></td>
<td>integer (short signed)</td>
</tr>
<tr>
<td><code>%u</code></td>
<td>integer (short unsigned)</td>
</tr>
<tr>
<td><code>%ld</code></td>
<td>integer (long signed)</td>
</tr>
<tr>
<td><code>%lu</code></td>
<td>integer (long unsigned)</td>
</tr>
<tr>
<td><code>%f</code></td>
<td>real (float)</td>
</tr>
<tr>
<td><code>%lf</code></td>
<td>real (double)</td>
</tr>
<tr>
<td><code>%c</code></td>
<td>character</td>
</tr>
<tr>
<td><code>%s</code></td>
<td>string</td>
</tr>
</tbody>
</table>
/* A portion of C program to illustrate formatted input and output */

int maths, science, english, total;
float percent;
clrscr(); /* A C library function to make the screen clear */
printf("Maths marks = "); /* Displays “Maths marks = ” */
scanf("%d", &maths); /* Accepts entered value and stores in variable “maths” */
printf("\n Science marks = "); /* Displays “Science marks = ” on next line because of \n */
scanf("%d", &science); /* Accepts entered value and stores in variable “science” */
printf("\n English marks = "); /* Displays “English marks = ” on next line because of \n */
scanf("%d", &english); /* Accepts entered value and stores in variable “english” */

total = maths + science + english;
percent = total/3; /* Calculates percentage and stores in variable “percent” */

printf("\n Percentage marks obtained = %f", percent); /* Displays “Percentage marks obtained = 85.66” on next line because of \n */

(Continued on next slide)
Formatted I/O Example

(Continued from previous slide..)

Output:
Maths marks = 92
Science marks = 87
English marks = 78
Percentage marks obtained = 85.66
Preprocessor Directives

- **Preprocessor** is a program that prepares a program for the C compiler.

- Three common preprocessor directives in C are:
  - **#include** – Used to look for a file and place its contents at the location where this preprocessor directives is used.
  - **#define** – Used for macro expansion.
  - **#ifdef..#endif** – Used for conditional compilation of segments of a program.
Examples of Preprocessor Directives

```c
#include <stdio.h>
#define PI 3.1415
#define AND &&
#define ADMIT printf(“The candidate can be admitted”);

#ifdef WINDOWS
  ...
  ...
  Code specific to windows operating system
  ...
  ...
#else
  ...
  ...
  Code specific to Linux operating system
  ...
  ...
#endif

Code common to both operating systems
```
## Standard Preprocessor Directives in C

<table>
<thead>
<tr>
<th>Preprocessor Directive</th>
<th>Meaning</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Null directive</td>
<td>Simple</td>
</tr>
<tr>
<td>#error message</td>
<td>Prints message when processed</td>
<td></td>
</tr>
<tr>
<td>#line linenum filename</td>
<td>Used to update code line number and filename</td>
<td></td>
</tr>
<tr>
<td>#pragma name</td>
<td>Compiler specific settings</td>
<td></td>
</tr>
<tr>
<td>#include filename</td>
<td>Includes content of another file</td>
<td>File</td>
</tr>
<tr>
<td>#define macro/string</td>
<td>Define a macro or string substitution</td>
<td>Macro</td>
</tr>
<tr>
<td>#undef macro</td>
<td>Removes a macro definition</td>
<td>Macro</td>
</tr>
<tr>
<td>#if expr macro</td>
<td>Includes following lines if expr is true</td>
<td>Conditional</td>
</tr>
<tr>
<td># elif expr</td>
<td>Includes following lines if expr is true</td>
<td></td>
</tr>
<tr>
<td>#else</td>
<td>Handles otherwise conditions of #if</td>
<td></td>
</tr>
<tr>
<td>#endif</td>
<td>Closes #if or #elif block</td>
<td></td>
</tr>
<tr>
<td>#ifdef macro</td>
<td>Includes following lines if macro is defined</td>
<td></td>
</tr>
<tr>
<td>#ifndef imacro</td>
<td>Includes following lines if macro is not defined</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>String forming operator</td>
<td>Operators</td>
</tr>
<tr>
<td>##</td>
<td>Token pasting operator</td>
<td></td>
</tr>
<tr>
<td>defined</td>
<td>same as #ifdef</td>
<td></td>
</tr>
</tbody>
</table>
Pointers

- C pointers allow programmers to directly access memory addresses where variables are stored.
- Pointer variable is declared by adding a ‘*’ symbol before the variable name while declaring it.
- If \( p \) is a pointer to a variable (e.g. `int i, *p = i;`)
  - Using \( p \) means address of the storage location of the pointed variable
  - Using \( *p \) means value stored in the storage location of the pointed variable
- Operator ‘&’ is used with a variable to mean variable’s address, e.g. `&i` gives address of variable \( i \)
Illustrating Pointers Concept

Address of \(i\) = 1000
Value of \(i\) = 62

```
int i = 62;
int *p;
int j;
p = &i;  /* p becomes 1000 */
j = *p;  /* j becomes 62 */
j = 0;   /* j becomes zero */
j = *(&i) /* j becomes 62 */
```
**Array**

- Collection of fixed number of elements in which all elements are of the same data type
- Homogeneous, linear, and contiguous memory structure
- Elements can be referred to by using their subscript or index position that is monotonic in nature
- First element is always denoted by subscript value of 0 (zero), increasing monotonically up to one less than declared size of array
- Before using an array, its type and dimension must be declared
- Can also be declared as multi-dimensional such as Matrix2D[10][10]
Illustrating Arrays Concept

<table>
<thead>
<tr>
<th>1010</th>
<th>92</th>
</tr>
</thead>
<tbody>
<tr>
<td>1008</td>
<td>63</td>
</tr>
<tr>
<td>1006</td>
<td>82</td>
</tr>
<tr>
<td>1004</td>
<td>66</td>
</tr>
<tr>
<td>1002</td>
<td>84</td>
</tr>
<tr>
<td>1000</td>
<td>45</td>
</tr>
<tr>
<td>1012</td>
<td>10.25</td>
</tr>
<tr>
<td>1008</td>
<td>250.00</td>
</tr>
<tr>
<td>1004</td>
<td>155.50</td>
</tr>
<tr>
<td>1000</td>
<td>82.75</td>
</tr>
<tr>
<td>1005</td>
<td>Y</td>
</tr>
<tr>
<td>1004</td>
<td>A</td>
</tr>
<tr>
<td>1003</td>
<td>B</td>
</tr>
<tr>
<td>1002</td>
<td>M</td>
</tr>
<tr>
<td>1001</td>
<td>O</td>
</tr>
<tr>
<td>1000</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int marks[6];</th>
<th>float price[4];</th>
<th>char city[6];</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each element</td>
<td>Each element</td>
<td>Each element</td>
</tr>
<tr>
<td>being an int</td>
<td>being a float</td>
<td>being a char</td>
</tr>
<tr>
<td>occupies 2 bytes</td>
<td>occupies 4 bytes</td>
<td>occupies 1 byte</td>
</tr>
<tr>
<td>marks[0] = 45</td>
<td>price[0] = 82.75</td>
<td>city[0] = ‘B’</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>(a) An array of integers having 6 elements</td>
<td>(b) An array of real numbers having 4 elements</td>
<td>(c) An array of characters having 6 elements</td>
</tr>
</tbody>
</table>
String

- One-dimensional array of characters terminated by a null character (‘\0’)
- Initialized at declaration as
  - `char name[] = “PRADEEP”;`
- Individual elements can be accessed in the same way as we access array elements such as `name[3] = ‘D’`
- Strings are used for text processing
- C provides a rich set of string handling library functions
## Library Functions for String Handling

<table>
<thead>
<tr>
<th>Library Function</th>
<th>Used To</th>
</tr>
</thead>
<tbody>
<tr>
<td>strlen</td>
<td>Obtain the length of a string</td>
</tr>
<tr>
<td>strlwr</td>
<td>Convert all characters of a string to lowercase</td>
</tr>
<tr>
<td>strupr</td>
<td>Convert all characters of a string to uppercase</td>
</tr>
<tr>
<td>strcat</td>
<td>Concatenate (append) one string at the end of another</td>
</tr>
<tr>
<td>strncat</td>
<td>Concatenate only first n characters of a string at the end of another</td>
</tr>
<tr>
<td>strcpy</td>
<td>Copy a string into another</td>
</tr>
<tr>
<td>strncpy</td>
<td>Copy only the first n characters of a string into another</td>
</tr>
<tr>
<td>strcmp</td>
<td>Compare two strings</td>
</tr>
<tr>
<td>strncmp</td>
<td>Compare only first n characters of two strings</td>
</tr>
<tr>
<td>strncmp</td>
<td>Compare two strings without regard to case</td>
</tr>
<tr>
<td>strnicmp</td>
<td>Compare only first n characters of two strings without regard to case</td>
</tr>
<tr>
<td>strdup</td>
<td>Duplicate a string</td>
</tr>
<tr>
<td>strchr</td>
<td>Find first occurrence of a given character in a string</td>
</tr>
<tr>
<td>strrchr</td>
<td>Find last occurrence of a given character in a string</td>
</tr>
<tr>
<td>strstr</td>
<td>Find first occurrence of a given string in another string</td>
</tr>
<tr>
<td>strset</td>
<td>Set all characters of a string to a given character</td>
</tr>
<tr>
<td>strnset</td>
<td>Set first n characters of a string to a given character</td>
</tr>
<tr>
<td>strrev</td>
<td>Reverse a string</td>
</tr>
</tbody>
</table>
User Defined Data Types (UDTs)

- UDT is composite data type whose composition is not included in language specification
- Programmer declares them in a program where they are used
- Two types of UDTs are:
  - Structure
  - Union
Structure

- UDT containing a number of data types grouped together
- Constituents data types may or may not be of different types
- Has continuous memory allocation and its minimum size is the sum of sizes of its constituent data types
- All elements (member variable) of a structure are publicly accessible
- Each member variable can be accessed using "." (dot) operator or pointer (EmpRecord.EmpID or EmpRecord → EmpID)
- Can have a pointer member variable of its own type, which is useful in creating linked list and similar data structures
struct Employee
{
    int EmpID;
    char EmpName[20];
};

Struct Employee EmpRecord;
Struct Employee *pEmpRecord = &EmpRecord
Union

- UDT referring to same memory location using several data types
- Mathematical union of all constituent data types
- Each data member begins at the same memory location
- Minimum size of a union variable is the size of its largest constituent data types
- Each member variable can be accessed using "\," (dot) operator
- Section of memory can be treated as a variable of one type on one occasion, and of another type on another occasion
unionNum
{
    int intNum;
    unsigned
unsNum';
};
union Num Number;
**Difference Between Structure and Union**

- Both group a number of data types together
- Structure allocates different memory space contiguously to different data types in the group
- Union allocates the same memory space to different data types in the group
Control Structures

- Control structures (branch statements) are decision points that control the flow of program execution based on:
  - Some condition test (conditional branch)
  - Without condition test (unconditional branch)
- Ensure execution of other statement/block or cause skipping of some statement/block
Conditional Branch Statements

- **if** is used to implement simple one-way test. It can be in one of the following forms:
  - if..stmt
  - if..stmt1..else..stmt2
  - if..stmt1..else..if..stmtn

- **switch** facilitates multi-way condition test and is very similar to the third if construct when primary test object remains same across all condition tests.
Examples of “if” Construct

- if (i <= 0)
  i++;
- if (i <= 0)
  i++;
  else
  j++;
- if (i <= 0)
  i++;
  else if (i >= 0)
  j++;
  else
  k++;
switch(ch)
{
    case ‘A’:
    case ‘B’:
    case ‘C’:
        printf("Capital");
        break;
    case ‘a’:
    case ‘b’:
    case ‘c’:
        printf("Small");
        break;
    default:
        printf("Not cap or small");
}

Same thing can be written also using *if* construct as:

```c
if (ch == ‘A’ || ch == ‘B’ || ch == ‘C’)
    printf("Capital");
else if (ch == ‘a’ || ch == ‘b’ || ch == ‘c’)
    printf("Small");
else
    printf("Not cap or small");
```
Break: Causes unconditional exit from *for*, *while*, *do*, or *switch* constructs. Control is transferred to the statement immediately outside the block in which *break* appears.

Continue: Causes unconditional transfer to next iteration in a *for*, *while*, or *do* construct. Control is transferred to the statement beginning the block in which *continue* appears.

Goto label: Causes unconditional transfer to statement marked with the label within the function.
Return [value/variable]: Causes immediate termination of function in which it appears and transfers control to the statement that called the function. Optionally, it provides a value compatible to the function’s return data type.
Loop Structures

- Loop statements are used to repeat the execution of statement or blocks
- Two types of loop structures are:
  - **Pretest**: Condition is tested before each iteration to check if loop should occur
  - **Posttest**: Condition is tested after each iteration to check if loop should continue (at least, a single iteration occurs)
Pretest Loop Structures

**for:** It has three parts:
- **Initializer** is executed at start of loop
- **Loop condition** is tested before iteration to decide whether to continue or terminate the loop
- **Incrementor** is executed at the end of each iteration

**While:** It has a *loop condition* only that is tested before each iteration to decide whether to continue to terminate the loop
Examples of “for” and “while” Constructs

- for (i=0; i < 10; i++)
  printf(“i = %d”, i);

- while (i < 10)
  {
    printf(“i = %d”, i);
    i++;
  }
Posttest Loop Construct
“do...while”

- It has a loop condition only that is tested after each iteration to decide whether to continue with next iteration or terminate the loop.

- Example of do...while is:

```c
do {
    printf("i = %d", i);
    i++;
} while (i < 10);
```
Functions

- Functions (or subprograms) are building blocks of a program
- All functions must be declared and defined before use
- Function declaration requires `functionname, argument list, and return type`
- Function definition requires coding the body or logic of function
- Every C program must have a `main` function. It is the entry point of the program
Example of a Function

```c
int myfunc ( int Val, int ModVal )
{
    unsigned temp;
    temp = Val % ModVal;
    return temp;
}
```

This function can be called from any other place using simple statement:

```c
int n = myfunc(4, 2);
```
Sample C Program (Program-1)

/* Program to accept an integer from console and to display
   whether the number is even or odd */

#include <stdio.h>
void main()
{
    int number, remainder;
    clrscr(); /* clears the console screen */
    printf("Enter an integer: ");
    scanf("%d", &number);
    remainder = number % 2;
    if (remainder == 0)
        printf("\n   The given number is even");
    else
        printf("\n   The given number is odd");

    getch();
}
Sample C Program (Program-2)

/* Program to accept an integer in the range 1 to 7 (both inclusive) from console and to display the corresponding day (Monday for 1, Tuesday for 2, Wednesday for 3, and so on). If the entered number is out of range, the program displays a message saying that */

#include <stdio.h>
#include <conio.h>

#define MON printf("\n Entered number is 1 hence day is MONDAY");
#define TUE printf("\n Entered number is 2 hence day is TUESDAY");
#define WED printf("\n Entered number is 3 hence day is WEDNESDAY");
#define THU printf("\n Entered number is 4 hence day is THURSDAY");
#define FRI printf("\n Entered number is 5 hence day is FRIDAY");
#define SAT printf("\n Entered number is 6 hence day is SATURDAY");
#define SUN printf("\n Entered number is 7 hence day is SUNDAY");
#define OTH printf("\n Entered number is out of range");

void main()
{
    int day;
    clrscr();
    printf("Enter an integer in the range 1 to 7");
    scanf("%d", &day);
    switch(day)
    {
    
    (Continued on next slide)


Sample C Program (Program-2)

(Continued from previous slide..)

```c
{ 
    Case 1: 
    MON; 
    break; 
    Case 2: 
    TUE; 
    break; 
    Case 3: 
    WED; 
    break; 
    Case 4: 
    THU; 
    break; 
    Case 5: 
    FRI; 
    break; 
    Case 6: 
    SAT; 
    break; 
    Case 7: 
    SUN; 
    break; 
    default: 
    OTH; 
} 
getch();
}```
/* Program to accept the radius of a circle from console and to calculate and display its area and circumference */

#include <stdio.h>
#include <conio.h>
#define PI 3.1415

void main()
{
    float radius, area, circum;
    clrscr();
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);
    area = PI * radius * radius;
    circum = 2 * PI * radius;
    printf("\n Area and circumference of the circle are %f and %f respectively", area, circum);
    getch();
}

(Continued on next slide)
Program to accept a string from console and to display the number of vowels in the string

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>

void main()
{
    char input_string[50]; /* maximum 50 characters */
    int len;
    int i = 0, cnt = 0;
    clrscr();
    printf(“Enter a string of less than 50 characters: \n”);
    gets(input_string);
    len = strlen(input_string);
    for (i = 0; i < len; i++)
    {
        switch (input_string[i])
        ...
    }
}```
Sample C Program (Program-4)

```
{ 
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
    case 'A':
    case 'E':
    case 'I':
    case 'O':
    case 'U':
        cnt++
    }
}
printf("\n Number of vowels in the string are: %d", cnt);
getch();
```
/ Program to illustrate use of a user defined function. The program initializes an array of n elements from 0 to n-1 and then calculates and prints the sum of the array elements. In this example n = 10 */

#include <stdio.h>
#define SIZE 10

int ArrSum(int *p, int n);
{
    int s, tot = 0;
    for(s = 0; s < n; s++)
    {
        tot += *p;
        p++;
    }
    return tot;
}

int main()
{
    int i = 0, sum = 0;
    int nArr[SIZE] = {0};
    while(i < SIZE)
    {
        nArr[i] = i;
        i++
    }
    sum = ArrSum(nArr, SIZE);
    printf("Sum of 0 to 9 = %d\n", sum);
    return 0;
}
Key Words/Phrases

- Arithmetic operators
- Arrays
- Assignment operators
- Bit-level manipulation
- Bitwise operators
- Branch statement
- Character set
- Comment statement
- Compound statement
- Conditional branch
- Conditional compilation
- Constants
- Control structures
- Format specifiers
- Formatted I/O
- Function
- Keywords
- Library functions
- Logical operators
- Loop structures
- Macro expansion
- Main function
- Member element
- Null statement
- Operator associativity
- Operator precedence
- Pointer
- Posttest loop
- Preprocessor directives
- Pretest loop
- Primitive data types
- Reserved words
- Simple statement
- Statement block
- Strings
- Structure data type
- Unconditional branch
- Union data type
- User-defined data types
- Variable name
- Variable type declaration
- Variables
Chapter 21
Introduction to C Programming Language

Learning Objectives

In this chapter you will learn about:

- Features of C
- Various constructs and their syntax
- Data types and operators in C
- Control and Loop Structures in C
- Functions in C
- Writing programs in C
Features

- Reliable, simple, and easy to use
- Has virtues of high-level programming language with efficiency of assembly language
- Supports user-defined data types
- Supports modular and structured programming concepts
- Supports a rich library of functions
- Supports pointers with pointer operations
- Supports low-level memory and device access
- Small and concise language
- Standardized by several international standards body

C Character Set

<table>
<thead>
<tr>
<th>Category</th>
<th>Valid Characters</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uppercase alphabets</td>
<td>A, B, C, ..., Z</td>
<td>26</td>
</tr>
<tr>
<td>Lowercase alphabets</td>
<td>a, b, c, ..., z</td>
<td>26</td>
</tr>
<tr>
<td>Digits</td>
<td>0, 1, 2, ..., 9</td>
<td>10</td>
</tr>
<tr>
<td>Special characters</td>
<td>~ `! @ # % ^ &amp; * ( ) _ ~ + = [ \ ] { } : ; &quot; ' &lt; &gt; . ? /</td>
<td>31</td>
</tr>
</tbody>
</table>

Total: 93
**Constants**

- Constant is a value that never changes
- Three primitive types of constants supported in C are:
  - Integer
  - Real
  - Character

**Rules for Constructing Integer Constants**

- Must have at least one digit
- + or − sign is optional
- No special characters (other than + and − sign) are allowed
- Allowable range is:
  - -32768 to 32767 for integer and short integer constants (16 bits storage)
  - -2147483648 to 2147483647 for long integer constants (32 bits storage)
- Examples are: 8, +17, -6
Rules for Constructing Real Constants in Exponential Form

- Has two parts – mantissa and exponent - separated by ‘e’ or ‘E’
- Mantissa part is constructed by the rules for constructing real constants in fractional form
- Exponent part is constructed by the rules for constructing integer constants
- Allowable range is -3.4e38 to 3.4e38
- Examples are: 8.6e5, +4.3E-8, -0.1e+4

Rules for Constructing Character Constants

- Single character from C character set
- Enclosed within single inverted comma (also called single quote) punctuation mark
- Examples are: ‘A’ ‘a’ ‘8’ ‘%’
**Variables**

- Entity whose value may vary during program execution
- Has a name and type associated with it
- Variable name specifies programmer given name to the memory area allocated to a variable
- Variable type specifies the type of values a variable can contain
- Example: In `i = i + 5`, `i` is a variable

**Rules for Constructing Variables Names**

- Can have 1 to 31 characters
- Only alphabets, digits, and underscore (as in `last_name`) characters are allowed
- Names are case sensitive (`nNum` and `nNUM` are different)
- First character must be an alphabet
- Underscore is the only special character allowed
- Keywords cannot be used as variable names
- Examples are: `I saving_2007 ArrSum`
### Data Types Used for Variable Type Declaration

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Minimum Storage Allocated</th>
<th>Used for Variables that can contain</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>2 bytes (16 bits)</td>
<td>integer constants in the range -32768 to 32767</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes (16 bits)</td>
<td>integer constants in the range -32768 to 32767</td>
</tr>
<tr>
<td>long</td>
<td>4 bytes (32 bits)</td>
<td>integer constants in the range -2147483648 to 2147483647</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes (32 bits)</td>
<td>real constants with minimum 6 decimal digits precision</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes (64 bits)</td>
<td>real constants with minimum 10 decimal digits precision</td>
</tr>
<tr>
<td>char</td>
<td>1 byte (8 bits)</td>
<td>character constants</td>
</tr>
<tr>
<td>enum</td>
<td>2 bytes (16 bits)</td>
<td>Values in the range -32768 to 32767</td>
</tr>
<tr>
<td>void</td>
<td>No storage allocated</td>
<td>No value assigned</td>
</tr>
</tbody>
</table>

### Variable Type Declaration Examples

```c
int count;
short index;
long principle;
float area;
double radius;
char c;
```
### Standard Qualifiers in C

<table>
<thead>
<tr>
<th>Category</th>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifetime</td>
<td>auto</td>
<td>Temporary variable</td>
</tr>
<tr>
<td></td>
<td>register</td>
<td>Attempt to store in processor register, fast access</td>
</tr>
<tr>
<td></td>
<td>static</td>
<td>Permanent, initialized</td>
</tr>
<tr>
<td></td>
<td>extern</td>
<td>Permanent, initialized but declaration elsewhere</td>
</tr>
<tr>
<td>Modifiability</td>
<td>const</td>
<td>Cannot be modified once created</td>
</tr>
<tr>
<td></td>
<td>volatile</td>
<td>May be modified by factors outside program</td>
</tr>
<tr>
<td>Sign</td>
<td>signed</td>
<td>+ or ~</td>
</tr>
<tr>
<td></td>
<td>unsigned</td>
<td>+ only</td>
</tr>
<tr>
<td>Size</td>
<td>short</td>
<td>16 bits</td>
</tr>
<tr>
<td></td>
<td>long</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

### Lifetime and Visibility Scopes of Variables

- Lifetime of all variables (except those declared as `static`) is same as that of function or statement block it is declared in.
- Lifetime of variables declared in global scope and static is same as that of the program.
- Variable is visible and accessible in the function or statement block it is declared in.
- Global variables are accessible from anywhere in program.
- Variable name must be unique in its visibility scope.
- Local variable has access precedence over global variable of same name.
Keywords

- **Keywords** (or reserved words) are predefined words whose meanings are known to C compiler.
- C has 32 keywords.
- Keywords cannot be used as variable names.

<table>
<thead>
<tr>
<th>auto</th>
<th>double</th>
<th>int</th>
<th>struct</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>else</td>
<td>long</td>
<td>switch</td>
</tr>
<tr>
<td>case</td>
<td>enum</td>
<td>register</td>
<td>typedef</td>
</tr>
<tr>
<td>char</td>
<td>extern</td>
<td>return</td>
<td>union</td>
</tr>
<tr>
<td>const</td>
<td>float</td>
<td>short</td>
<td>unsigned</td>
</tr>
<tr>
<td>continue</td>
<td>for</td>
<td>signed</td>
<td>void</td>
</tr>
<tr>
<td>default</td>
<td>goto</td>
<td>sizeof</td>
<td>volatile</td>
</tr>
<tr>
<td>do</td>
<td>if</td>
<td>static</td>
<td>while</td>
</tr>
</tbody>
</table>

Comments

- Comments are enclosed within `/*` and `*/`.
- Comments are ignored by the compiler.
- Comment can also split over multiple lines.
- Example: `/* This is a comment statement */`
Operators

- Operators in C are categorized into data access, arithmetic, logical, bitwise, and miscellaneous.

- **Associativity** defines the order of evaluation when operators of same precedence appear in an expression.
  - For example, `a = b = c = 15`, `=` has right-to-left (R→L) associativity.
  - First `c = 15`, then `b = c`, then `a = b` is evaluated.

- **Precedence** defines the order in which calculations involving two or more operators is performed.
  - For example, `x + y * z`, `*` is performed before `+`.

Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition; <code>x + y</code></td>
<td>L→R</td>
<td>4</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction; <code>x - y</code></td>
<td>L→R</td>
<td>4</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication; <code>x * y</code></td>
<td>L→R</td>
<td>3</td>
</tr>
<tr>
<td>/</td>
<td>Division; <code>x / y</code></td>
<td>L→R</td>
<td>3</td>
</tr>
<tr>
<td>%</td>
<td>Remainder (or Modulus); <code>x % y</code></td>
<td>L→R</td>
<td>3</td>
</tr>
<tr>
<td>++</td>
<td>Increment; <code>x++</code></td>
<td>L→R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>x++ means post-increment (increment the value of x by 1 after using its value);</td>
<td>L→R</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>++x means pre-increment (increment the value of x by 1 before using its value)</td>
<td>R→L</td>
<td>2</td>
</tr>
</tbody>
</table>
### Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>Decrement;</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>x--</td>
<td>means post-decrement (decrement the value of x by 1 after using its value);</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>-=x</td>
<td>means pre-decrement (decrement the value of x by 1 before using its value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>x = y means assign the value of y to x</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>+=</td>
<td>x += 5 means x = x + 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>-=</td>
<td>x -= 5 means x = x - 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>*=</td>
<td>x *= 5 means x = x * 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>/=</td>
<td>x /= 5 means x = x / 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>%=</td>
<td>x %= 5 means x = x % 5</td>
<td>R → L</td>
<td>14</td>
</tr>
</tbody>
</table>

### Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Reverse the logical value of a single variable; !x means if the value of x is non-zero, make it zero; and if it is zero, make it one</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than; x &gt; y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than; x &lt; y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to; x &gt;= y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to; x &lt;= y</td>
<td>L → R</td>
<td>6</td>
</tr>
<tr>
<td>==</td>
<td>Equal to; x == y</td>
<td>L → R</td>
<td>7</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to; x != y</td>
<td>L → R</td>
<td>7</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>AND; x &amp;&amp; y means both x and y should be true (non-zero) for result to be true</td>
<td>L → R</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OR; x</td>
</tr>
<tr>
<td>z?x:y</td>
<td>If z is true (non-zero), then the value returned is x, otherwise the value returned is y</td>
<td>R → L</td>
<td>13</td>
</tr>
</tbody>
</table>
## Bitwise Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>Complement; (~x) means All 1s are changed to 0s and 0s to 1s</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>&amp;</td>
<td>AND; (x &amp; y) means (x) AND (y)</td>
<td>L → R</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>OR; (x \mid y) means (x) OR (y)</td>
<td>L → R</td>
<td>10</td>
</tr>
<tr>
<td>^</td>
<td>Exclusive OR; (x \spadesuit y) means (x) OR (y)</td>
<td>L → R</td>
<td>9</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>Left shift; (x &lt;&lt; 4) means shift all bits in (x) four places to the left</td>
<td>L → R</td>
<td>5</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Right shift; (x &gt;&gt; 3) means shift all bits in (x) three places to the right</td>
<td>L → R</td>
<td>5</td>
</tr>
<tr>
<td>&amp;=</td>
<td>(x &amp;= y) means (x = x &amp; y)</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>=</td>
<td>(x</td>
<td>= y) means (x = x</td>
</tr>
<tr>
<td>^=</td>
<td>(x ^= y) means (x = x ^ y)</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>&lt;&lt;=</td>
<td>(x &lt;&lt;= 4) means shift all bits in (x) four places to the left and assign the result to (x)</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>&gt;&gt;=</td>
<td>(x &gt;&gt;= 3) means shift all bits in (x) three places to the right and assign the result to (x)</td>
<td>R → L</td>
<td>14</td>
</tr>
</tbody>
</table>

## Data Access Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x[y])</td>
<td>Access (y^{th}) element of array (x); (y) starts from zero and increases monotonically up to one less than declared size of array</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>(x.y)</td>
<td>Access the member variable (y) of structure (x)</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>(x \leftrightarrow y)</td>
<td>Access the member variable (y) of structure (x)</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>&amp;x</td>
<td>Access the address of variable (x)</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>*(x)</td>
<td>Access the value stored in the storage location (address) pointed to by pointer variable (x)</td>
<td>R → L</td>
<td>2</td>
</tr>
</tbody>
</table>
### Miscellaneous Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>x(y)</td>
<td>Evaluates function x with argument y</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>sizeof (x)</td>
<td>Evaluate the size of variable x in bytes</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>sizeof (type)</td>
<td>Evaluate the size of data type “type” in bytes</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>(type) x</td>
<td>Return the value of x after converting it from declared data type “type”</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>x,y</td>
<td>Sequential operator (x then y)</td>
<td>L → R</td>
<td>15</td>
</tr>
</tbody>
</table>

### Statements

- C program is a combination of statements written between { and } braces
- Each statement performs a set of operations
- Null statement, represented by “;” or empty {} braces, does not perform any operation
- A simple statement is terminated by a semicolon “;”
- Compound statements, called statement block, perform complex operations combining null, simple, and other block statements
Examples of Statements

```c
float a = (x + y) * 10; /* simple statement */
if (sell > cost) /* compound statement
    follows */
{
    profit = sell - cost;
    printf("profit is %d", profit);
}
else /* null statement follows */
{
}
```
### Basic Library Functions for I/O Operations

<table>
<thead>
<tr>
<th>I/O Library Functions</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getch()</code></td>
<td>Inputs a single character (most recently typed) from standard input (usually console).</td>
</tr>
<tr>
<td><code>getche()</code></td>
<td>Inputs a single character from console and echoes (displays) it.</td>
</tr>
<tr>
<td><code>getchar()</code></td>
<td>Inputs a single character from console and echoes it, but requires Enter key to be typed after the character.</td>
</tr>
<tr>
<td><code>putchar()</code> or <code>putch()</code></td>
<td>Outputs a single character on console (screen).</td>
</tr>
<tr>
<td><code>scanf()</code></td>
<td>Enables input of formatted data from console (keyboard). Formatted input data means we can specify the data type expected as input. Format specifiers for different data types are given in Figure 21.6.</td>
</tr>
<tr>
<td><code>printf()</code></td>
<td>Enables obtaining an output in a form specified by programmer (formatted output). Format specifiers are given in Figure 21.6. Newline character &quot;\n&quot; is used in <code>printf()</code> to get the output split over separate lines.</td>
</tr>
<tr>
<td><code>gets()</code></td>
<td>Enables input of a string from keyboard. Spaces are accepted as part of the input string, and the input string is terminated when Enter key is hit. Note that although <code>scanf()</code> enables input of a string of characters, it does not accept multi-word strings (spaces in-between).</td>
</tr>
<tr>
<td><code>puts()</code></td>
<td>Enables output of a multi-word string</td>
</tr>
</tbody>
</table>

### Basic Format Specifiers for `scanf()` and `printf()`

<table>
<thead>
<tr>
<th>Format Specifiers</th>
<th>Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%d</code></td>
<td>integer (short signed)</td>
</tr>
<tr>
<td><code>%u</code></td>
<td>integer (short unsigned)</td>
</tr>
<tr>
<td><code>%ld</code></td>
<td>integer (long signed)</td>
</tr>
<tr>
<td><code>%lu</code></td>
<td>integer (long unsigned)</td>
</tr>
<tr>
<td><code>%f</code></td>
<td>real (float)</td>
</tr>
<tr>
<td><code>%lf</code></td>
<td>real (double)</td>
</tr>
<tr>
<td><code>%c</code></td>
<td>character</td>
</tr>
<tr>
<td><code>%s</code></td>
<td>string</td>
</tr>
</tbody>
</table>
/* A portion of C program to illustrating formatted input and output */

```
int maths, science, english, total;
float percent;
clrscr();    /* A C library function to make the screen clear */
printf( "Maths marks = " ); /* Displays "Maths marks = " */
scanf( "%d", &maths ); /* Accepts entered value and stores in variable "maths" */
printf( "Science marks = " ); /* Displays "Science marks = " on next line because of \n */
scanf( "%d", &science ); /* Accepts entered value and stores in variable "science" */
printf( "English marks = " ); /* Displays "English marks = " on next line because of \n */
scanf( "%d", &english ); /* Accepts entered value and stores in variable "english" */
total = maths + science + english; /* Calculates percentage and stores in variable "percent" */
percent = total/3; /* Displays "Percentage marks obtained = %f", percent); */

printf( "Percentage marks obtained = %f", percent); /* Displays "Percentage marks obtained = 85.66" on next line because of \n */
```

(Continued on next slide)

### Output:

Maths marks = 92  
Science marks = 87  
English marks = 78  
Percentage marks obtained = 85.66
Preprocessor Directives

Preprocessor is a program that prepares a program for the C compiler.

Three common preprocessor directives in C are:

- `#include` – Used to look for a file and place its contents at the location where this preprocessor directives is used.
- `#define` – Used for macro expansion.
- `#ifdef..#endif` – Used for conditional compilation of segments of a program.

Examples of Preprocessor Directives

```c
#include <stdio.h>
#define PI 3.1415
#define AND &&
#define ADMIT printf("The candidate can be admitted");
#ifdef WINDOWS
  ...
  Code specific to windows operating system
  ...
#else
  ...
  Code specific to Linux operating system
  ...
#endif

else
  ...

endif
```

Code common to both operating systems.
### Standard Preprocessor Directives in C

<table>
<thead>
<tr>
<th>Preprocessor Directive</th>
<th>Meaning</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Null directive</td>
<td></td>
</tr>
<tr>
<td>#error message</td>
<td>Prints message when processed</td>
<td></td>
</tr>
<tr>
<td>#line linenumber filename</td>
<td>Used to update code line number and filename</td>
<td></td>
</tr>
<tr>
<td>#pragma name</td>
<td>Compiler specific settings</td>
<td></td>
</tr>
<tr>
<td>#include filename</td>
<td>Includes content of another file</td>
<td></td>
</tr>
<tr>
<td>#define macro/string</td>
<td>Define a macro or string substitution</td>
<td></td>
</tr>
<tr>
<td>#undef macro</td>
<td>Removes a macro definition</td>
<td></td>
</tr>
<tr>
<td>#if expr</td>
<td>Includes following lines if expr is true</td>
<td>Macro</td>
</tr>
<tr>
<td>#elif expr</td>
<td>Includes following lines if expr is true</td>
<td></td>
</tr>
<tr>
<td>#else</td>
<td>Handles otherwise conditions of #if</td>
<td></td>
</tr>
<tr>
<td>#endif</td>
<td>Closes #if or #else block</td>
<td></td>
</tr>
<tr>
<td>#ifdef macro</td>
<td>Includes following lines if macro is defined</td>
<td></td>
</tr>
<tr>
<td>#ifndef macro</td>
<td>Includes following lines if macro is not defined</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>String forming operator</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Token pasting operator</td>
<td></td>
</tr>
<tr>
<td>defined</td>
<td>same as #ifdef</td>
<td>Operators</td>
</tr>
</tbody>
</table>

### Pointers

- C pointers allow programmers to directly access memory addresses where variables are stored.
- Pointer variable is declared by adding a ‘*’ symbol before the variable name while declaring it.
- If `p` is a pointer to a variable (e.g. `int i, *p = i;`)
  - Using `p` means address of the storage location of the pointed variable
  - Using `*p` means value stored in the storage location of the pointed variable
- Operator ‘&’ is used with a variable to mean variable’s address, e.g. `&i` gives address of variable `i`
Illustrating Pointers Concept

Address of i = 1000
Value of i = 62

```c
int i = 62;
int *p;
int j;
p = &i; /* p becomes 1000 */
j = *p; /* j becomes 62 */
j = 0; /* j becomes zero */
j = *(&i); /* j becomes 62 */
```

Array

- Collection of fixed number of elements in which all elements are of the same data type
- Homogeneous, linear, and contiguous memory structure
- Elements can be referred to by using their subscript or index position that is monotonic in nature
- First element is always denoted by subscript value of 0 (zero), increasing monotonically up to one less than declared size of array
- Before using an array, its type and dimension must be declared
- Can also be declared as multi-dimensional such as Matrix2D[10][10]
Illustrating Arrays Concept

(a) An array of integers having 6 elements
(b) An array of real numbers having 4 elements
(c) An array of characters having 6 elements

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>100</td>
<td>92</td>
<td>1008</td>
<td>1006</td>
<td>82</td>
<td>1004</td>
</tr>
<tr>
<td>1006</td>
<td>66</td>
<td>82</td>
<td>1008</td>
<td>250.00</td>
<td>66</td>
<td>1004</td>
</tr>
<tr>
<td>1004</td>
<td>84</td>
<td>1008</td>
<td>155.50</td>
<td>84</td>
<td>1004</td>
<td>1000</td>
</tr>
<tr>
<td>1000</td>
<td>45</td>
<td>1000</td>
<td>10.25</td>
<td>45</td>
<td>1000</td>
<td>1005</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

int marks[6]; float price[4]; char city[6];

Each element being an int occupies 2 bytes
marks[0] = 45
marks[1] = 84
marks[2] =
marks[3] =
marks[4] =
marks[5] = 92

Each element being a float occupies 4 bytes
price[0] = 82.75
price[1] = 155.50
price[2] =
price[3] = 10.25

Each element being a char occupies 1 byte
city[0] = ‘B’
city[1] = ‘O’
city[2] =
city[3] =
city[4] =
city[5] = ‘Y’

String

- One-dimensional array of characters terminated by a null character (‘\0’)
- Initialized at declaration as
  - char name[] = “PRADEEP”;
- Individual elements can be accessed in the same way as we access array elements such as name[3] = ‘D’
- Strings are used for text processing
- C provides a rich set of string handling library functions
Library Functions for String Handling

<table>
<thead>
<tr>
<th>Library Function</th>
<th>Used To</th>
</tr>
</thead>
<tbody>
<tr>
<td>strlen</td>
<td>Obtain the length of a string</td>
</tr>
<tr>
<td>strlwr</td>
<td>Convert all characters of a string to lowercase</td>
</tr>
<tr>
<td>strupr</td>
<td>Convert all characters of a string to uppercase</td>
</tr>
<tr>
<td>strcat</td>
<td>Concatenate (append) one string at the end of another</td>
</tr>
<tr>
<td>strncat</td>
<td>Concatenate only first n characters of a string at the end of another</td>
</tr>
<tr>
<td>strncpy</td>
<td>Copy a string into another</td>
</tr>
<tr>
<td>strncpy</td>
<td>Copy only the first n characters of a string into another</td>
</tr>
<tr>
<td>strcmp</td>
<td>Compare two strings</td>
</tr>
<tr>
<td>stricmp</td>
<td>Compare two strings without regard to case</td>
</tr>
<tr>
<td>strncpy</td>
<td>Compare only first n characters of two strings</td>
</tr>
<tr>
<td>strncpy</td>
<td>Compare two strings without regard to case</td>
</tr>
<tr>
<td>strdup</td>
<td>Duplicate a string</td>
</tr>
<tr>
<td>strchr</td>
<td>Find first occurrence of a given character in a string</td>
</tr>
<tr>
<td>strchr</td>
<td>Find last occurrence of a given character in a string</td>
</tr>
<tr>
<td>strstr</td>
<td>Find first occurrence of a given string in another string</td>
</tr>
<tr>
<td>strset</td>
<td>Set all characters of a string to a given character</td>
</tr>
<tr>
<td>strnset</td>
<td>Set first n characters of a string to a given character</td>
</tr>
<tr>
<td>strrev</td>
<td>Reverse a string</td>
</tr>
</tbody>
</table>

User Defined Data Types (UDTs)

- UDT is composite data type whose composition is not include in language specification
- Programmer declares them in a program where they are used
- Two types of UDTs are:
  - Structure
  - Union
Structure

- UDT containing a number of data types grouped together
- Constituents data types may or may not be of different types
- Has continuous memory allocation and its minimum size is the sum of sizes of its constituent data types
- All elements (member variable) of a structure are publicly accessible
- Each member variable can be accessed using "." (dot) operator or pointer (EmpRecord.EmpID or EmpRecord → EmpID)
- Can have a pointer member variable of its own type, which is useful in crating linked list and similar data structures

Structure (Examples)

```c
struct Employee
{
    int EmpID;
    char EmpName[20];
};

Struct Employee EmpRecord;
Struct Employee *pEmpRecord = &EmpRecord
```

```c
struct Employee
{
    int EmpID;
    char EmpName[20];
};

EmpRecord;
```
**Union**

- UDT referring to same memory location using several data types
- Mathematical union of all constituent data types
- Each data member begins at the same memory location
- Minimum size of a union variable is the size of its largest constituent data types
- Each member variable can be accessed using "," (dot) operator
- Section of memory can be treated as a variable of one type on one occasion, and of another type on another occasion

```c
union Num
{
    int intNum;
    unsigned unsNum;
};

unionNum Num Number;
```
### Difference Between Structure and Union

- Both group a number of data types together
- Structure allocates different memory space contiguously to different data types in the group
- Union allocates the same memory space to different data types in the group

### Control Structures

- **Control structures** (branch statements) are decision points that control the flow of program execution based on:
  - Some condition test (conditional branch)
  - Without condition test (unconditional branch)
- Ensure execution of other statement/block or cause skipping of some statement/block
Conditional Branch Statements

- **if** is used to implement simple one-way test. It can be in one of the following forms:
  - if..stmt
  - if..stmt1..else..stmt2
  - if..stmt1..else..if..stmtn

- **switch** facilitates multi-way condition test and is very similar to the third *if* construct when primary test object remains same across all condition tests

Examples of “if” Construct

- **if** (i <= 0)
  
  i++;

- **if** (i <= 0)
  
  else
  
  j++;

- **if** (i <= 0)
  
  i++;
  
  else if (i >= 0)
  
  j++;
  
  else
  
  k++;
Example of “switch” Construct

```c
switch(ch) {
    case 'A':
    case 'B':
    case 'C':
        printf("Capital");
        break;
    case 'a':
    case 'b':
    case 'c':
        printf("Small");
        break;
    default:
        printf("Not cap or small");
}
```

Same thing can be written also using `if` construct as:

```c
if (ch == 'A' || ch == 'B' || ch == 'C')
    printf("Capital");
else if (ch == 'a' || ch == 'b' || ch == 'c')
    printf("Small");
else
    printf("Not cap or small");
```

Unconditional Branch Statements

- **Break**: Causes unconditional exit from `for`, `while`, `do`, or `switch` constructs. Control is transferred to the statement immediately outside the block in which `break` appears.

- **Continue**: Causes unconditional transfer to next iteration in a `for`, `while`, or `do` construct. Control is transferred to the statement beginning the block in which `continue` appears.

- **Goto label**: Causes unconditional transfer to statement marked with the label within the function.

(Continued on next slide)
Unconditional Branch Statements

(Continued from previous slide)

- Return [value/variable]: Causes immediate termination of function in which it appears and transfers control to the statement that called the function. Optionally, it provides a value compatible to the function’s return data type.

Loop Structures

- Loop statements are used to repeat the execution of statement or blocks

- Two types of loop structures are:
  - **Pretest**: Condition is tested before each iteration to check if loop should occur
  - **Posttest**: Condition is tested after each iteration to check if loop should continue (at least, a single iteration occurs)
Pretest Loop Structures

**for**: It has three parts:
- **Initializer** is executed at start of loop
- **Loop condition** is tested before iteration to decide whether to continue or terminate the loop
- **Incrementor** is executed at the end of each iteration

**While**: It has a **loop condition** only that is tested before each iteration to decide whether to continue to terminate the loop

Examples of "for" and "while" Constructs

```c
§ for (i=0; i < 10; i++)
    printf("i = %d", i);

§ while (i < 10)
    {
        printf("i = %d", i);
        i++;
    }
```
Posttest Loop Construct
“do...while”

- It has a loop condition only that is tested after each iteration to decide whether to continue with next iteration or terminate the loop.

- Example of do...while is:

```
do {
    printf("i = %d", i);
    i++;
}while (i < 10);
```

Functions

- Functions (or subprograms) are building blocks of a program.
- All functions must be declared and defined before use.
- Function declaration requires `functionname, argument list, and return type`.
- Function definition requires coding the body or logic of function.
- Every C program must have a `main` function. It is the entry point of the program.
Example of a Function

```c
int myfunc( int Val, int ModVal )
{
    unsigned temp;
    temp = Val % ModVal;
    return temp;
}
```

This function can be called from any other place using simple statement:

```c
int n = myfunc(4, 2);
```

Sample C Program (Program-1)

```c
/* Program to accept an integer from console and to display whether the number is even or odd */

#include <stdio.h>

void main()
{
    int number, remainder;
    clrscr(); /* clears the console screen */
    printf("Enter an integer: ");
    scanf("%d", &number);
    remainder = number % 2;
    if (remainder == 0)
        printf("\n The given number is even");
    else
        printf("\n The given number is odd");
    getch();
}
```
Program to accept an integer in the range 1 to 7 (both inclusive) from console and to display the corresponding day (Monday for 1, Tuesday for 2, Wednesday for 3, and so on). If the entered number is out of range, the program displays a message saying that:

```c
#include <stdio.h>
#include <conio.h>

#define MON printf("Entered number is 1 hence day is MONDAY");
#define TUE printf("Entered number is 2 hence day is TUESDAY");
#define WED printf("Entered number is 3 hence day is WEDNESDAY");
#define THU printf("Entered number is 4 hence day is THURSDAY");
#define FRI printf("Entered number is 5 hence day is FRIDAY");
#define SAT printf("Entered number is 6 hence day is SATURDAY");
#define SUN printf("Entered number is 7 hence day is SUNDAY");
#define OTH printf("Entered number is out of range");

void main()
{
    int day;
    clrscr();
    printf("Enter an integer in the range 1 to 7");
    scanf("%d", &day);
    switch(day)
    {
        Case 1: MON; break;
        Case 2: TUE; break;
        Case 3: WED; break;
        Case 4: THU; break;
        Case 5: FRI; break;
        Case 6: SAT; break;
        Case 7: SUN; break;
        default: OTH;
    }
    getch();
}
```
Sample C Program (Program-3)

/* Program to accept the radius of a circle from console and to calculate
 and display its area and circumference */

#include <stdio.h>
#include <conio.h>
define PI 3.1415

void main()
{
    float radius, area, circum;
clearscr();
printf("Enter the radius of the circle: ");
scanf("%f", &radius);
area = PI * radius * radius;
circum = 2 * PI * radius;
printf("Area and circumference of the circle are %f
   and %f respectively", area, circum);
getch();
}

(Continued on next slide)

Sample C Program (Program-4)

/* Program to accept a string from console and to display the number of
vowels in the string */

#include <stdio.h>
#include <conio.h>
#include <string.h>

void main()
{
    char input_string[50]; /* maximum 50 characters */
    int len;
    int i = 0, cnt = 0;
clearscr();
printf("Enter a string of less than 50 characters: \n");
gets (input_string);
len = strlen (input_string);
for (i = 0; i < len; i++)
{
    switch (input_string[i])

(Continued on next slide)
Sample C Program (Program-4)

```c
{ 
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
    case 'A':
    case 'E':
    case 'I':
    case 'O':
    case 'U':
        cnt++
    }
    printf("\n Number of vowels in the string are: %d", cnt);
    getch();
}
```

Sample C Program (Program-5)

```c
/* Program to illustrate use of a user defined function. The program initializes an array of n elements from 0 to n-1 and then calculates and prints the sum of the array elements. In this example n = 10 */
#include <stdio.h>
#define SIZE 10
int ArrSum(int *p, int n)
{
    int s, tot = 0;
    for(s = 0; s < n; s++)
    {
        tot += *p;
        p++;
    }
    return tot;
}
int main()
{
    int i = 0, sum = 0;
    int okr[SIZE] = {0};
    while(i < SIZE)
    {
        nArr[i++] = i;
    }
    sum = ArrSum(okr, SIZE);
    printf("Sum of 0 to 9 = \%d\n", sum);
    return 0;
}
```
Key Words/Phrases

- Arithmetic operators
- Arrays
- Assignment operators
- Bit-level manipulation
- Bitwise operators
- Branch statement
- Character set
- Comment statement
- Compound statement
- Conditional branch
- Conditional compilation
- Constants
- Control structures
- Format specifiers
- Formatted I/O
- Function
- Keywords
- Library functions
- Logical operators
- Loop structures
- Macro expansion
- Main function
- Member element
- Null statement
- Operator associativity
- Operator precedence
- Pointer
- Posttest loop
- Preprocessor directives
- Pretest loop
- Primitive data types
- Reserved words
- Simple statement
- Statement block
- Strings
- Structure data type
- Unconditional branch
- Union data type
- User-defined data types
- Variable name
- Variable type declaration
- Variables
Chapter 21
Introduction to C
Programming Language

Learning Objectives

In this chapter you will learn about:

- Features of C
- Various constructs and their syntax
- Data types and operators in C
- Control and Loop Structures in C
- Functions in C
- Writing programs in C

Features

- Reliable, simple, and easy to use
- Has virtues of high-level programming language with
efficiency of assembly language
- Supports modular and structured programming concepts
- Supports a rich library of functions
- Supports pointers with pointer operations
- Supports low-level memory and device access
- Small and concise language
- Standardized by several international standards body
## C Character Set

<table>
<thead>
<tr>
<th>Category</th>
<th>Valid Characters</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uppercase alphabets</td>
<td>A, B, C, ..., Z</td>
<td>26</td>
</tr>
<tr>
<td>Lowercase alphabets</td>
<td>a, b, c, ..., z</td>
<td>26</td>
</tr>
<tr>
<td>Digits</td>
<td>0, 1, 2, ..., 9</td>
<td>10</td>
</tr>
<tr>
<td>Special characters</td>
<td>~ ! ` # $ % ^ &amp; ( ) _ - + * = &lt; &gt;</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>93</td>
</tr>
</tbody>
</table>

### Constants

- Constant is a value that never changes
- Three primitive types of constants supported in C are:
  - Integer
  - Real
  - Character

### Rules for Constructing Integer Constants

- Must have at least one digit
- + or – sign is optional
- No special characters (other than + and – sign) are allowed
- Allowable range is:
  - -32768 to 32767 for integer and short integer constants (16 bits storage)
  - -2147483648 to 2147483647 for long integer constants (32 bits storage)
- Examples are: 8, +17, -6
Rules for Constructing Real Constants in Exponential Form

- Has two parts - mantissa and exponent - separated by 'e' or 'E'
- Mantissa part is constructed by the rules for constructing real constants in fractional form
- Exponent part is constructed by the rules for constructing integer constants
- Allowable range is -3.4e38 to 3.4e38
- Examples are: 8.6e5, +4.3E-8, -0.1e+4

Rules for Constructing Character Constants

- Single character from C character set
- Enclosed within single inverted comma (also called single quote) punctuation mark
- Examples are: 'A', 'a', '8', '%'

Variables

- Entity whose value may vary during program execution
- Has a name and type associated with it
- Variable name specifies programmer given name to the memory area allocated to a variable
- Variable type specifies the type of values a variable can contain
- Example: In i = i + 5, i is a variable
Rules for Constructing Variables

- Can have 1 to 31 characters
- Only alphabets, digits, and underscore (as in last_name) characters are allowed
- Names are case sensitive (nNum and nNUM are different)
- First character must be an alphabet
- Underscore is the only special character allowed
- Keywords cannot be used as variable names
- Examples are: I saving_2007 ArnSum

Data Types Used for Variable Type Declaration

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Minimum Storage Allocated</th>
<th>Used for Variables that can contain</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>2 bytes (16 bits)</td>
<td>integer constants in the range -32768 to 32767</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes (16 bits)</td>
<td>integer constants in the range -32768 to 32767</td>
</tr>
<tr>
<td>long</td>
<td>4 bytes (32 bits)</td>
<td>integer constants in the range -2147483648 to 2147483647</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes (32 bits)</td>
<td>real constants with minimum 6 decimal digits precision</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes (64 bits)</td>
<td>real constants with minimum 10 decimal digits precision</td>
</tr>
<tr>
<td>char</td>
<td>1 byte (8 bits)</td>
<td>character constants</td>
</tr>
<tr>
<td>enum</td>
<td>2 bytes (16 bits)</td>
<td>values in the range -32768 to 32767</td>
</tr>
<tr>
<td>void</td>
<td>No storage allocated</td>
<td>no value assigned</td>
</tr>
</tbody>
</table>

Variable Type Declaration Examples

```c
int count;
short index;
long principle;
float area;
double radius;
char c;
```
### Standard Qualifiers in C

<table>
<thead>
<tr>
<th>Category</th>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifetime</td>
<td>auto</td>
<td>Temporary variable</td>
</tr>
<tr>
<td></td>
<td>register</td>
<td>Attempt to store in processor register, fast access</td>
</tr>
<tr>
<td></td>
<td>static</td>
<td>Permanent, initialized</td>
</tr>
<tr>
<td></td>
<td>extern</td>
<td>Permanent, initialized but declaration statements may be modified</td>
</tr>
<tr>
<td>Modifiability</td>
<td>const</td>
<td>Cannot be modified once created</td>
</tr>
<tr>
<td></td>
<td>volatile</td>
<td>May be modified by factors outside program</td>
</tr>
<tr>
<td>Sign</td>
<td>signed</td>
<td>+ or -</td>
</tr>
<tr>
<td></td>
<td>unsigned</td>
<td>+ only</td>
</tr>
<tr>
<td>Size</td>
<td>short</td>
<td>16 bits</td>
</tr>
<tr>
<td></td>
<td>long</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

### Lifetime and Visibility Scopes of Variables

1. Lifetime of all variables (except those declared as static) is same as that of function or statement block it is declared in.
2. Lifetime of variables declared in global scope and static is same as that of the program.
3. Variable is visible and accessible in the function or statement block it is declared in.
4. Global variables are accessible from anywhere in program.
5. Variable name must be unique in its visibility scope.
6. Local variable has access precedence over global variable of same name.

### Keywords

1. Keywords (or reserved words) are predefined words whose meanings are known to C compiler.
2. C has 32 keywords.
3. Keywords cannot be used as variable names.

- auto
- break
- case
- char
- const
- continue
- default
- do
- double
- else
- enum
- float
- for
- goto
- int
- long
- register
- return
- short
- signed
- sizeof
- struct
- switch
- typedef
- union
- unsigned
- void
- while
- volatile
- void
- static
- inline
- typedef
- volatile
- switch
- typedef
- volatile
- void
- static
- inline
Comments

- Comments are enclosed within `*/` and `*/`
- Comments are ignored by the compiler
- Comment can also split over multiple lines
- Example: `//` This is a comment statement `*/`

Operators

- Operators in C are categorized into data access, arithmetic, logical, bitwise, and miscellaneous
- **Associativity** defines the order of evaluation when operators of same precedence appear in an expression
  - `a = b = c = 15`, `=` has `R→L` associativity
  - First `c = 15`, then `b = c`, then `a = b` is evaluated
- **Precedence** defines the order in which calculations involving two or more operators is performed
  - `x + y * z`, `*` is performed before `+`

Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>Increment; <code>x++</code> means post-increment (increment the value of <code>x</code> by <code>1</code> after using its value); <code>x+=1</code> means pre-increment (increment the value of <code>x</code> by <code>1</code> before using its value)</td>
<td><code>R→L</code></td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction; <code>x - y</code></td>
<td><code>L→R</code></td>
<td>4</td>
</tr>
<tr>
<td>/</td>
<td>Division; <code>x / y</code></td>
<td><code>L→R</code></td>
<td>3</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication; <code>x * y</code></td>
<td><code>L→R</code></td>
<td>2</td>
</tr>
<tr>
<td>%</td>
<td>Remainder (or Modulus); <code>x % y</code></td>
<td><code>L→R</code></td>
<td>3</td>
</tr>
</tbody>
</table>
### Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>-=</td>
<td>x -= 5 means x = x - 5</td>
<td>L → R</td>
<td>14</td>
</tr>
<tr>
<td>/=</td>
<td>x /= 5 means x = x / 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>*=</td>
<td>x *= 5 means x = x * 5</td>
<td>L → R</td>
<td>14</td>
</tr>
<tr>
<td>-=</td>
<td>x -= 5 means x = x - 5</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>%=</td>
<td>x %= 5 means x = x % 5</td>
<td>R → L</td>
<td>14</td>
</tr>
</tbody>
</table>

### Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>If z is true (non-zero), then the value returned is x, otherwise the value returned is y</td>
<td>R → L</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>x &amp;&amp; y means both x and y should be true (non-zero) for result to be true</td>
<td>R → L</td>
<td>7</td>
</tr>
<tr>
<td>!=</td>
<td>x != y means x is not equal to y</td>
<td>R → L</td>
<td>7</td>
</tr>
<tr>
<td>==</td>
<td>x == y means x is equal to y</td>
<td>R → L</td>
<td>6</td>
</tr>
<tr>
<td>&lt;=</td>
<td>x &lt;= y means x is less than or equal to y</td>
<td>R → L</td>
<td>6</td>
</tr>
<tr>
<td>&gt;=</td>
<td>x &gt;= y means x is greater than or equal to y</td>
<td>R → L</td>
<td>6</td>
</tr>
<tr>
<td>&lt;</td>
<td>x &lt; y means x is less than y</td>
<td>R → L</td>
<td>6</td>
</tr>
<tr>
<td>&gt;</td>
<td>x &gt; y means x is greater than y</td>
<td>R → L</td>
<td>6</td>
</tr>
<tr>
<td>!x</td>
<td>Reverse the logical value of a single variable; !x means if the value of x is non-zero, make it zero; and if it is zero, make it one</td>
<td>R → L</td>
<td>4</td>
</tr>
</tbody>
</table>

### Bitwise Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>~</td>
<td>~x means All 1s are changed to 0s and 0s to 1s</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>&amp;</td>
<td>x &amp; y means x AND y</td>
<td>L → R</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>y means x OR y</td>
</tr>
<tr>
<td>^</td>
<td>x ^ y means x XOR y</td>
<td>L → R</td>
<td>9</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>x &lt;&lt; 4 means shift all bits in x four places to the left</td>
<td>L → R</td>
<td>5</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>x &gt;&gt; 3 means shift all bits in x three places to the right</td>
<td>L → R</td>
<td>5</td>
</tr>
<tr>
<td>&amp;=</td>
<td>x &amp;= y means x = x &amp; y</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>=</td>
<td>x</td>
<td>= y means x = x</td>
</tr>
<tr>
<td>^=</td>
<td>x ^= y means x = x ^ y</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>&lt;&lt;=</td>
<td>x &lt;&lt;= 4 means shift all bits in x four places to the left and assign the result to x</td>
<td>R → L</td>
<td>14</td>
</tr>
<tr>
<td>&gt;&gt;=</td>
<td>x &gt;&gt;= 3 means shift all bits in x three places to the right and assign the result to x</td>
<td>R → L</td>
<td>14</td>
</tr>
</tbody>
</table>
Data Access Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>x[y]</td>
<td>Access yth element of array x; y starts from zero and increases monotonically up to one less than declared size of array</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>.p</td>
<td>Access the member variable p of structure s</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>s-&gt;y</td>
<td>Access the member variable y of structure s</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>&amp;x</td>
<td>Access the address of variable s</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>*x</td>
<td>Access the value stored in the storage location (address) pointed to by pointer variable x</td>
<td>R → L</td>
<td>2</td>
</tr>
</tbody>
</table>

Miscellaneous Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning with Example</th>
<th>Associativity</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>h(x)</td>
<td>Evaluate function h with argument x</td>
<td>L → R</td>
<td>1</td>
</tr>
<tr>
<td>sizeof(x)</td>
<td>Evaluate the size of variable x in bytes</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>sizeof(type)</td>
<td>Evaluate the size of data type &quot;type&quot; in bytes</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>(type)x</td>
<td>Return the value of x after converting it from declared data type of variable x to the new data type &quot;type&quot;</td>
<td>R → L</td>
<td>2</td>
</tr>
<tr>
<td>n,y</td>
<td>Sequential operator (n then y)</td>
<td>L → R</td>
<td>15</td>
</tr>
</tbody>
</table>

Statements

6. C program is a combination of statements written between ( and ) braces
6. Each statement performs a set of operations
6. Null statement, represented by ";" or empty {} braces, does not perform any operation
6. A simple statement is terminated by a semicolon ";"
6. Compound statements, called statement block, perform complex operations combining null, simple, and other block statements
Examples of Statements

\[ a = (x + y) \times 10; \quad /\!* \text{simple statement} */\]
\[ a \text{ if } (sell > cost) /\!* \text{compound statement} */\]
\[ \begin{cases} 
\text{profit} = sell - cost; \\
\text{printf}("\text{profit is } \%d\), \text{profit}); \\
\text{else} \quad /\!* \text{null statement follows} */
\end{cases} \]

Simple I/O Operations

- C has no keywords for I/O operations
- Provides standard library functions for performing all I/O operations

Basic Library Functions for I/O Operations

<table>
<thead>
<tr>
<th>I/O Library Function</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>puts()</td>
<td>Outputs a single character (most recently typed) from standard output (usually screen).</td>
</tr>
<tr>
<td>gets()</td>
<td>Enables output of a multi-word string.</td>
</tr>
<tr>
<td>printf()</td>
<td>Enables output of a form specified by programmer (format string) and data members.</td>
</tr>
<tr>
<td>scanf()</td>
<td>Enables input of a multi-word string.</td>
</tr>
<tr>
<td>putchar() / putch()</td>
<td>Enables single character output (screen).</td>
</tr>
<tr>
<td>getchar() / getch()</td>
<td>Enables single character input.</td>
</tr>
</tbody>
</table>

Meanings

- I/O Library: Input/Output Library
- Function: Various functions for input/output operations
- Meanings: Descriptions of what each function does
Basic Format Specifiers for `scanf()` and `printf()`

<table>
<thead>
<tr>
<th>Format Specifiers</th>
<th>Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>%d</td>
<td>integer (short signed)</td>
</tr>
<tr>
<td>%u</td>
<td>integer (short unsigned)</td>
</tr>
<tr>
<td>%ld</td>
<td>integer (long signed)</td>
</tr>
<tr>
<td>%lu</td>
<td>integer (long unsigned)</td>
</tr>
<tr>
<td>%f</td>
<td>real (float)</td>
</tr>
<tr>
<td>%lf</td>
<td>real (double)</td>
</tr>
<tr>
<td>%c</td>
<td>character</td>
</tr>
<tr>
<td>%s</td>
<td>string</td>
</tr>
</tbody>
</table>

Formatted I/O Example

```c
// A C program to illustrate formatted input and output

#include <stdio.h>

int main() {
    int maths, science, english, total;
    float percent;
    clrscr();    /* A C library function to make the screen clear
    printf( "Maths marks = ");    /* Displays "Maths marks = 
    scanf( "%d", &maths);    /* Accepts entered value and stores in variable "maths"
    printf( "Science marks = ");    /* Displays "Science marks = " on next line because of 
    scanf( "%d", &science);    /* Accepts entered value and stores in variable "science"
    printf( "English marks = ");    /* Displays "English marks = " on next line because of 
    scanf( "%d", &english);    /* Accepts entered value and stores in variable "english"
    total = maths + science + english;
    percent = total/3;    /* Calculates percentage and stores in variable "percent"
    printf( "Percentage marks obtained = %f", percent);    /* Displays "Percentage marks obtained = 85.66" on next line because of "'
    return 0;
} /* End of main()*/
```

Output:

Maths marks = 92
Science marks = 87
English marks = 78
Percentage marks obtained = 85.66

(Continued from previous slide..)
Preprocessor Directives

Preprocessor is a program that prepares a program for the C compiler.

Three common preprocessor directives in C are:

- **#include** - Used to look for a file and place its contents at the location where this preprocessor directives is used

- **#define** - Used for macro expansion

- **#ifdef...#endif** - Used for conditional compilation of segments of a program

Examples of Preprocessor Directives

```c
#include <stdio.h>
#define PI 3.1415
#define AND &&
#define ADMIT printf("The candidate can be admitted");
#if defined WINDOWS
    ...
    Code specific to windows operating system
    ...
#endif
 else
    ...
    Code specific to Linux operating system
    ...
#endif
 Code common to both operating systems
```

Standard Preprocessor Directives in C

<table>
<thead>
<tr>
<th>Preprocessor Directive</th>
<th>Meaning</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Null directive</td>
<td>Simple</td>
</tr>
<tr>
<td>ifndef</td>
<td>Finds filename when processed</td>
<td>Simple</td>
</tr>
<tr>
<td>define</td>
<td>Define a macro or string substitution</td>
<td>Macro</td>
</tr>
<tr>
<td>include</td>
<td>Include a file or string substitution</td>
<td>File</td>
</tr>
<tr>
<td>ifdef</td>
<td>If macro is defined</td>
<td>Macro</td>
</tr>
<tr>
<td>ifndef</td>
<td>If macro is not defined</td>
<td>Macro</td>
</tr>
<tr>
<td>if</td>
<td>If expression is true</td>
<td>Conditional</td>
</tr>
<tr>
<td>else</td>
<td>If expression is false</td>
<td>Conditional</td>
</tr>
<tr>
<td>elif</td>
<td>If expression is true</td>
<td>Conditional</td>
</tr>
<tr>
<td>#if expr</td>
<td>If expression is true</td>
<td>Macro</td>
</tr>
<tr>
<td>#ifdef macro</td>
<td>Include following lines if macro is defined</td>
<td>Macro</td>
</tr>
<tr>
<td>#ifndef macro</td>
<td>Include following lines if macro is not defined</td>
<td>Macro</td>
</tr>
<tr>
<td>#</td>
<td>String forming operator</td>
<td>оператор</td>
</tr>
<tr>
<td>define</td>
<td>Define as &quot;define&quot;</td>
<td>Simple</td>
</tr>
</tbody>
</table>

Pointers

- C pointers allow programmers to directly access memory addresses where variables are stored.
- Pointer variable is declared by adding a ‘*’ symbol before the variable name while declaring it.
- If p is a pointer to a variable (e.g., int i, *p = i;)
  - Using *p means address of the storage location of the pointed variable.
  - Using *p means value stored in the storage location of the pointed variable.
- Operator ‘&’ is used with a variable to mean variable’s address, e.g., &i gives address of variable i.

Illustrating Pointers Concept

- Address of i = 1000
- Value of i = 62
- int i = 62;
- int *p;
- j = *p;
- j = 0;
- j = (*(&i));

Array

- Collection of fixed number of elements in which all elements are of the same data type.
- Homogeneous, linear, and contiguous memory structure.
- Elements can be referred to by using their subscript or index position that is monotonic in nature.
- First element is always denoted by subscript value of 0 (zero), increasing monotonically up to one less than declared size of array.
- Before using an array, its type and dimension must be declared.
- Can also be declared as multi-dimensional such as Matrix2D[10][10].
Illustrating Arrays Concept

<table>
<thead>
<tr>
<th>int</th>
<th>marks[6]</th>
<th>Each element being an int occupies 2 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>marks[0] = 45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>marks[1] = 84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>marks[5] = 92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>float</th>
<th>price[4]</th>
<th>Each element being a float occupies 4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>price[0] = 82.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>price[1] = 155.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>price[3] = 10.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>char</th>
<th>city[6]</th>
<th>Each element being a char occupies 1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>city[0] = 'B'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>city[1] = 'O'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>city[5] = 'Y'</td>
</tr>
</tbody>
</table>

String

- One-dimensional array of characters terminated by a null character ('\0')
- Initialized at declaration as:
  ```
  char name[] = "PRADEEP";
  ```
- Individual elements can be accessed in the same way as we access array elements such as name[3] = 'D'
- Strings are used for text processing
- C provides a rich set of string handling library functions

Library Functions for String Handling

<table>
<thead>
<tr>
<th>Library Function</th>
<th>Used For</th>
</tr>
</thead>
<tbody>
<tr>
<td>strlen</td>
<td>Used to determine the length of a string</td>
</tr>
<tr>
<td>strcpy</td>
<td>Copy a string into another</td>
</tr>
<tr>
<td>strncpy</td>
<td>Copy only the first n characters of a string</td>
</tr>
<tr>
<td>strcmp</td>
<td>Compare two strings</td>
</tr>
<tr>
<td>strncmp</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcmp</td>
<td>Compare two strings</td>
</tr>
<tr>
<td>strncat</td>
<td>Concatenate only the first n characters of a string</td>
</tr>
<tr>
<td>strcat</td>
<td>Concatenate (append) one string at the end of another</td>
</tr>
<tr>
<td>strupr</td>
<td>Convert all characters to uppercase</td>
</tr>
<tr>
<td>strwr</td>
<td>Convert all characters to lowercase</td>
</tr>
<tr>
<td>strrev</td>
<td>Reverse a string</td>
</tr>
<tr>
<td>strstr</td>
<td>Find first occurrence of a given string in another</td>
</tr>
<tr>
<td>strrchr</td>
<td>Find last occurrence of a given character in a string</td>
</tr>
<tr>
<td>strnstrr</td>
<td>Find last occurrence of a given substring in another</td>
</tr>
<tr>
<td>strcasrmp</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasncmp</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp2</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp4</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp6</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp8</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp10</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp12</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp14</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp16</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp18</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp20</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp22</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp24</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp26</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp28</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp30</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp32</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp34</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp36</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp38</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp40</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp42</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp44</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp46</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp48</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp50</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp52</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp54</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp56</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp58</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp60</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp62</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp64</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp66</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp68</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp70</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp72</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp74</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp76</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp78</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp80</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp82</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp84</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp86</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp88</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp90</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp92</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp94</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp96</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp98</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp100</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp102</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp104</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp106</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp108</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp110</td>
<td>Compare only the first n characters of two strings</td>
</tr>
<tr>
<td>strcasmp112</td>
<td>Compare only the first n characters of two string...</td>
</tr>
</tbody>
</table>
User Defined Data Types (UDTs)

- UDT is a composite data type whose composition is not included in language specification
- Programmer declares them in a program where they are used
- Two types of UDTs are:
  - Structure
  - Union

Structure

- UDT containing a number of data types grouped together
- Constituents data types may or may not be of different types
- Has continuous memory allocation and its minimum size is the sum of sizes of its constituent data types
- All elements (member variable) of a structure are publicly accessible
- Each member variable can be accessed using "." (dot) operator or pointer (EmpRecord.EmpID or EmpRecord ∗ EmpID)
- Can have a pointer member variable of its own type, which is useful in creating linked list and similar data structures

Structure (Examples)

```c
struct Employee {
    int EmpID;
    char EmpName[20];
};

struct Employee EmpRecord;
Struct Employee ∗pEmpRecord = &EmpRecord
```
Union

- UDT referring to same memory location using several data types
- Mathematical union of all constituent data types
- Each data member begins at the same memory location
- Minimum size of a union variable is the size of its largest constituent data types
- Each member variable can be accessed using "," (dot) operator
- Section of memory can be treated as a variable of one type on one occasion, and of another type on another occasion

Union Example

```c
union Num
{
    int intNum;
    unsigned unsNum;
};
union Num Number;
```

Difference Between Structure and Union

- Both group a number of data types together
- Structure allocates different memory space contiguously to different data types in the group
- Union allocates the same memory space to different data types in the group
Control Structures

- Control structures (branch statements) are decision points that control the flow of program execution based on:
  - Some condition test (conditional branch)
  - Without condition test (unconditional branch)
  - Ensure execution of other statement/block or cause skipping of some statement/block

Conditional Branch Statements

- if is used to implement simple one-way test. It can be in one of the following forms:
  - if..stmt
  - if..stmt1..else..stmt2
  - if..stmt1..else..if..stmtn

- switch facilitates multi-way condition test and is very similar to the third if construct when primary test object remains same across all condition tests

Examples of “if” Construct

- if (i <= 0)
  i++;
- if (i <= 0)
  j++;
  else if (i >= 0)
  j++;
  else
  k++;
Example of “switch” Construct:

```c
switch(ch) {
    case 'A':
    case 'B':
    case 'C':
        printf("Capital");
        break;
    case 'a':
    case 'b':
    case 'c':
        printf("Small");
        break;
    default:
        printf("Not cap or small");
}
```

Same thing can be written also using if construct as:

```c
if (ch == 'A' || ch == 'B' || ch == 'C')
    printf("Capital");
else if (ch == 'a' || ch == 'b' || ch == 'c')
    printf("Small");
else
    printf("Not cap or small");
```

Unconditional Branch Statements

- **Break**: Causes unconditional exit from for, while, do, or switch constructs. Control is transferred to the statement immediately outside the block in which break appears.

- **Continue**: Causes unconditional transfer to next iteration in a for, while, or do construct. Control is transferred to the statement beginning the block in which continue appears.

- **Goto label**: Causes unconditional transfer to statement marked with the label within the function.

(Continued from previous slide)

**Return** [value/variable]: Causes immediate termination of function in which it appears and transfers control to the statement that called the function. Optionally, it provides a value compatible to the function’s return data type.

(Continued on next slide)
Loop Structures

Loop statements are used to repeat the execution of statement or blocks

Two types of loop structures are:

- **Pretest**: Condition is tested before each iteration to check if loop should occur
- **Posttest**: Condition is tested after each iteration to check if loop should continue (at least, a single iteration occurs)

Pretest Loop Structures

- **for**: It has three parts:
  - **Initializer** is executed at start of loop
  - **Loop condition** is tested before iteration to decide whether to continue or terminate the loop
  - **Incrementor** is executed at the end of each iteration
- **While**: It has a **loop condition** only that is tested before each iteration to decide whether to continue to terminate the loop

Examples of “for” and “while” Constructs

```c
for (i=0; i < 10; i++)
    printf("i = %d", i);

while (i < 10)
    {
        printf("i = %d", i);
        i++;
    }
```
Posttest Loop Construct
“do…while”

- It has a loop condition only that is tested after each iteration to decide whether to continue with next iteration or terminate the loop.
- Example of do...while is:

```c
    do {
        printf("i = %d", i);
        i++;  
    } while (i < 10) ;
```

Functions

- Functions (or subprograms) are building blocks of a program.
- All functions must be declared and defined before use.
- Function declaration requires `functionname`, `argument list`, and `return type`.
- Function definition requires coding the body or logic of function.
- Every C program must have a `main` function. It is the entry point of the program.

Example of a Function

```c
int myfunc ( int Val, int ModVal )
{
    unsigned temp;
    temp = Val % ModVal;
    return temp;
}
```

This function can be called from any other place using simple statement:

```c
    int n = myfunc(4, 2);
```
Sample C Program (Program-1)

/* Program to accept an integer from console and to display whether the number is even or odd */
#include <stdio.h>
int number, remainder;
void main()
{
    clrscr(); /* clears the console screen */
    printf("Enter an integer: ");
    scanf("%d", &number);
    remainder = number % 2;
    if (remainder == 0)
        printf("The given number is even");
    else
        printf("The given number is odd");
    getch();
}

Sample C Program (Program-2)

/* Program to accept an integer in the range 1 to 7 (both inclusive) from console and to display the corresponding day (Monday for 1, Tuesday for 2, Wednesday for 3, and so on). If the entered number is out of range, the program displays a message saying that */
#include <stdio.h>
#include <conio.h>
define MON printf("Entered number is 1 hence day is MONDAY");
define TUE printf("Entered number is 2 hence day is TUESDAY");
define WED printf("Entered number is 3 hence day is WEDNESDAY");
define THU printf("Entered number is 4 hence day is THURSDAY");
define FRI printf("Entered number is 5 hence day is FRIDAY");
define SAT printf("Entered number is 6 hence day is SATURDAY");
define SUN printf("Entered number is 7 hence day is SUNDAY");
define OTH printf("Entered number is out of range");

void main()
{
    int day;
    clrscr(); /* clears the console screen */
    printf("Enter an integer in the range 1 to 7");
    scanf("%d", &day);
    switch(day)
    {
        Case 1:
            MON;
            break;
        Case 2:
            TUE;
            break;
        Case 3:
            WED;
            break;
        Case 4:
            THU;
            break;
        Case 5:
            FRI;
            break;
        Case 6:
            SAT;
            break;
        Case 7:
            SUN;
            break;
        default:
            OTH;
    }
    getch();
}
Program to accept the radius of a circle from console and to calculate and display its area and circumference

```c
#include <stdio.h>
#include <conio.h>
#define PI 3.1415

void main()
{
    float radius, area, circum;
    clrscr();
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);
    area = PI * radius * radius;
    circum = 2 * PI * radius;
    printf("Area and circumference of the circle are %f and %f respectively", area, circum);
    getch();
}
```

Program to accept a string from console and to display the number of vowels in the string

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>

void main()
{
    char input_string[50];
    printf("Enter a string of less than 50 characters: \
");
    gets(input_string);
    int len = strlen(input_string);
    int i = 0, cnt = 0;
    printf("Enter a string of less than 50 characters: \n");
    for (i = 0; i < len; i++)
    {
        switch (input_string[i])
        {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
            case 'A':
            case 'E':
            case 'I':
            case 'O':
            case 'U':
                cnt++;
                break;
        }
    }
    printf("Number of vowels in the string are: %d", cnt);
    getch();
}
```
A Program to illustrate use of a user defined function. The program initializes an array of elements from 0 to \(n-1\) and then calculates and prints the sum of the array elements. In this example \(n = 10\).

```c
#include <stdio.h>
#define SIZE 10
intArrSum(int*p, int n);
{
  int s, tot = 0;
  for (s = 0; s < n; s++)
  {
    tot += *p;
    p++;
  }
  return tot;
}

int main()
{
  int i = 0, sum = 0;
  int nArr[SIZE] = {0};
  while (i < SIZE)
  {
    nArr[i] = i;
    i++;
  }
  sum = ArrSum(nArr, SIZE);
  printf("Sum of 0 to 9 = %d\n", sum);
  return 0;
}
```

Sample C Program (Program-5)