# UNIT I

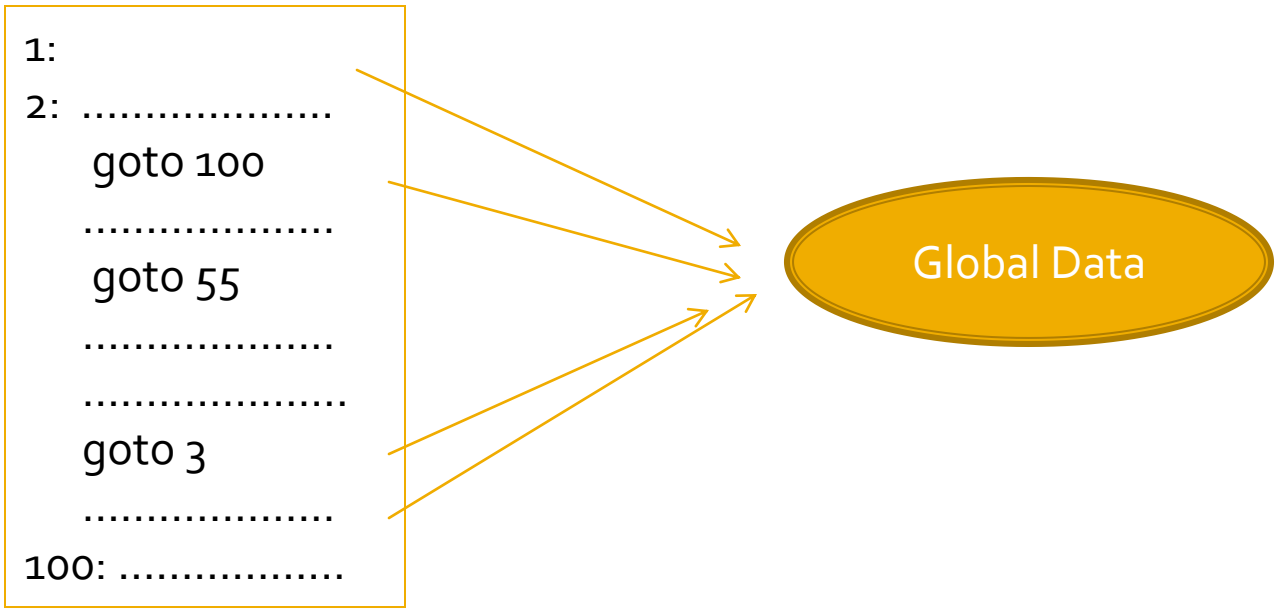
# EVOLUTION OF PROGRAMMING PARADIGM

- Programming paradigm can be categorised as:
  - Monolithic Programming
  - Procedural Programming
  - Structured Programming
  - Object Oriented Programming

# Monolithic Programming

- Contains only global data and sequential code.
- Jumps statements are used.
- Code is duplicated.

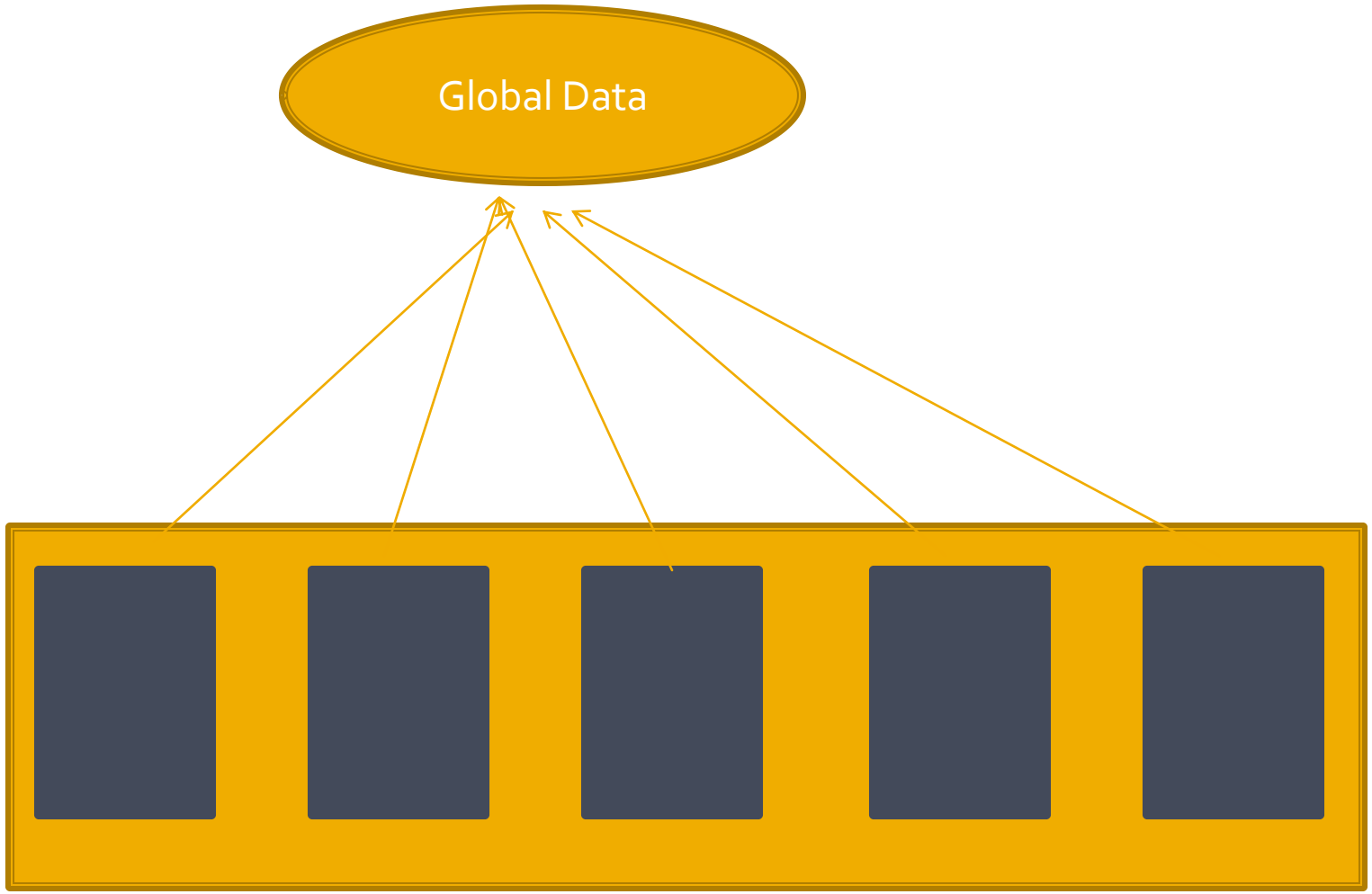
**Assembly Language and BASIC**


1:
2: .....
goto 100
.....
goto 55
.....
.....
goto 3
.....
100: .....
Global Data

# Procedural Programming

- Data items are global.
- Program organised in the form of sub routines.
- Controls through jump statements.
- Suitable for medium sized applications.
- Difficult to maintain.

**Example: FORTRAN and COBOL**



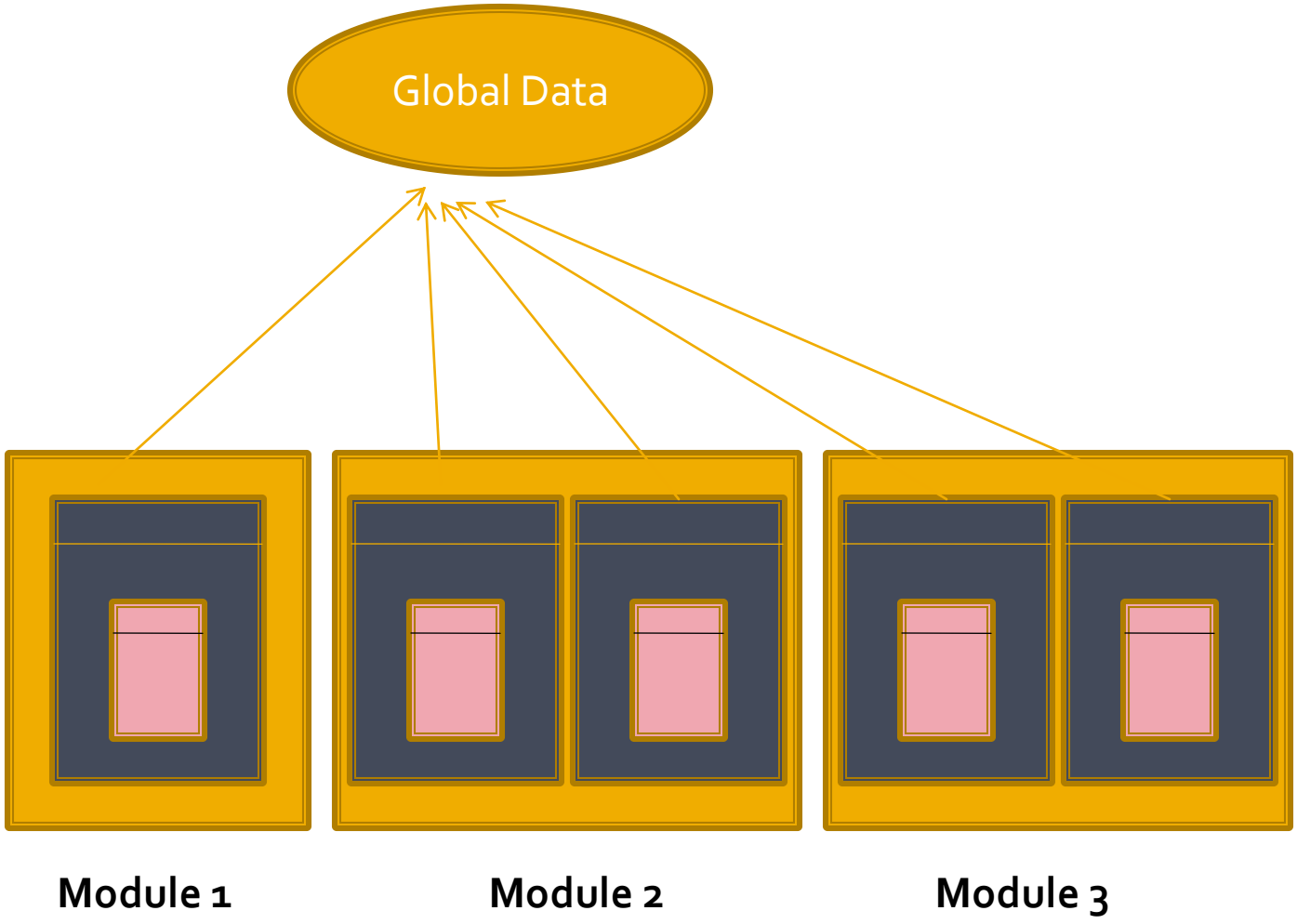

**Sub programs**

# Structured Programming

- Emphasis on algorithm than data.
- Divided into individual procedures.
- Independent of each other.
- Have their own local data and processing logic.
- Supports modular programming.
- Maintenance of large software system is costly.
- Concepts of user defined data type.

**Example : PASCAL and C**

Sub programs

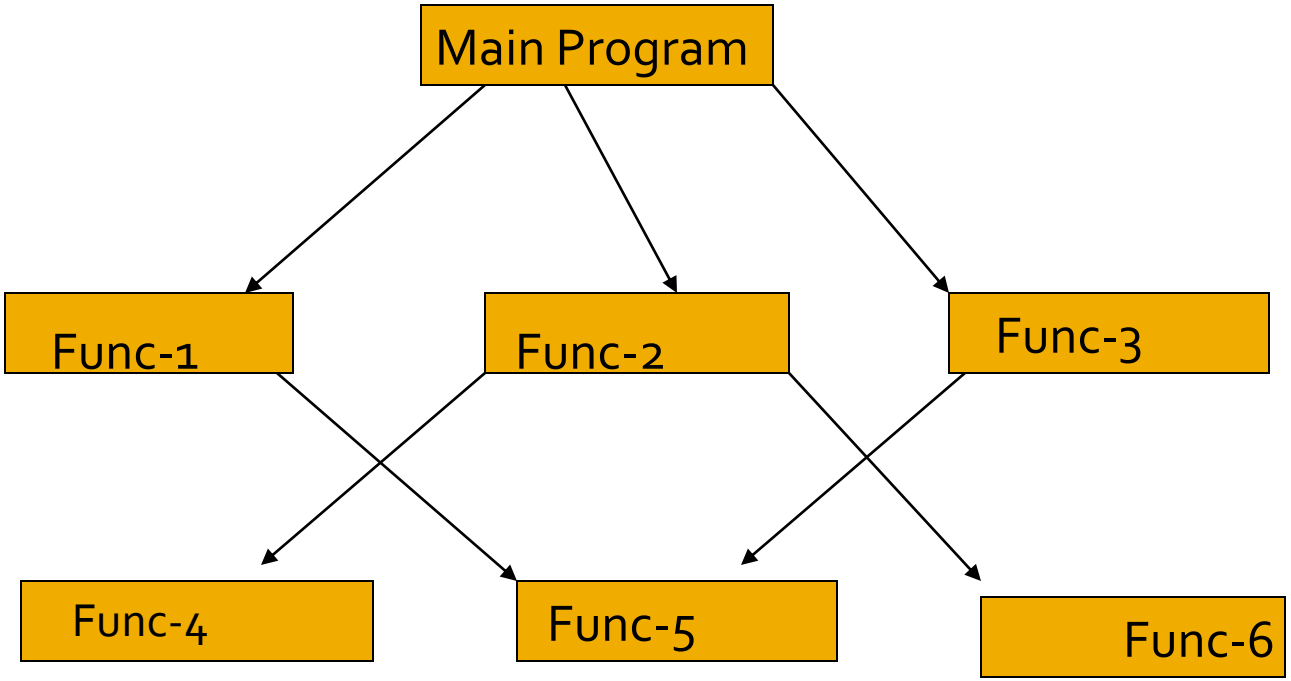


# Object Oriented Programming

- Improvement over structured programming paradigm.
- Emphasis on data rather than algorithm.
- Data abstraction is introduced.
- Data and associated operations are unified into single unit.

**Examples: C++, Smalltalk, Java.**

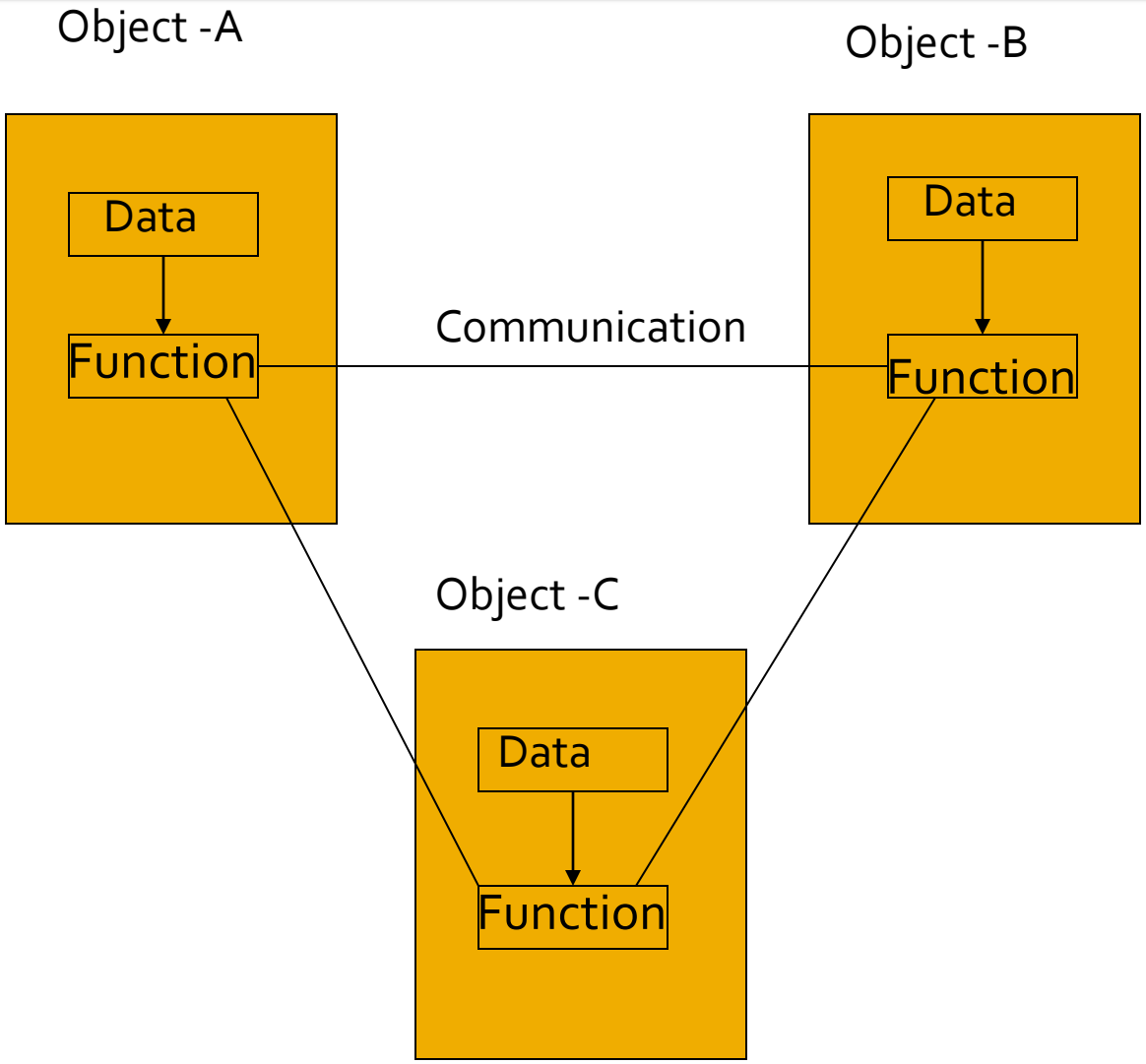
# Structure of procedure-Oriented Programs



# Characteristics of POP

- Emphasis is on doing things (Algorithms).
- Large programs are divided into smaller programs known as function.
- Most of the functions share global data.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.
- Employs top-down approach in program design.

# Structure of OOP



# Characteristics of OOP

- Emphasis is on data rather than procedure.
- Programs are divided into what are known as Objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together in the data structure.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- New data and functions can be easily added whenever necessary.
- Follows bottom-up approach in program design.

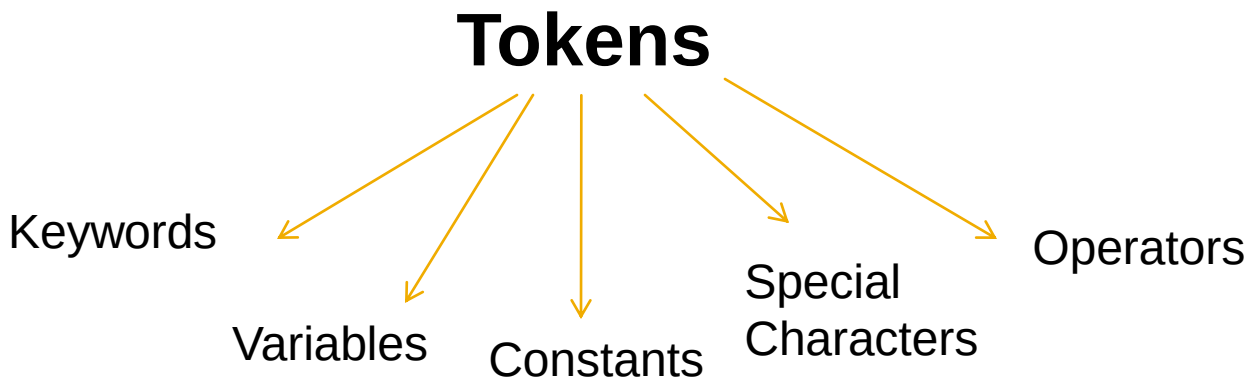
# Procedural vs. Object Oriented

- Procedural languages express programs as a collection of procedures (subroutines).
- Object Oriented languages express programs as a collection of object types (classes).

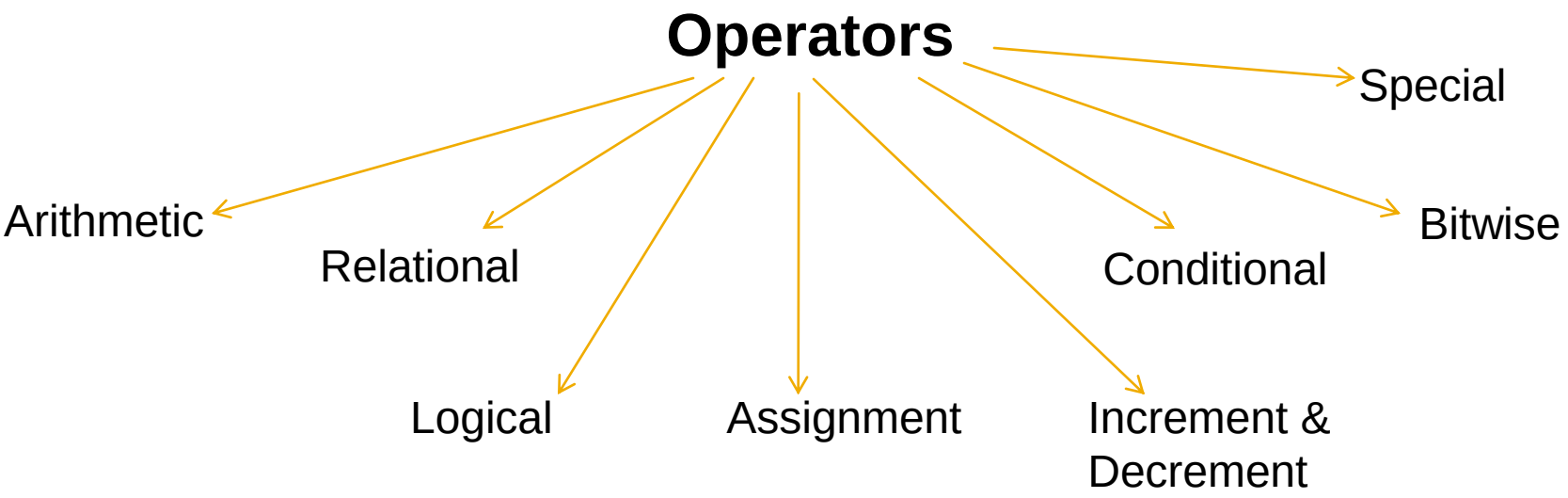
# Moving from C to C++

- Comments
- Stream based I/O
- Scope Resolution Operator
- Variable definition at the point of use
- Strict type checking
- Type Conversion
  - Syntax: Data type(Variable)

- Character Set




- Variables (Definition and initialization)
- Data type and sizes




- Qualifiers
- Compound Assignment Operator
- Constants (const, #define, enum)

- Branching Statements
  - if statement
  - if-else statement
  - switch statement
  - goto statement
- Looping Statements
  - for statement
  - while statement
  - do-while statement

- Statements and block
- Example

```
{
  int a;
  int b=10;
  a=b+10;
  .....
}
```

- One dimensional integer array
  - Searching
    - Linear Searching
    - Binary Searching
  - Sorting
    - Bubble Sorting
    - Insertion Sorting
    - Selection Sorting

- Two dimensional array
  - Matrix addition
  - Matrix Multiplication

- Strings
  - Various Programs
  - String Manipulation Functions

# Functions

- Advantages:
  - Modular Programming
  - Amount of work and development time reduced
  - Debugging is easy
  - Code Reusability
  - Reduction in size
  - Library can be designed

# Function Components

- Function Declaration/ Prototype
- Parameters
- Function Definition
- Return statement
- Function Call

# Function Prototype

- Function Name
- Return Type
- Parameters type

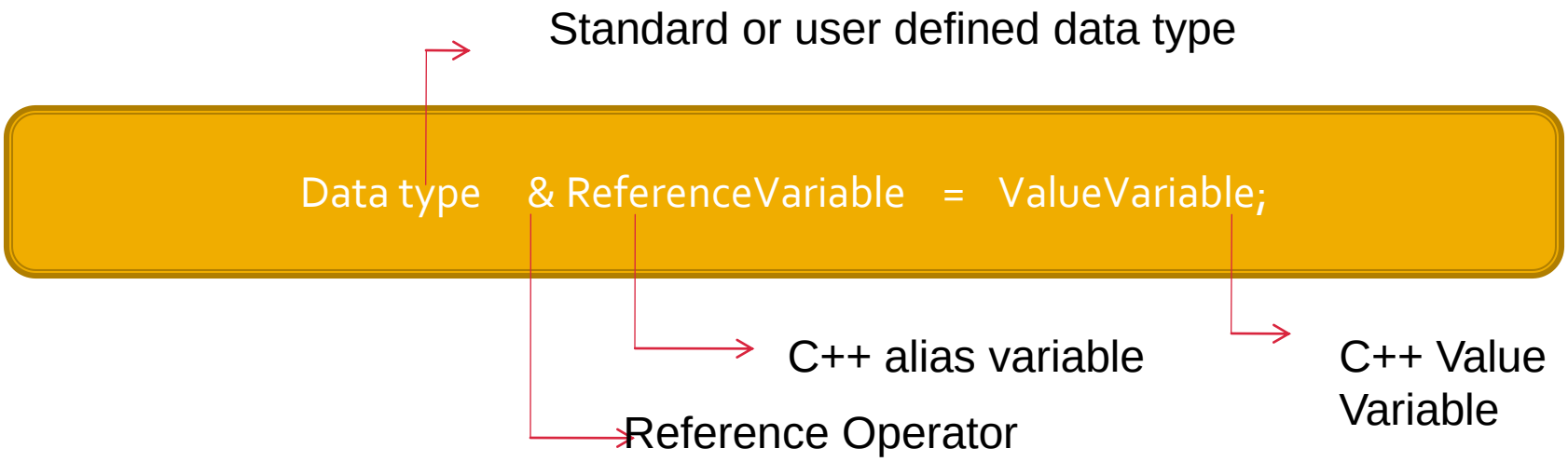
# Function Definition

- Function Header/ Function Declarator
- Function Body

# Reference Variable

- Type of variables in C++
  - Value Variable
  - Address Variable
  - Reference Variable(only in C++ not in C)

- Simple as value variable
- Powerful as pointer variable
- Syntax



# Default Arguments

- One or more arguments can be omitted in C++
- Default values can be provided in the function prototype

# Inline Functions

- Use inline keyword before the function header in function definition

# Function Overloading/ Polymorphism

- Multiple functions share same name.
- Different arguments.
- Not permitted if only return type is different.

- If an exact match is not found then the compiler converts the arguments as follows.
  - char to int
  - float to double
  - int to float or double
  - All these conversions take place to find a match.