LECTURE 14

ONE DIMENSIONAL ARRAYS

Array :

An array is a fixed sized sequenced collection of related data items of same data type.

In its simplest form an array can be used to represent a list of numbers or a list of names using a single name.

We can use an array name salary to represent a set of salaries of a group employees in an organization. We can refer to the individual salaries by writing a number called index or subscript in brackets after the array name. for example salary[10] represents the salary of 10th employee.

While the complete set of values referred to as an array, individual values are called elements.

C supports the following types of arrays :

1. One-dimensional arrays

2. Two-dimensional arrays

3. multidimensional arrays

One-dimensional arrays :

A list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable or one-dimensional array.

Declaration of one-dimensional arrays:

Like any other variable, arrays must be declared before they are used.

The general form of array declaration is :

type variable_name[size];

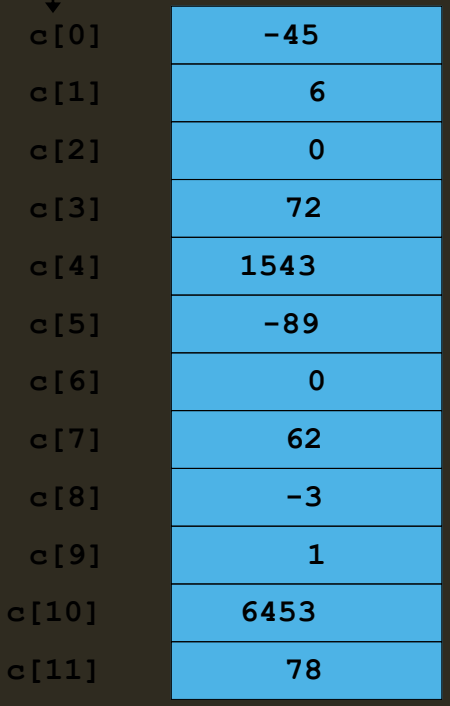
type specifies the type of elements of array, such as int, float or char and the size indicates the maximum number of elements that can be stored inside the array.

For example :

float height[10];

declares the height to be an array containing 10 real elements.

Name of array (Note that all elements of this array have the same name, **c**)



Position number of the element within array **c**

LECTURE 15

Arrays

- Array elements are like normal variables

```
c[ 0 ] =  3;
printf( "%d", c[ 0 ] );
```

  - Perform operations in subscript. If **x** equals **3**

```
c[ 5 - 2 ] == c[ 3 ] == c[ x ]
```

Declaring Arrays

- When declaring arrays, specify
  - Name
  - Type of array
  - Number of elements

```
arrayType arrayName[ numberOfElements ];
```

  - Examples:

```
int c[ 10 ];
float myArray[ 3284 ];
```

- Declaring multiple arrays of same type
  - Format similar to regular variables
  - Example:

```
int b[ 100 ], x[ 27 ];
```

Examples Using Arrays

- Initializers

```
int n[ 5 ] = { 1, 2, 3, 4, 5 };
```

  - If not enough initializers, rightmost elements become **0**

```
int n[ 5 ] = { 0 }
```

    - All elements 0
  - If too many a syntax error is produced syntax error
  - C arrays have no bounds checking
- If size omitted, initializers determine it

```
int n[ ] = { 1, 2, 3, 4, 5 };
```

  - 5 initializers, therefore 5 element array

```c
/* Fig. 6.8: fig06_08.c
   Histogram printing program */
#include <stdio.h>
#define SIZE 10

int main()
{
   int n[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
   int i, j;

   printf( "%s%13s%17s\n", "Element", "Value", "Histogram" );

   for ( i = 0; i <= SIZE - 1; i++ ) {
      printf( "%7d%13d        ", i, n[ i ]) ;

      for ( j = 1; j <= n[ i ]; j++ )   /* print one bar */
         printf( "%c", '*' );

      printf( "\n" );
   }

   return 0;
}
```

1. Initialize array

2. Loop

3. Print

Program Output

```
Element         Value         Histogram
      0            19         ********************
      1             3         ***
      2            15         ***************
      3             7         *******
      4            11         ***********
      5             9         *********
      6            13         *************
      7             5         *****
      8            17         *****************
      9             1         *
```

LECTURE 16

TWO DIMENSIONAL ARRAYS

Two-dimensional arrays :

One-dimensional array can store a list of values. There could be situations where a table of values will have to be stored.

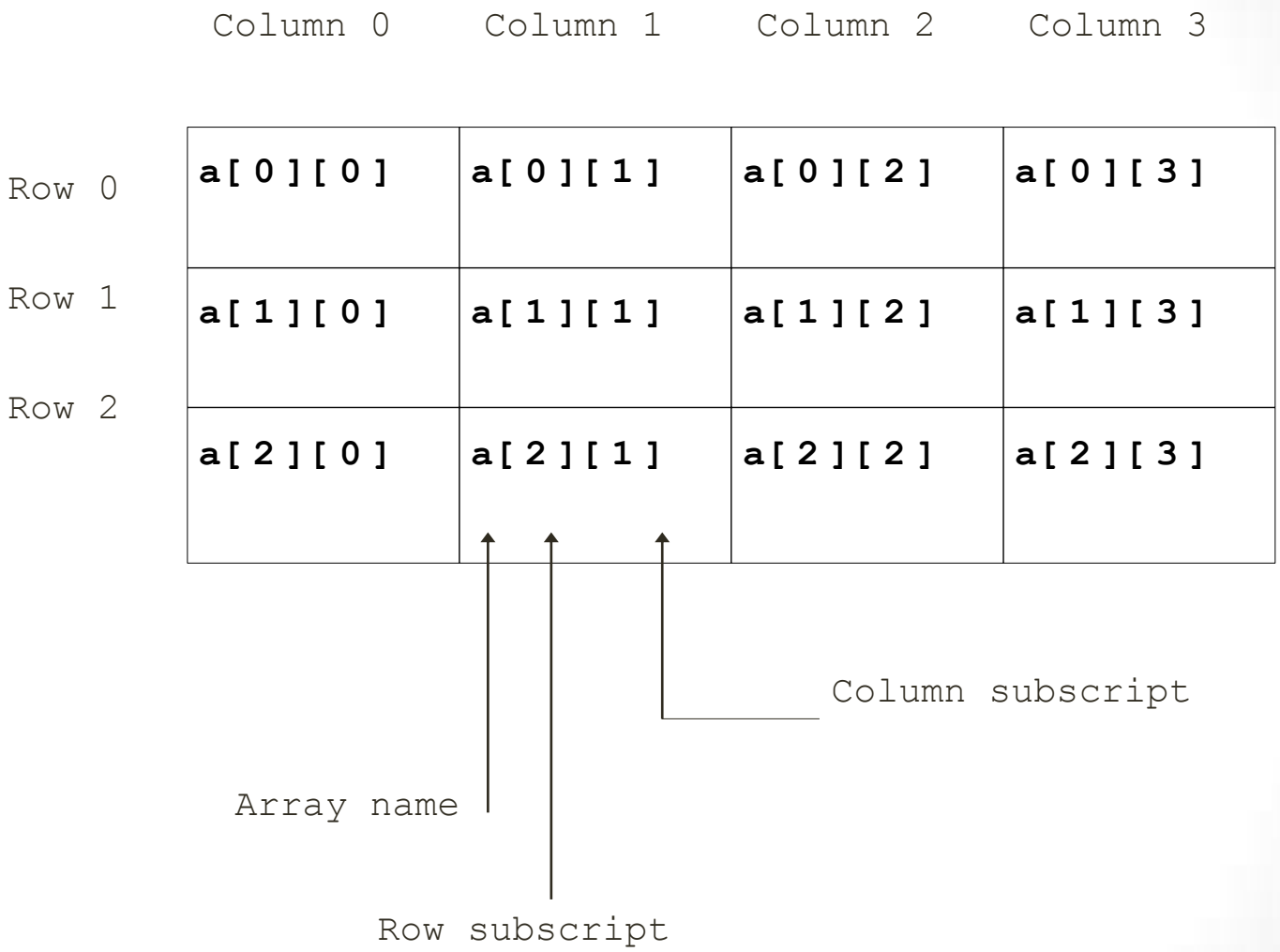
We can think of a table as a matrix consist of rows and columns.

C allows to define such table of items by using two-dimensional array such as v[3][4].

The general form of declaration of a two-dimensional array is :

type array_name[row_size][column_size];

Each dimension of array is indexed from zero to its maximum size minus one, the first index selects the row and the second index selects the column within that row.



- Initialization
  - `int b[2][2]={{1,2},{3,4}};`
  - Initializers grouped by row in braces
  - If not enough, unspecified elements set to zero

    ```
    int b[2][2]={{1},{3,4}};
    ```
- Referencing elements
  - Specify row, then column

    ```
    printf("%d",b[0][1]);
    ```

LECTURE 17

MULTIDIMENSIONAL ARRAYS

Multi-dimensional array :

C allows arrays of three or more dimensions. The exact limit is determined by the complier.

The general form of multi-dimensional array is :

type array_name[s1][s2][s3[]…….[sm];

Where si is the size of ith dimension .

For example :

int survey[3][5][12];

float table[5][4][5][3];

survey is a three-dimensional array, whereas table is a four-dimensional array.

LECTURE 18

STRINGS

Strings

A string is sequence of characters that is treated as a single data item. Any group of characters defined between double quotation marks is a string constant.

Declaring strings :

C does not support strings as a data type. However, it allows us to represent strings as character arrays. In C therefore, a string variable is any valid C vaiable name and is always declared as an array of characters.

char string_name[size];

the size determines the number of characters in string.

For example :

char city[10];

when compiler assigns a character string to a character array, it automatically supplies a null character ('\0') at the end of the string. Therefore, the size should be equal to the maximum number of characters in the string plus one.

Reading strings from terminal

Using scanf function :

The familiar input function scanf can be used with %s format specification to read in a string of characters.
For example :

```
char   address[15];
scanf("%s",address);
```

the problem with the scanf function is that it terminates its input on the first white space it finds.

Reading with getchar and gets functions :

A single character can be read from the keyboard using getchar. We can use this function repeatedly to read successive single characters from the input and place them into a character array. The reading is terminated when newline character ('\n') is entered and the null character is then inserted at the end of the string.

The getchar function call takes the form :

```
char ch;
ch=getchar( );
```

Another and more convenient method of reading a string of text containing whitespaces is to use the library function gets available in the <stdio.h> header file. This is simple function with one string parameter and called as under:

```
gets(str);
```

str is a string variable declared properly. It reads characters into str from the keyboard until a newline character is encountered and then appends a null character to the string. Unlike scanf, it does not skip whitespaces.

LECTURE 19

ARITHMATIC OPERATIONS ON STRINGS

C allows us to manipulate characters the same way we do with numbers.

Whenever a character constant or character variable is used in an expression, it is automatically converted into integer value by the system.

To write a character in its integer representation, we may write it as an integer.

For example :

```
char x='a';
printf("%d",x);
```

Will display 97 ASCII value of a on the screen.

It is also possible to perform arithmetic operations on the characters and variables

For example :

```
int x;
x='z'-1;
```

is a valid statement. The ASCII value of z is 122 and therefore the statement will assign 121 to x.

LECTURE 20

String-Handling Functions

C library supports a large number of string-handling functions that can be used to carry out many of the string manipulations.

Most commonly used string functions are :

strcat( ) – concatenates two strings

strcmp( ) – compares two strings

strcpy – copies one string into another

strlen – finds the length of a string

These functions are defined in string.h header file.

strcat function

The strcat function joins two strings together.

It takes the following form :

strcat(string1,string2);

Where string1 & string two are character arrays.

When the strcat function is excuted, string two is appended to string1

It does so by removing the null character at the end of string1 and placing string2 from there.

The string at string2 remains unchanged.

Size of string1 should be enough to accommodate the final string.

strcat function may also append a string constant to a string variable.

strcmp() function

The strcmp compares two functions identified by the arguments and has a value of 0 if they are equal.

If they are not, it has numeric difference between the first nonmatching characters in the strings.

It takes the form :

strcmp(string1,string2);

string1 and string2 may be string variables or string constants.

strcpy function

strcpy copies one string into another.

It takes the form :

strcpy(string1,string2);

It assigns the contents of string2 to string1.

string2 may be a character array variable or a string constant.

strlen function

This function counts and returns the number of characters in a string.

It takes the form :

n=strlen(string);

Where n is an integer variable, which receives the value of the length of the string.

The argument may be a string constant.

The counting ends at the first null character.

Other string handling functions

strncpy – it copies the left-most n characters of the sourse string to the target string variable.

This is a three parameter function & has the following form :

```
strncpy(string1,string2,n);
```

strncmp – this fuction has three parameter and has the following form :

```
strncmp(string1,string2,n);
```

This compares the left-most n characters of string1 to string2 and returns :

a) O if they are equal.

b) Negative number, if string1 is less than string2.

c) Positive number, otherwise.

strncat - This function concatenates the left-most n characters of second string to the end of first string.

It takes the following form :

strncat(string1,string2,n);

strstr – It is a two-parameter function that can be used to locate a sub-string in a string.

It takes the form :

strstr(string1,string2);

The function strstr searches the string1 to see whether the string2 is contained in string1.

if yes function returns the position of the first occurrence of the sub-string. Otherwise, it returns a NULL pointer.