Chapter - 9

ARRAYS

ARRAY

- An array is a structured data type consisting of a group of elements of the same type.

Defining an Array

An array must be declared before it is used .It is declared by giving the type of its elements, name and size. The syntax for declaring an array is

datatype array_name[size];

- Where data type declares the type of elements to be stored in the array such as int,float,char,double
- Array_name is the name of the array which follows the rules of constructing an

- Size specifies the maximum number of elements that can be stored in the array.

Example

```
int x[100];

char text[80];

Float sal[10];
```

Important characteristics of an Array

- All the elements of an array must be of the same type.
- Each element of an array is referred to by specifying the array name followed by one or more subscipt.
- The subscript gives the position of the an array element.
- Each subscript must be expressed as a positive integer.

- Array elements are always stored in sequential memory locations.
- The number of subscripts determines the dimension of an array.

One Dimensional array

- Arrays which have elements with single subscript are known as one dimensional array.

The general syntax is

data type array_name[size];

e-x

int a[10];

Float b[20];

Char c[100];

```
/* program to find the sum, average of n number */
#include<stdio.h>
void main()
{
int a[10],i,n,sum=0,avg;
printf("\n Enter the number of
```

Two dimensional array

- A two dimensional array consists of rows and columns. Each element is accessed by two subscripts. The first subscript refers to the row and second subscript refers to the column.
- The general form of declaration is

  datatype array_name[rowsize][colsize];

- Where row size and col size indicates the maximum number of rows and columns.
- Data type may be int,float,double and char
- Array_name is the name of the array.

Example

```
int a[5][5];

float b[3][4];
```

Initializing two-dimensional array

- int mat[3][3] = { {10,11,12},{25,34,67},{0,6,7}};

Multi-dimensional array

- More than two dimensions are also possible in c. The general form of a multi-dimensional array is:

datatype array_name [s1][s2]…[s3];

Datatype may be int,float,char,double

Array_name is the name of the array.

- Where s1,s2,s3…sn are positive integers indicating the number of array elements associated with each subscript.
- E-x

Int s[3][3][4];

Float tri[3][3][3];

Int table [4][4][5][2];

Chapter – 10 User defined functions

- A function can be defined as a subprogram which is used for doing a specific task.
- Programs are divided into smaller modules called functions. Each function can be coded separately.

Categories of functions

- Library functions

These functions are built-in functions and ready to use. E-x mathematical functions – sqrt,abs

- User defined functions

These functions must be developed by the users.

Need for writing user-defined functions

- When same set of statements must be repeated several times in various parts of the program, these statements can be placed in a function and invoked whenever required.
- The number of lines of code will be decreased
- Debugging becomes easier.

Structure of c functions

- The general structure of a c function is

```
data-type function-name(type1 arg1,type2
                arg2,…….typen argn)
{
        local variable declaration;
 executable statement;
 ----------
 return(expression);
```

- Data-type may be int,float,char,double.
- Function-name represents the name of the function.
- Argument list – List of variables and their data types.
- Return statement is used to return the result value to the main function.

Example for function

```
/*  function to add two numbers */
int  add(int a,int b)
{
int sum;
sum = a+b;
return(sum);
}
```

```
/* function to find maximum of 3 integers */
int max(int a ,int b)
{
int big;
If (a>b)
Big=a;
Else
Big = b;
```

Calling a function

- A function can be called by specifying its name, followed by a list or arguments enclosed in parentheses. The function call is generally of the form

variable=function_name(argument-list);

Return statement

- The return statement is used to send back values from the sub program or function to the main function. The general form of return statement is

  return;

  or

  return(expression);

Function prototype

- In computer programming, a function prototype is a declaration of a function that specifies the function's name and return type and type of the arguments. The syntax of the function prototype is

return-data-type function-name (type1,type2 …..typen);

E-x for function prototypes

Addition of two numbers

int add(int , int);

Area of the circle

float area_circle(float)

Factorial of a number

int   fact(int)

Actual arguments

- The arguments appearing in the function call are known as actual arguments.
- The arguments may be constants or variables.

In the function call

result = add(a,b)

a and b are actual arguments. The actual arguments have some values stored in them before function call.

Formal arguments

- The arguments that appear in the function header is called as formal arguments. Formal arguments get their values from the calling function. E-x

  int fact(int n1)

  n1 is the formal argument

Difference between actual and formal arguments

Actual arguments	Formal arguments
Actual arguments may be constants or variables	Formal arguments must always be variables
Actual arguments are used in function call	Formal arguments are used in function heading
Actual arguments are supplied values to the formal arguments	Formal arguments are received values form actual arguments

Category of functions

- Functions with no arguments and no return value.
- Functions with arguments but no return values.
- Functions with arguments and return values.
- Functions with no arguments and with return values.
- Recursive functions.

Functions with no arguments and no return values

- The function does not contain arguments and return values.

```
Main()
{
-------
-------
print_message();
}

print_message()
{
printf(“welcome to new horizon college”);
```

Function with argument but no return values

The main program calls another function by passing arguments and does not return value from the called function.

```
 void main()
{
 ----------
area_circle(r);
----------
}

area_circle(r)
{
---------
```

Function with Argument and return value

- The main program calls another function by passing arguments and returns value to the main program.

```
Void main()
{
-----
Result = Area_triangle(b,h);
-----
}
Float   Area_triangle(b,h)
{
Return(area);
}
```

Recursive functions

- Recursion is a process by which a function calls itself repeatedly until some specified condition is satisfied.
- To write a recursive function, two condition must be satisfied.

  1. The program must be in recursive form.
  2. It must include a terminating condition.

```
function1(int x)
{
 ------------
 function1(x);
  ------------
 }
```

Factorial of a number using recursion

```
#include<stdio.h>
Int fact(int);          // function prototype
 main ( )
 {
 int result, n;
```

```
int  fact(int n)
{
If (n<=1)
   return(1);
else
    return(n*fact(n-1));
}
```

Fibonacci series using recursive function

```
#include<stdio.h>
 int fibonacci (int);
 void main ( )
 {
 int i, n;
```

```
int fibonacci(int n)
{
If(n==0)
return(0);
If(n==1)
return(1);
else
return((fib(n-1)+fib(n-2));
}
```

Storage classes

Storage classes

- All the variables not only have data types but also should have storage class
- The general syntax is

<storage class> <data type> <variable name>

Storage class

Auto

Static

Register

The storage class tells us

- Where the variable should be stored
- What will be the initial value
- What is the scope of the variable (in which function the value of the variable would be available)
- What is the life of the variable

Auto storage class

- The variables which are declared as auto are referred to as automatic variables or local variables.
- The automatic variables are defined at the beginning of a function in which they are used.
- When the function is called, the memory is allocated to these variables and when control comes out of the function, the memory is deallocated.

- If the automatic variables are not initialized , then it will be initialized to some other values or meaningless values.
- The scope of the variable is limited only to the function in which they are declared and cannot be accessed outside the function.
- E-x auto int a; auto float b;

Register storage class

- The register variables are same as automatic variables except that variables are stored in CPU registers than in memory.
- The access time of the CPU register is less than that of memory. Then the program will run faster.
- The scope of the variables is limited only to the function.

e.x register int i; register float a;

Extern storage class

- The external variables are called as global variables.
- They are declared and defined outside of the function.
- External variables are initializes to zero and can be accessible any where in the program.
- The scope of the variable is throughout the execution of the program and can be shared by different modules of the same

Static storage class

- The memory allocated for static variable will not be destroyed when the control goes out of the function.
- If the static variables are not initialized ,then by default it contains zero.
- The value of the static variable persist between the function calls.
- E-x static int a; static float b;

```
#include<stdio.h>
void main()
{
Function1();
Function1();
Function1();
}
Function1()
{
static int a=0;
```

Difference between local and global variables

Global variables	Local Variables
They are normally defined outside the main program	Local variables are defined inside a function
Their default initial value is 0	Their default initial value would be unpredictable value
The life of global variables is as long as the duration of the execution of the program	The life of local variable is as long as the duration of execution of the function within which they are declared