**FUNCTION**

A function is a self contained block of codes or sub programs with a set of statements that perform some specific task or coherent task when it is called.

It is something like to hiring a person to do some specific task like, every six months servicing a bike and hand over to it.

**Any ‘C’ program contain at least one function i.e main().**

There are basically two types of function those are

1. Library function

2. User defined function

The user defined functions defined by the user according to its requirement System defined function can’t be modified, it can only read and can be used.

These function are supplied with every C compiler Source of these library function are pre complied and only object code get used by the user by linking to the code by linker .Here in system defined function description:

**Function definition** : predefined, precompiled, stored in the library

**Function declaration** : In header file with or function prototype.

**Function call :** By the programmer

**Function declaration:-**

Function declaration is also known as function prototype. It inform the compiler about three thing, those are name of the function, number and type of argument received by the function and the type of value returned by the function.

While declaring the name of the argument is optional and the function prototype always terminated by the semicolon.

**Function definition:-**

* Function definition consists of the whole description and code of the function.
* It tells about what function is doing what are its inputs and what are its output.
* It consists of two parts function header and function body.
* The return type denotes the type of the value that function will return and it is optional and if it is omitted, it is assumed to be int by default.
* The body of the function is the compound statements or block which consists of local variable declaration statement and optional return statement.
* The local variable declared inside a function is local to that function only. It can’t be used anywhere in the program and its existence is only within this function.
* The arguments of the function definition are known as formal arguments.

**Function Call**

* When the function get called by the calling function then that is called, function call.
* The compiler execute these functions when the semicolon is followed by the function name.

**Example:-**

function(arg1,arg2,arg3);

**Formal Arguments**

* The arguments which are mentioned in function definition are called formal arguments or dummy arguments .
* These arguments are used to just hold the copied of the values that are sent by the calling function through the function call.
* These arguments are like other local variables which are created when the function call starts and destroyed when the function ends.

***The basic difference between the formal argument and the actual argument are***

1) The formal argument are declared inside the parenthesis where as the local variable declared at the beginning of the function block.

2). The formal argument are automatically initialized when the copy of actual arguments are passed while other local variable are assigned values through the statements.

3)Order number and type of actual arguments in the function call should be match with the order number and type of the formal arguments.

**Return type**

It is used to return value to the calling function. It can be used in two way as

1. return
2. Or return(expression);

**Advantage of function**

* By using function large and difficult program can be divided in to sub programs and solved.
* When we want to perform some task repeatedly or some code is to be used more than once at different place in the program, then function avoids this repetition or rewritten over and over.
* Due to reducing size, modular function it is easy to modify and test.

**Notes:-**

* C program is a collection of one or more function.
* A function is get called when function is followed by the semicolon.
* A function is defined when a function name followed by a pair of curly braces
* A function can call itself again and again and this process is called recursion.
* A function can be called from other function but a function can’t be defined in another function.

**Call by value and call by reference**

There are two way through which we can pass the arguments to the function such as call by value and call by reference.

**1. Call by value**

* In the call by value copy of the actual argument is passed to the formal argument and the operation is done on formal argument.
* When the function is called by ‘call by value’ method, it doesn’t affect content of the actual argument.
* Changes made to formal argument are local to block of called function so when the control back to calling function the changes made is vanish.

1. **Call by reference**

* Instead of passing the value of variable, address or reference is passed and the function operate on address of the variable rather than value.
* Here formal argument is alter to the actual argument, it means formal arguments calls the actual arguments.

**Local, Global and Static variable**

**Local variable:-**

* variables that are defined with in a body of function or block.
* The local variables can be used only in that function or block in which they are declared.

Same variables may be used in different functions such as

function()

{

int a,b;

function 1();

}

function2 ()

{

int a=0;

b=20;

}

**Global variable:**

* The variables that are defined outside of the function is called global variable.
* All functions in the program can access and modify global variables. Global variables are automatically initialized at the time of initialization.
* Static variables: static variables are declared by writing the key word static.

-**syntax:-**

static data type variable name;

static int a;

-the static variables initialized only once and it retain between the function call. If its variable is not initialized, then it is automatically initialized to zero.

**Recursion**

* When function calls itself (inside function body) again and again then it is called as recursive function.
* In recursion calling function and called function are same.
* It is powerful technique of writing complicated algorithm in easiest way.
* According to recursion problem is defined in term of itself.
* Here statement with in body of the function calls the same function and same times it is called as circular definition.
* In other words recursion is the process of defining something in form of itself.

**POINTER**

A pointer is a variable that store memory address or that contains address of another variable where addresses are the location number always contains whole number.

So, pointer contain always the whole number. It is called pointer because it points to a particular location in memory by storing address of that location.

**Syntax-**

Data type \*pointer name;

Here \* before pointer indicate the compiler that variable declared as a pointer.

e.g.

int \*p1; //pointer to integer type

float \*p2; //pointer to float type

char \*p3; //pointer to character type

* When pointer declared, it contains garbage value i.e. it may point any value in the memory.
* **Two operators are used in the pointer i.e. address operator(&) and indirection operator or dereference operator (\*).**
* Indirection operator gives the values stored at a particular address.
* Address operator cannot be used in any constant or any expression.

**Example**:

void main()

{

int i=105;

int \*p;

p=&i;

t

printf(“value of i=%d”,\*p);

printf(“value of i=%d”,\*/&i);

printf(“address of i=%d”,&i);

printf(“address of i=%d”,p);

printf(“address of p=%u”,&p);

}

declaration tells the compiler that P will be used to store the address of integer value or in other word P is a pointer to an integer and \*p reads the value at the address contain in p.

**Sructure**

It is the collection of dissimilar data types or heterogenous data types grouped together. It means the data types may or may not be of same type.

**Structure declaration-**

struct tagname

{

Data type member1;

Data type member2;

Data type member3;

………

………

Data type member n;

};

OR

struct

{

Data type member1;

Data type member2;

Data type member3;

………

………

Data type member n;

};

OR

struct tagname

{

struct element 1;

struct element 2;

struct element 3;

………

………

struct element n;

};

Structure variable declaration;

struct student

{

int age;

char name[20];

char branch[20];

};

struct student s;

**UNION**

* Union is derived data type contains collection of different data type or dissimilar elements.
* All definition declaration of union variable and accessing member is similar to structure, but instead of keyword struct the keyword union is used, the main difference between union and structure is Each member of structure occupy the memory location, but in the unions members share memory.
* Union is used for saving memory and concept is useful when it is not necessary to use all members of union at a time.
* Where union offers a memory treated as variable of one type on one occasion where (struct), it read number of different variables stored at different place of memory.