Ch-1 # Advanced Features of C

Function:- A function is a sub-program which act on data and produce a result. In c, the function should be defined before its usage in the program.

General Syntax:-

type function-name (formal-argument list)

Every C program has at least one main function that is main(). When the program starts, the function main() is automatically called and this can call other function/ functions but it cannot be called by any other function.

Functions are of two types:

(i) Built-in functions.
(ii) User-defined functions

Built in functions are called library functions and they are the part of compiler package. User define functions are used according to the requirement of the program and created by the programmer himself.

Declaring and Defining a function.

First of all declare the function in the program and then define it.

This declaration is called function prototype and this tells the compiler about name, return type, parameters of the function. The function defination tells the compiler how a function works in a program. Many built-in functions have their own prototypes. The files having these prototypes are included in the program by using - #include.

The defination of a functions consists of function header. The statements enclosed within { and } is called the body of the function.

General Syntax:-

Type function-name (formal argument list)

{ opening Brace
  Statement Body of
  return expression; function.

} closing Brace.

Function Prototypes

All functions should be declared before their use in a program. The prototype tells the computer about characteristics of function being used in the program. Many built in functions have their function prototypes already written. The files having these prototypes should be included in the program by using #include directive.

General form. `<type> <name> (arguments);`

The function prototype ends with a semicolon.

There are 3 main components in function prototype.

(1) Function name (2) function Type (3) function Arguments.

(1) Function Name :- It is user defined any legal identifiers followed by parantheses without any space.

(2) function Type :- The function 'type' is the type of value to be returned. Every function has a return type. If a function does n't return a value, it return-type will be void.

(3) function Arguments :- function arguments is a list of parameters. The arguments are placed in the parantheses. All function arguments need not be of same type. A function can have integer, character as its arguments. Any valid C expression may be a function argument that includes ~~integ~~ constants, logical ~~to its arguments~~ expression and other functions that return a value.

```
void function_name (parameter list)   // does n't return a value.
type; function name (void);           // does n't require parameters
```

ex of function prototypes.

```
void volume();
float volume (int l, float b, float h);
int volume (int a, int b, int c);
```

Accessing a function :-

It means a function can be called or invoked in/from the main program by using function name followed by the parameters enclosed in parantheses separated by commas. It should be noted that in the function defination in the beginning, the semicolon is not used at the end. If semicolon is absent, it means

that you are defining a function and not calling it. Therefore, when a function is called, it is necessary to keep semicolon at the end. A function call statement may be written as follows -

```
volume ();
```

The statement will transfer the control to function volume(). By using the following statement we can call the function clear.

```
clear ();
```

when a call statement is encountered, the control is transferred to the function.

write a program which illustrates function accessing and prototyping.

```
#include <math.h>
#include <comanip.h>
main()
{
  float square (float); // function prototype
  float a, b;
  Puts ("Enter the number;");
  scanf("%f'&a);
  b = square(a);
  Printf("The square of the number is = %f", b);
  return 0;
}
float square(float c)
{
  return c * c ;
}
```

Run of the program -

```
Enter the number : 5
The square of the Number is = 25
```

When the function call statement is given then the control automatically transfers to functions body and statements are executed.

Passing Arguments to Functions

In C program we can call a function or a number of functions from another functions. The various

functions can be called from main() function. If the communication between these functions (calling function and called function) is to be made in such a way that there should be no error then the mechanism of passing variables from one function to other should be correct one.

Recursion:-

A function has capability to call itself in a process. This process is called recursion. Recursion may be direct or indirect. When a function call itself, then it is called direct recursion. ~~Sometimes~~ If it calls another function which then calls first function then it is indirect recursion.

The function that have parameters can be accessed in the following ways:-

(1) call by value
(2) call by Reference.

eg:- Call by value.

```
#include <stdio.h>
int main()
{ int square(int);
  int volume, side=4;
  - - - - -
  - - - - -
  volume= square(side);
  - - - - -
  - - - - -
  puts("volume");
  return 0;
}
int square(int s)
{ return s*s;
}
```

The function square() uses the value of side(i.e. 4) here (call by value)

WAP to find factorial by recursion

```
#include <stdio.h>
long int fact(int);
main()
{
int n;
Printf("Enter the Number:");
scanf("%d", &n);
Printf("\n factorial of %d is %ld \n", n, fact(n));
}
long int fact(int n)
{ if(n==0)
return(1);
else
return(n * fact(n-1));
}
```

Program to illustrate call by reference.

```
# include <stdio.h>
void swap(int *a, int *b);
void main()
{ int a=10;
int b=20;
Printf("Before swapping \n");
Printf("a=%d \n", a);
Printf("b=%d \n", b);
//using function swap
swap(&a,&b);
Printf("After swapping \n");
Printf("a=%d \n", a);
Printf("b=%d \n", b);
}
void swap(int *a, int *b)
{ int temp = *a;
*a = *b;
*b = temp;
}
```

Output

Before swapping x = 5, y = 10

After swapping x = 10, y = 5

Default arguments.

A C function prototype can be declare that one or more arguments of the function have default values. The default values are specified at the time of function declaration. when a call to the function omits the corresponding parameter argument, the compiler inserts the default values.

A function which doesn't have explicit return value by default return a value of type int. The function that doesn't return a value should declare a function type of void. In C program, the function which doesn't indicate a return an integer value.

Constant Arguments.

Sometimes when arguments are not to be modified in the function, these are declared as constant arguments. If constant values are assigned to the function, then the function is not able to modify the values. Constant arguments are useful when the functions is called by reference.

Call by value :-

The function having arguments can be accessed in two ways. one is call by value and other is call by reference It is better to first know what is actual and formal parameter. Actual parameter means the parameter which happens to be present in a function call statement and formal parameter means the parameter which happens to be in function defination.

Call by Reference.

Call by reference is frequently used for passing arguments or parameters by reference. Instead of passing a value to the function being called, reference is passed by original variable. The same variable value can be called by any of the two names ie. original variables name and the reference name.

Libraries :-

A library is a collection of linkable files or programs or functions that can be used by other programs. Library functions are predefined functions provided by C. These functions are beneficial for the programmer because a lot of time is saved due to pre-definedness of these functions and the programmer is saved of writing the codes of these functions.

Features of Library function.

(1) Time saving :- Library functions being pre-defined save time. The efforts of redevelopment can be avoided.

(2) Fast Process :- Program development is faster.

(3) Sharing - Library functions can be shared by different users.

(4) Less Disk space :- These functions occupy a lesser space on the disk. Only one copy to be maintd.

Various Header Files:

(1) stdio.h (2) string.h (3) math.h (4) stdlib.h
(5) iostream.h (6) iomanip.h (7) ctype.h (8) process.h
(9) time.h (10) values.h (11) dir.h (12) stat.h
(13) share.h (14) dos.h.

Scope of variable and Storage classes.

A storage class defines the scope (visibility) and lifetime of variables and/or functions within a C program. Also. 'storage' refers to memory allocated by the compiler to store that variable. Scope of a variable is the boundary within which a variable can be used.

Following Storage classes which can be used in a C program

- auto
- register
- static
- extern.

auto - storage class.

auto is the default storage class for all local variables.

Syntax :- auto [data_type] [variable_name];

Example:
```
{
    int count;
    auto int Month;
}
```

The example above defines two variables with the same storage class, auto can only be used within functions ie. local variables.

register - storage class

register is used to define local variables that should be stored in a register instead of RAM. This means that the variable has a maximum size equal to register size

Syntax

register [data_type] [variable_name];

example:

```
{
    register int Miles;
}
```

Static storage class:-

There are two types of static variables as

(a) local static variable.

(b) Global static variable.

static is a default storage class for global variables. static variables can be 'seen' within a functions in this source file. At link time, the static variables defined here will not be seen by the object modules that are brought in.

static can also be defined within a function. If this is done the variable is initialised at runtime but is not reinitialized when the function is called.

Syntax:- static [data_type] [variable_name];

Ex:-
```
static int count;
    int Road;
    {
        printf("%d\n", Road);
    }
```

Extern storage class.

extern is used to give a reference of a global variable that is visible to ALL the program files. When you use 'extern' the variable cannot be initialized as all it does is point the variable name and storage location that has been previously defined.

Syntax:- extern [data_type] [variable_name];

example:- extern int a;

The variable access time is very fast as compared to other storage classes.

Recursion function.

A function that calls itself is known as recursive function and the process of calling function itself is known as recursion in C programming.

C program to reverse a sentence using recursion

```
#include<stdio.h>
void Reverse();
int main()
  { Printf("Enter a sentence:");
    Reverse();
    return 0;
  }
  void Reverse()
   { char c;
     scanf("%c",&c);
     if(c!='\n')
     { Reverse();
       Printf("%c",c);
     }
   }
```

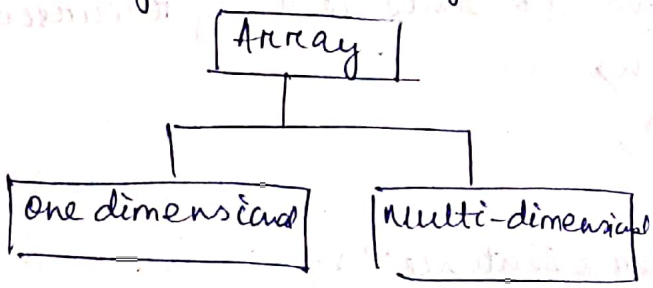
O/P:- Enter a sentence margorp emosewa
awesome preogram.

Advantages and Disadvantages of Recursion

Recursion is more elegant and requires few variables which make program clean. Recursion can be used to replace complex nesting code by dividing the problem into same problem of its sub-type. In other hand, it is hard to think the logic of a recursive function. It is also difficult to debug the code containing recursion.

Array:→

An array is a data structure and fixed-sized collection of finite number of objects and data values that are of the same type. An array is a collection of data storage locations. Each locations holds the same type of data. The storage location is called an element or component of the array. The individual element within the array is called subscript. The subscript is the number of elements in the array surrounded by square brackets [ ].



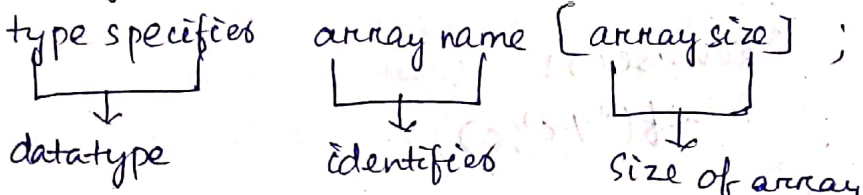
One-Dimensional Array:

In one-dimensional array the elements are specified by a single subscript. This is the simplest form of the array.

Declaration of One-Dimensional Array:

The declaration of one-dimensional array has been given here under:

General Syntax:



Datatype may be of following types:-

int
float
char
double.

The following are the valid statements of one dimensional array declaration.

```
int number[25];
char letter[10];
float sum[5];
```

Initialization of single dimensional Array

Initialization means ~~assigning~~ assigning initial values to the elements of an array. Array can be initialized only after it is declared. After the array name, equal sign can be put and a list of comma-separated values is written within the braces.

General Syntax –

type specifier array name [array size] = {initial data values};

Ex:- i) int array[8] = {5, 10, 15, 20, 25, 30, 35, 40}

ii) char vowels[5] = {a, e, i, o, u}.

Multi Dimensional Arrays:-

Arrays having more than one-dimensions are called multi-dimensional arrays.

Two-dimensional array contains 2 subscripts and a[],[] Three-dimensional array contains 3 subscripts and so on a[], b[], c[]

Strings operations

When you press any key from the keyboard, then it is said to be a character, But when you press more than one key, then it becomes a string. So combination of characters (group of character) is called string.

"I am a good boy" is a string. We can print or display the string by using the printf() function as:

Printf("\n I am a good boy \n");

Operations on string

(i) Initialization of string
(ii) Reading and writing of string
(iii) Combining strings together.
(iv) Copy one string to another
(v) Comparing two string.
(vi) Extracting a portion of string.

Pointer.

Pointer is a variable which hold the address of memory location rather than value at the location.

int num = 45

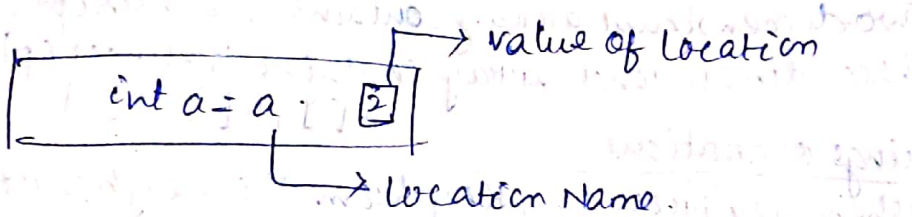
Pointer Notation.

The actual address of a variable is not known immediately. we can determine the address of a variable using address of operator (&).

The Address Operator(&)

The address operator is represent by '&' and is used to get the address of variable. The address operator & is a unary operator and returns the memory address of its operand. When a variable 'a' is declared in the program, a storage location 'a' becomes available in the memory.

```
int a;
a = 2.
```



Structure and Union.

Structure :- A structure is hetrogeneous collection of related fields. Here fields are called structure member or structure element. Every field has diff. type-specifier (datatype).

Structure defination :

```
struct tag-name
{ datatype member1;
  datatype member2;
  ....;
  ....;
};
```

Union

A union is a variable that declares a set of multiple variables called members of different type specifiers, sharing the same area of memory.

Union declaration.

```
union number
{ int num1;
  long num2;
  float num3;
  double num4;
} data;
```

Structure & Union Difference.

Structure	Union.
(1) Every member has its own memory	(1) All members use the same memory.
(2) Keyword "struct" is used	(2) Keyword "union" is used
(3) All members may be initialized.	(3) Only its first member may be initialized.
(4) Different interpretation of the same memory location are not possible	(4) Different interpretation of same memory location are possible
(5) Consumes more space compared to union	(5) Conservation of memory is possible.