Unit 4
Programming in C

1. Functions

The Modular programming where we can break large programs into small subprograms or modules which makes program simpler and easier to understand. These Subprograms are called function. It makes program more efficient and reduces complexity so that programmer can easily create, modify and debug programs. Function is categorized into two different types i.e.; library function and user define function.

a. Library Function: The functions that is already defined by a programming language itself are called library function. They are predefined functions which has its own specific work. Examples Printf();, scanf();, getch();,

b. User Define Function: The function that is defined by a user according to their own needs is called User Define Function. A function is a self-contained block of codes that can be called from any other function. When a function is called, the control transfers to the called function, which will be executed, and then again transfers the control back to the calling function.

Difference between library function and user define function:

- In library function the functions are predefined where in user define function the not predefined and user can create as per their requirement.
- library functions are stored in library files where in user define function the not stored.
- To use library function we have to add library of that function in header file where as In user define function no such requirement.
- Examples of library function is printf();, scanf();, sqrt(); etc where as examples of user define function is sum();, add();, area_of_circle();

i. Declaration of a function (Prototype of a function)

The function has to be declared before the main function which specify the function name, return type and argument list for the function definition.

Example of function declaration:

void add(); *// function declaration where **void** specify without return statement, null in bracket specify without argument and **add** specify function name.*

void add(int, int); *// function declaration where **void** specify without return statement, **(int, int)** in bracket specify argument and **add** specify function name.*

int add (); *// function declaration where **int** specify return statement, null in bracket specify without argument and **add** specify function name.*

int add (int, int); *// function declaration where **int** specify return statement, **(int, int)** in bracket specify argument and **add** specify function name.*

ii. Calling function

Calling function is done in main function which transfer the control to the called function.

Example of calling function

```
void main ()
{
add(); // this is calling a function
}
```

iii. Called function

It is a definition of function which is defined as the function is declared before with curly brackets. All the block of statements are written inside that curly bracket.

Example of function definition or called function

```
int add()   // this a function definition with return statement without arguments
{
Block of statements….
}
```

iv. Return statement

A function returns value to the calling function which is done through **return** statement. If function doesn't return value, then only control is transferred to calling function.

Example of return statement

Example 1:

```
int add()
{
int a=5, b=6,c;
c=a+b;
return c; //   It will return the value of c to the calling function
}
```

Example 2:

```
int add()
{
int a=5, b=6,c;
c=a+b;
printf(" The sum id %d", c);
                    //  There is no return statement so only control will be transferred to calling function.
}
```

v. Void statement

Void means nothing or empty. If void is used Infront of the function name that means the function is not returning any value. So if you want a function without return value then you can void statement. If you want to return value then you can use int, float etc.

Accessing a Function

A function ca be accessed by four ways which are given below:

a) Function with no argument and no return value.
b) Function with no argument with return value
c) Function with argument and no return value
d) Function with argument and with return value

a) Function with no argument and no return value.

```
#include<stdio.h>
#include<conio.h>
//A program to calculate simple interest
void si(); // function declaration without return and argument

void main()
{
si(); // function call
getch();
}

void si()   // function defination without return and argument
{
        int p,t,r,i;
        printf("Enter principle");
        scanf("%d",&p);
                printf("Enter rate");
        scanf("%d",&r);
                printf("Enter time");
        scanf("%d",&t);
        i=(p*t*r)/100;
        printf("The simple interest is %d",i);
        }
```

b) Function with no argument with return value

```
#include<stdio.h>
#include<conio.h>
//A program to calculate simple interest
int si(); // function declaration with return value without argument
void main()
{
int x;
x=si();        // function call
printf(" The simple interest is %d",x);
        getch();
}

int si() // function defination with return value without argument
{
        int p,t,r,i;
```

```
        printf("Enter princple");
        scanf("%d",&p);
        printf("Enter rate");
        scanf("%d",&r);
        printf("Enter time");
        scanf("%d",&t);
        i=(p*t*r)/100;
        return i;  // returning value to calling function
}
```

c) Function with argument and no return value

```
#include<stdio.h>
#include<conio.h>
//A program to calculate simple interest
void si(int,int,int);  // function declaration without return with argument
void main ()
{
int p,t,r;
printf("Enter principle");
scanf("%d",&p);
printf("Enter time");
scanf("%d",&t);
printf("Enter rate");
scanf("%d",&r);
si(p,t,r);          // function call
getch();
}
void si(int p, int t, int r) // function defination without return with argument
{
        int i;
        i=(p*t*r)/100;
        printf(" The simple interest %d",i);
        }
```

d) Function with argument and with return value

```
#include<stdio.h>
#include<conio.h>
//A program to calculate simple interest
int si(int,int,int); //function declaration with return value and with argument

 void main()
```

```
  {
        int p,t,r,x;
        printf("Enter principle");
        scanf("%d",&p);
              printf("Enter rate");
        scanf("%d",&r);
              printf("Enter time");
        scanf("%d",&t);
        x=si(p,t,r) ; // function call
        printf("The simple interest is %d",x);
 getch();
 }
 int si(int p,int t, int r) //function definition with return value and with argument
 {
 int i;
 i = (p*t*r)/100;
 return i;
  }
```

Some other examples of function

```
#include<stdio.h>
#include<conio.h>
// A program to check either the given number is even or odd
int even_odd(int); // function decleration

void main()
{
int num,ret;
printf("Enter number ")        ;
scanf("%d",&num);
ret=even_odd(num);
if(ret==1)
{
printf("The number is even");
}
else
{
printf("The number is odd");
}
getch();
}

int even_odd(int num)
```

```
{
int r;
r=num%2;
if(r==0)
        {
        return 1;
        }
else
{
return 2;
}
}
```

Same program can be done with this method too….

```
#include<stdio.h>
#include<conio.h>
// A program to check either the given number is even or odd using function.
int even_odd(int); // function decleration

void main()
{
int num,ret;
printf("Enter number ")      ;
scanf("%d",&num);
ret=even_odd(num);
if(ret==0)
{printf("The number is even");}
else
{       printf("The number is odd");}
        getch();
}
int even_odd(int num)
{
        int r;
        r=num%2;
return r;
}
```

2. Ways of passing Arguments of Function

We can pass arguments on the basis of following two types

a. Call by Value

When we pass the value of variable it is said as call by value. To understand call by value we have to understand actual argument and formal argument. When an argument is declared in a function that is called formal parameter and when the argument is called in main function that is called actual parameter. And the value of actual argument is passed in formal parameter the process is called argument call by a value. The changes made in formal parameter doesn't affect actual parameter.

Example:

```
#include<stdio.h>
#include<conio.h>
void chk(int);  // formal argument

void main()
{
int a = 100;
chk(a);              // actual argument that is passing value of the variable
printf("%d",a);
   getch();
}

void chk(int x)  // formal argument
{
x=x+1;   // changes in formal argument doesn't affect actual argument
}
```

Here the value of actual parameter a=100 remain same although changes made in formal parameter x=x+1

b. Call by reference

When we pass the address of variable it is said as call by reference. the address of actual argument is passed in formal parameter the process is called argument call by a reference. The changes made in formal parameter affect actual parameter.

Example:

```
#include<stdio.h>
#include<conio.h>
void chk(int *);  // formal argument with pointer

void main()
{
int a = 100;
chk(&a);  // actual argument that is passing the address of the variable
printf("%d",a);
```

```
    getch();
}

void chk(int *x)  // formal argument which is holding the address of actual argument
{
*x=*x+1; // changes in formal argument doesn't affect actual argument
}
```

*Here the value of actual parameter a=100 changes because changes are made in formal parameter *x=*x+1 . So the value of a becomes 101 i.e. a=101*

3. Scope of variables (Local & Global Variable)

a. Local Variable

The variable which is located within the function which is limited within that function is called local variable.

Example:

```
#include<stdio.h>
#include<conio.h>
void fun1();
void fun2();

void main()
{
fun1();
fun2();
}

void fun1()
{
int a=10,b=5; // local variable which can be used only in this function
int c;
c=a+b;
printf("%d",c);
}

void fun2()
{
int a=10,b=5; // local variable which can be used only in this function
int c;
c=a-b;
printf("%d",c);
}
```

b. Global Variable

The variable which is declared outside of the function and can be used in any function within the program that is called global variable. The global variable is declared before main function.

Example:

```
#include<stdio.h>
#include<conio.h>
void fun1();
void fun2();
int a=10,b=5; // global variable which can be used in any  function
void main()
{
fun1();
fun2();
}

void fun1()
{
int c;
c=a+b;
printf("%d",c);
}
void fun2()
{

int c;
c=a-b;
printf("%d",c);
}
```

4. Recursion

If the function calls itself, within the body then it is called recursion. In other word if a program allows you to call the function inside the same function is called recursive. When recursion occurs, it is necessary to stop recursive function using terminating condition otherwise it will go to a infinite loop.

Example:

```
#include<stdio.h>
#include<conio.h>
int fun(int);

void main()
{
int ret;
```

```
ret=fact(4);  // calling a value 4 to the function defination
printf("The value is %d",ret);
                getch();
}

int fun(int n)
{
    if (n==0)
    {
    return 1;
    }
    else
    {
    return 7 +fun(n-2);
    }
}
```

Lets Understand:

Here is the given program value 4 is called from the main function. So, the number 4 is checked in if condition where, 2nd condition is true so, it return 7 to the function with a call value 4-2=2. Again 2 is checked in if condition where, 2nd condition is true so, it return 7 to the function with a call value 2-2=0. Now 0 is checked in if condition now 1st condition is true so, it return 1 to the function and the program is terminated. So the result will be 7+7+1=15.

Let's understand recursion with the help of a program that display the factorial of any number.

```
#include<stdio.h>
#include<conio.h>
// to display factorial of any number
int fact(int);

void main()
{
    int ret;
ret=fact(5); // calling a value 5 to the function defination
printf("The factorial is %d",ret)    ;
getch();
}

int fact (int no)
{
    int f;
    if (no==0)
    {return 1;       }
    else
```

```
   {
         f=no*fact(no-1);  // recursion
         return f;
   }

}
```

Program to display factorial of any number in simple main function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int i,n,f=1;
   printf("Enter any number");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
   {
        f=f*i;
   }
        printf("The factorial of given number is %d",f);
             getch();
}
```

Program to display factorial of any number using user define function

```
#include<stdio.h>
#include<conio.h>
int fact(int); // function decleration with return type and argument.
void main()
{
   int n, ret;
   printf("Enter any number");
   scanf("%d",&n);
   ret=fact(n);
   printf("The factorial of given number is %d",ret);
             getch();
}
int fact(int n)
{
   int i, f=1;
   for(i=1;i<=n;i++)
   {
        f=f*i;
   }
   return f;
```

}

5. Structure

A Structure is a collection of one or more variables of different data types (int, float, char) grouped together under a single name which can store multiple data of different types. Once the structure has been declared, we can create a variable of its type. We use struct keyword to create a structure in c. for eg:

```
struct struct_name
{
  DataType member1_name;
  DataType member2_name;
  …
};
```

Then we have to declare variable of a structure in Main function. Eg:

```
Struct struct_name variable_name;
```

Or we can declare variable after structure declaration before semi colon;Eg:

```
struct struct_name {
  DataType member1_name;
  DataType member2_name;
    …
} variable_name;
```

Then we have to assign values to those structure. Eg:

```
Variable_name.member_name=value;
```

Small program to understand structure.

```
#include<stdio.h>
#include<conio.h>
struct student
{
int rn;
char *name;
float marks;
};
void main()
{
struct student info={1,"Satish", 45.5};

printf("The roll is %d\n",info.rn);
printf("The name is %s\n",info.name);
printf("The marks is %f",info.marks);
     getch();
}
```

Or,

```
#include<stdio.h>
#include<conio.h>

struct student
```

```
{
int rn;
char name[55];
float marks;
} info={1,"Satish", 45.5};

void main()
{
printf("The roll is %d\n",info.rn);
printf("The name is %s\n",info.name);
printf("The marks is %f",info.marks);
     getch();
}
```

Or,

```
#include<stdio.h>
#include<conio.h>

struct student  // structure declaration with member name
{
int rn;                  // member name
char *name;                 // member name
float marks;            // member name
};

void main()
{
struct student info;       // variable declaration with name info

info.rn=1;            //storing integer value in the variable
info.name="Satish";   //storing string value in the variable
info.marks=45.5  ;    //storing floating value in the variable


printf("The roll is %d\n",info.rn);  // displaying value
printf("The name is %s\n",info.name); // displaying value
printf("The marks is %f",info.marks); // displaying value
     getch();
}
```

Example of structure to input value from user and display it

```
#include<stdio.h>
#include<conio.h>
struct student
{
int rn;
char name[55];
```

```
float marks;
};
void main()
{
struct student info;
printf("Enter the Roll Number of student ");
scanf("%d",&info.rn);
printf("Enter the name of student ");
scanf("%s",info.name);
printf("Enter the marks of student ");
scanf("%f",&info.marks);

printf("The roll is %d\n",info.rn);
printf("The name is %s\n",info.name);
printf("The marks is %f",info.marks);
     getch();
}
```

6. Array of Structure:

Example of structure to input value from user and display it using array.

```
#include<stdio.h>
#include<conio.h>
struct student
{
int rn;
char name[55];
float marks;
};
void main()
{
struct student info[2];    // array in structure that will store the value of two
person
int i;
for (i=0;i<2;i++)
{
printf("Enter the Roll Number of student ");
scanf("%d",&info[i].rn);
printf("Enter the name of student ");
scanf("%s",info[i].name);
printf("Enter the marks of student ");
scanf("%f",&info[i].marks);
}

for (i=0;i<2;i++)
{
printf("The roll is %d\n",info[i].rn);
```

```
printf("The name is %s\n",info[i].name);
printf("The marks is %f\n",info[i].marks);
}
      getch();
}
```

Example of structure to input value from user and display it using array as user wants.

Program to display factorial of any number in simple main function.

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int i,n,f=1;
   printf("Enter any number");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
   {
        f=f*i;
   }
        printf("The factorial of given number is %d",f);
             getch();
}
```

Program to display factorial of any number using user define function

```
#include<stdio.h>
#include<conio.h>
int fact(int); // function decleration with return type and argument.
void main()
{
   int n, ret;
   printf("Enter any number");
   scanf("%d",&n);
   ret=fact(n);
   printf("The factorial of given number is %d",ret);
             getch();
}
int fact(int n)
{
   int i, f=1;
   for(i=1;i<=n;i++)
   {
        f=f*i;
```

```
    }
    return f;
  }
```

5. Structure

A Structure is a collection of one or more variables of different data types (int, float, char) grouped together under a single name which can store multiple data of different types. Once the structure has been declared, we can create a variable of its type. We use struct keyword to create a structure in c. for eg:

```
struct struct_name
{
  DataType member1_name;
  DataType member2_name;
  …
};
```

Then we have to declare variable of a structure in Main function. Eg:

```
Struct struct_name variable_name;
```

Or we can declare variable after structure declaration before semi colon;Eg:

```
struct struct_name {
  DataType member1_name;
  DataType member2_name;
    …
} variable_name;
```

Then we have to assign values to those structure. Eg:

```
Variable_name.member_name=value;
```

Small program to understand structure.

```
#include<stdio.h>
#include<conio.h>
struct student
{
int rn;
char *name;
float marks;
};
void main()
{
struct student info={1,"Satish", 45.5};

printf("The roll is %d\n",info.rn);
printf("The name is %s\n",info.name);
printf("The marks is %f",info.marks);
     getch();
}
```

Or,

```
#include<stdio.h>
#include<conio.h>
```

```
struct student
{
int rn;
char name[55];
float marks;
} info={1,"Satish", 45.5};

void main()
{
printf("The roll is %d\n",info.rn);
printf("The name is %s\n",info.name);
printf("The marks is %f",info.marks);
     getch();
}
```

Or,

```
#include<stdio.h>
#include<conio.h>

struct student  // structure declaration with member name
{
int rn;                  // member name
char *name;                 // member name
float marks;           // member name
};

void main()
{
struct student info;       // variable declaration with name info

info.rn=1;            //storing integer value in the variable
info.name="Satish";   //storing string value in the variable
info.marks=45.5  ;    //storing floating value in the variable


printf("The roll is %d\n",info.rn);  // displaying value
printf("The name is %s\n",info.name); // displaying value
printf("The marks is %f",info.marks); // displaying value
     getch();
}
```

Example of structure to input value from user and display it

```
#include<stdio.h>
#include<conio.h>
struct student
{
```

```
int rn;
char name[55];
float marks;
};
void main()
{
struct student info;
printf("Enter the Roll Number of student ");
scanf("%d",&info.rn);
printf("Enter the name of student ");
scanf("%s",info.name);
printf("Enter the marks of student ");
scanf("%f",&info.marks);

printf("The roll is %d\n",info.rn);
printf("The name is %s\n",info.name);
printf("The marks is %f",info.marks);
      getch();
}
```

6. Array of Structure:

Example of structure to input value from user and display it using array.

```
#include<stdio.h>
#include<conio.h>
struct student
{
int rn;
char name[55];
float marks;
};
void main()
{
struct student info[2];    // array in structure that will store the value of two
person
int i;
for (i=0;i<2;i++)
{
printf("Enter the Roll Number of student ");
scanf("%d",&info[i].rn);
printf("Enter the name of student ");
scanf("%s",info[i].name);
printf("Enter the marks of student ");
scanf("%f",&info[i].marks);
}

for (i=0;i<2;i++)
```

```
{
printf("The roll is %d\n",info[i].rn);
printf("The name is %s\n",info[i].name);
printf("The marks is %f\n",info[i].marks);
}
     getch();
}
```

7. Unions

Unions are similar to structures in all manner but different is that a union occupy the same memory location Hence, the member which has been updated at last is available at any given time.

Small program to understand Union.

```
#include<stdio.h>
#include<conio.h>
union student
{
int rn;
char *name;
float marks;
};
void main()
{
union student info={1,"Ramesh", 45.5};

printf("The roll is %d\n",info.rn);
printf("The name is %s\n",info.name);
printf("The marks is %f",info.marks);
     getch();
}
```

Difference between Array and Structure

Array	Structure
Array is a collection of same data type (homogeneous data).	Structure is a collection of different data type (heterogeneous data).
Array is derived data type.	Structure is user define data type.
Array element access take less time.	Structure element access take more time.
Data_type array_name[size];	struct struct_name { Members….. }structure variable_name;
int a[5];	struct student { Int rn; Char name[50];

	}info;

Structure	Union
Each member within a structure is assigned its own unique storage.	All members within a union share the same storage.
Takes more memory	Takes less memory
Array element access take less time	Structure element access take more time
The amount of memory required to store a structure is the sum of the size of entire member	The amount of memory required to store an union is same as member that has largest memory
struct student **{** **Int rn;** **Char name[50];** **}info;**	union student { Int rn; Char name[50]; }info;

8. Pointers

A pointer is a variable that holds an address of any other variable rather than a value. It reduces length and complexity of program and save the space in memory. It helps to implement dynamic memory allocation.

Pointer Declaration

Datatype *pointer_variable;

A program to display the address and value of a variable using pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i=3, *j;
j=&i;
printf("Address of i= %d\n",&i);
printf("Address of j= %d\n",&j);
printf("Value of i= %d\n",i);
printf("Value of j= %d\n",*j);
getch();
}
```

Output: Address of i= 6487580
Address of j= 6487568
Value of i= 3
Value of j= 3 // because j is holding the address of i so its pointing to the value of i.

9.Working with files

It allows user to store data permanently and read data files from an auxiliary storage device (Secondary Storage device like Hard disk. While program processing the content are stored in RAM temporarily and data may erase after the computer is shutdown so, data file save the content in secondary device and retrieve whenever we want.

Modes of data file

r = Reading file

w = Writing files

a = Add data to existing file.

EOF(End of file) = Checks the end of file

fopen() = Used to open file for reading, writing or adding mode.

fprintf() = Used to write the content to the data file.

fscanf() = Used to read the content from the data file.

fclose() = close or stop reading and writing mode.

Write a program to write person name, address and telephone number in a data file.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char name[50];
char address[50];
int tel;
int i,n;
FILE *fp;
fp=fopen("rec.dat", "w");
printf("Enter how may many records you want to write");
scanf("%d",&n);
for (i=1;i<=n;i++)
{
      printf("Enter your name");
      scanf("%s",name);
      printf("Enter your address");
      scanf("%s",address);
      printf("Enter telephone number");
      scanf("%d",&tel);
      fprintf(fp,"%s %s %d",name,address,tel);
}
fclose(fp);
getch();
```

```
}
```

Write a program to read person name, address and telephone number from a data file.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char name[50];
char address[50];
int tel;
FILE *fp;
fp=fopen("rec.dat", "r");
printf("Name Address Telephone no.");
while(fscanf(fp,"%s" "%s" "%d",name,address,tel)!=EOF)
{
      printf("%s\t" "%s\t" "%d\n",name,address,tel);
}
fclose(fp);
getch();
}
```

Write a program to add person name, address and telephone number in a exixting data file.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char name[50];
char address[50];
int tel;
int i,n;
FILE *fp;
fp=fopen("rec.dat", "a");
printf("Enter how may many records you want to write");
scanf("%d",&n);
for (i=1;i<=n;i++)
{
      printf("Enter your name");
      scanf("%s",name);
      printf("Enter your address");
      scanf("%s",address);
      printf("Enter telephone number");
      scanf("%d",&tel);
      fprintf(fp,"%s %s %d",name,address,tel);
```

```
}
fclose(fp);
getch();
}
```

Write a program to write and read name, address, telephone number in a data file.

```
#include<stdio.h>
#include<conio.h>
void main()
{
char name[50];
char address[50];
int tel;
int i,n;
FILE *fp;
fp=fopen("rec.dat", "w");
printf("Enter how may many records you want to write");
scanf("%d",&n);
for (i=1;i<=n;i++)
{
      printf("Enter your name");
      scanf("%s",name);
      printf("Enter your address");
      scanf("%s",address);
      printf("Enter telephone number");
      scanf("%d",&tel);
      fprintf(fp,"%s\t" "%s\t" "%d\n",name,address,tel);
}
fclose(fp);
fp=fopen("rec.dat", "r");
printf("Name Address Telephone no.");
while(fscanf(fp,"%s" "%s" "%d",name,address,tel)!=EOF)
{
      printf("%s\t" "%s\t" "%d\n",name,address,tel);
}
fclose(fp);
getch();
}
```

Write a program to read the number of passed students and number of failed students from a data file .

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
char name[50];
int mark;
int p=0,f=0;
FILE *fp;
fp=fopen("rec.dat", "r");
while(fscanf(fp,"%s\t" "%d",name,mark)!=EOF)
{
if (mark>=35)
{
      p=p+1;
}
else
{
      f=f+1;
}
}
printf("Total Passed students = %d",p);
printf("Total Failed students = %d",f);
fclose(fp);
getch();
}
```