Chapter 4

Function in C++

C++ Program is also collection of different functions dividing a program into function makes program easy to understand and develop. By using function length of program get reduced. In C++ some more features are added in function.

Main( ) Function

Every C++ program must have a main( ) function. This is because the program execution always starts with main( ) function. In C++ main( ) also return a value of int type. Therefore the main( ) function must be written as follows.

```
int main( )
{
    Statements;
    :
    :
}
```

The function that return a value should use the statement return( ), for the termination of program. Therefore main( ) in C++ should be defined as follows.

```
int main( )
{
    Statements;
    :
    :
    return(0);
}
```

Since the return type of every main( ) function is integer by default. The keyword int in header part of main( ) function is optional. Most C++ program will generate a warning "Function should return a value", if we forget write to return( ) at the end of main( ) function. If we want to remove this warning then always write a return(0) statement or write return type void in a header part of main( ) function.

Function Prototype

Function prototype is one of important feature added in C++. In C++ variable declaration is compulsory part, similarly function declaration is also compulsory part. The function declaration statement is known as function prototype.

The prototype statement tells the complier that what is the function name, how many arguments will be received by the function & what is the sequence of argument. The prototype statement also tells that what type of value will be return by the function. When a function is called complier check a prototype statement to ensure that the proper argument are pass to the function or not.

Here argument list contain the types & name of arguments that must be pass to the function. All the argument must be declared independently.

The prototype statement is defined as follows.

Return type Function name(Argument List);

e.g.

float add(int x,int y,float z);

In function declaration the names of the arguments are not compulsory therefore they are optional i.e. the above prototype statement can also be declared as follows.

float add(int,int,float);

Write a Program to Addition of Two Number Using Function Prototype.

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

int add(int,int);

main()
{
clrscr();
int a,b,c;
cout<<"Enter two no:-";
cin>>a>>b;
c=add(a,b);
cout<<"Addtion = "<<c;
getch();
return(0);
}

int add(int p,int q)
{
return(p+q);
}
```

Write a Program to Calculate Simple Interest Using Function Prototype.

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

float simple(float,float,float);

main()
{
clrscr();
float p,n,r;
cout<<"Enter Principle Amount = ";
```

```
cin>>p;
cout<<"Enter the no. of Year = ";
cin>>n;
cout<<"Enter the Rate = ";
cin>>r;
float si;
si=simple(p,n,r);
cout<<"Simple Interest = "<<si;
getch();
return(0);
}

float simple(float a,float b,float c)
{
return(a*b*c/100);
}
```

Write a Program to Addition, Subtraction, Multiplication, Division of two nos.

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

int add(int,int);
int sub(int,int);
int mul(int,int);
float div(int,int);

main()
{
clrscr();
int a,b;
cout<<"Enter two no = ";
cin>>a>>b;
int c;
c=add(a,b);
cout<<"Addition = "<<c<<endl;
int s;
s=sub(a,b);
cout<<"Substraction = "<<s<<endl;
int m;
m=mul(a,b);
cout<<"Multiplication = "<<m<<endl;
float d;
d=div(a,b);
cout<<"Division = "<<d<<endl;
getch();
return(0);
}
```

```
int add(int p,int q)
{
return(p+q);
}
int sub(int c,int d)
{
return(c-d);
}

int mul(int x,int y)
{
return(x*y);
}

float div(int w,int z)
{
return(w/z);
```

Write a Program to Find Reverse of Given Number.

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

int rev(int);

main()
{
clrscr();
int n;
cout<<"Enter the no = ";
cin>>n;
int r;
r=rev(n);
cout<<"Reverse no = "<<r;
getch();
return(0);
}

int rev(int a)
{
int sum=0;
while(a>0)
{
   sum=sum*10+a%10;
   a=a/10;
}
return(sum);
}
```

Write a Program to Find Factorial of Given Number.

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

int fact(int);

main()
{
clrscr();
int no,f;
cout<<"Enter the no = ";
cin>>no;
f=fact(no);
cout<<"Factorial = "<<f;
getch();
return(0);
}

int fact(int n)
{
int s=1;
while(n>0)
{
   s=s*n;
   n=n-1;
}
return(s);
}
```

Inline Function

One of the objective behind using a function in program is to save some memory space when function likely to call more than one time. But every time function is called it takes lot of extra time in executing the statement as jumping to the specific function. Saving variables of the function saving the return address and after returning recalling of the variables of the calling functions. When a function is small in size a lot of execution time is a spend in such activities.

C++ provides a solution to this problem known as inline function. Inline function is a function that is expanded in a single line when the function is called i.e. complier replaced the function call statement with corresponding statement with a function .

Syntax

```
inline return type function name(argument list)
{
    Function Statement;
}
```

e.g.
```
inline int add(int p,int q)
{
   add(p+q);
}
```

Some of the situations where Inline function expansion may not be work are:

1. For function returning values, if a loop, a switch, or a goto exists.
2. For functions not returning values, if a return statement exists.
3. If function contain static variables.
4. If Inline function are recursive.

Note

Inline Function must be defined before they are used.

Write a Program to Addition of two nos. Using Inline Function

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

inline int add(int p,int q)
{
    return(p+q);
}

main()
{
clrscr();
int a,b,c;
cout<<"Enter two nos = ";
cin>>a>>b;
c=add(a,b);
cout<<"Addition = "<<c<<endl;
int x,y,z;
cout<<"Enter two no = ";
cin>>x>>y;
z=add(x,y);
cout<<"Addition = "<<z;
getch();
return(0);
}
```

Write a Program to Find Square of given no. Using Inline Function

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

inline int square(int n)
{
   return(n*n);
}
```

```
main()
{
clrscr();
int a,sq;
cout<<"Enter the no = ";
cin>>a;
sq=square(a);
cout<<"Square of given no = "<<sq<<endl;
int b,c;
cout<<"Enter another no = ";
cin>>b;
c=square(b);
cout<<"Square of given no = "<<c;
getch();
return(0);
}
```

Write a Program to Find Simple Interest Using Inline Function

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

inline int simple(int p,int n,int r)
{
   return(p*n*r/100);
}
main()
{
clrscr();
int a,b,c,s1;
cout<<"Enter Principle Amount = ";
cin>>a;
cout<<"Enter the no. of Year = ";
cin>>b;
cout<<"Enter the Rate = ";
cin>>c;
s1=simple(a,b,c);
cout<<"Simple Interest = "<<s1<<endl;
int d,e,f,s2;
cout<<"Enter Principle Amount = ";
cin>>d;
cout<<"Enter the no. of Year = ";
cin>>e;
cout<<"Enter the Rate = ";
cin>>f;
s2=simple(d,e,f);
cout<<"Simple Interest = "<<s2;
getch();
return(0);
}
```

Write a Program to Find Area of Circle Using Inline Function

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

inline float area(float r)
{
return(3.14*r*r);
}

main()
{
float b,a;
clrscr();
cout<<"Enter the radius = ";
cin>>b;
a=area(b);
cout<<"Area of circle = "<<a<<endl;
float y,z;
cout<<"Enter the radius = ";
cin>>y;
z=area(y);
cout<<"Area of circle = "<<z;
getch();
return(0);
}
```

Default Argument

C++ allows to call a function without specifying all its argument. In such a case a default value is assign to the argument in a prototype statement. Which is not specified while calling function. Default value are specified in function declaration statement (Prototype Statement).

The complier look's like at a prototype statement to check how many arguments a function received and how many arguments are specified while calling function. If an argument is not specified it alert for the possible default value.

e.g.

float simple(int,int,int r=4);

The above prototype statement content default value of 4 which is assign to the last argument in the function definition. When a function is called by following statement.

si=simple(p,n);
si=simple(2000,5);

A value 2000 is assign to the first argument, the value 5 is assign to second argument. There is no third argument specified while function call. In that case third argument will get the value 4 i.e. default value.

Note

While calling function if value for a default argument is explicitly specifies then default value will be ignored. The default argument are always specified from right to left i.e. we can not provide a default value to particular argument in the middle of the argument list.

e.g.

int simple(int,int,int r=4);	Legal
int simple(int r=5,int,int);	Illegal
int simple(int,int n=3,int r=4);	Legal
int simple(int n=2,int,int r=4);	Illegal
int simple(int p=1000,int n=2,int r=4);	Legal

Write a Program to Demonstration of Default Argument.

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

float simple(int,int,int z=4);

main()
{
clrscr();
int p,n;
float si;
cout<<"Enter Principle Amount = ";
cin>>p;
cout<<"Enter the no. of Year = ";
cin>>n;
si=simple(p,n);
cout<<"Simple Interest = "<<si;
getch();
return(0);
}

float simple(int x,int y,int z)
{
   return(x*y*z/100);
}
```

Function Overloading

Overloading refers to the use of same thing for different purpose. Overloading of function means we can the same name to different functions which perform different task. This is also known as function polymorphism.

By using this concept we can create a number of function with one function name but with different argument list. The function would perform different operation depending upon number of arguments in function call. The correct function will be executed by the complier checking the number of argument & type of argument. The complier will not differentiate between base on the return type of function.

The following prototype statement shows function overloading.

```
int add(int,int);
float add(float,float);
int add(int,int,int);
float add(int,float);
```

When a function is called complier first match the function call statement with prototype statement when number of argument & type of argument are match with specific prototype statement then appropriate function is called.

Write a Program to Addition of two nos. using Function Overloading.

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

int add(int,int);
float add(float,float);
int add(int,int,int);

main()
{
clrscr();
int a,b,c;
cout<<"Enter two no = ";
cin>>a>>b;
c=add(a,b);
cout<<"Addition = "<<c<<endl;
float x,y,z;
cout<<"Enter two no = ";
cin>>x>>y;
z=add(x,y);
cout<<"Addition = "<<z<<endl;
int p,q,r,s;
cout<<"Enter three no = ";
cin>>p>>q>>r;
s=add(p,q,r);
cout<<"Addition = "<<s;
getch();
return(0);
}

int add(int m,int n)
{
   return(m+n);
}

float add(float d,float e)
{
   return(d+e);
}
```

```
int add(int f,int g,int h)
{
   return(f+g+h);
}
```

Write a Program to Calculate Volume of Cube, Rectangle box, Cylinder.

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

int vol(int);
int vol(int,int,int);
float vol(int,int);

main()
{
int s,volc;
clrscr();
cout<<"Enter the side = ";
cin>>s;
volc=vol(s);
cout<<"Volume of cube = "<<volc<<endl;
int l,b,h1,volb;
cout<<"Enter the length, breadth & height = ";
cin>>l>>b>>h1;
volb=vol(l,b,h1);
cout<<"Volume of box = "<<volb<<endl;
int r,h;
float volcy;
cout<<"Enter the radius & height = ";
cin>>r>>h;
volcy=vol(r,h);
cout<<"Volume of cylinder = "<<volcy;
getch();
return(0);
}

int vol(int a)
{
   return(a*a*a);
}

int vol(int x,int y,int z)
{
   return(x*y*z);
}

float vol(int p,int q)
{
   return(3.14*p*q);
}
```

Write a Program to Calculate Simple Interest using Function Overloading.

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

float simple(int,int,int);
float simple(float,float,float);

main()
{
clrscr();
int p,n,r;
float si;
cout<<"Enter Principle Amount = ";
cin>>p;
cout<<"Enter the no. of Year = ";
cin>>n;
cout<<"Enter the rate = ";
cin>>r;
si=simple(p,n,r);
cout<<"Simple Interest = "<<si<<endl;
float a,b,c;
float sim;
cout<<"Enter Principle Amount = ";
cin>>a;
cout<<"Enter the no. of Year = ";
cin>>b;
cout<<"Enter the rate = ";
cin>>c;
sim=simple(a,b,c);
cout<<"Simple Interest = "<<sim;
getch();
return(0);
}

float simple(int x,int y,int z)
{
   return(x*y*z/100);
}

float simple(float x,float y,float z)
{
   return(x*y*z/100);
}
```

Const Arguments

In C++, an argument to a function can be declared a const as shown below. The qualifier const tells the complier that the function should not modify the argument. The complier will generate an error when this condition is violated. This type of declaration is significant only when we pass arguments be reference or pointer.

e.g.

```
int strlen(const char *p);
int strlen(const string &s);
```

Write a Program to area of Rectangle, Square & Triangle using Function Overloading.

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

int area(int,int);
int area(int);
float area(int,int,int);

main()
{
int l,b,ar;
clrscr();
cout<<"Enter the length & breth = ";
cin>>l>>b;
ar=area(l,b);
cout<<"Area of Rectangle = "<<ar<<endl;
int s,as;
cout<<"Enter the side = ";
cin>>s;
as=area(s);
cout<<"Area of Square = "<<as<<endl;
int x,y,z;
float at;
cout<<"Enter the value of three side = ";
cin>>x>>y>>z;
at=area(x,y,z);
cout<<"Area of Triangle = "<<at;
getch();
return(0);
}

int area(int a,int b)
{
   return(a*b);
}

int area(int x)
{
   return(x*x);
}

float area(int p,int q,int r)
{
   return((p*q*r)/2);
}
```

Write a C++ Program using inline function to calculate area of triangle.

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

inline float area(float b,float h)
{
  return(0.5*b*h);
}

main()
{
float x,y,z;
clrscr();
cout<<"Enter the Base & Hieght = ";
cin>>x>>y;
z=area(x,y);
cout<<"Area of Triangle = "<<z;
getch();
return(0);
}
```