CLASS AND OBJECT

A class is a unit, which combines both data and the functions that operates on the data. The internal data of the class is called as member data and the functions are called member function. The member functions mostly manipulate the internal data of a class the variables of a class are called objects or instances of a class.

Class Declaration

```
class <classname>
{
      access specifier:
                  data type member-data;
                  member function
};
```

There are there types of access specifier private, public and protected.

Private: In private section, the member functions and friend of this class can only access a member data. The member functions and friends of this class can always read or write private data members. The private data member is not accessible from the outside of the class.

Public: The members which are declared as public, can accessed by any function in the outside of the class. So any outside functions can read and write the public members of the class.

Protected: These members can access by the member functions and friend function and also by the member functions and friends derived from this class. It is not accessible by the outside function

Defining the object of a class

1. `<class name> object1, object2 ……. ;`
2. Object can also be created when a class is defined by placing their names immediately after the closing brace.

Accessing class members:

The data or function members of a class construct are accessed using the operator (member selection operator).

The general syntax for accessing a member of class is

```
Object-name.data-member;

Object-name.function-member;
```

Program:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  int a,b,c;
};
void main()
{
 sum obj;
 clrscr();
 cin>>obj.a>>obj.b;
 obj.c=obj.a+obj.b;
 cout<<"Sum="<<obj.c<<endl;
 getch();
}
output:
10
20
Sum=30
```

Member functions can be defined in two places

1. outside the class definitions.
2. inside the class definition.

Defining member function outside the class

The general syntax to define the member function outside class is

```
Return type class-name : : function-name( )

{

      function body

}
```

Function definition inside the class

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  void add(void)
  {
   int a,b,c;
   cin>>a>>b;
   c=a+b;
   cout<<"Sum="<<c;
  }
};
void main()
{
 sum obj;
 clrscr();
 obj.add();
 getch();
}
output:
10
20
Sum=30
```

Function definition outside the class

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  void add(void);
};
void sum :: add(void)
{
 int a,b,c;
 cin>>a>>b;
 c=a+b;
 cout<<"Sum="<<c;
}
void main()
{
 sum obj;
 clrscr();
 obj.add();
 getch();
}
output:
10
20
Sum=30
```

Example for public data-member and public member-function

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  int a,b;
  int add(int x,int y);
};
int sum :: add(int x,int y)
{
 return(x+y);
}
void main()
{
 sum obj;
 clrscr();
 cin>>obj.a>>obj.b;
 cout<<"Sum="<<obj.add(obj.a,obj.b);
 getch();
}
output:
10
20
Sum=30
```

Example for private data-member:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 private:
  int a,b,c;
 public:
  void add(void);
};
void sum :: add(void)
{
 cin>>a>>b;
 c=a+b;
 cout<<"Sum="<<c;
}
void main()
{
 sum obj;
 clrscr();
 obj.add();
 getch();
}
output:
10
20
Sum=30
```

Example for private member-function:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 private:
  int mul(int a,int b);
 public:
  void add(void);
};
int sum :: mul(int a,int b)
{
 return(a*b);
}
void sum :: add(void)
{
 int a,b,c,d;
 cin>>a>>b;
 c=a+b;
 cout<<"Sum="<<c<<”  “;
 d=mul(a,b);
 cout<<"Mul="<<d<<endl;
}
void main()
{
 sum obj;
 clrscr();
 obj.add();
 getch();
}
```

```
output:
10
20
Sum=30  Mul=200
```

Member Function Overloading:

Member function of the class can be overloaded. They may differ in type of arguments or different number of arguments

Program:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  int add(int a,int b);
  int add(int a,int b,int c);
  double add(double a,double b);
  double add(int a,double b);
  double add(double a,int b);
};
int sum :: add(int a,int b)
{
 return(a+b);
}
int sum :: add(int a,int b,int c)
{
 return(a+b+c);
}
double sum :: add(double a,double b)
{
 return(a+b);
}
double sum :: add(int a,double b)
{
 return(a+b);
}
double sum :: add(double a,int b)
{
 return(a+b);
}
void main()
{
 sum obj;
 clrscr();
 cout<<obj.add(5,4)<<endl;
 cout<<obj.add(6,5,4)<<endl;
 cout<<obj.add(6.7,4.5)<<endl;
 cout<<obj.add(4,5.6)<<endl;
 cout<<obj.add(4.5,6)<<endl;
 getch();
}
output:
9
15
11.2
9.6
10.5
```

Making an outside function inline

We can define a member function outside the class definition and still make it inline by just using the qualifier inline the header line of function definition.

Program:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  int add(int a,int b);
};
inline int sum :: add(int a,int b)
{
 return(a+b);
}
void main()
{
 sum obj;
 clrscr();
 cout<<obj.add(5,4)<<endl;
 getch();
}
output:
9
```

Arrays of objects

We know that an array can be of any data type including strcut. Similarly, we can also have arrays of variables that are of the type class. Such variables are called arrays of objects.

Program:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  void add(void);
};
void sum :: add(void)
{
 int a,b,c;
 cin>>a>>b;
 c=a+b;
 cout<<"Sum="<<c<<endl;
}
void main()
{
 sum obj[5];
 int i;
 clrscr();
 for(i=0;i<5;i++)
 {
  obj[i].add();
 }
 getch();
}
output:
10 20
Sum=30
15 20
Sum=35
100 200
Sum=300
18 20
Sum=38
13 25
Sum=38
```

Friend Function

A function that although not a member of a class is able to access the private members of that class.

A friend function possess certain special characteristics:

1. Friend function should be preceded by the key word friend.
2. We have to declare the same function in another class, what we have declared in the friend class.
3. It can be declare either in the public or the private part of a class without affecting its meaning.
4. Usually, it has the objects as argument.
5. The friend function should be defined as a normal function.
6. It can be invoked like a normal function without the help of any object.

Program:

```
#include<iostream.h>
#include<conio.h>
class ABC
{
 private:
  int a,b;
 public:
  friend int sum(ABC p);
};
class XYZ
{
 public:
  friend int sum(ABC p);
};
int sum(ABC p)
{
 cin>>p.a>>p.b;
 return(p.a+p.b);
}
void main()
{
 ABC abc;
 clrscr();
 cout<<sum(abc)<<endl;
 getch();
}
output:
10 15
25
```

Static Data Member

1. It is initialized to zero when the first object of its class is created. No other initialization is permitted.
2. It is visible only within the class, but its lifetime is the entire program.

Program:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  static int c;
};
int sum :: c;
void main()
{
 sum obj;
 clrscr();
 cout<<obj.c<<endl;
 getch();
}
output:
0
```

Static Member Function

1. A static Function can have access to only other static members (function or variables) declared in the same class.
2. A static member function can be called using the class name (instead of its objects) as follows.

```
      Class-name : : function name.
```

Program:

```
#include<iostream.h>
#include<conio.h>
class sum
{
 public:
  static int a,b,c;
  static void add()
  {
   c=a+b;
   cout<<c<<endl;
  }
};
int sum :: a=10;
int sum :: b=20;
int sum :: c;
void main()
{
 clrscr();
 sum :: add();
 getch();
}
output:
30
```