Classes

Encapsulate, binds together the data and member function. Hides the internal detail from outside world. Only the object of a class can access the data through the methods.

syntax

```
class <class_name>
{
        Data member declation;
Public:
        member functions;
};
void main()
{
<class_name>    obj;
Obj.memberfunctio();
}
```

WAP to initialize the public member data of a class from main function

```
#include<iostream.h>
#include<conio.h>
class xyz
{
 public:
        int a, b;
};
 void main()
 {
```

```
        xyz obj;
        obj.a=10;
        obj.b=20;
        cout<<"values are "<<obj.a<<" and "<<obj.b;
        getch();
}
```

WAP to input data and display it on the screen using member functions which have inside and outside class definition.

```
#include<iostream.h>
#include<conio.h>
class xyz
{
        int a, b;
 public:
 void get_data()
 {
      cout<<"enter two numbers\n"<<endl;
      cin>>a>>b;
 }
 void put_data();
 };
void xyz :: put_data()
{
        cout<< "the vlues are:\n "<<a <<" and " <<b;
}
void main()
```

```
{
        xyz obj;
        obj.get_data();
        obj.put_data();
        getch();
}
```

Passing value to functions

```
void xyz :: get_data( int x, int y)
 {
     cout<<"initializing the values of a and b\n"<<endl;
         a=x;
         b=y;
 }
```

Then we may call the function from main() function with parameters as:

```
Obj.get_data(100, 200);
```

Nesting of function

Nesting of function find a lot of applications in c++. One function may be called from inside of another function. Even the Private function can be called from inside the public functions.

```
#include<iostream.h>
#include<conio.h>
class xyz
{
```

```
 int a, b;
 int large();
 public:
 void get_data()
 {
      cout<<"enter two numbers\n"<<endl;
      cin>>a>>b;
 }
 void put_data();
};
int xyz:: large()
 {
  if (a > b)
            return a;
  else
            return b;
  }

 void xyz :: put_data()
 {
  cout<< "the largest of two are "<<large();
 }
void xyz :: put_data()
 {
  cout<< "the largest of two are "<<large();
```

```
}
void main()
{
 xyz obj;
 obj.get_data();
 obj.put_data();
 getch();
}
```

Passing Objects to functions

```
#include<iostream.h>
#include<conio.h>
class time
{
int hrs, min;
public:
void get_data(int x,int y)
        {
         hrs=x;
         min=y;
        }
void add_time(time x1, time x2)
        {
         min=x1.min + x2.min;
```

```
            hrs = min/60;
             min = min%60;
             hrs = hrs + x1.hrs + x2.hrs;
            }
void disp_time()
            {
            cout<<"hour="<<hrs<<"  min "<<min<<endl;
            }
};
void main()
{
             time t1,t2,t3;
             t1.get_data(5,10);
             t2.get_data(6,20);
            t3.add_time(t1,t2);
             t1.disp_time();
             t2.disp_time();
             t3.disp_time();
             getch();
}
```

Functions Returning Objects

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

```
{
int real, img;
public:
void get_data(int x,int y)
        {
         real=x;
         img=y;
        }
complex add_time(complex, complex);
void disp_items()
        {
        cout<<real<<" + i"<<img<<endl;
        }
};
complex complex::add_items(complex x1, complex x2)
        {
         complex temp;
         temp.img=x1.img + x2.img;
         temp.real = x1.real + x2.real;
         return temp;
        }
void main()
{
         clrscr();
         complex t1,t2,t3,t4;
```

```
        t1.get_data(5,10);
        t2.get_data(6,20);
        t4=t3.add_items(t1,t2);
        t1.disp_items();
        t2.disp_items();
        t4.disp_items();
         getch();
}
```