

Class Inheritance

Class Inheritance

One of the main pillars in of OOP. Inheritance is the process by which new classes called **derived classes** are created from existing classes called **base classes**. The derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class.

Inheritance is the ability to create a class using, or based on another. The new or inherited class is also said to derive, or be derived, from the other class.

For example, a programmer can create a *base* class named fruit and define *derived* classes as mango, orange, banana, etc.

Each of these derived classes, (mango, orange, banana, etc.) has all the features of the *base* class (fruit) with additional attributes or features specific to these newly created derived classes.

Mango would have its own defined features, orange would have its own defined features, banana would have its own defined features, etc.

If the *base* class is *fruit* and the derived class is mango it is specified as:

General Form

class derived class name : access specifies basic class name

class mango: public fruit

Example

```
#include<iostream>
using namespace std;

class animal
{
public:
void eat();
void sleep();
void breath(); };

void animal :: eat()
{ cout<<"Eating"<<endl; }
void animal :: sleep()
{ cout<<"Sleeping"<<endl; }
void animal :: breath()
{ cout<<"Breathing"<<endl; }

int main()
{ animal cat;
cat.eat();
cat.sleep();
cat.breath();
return 0;}
```

Consider a base class **Shape** and its derived class **Rectangle** as follows:

```
#include <iostream>

using namespace std;

// Base class
class Shape
{
    public:
        void setWidth(int w)
        {
            width = w;
        }
        void setHeight(int h)
        {
            height = h;
        }
    protected:
        int width;
        int height;
};
```

```
// Derived class
class Rectangle: public Shape
{
    public:
        int getArea()
        {
            return (width * height);
        }
};

int main(void)
{
    Rectangle Rect;

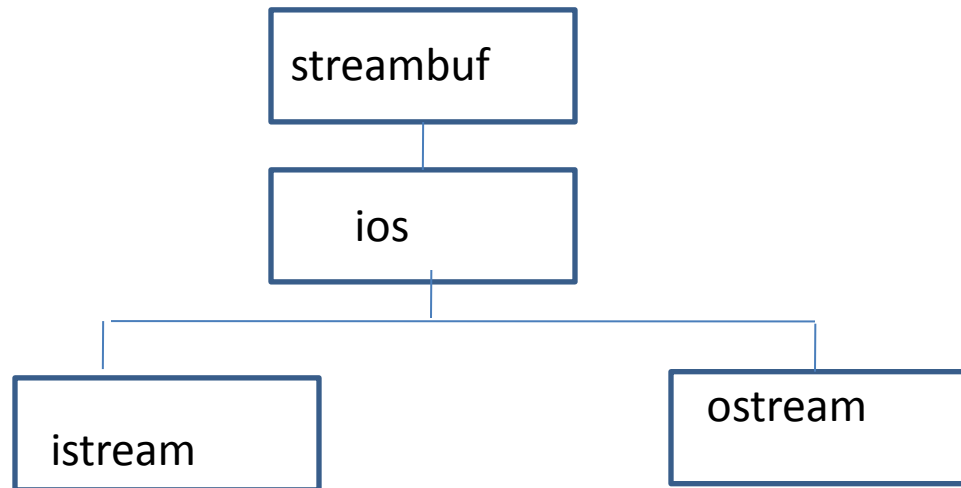
    Rect.setWidth(5);
    Rect.setHeight(7);

    // Print the area of the object.
    cout << "Total area: " << Rect.getArea() <<
endl;

    return 0; }
```

Input/Output stream

The input/output of C++ are known as streams. A stream either produces or receives information. The standard classes and objects for normal key board input and console output are defined in header file `iostream.h`



`cin>>x` (object of class `istream`)
`cout<<y` (object of class `ostream`)

Member function – put()

Stream `cout` is an object of `ostream`. Member function `put()` of stream `ostream` output one character at a time.

`cout.put('X')` // X display on the screen

Member function write()

Function **write ()** outputs some bytes from a character array. Syntax of statement in which object cout calls member function **write()** is given below.

cout. write(str,k)

str – pointer to the string whose character are to be displayed

k – Number of bytes to be displayed

Example:

```
#include<iostream>
using namespace std;

int main()
{

char str[]="Univotec";
cout.write (str,5);
return 0;
}
```

Member function getline()

syntax:

cin.getline(str,max)

str is the pointer to the character string to be read

max is the maximum number of characters in the string

```
#include<iostream>
using namespace std;

int main()
{
    char str[50];
    cout<<"Enter a string";
    cin.getline(str,50);
    cout<<"The string is:"<<str;
    return 0
}
```

Member function gcount()

Member function gcount() returns the number of characters read by the last input statement


```
#include<iostream>
using namespace std;

int main()
{
char str[50];
int k;
cout<<"Enter a string";
cin.getline(str,50);
k=cin.gcount();
cout<<"Number of characters read:"<<k;
return 0;
}
```