Part1: Composition

A class can have one or more objects of other classes as members. A class is written in such a way that object of another existing class becomes a member of the new class. This interconnection  between classes is known as C++ Composition. It is also known as containment, part-whole, or has-a relationship. A common form of software reusability is C++ Composition.

In C++ Composition, an object is a part of another object. The object that is a part of another object is known as sub object. When a C++ Composition is destroyed, then all of its sub objects are destroyed as well. Such as when car is destroyed, then its motor, frame, and other part are also destroyed with it. It has do and die relationship.

Example

#include <iostream>

using namespace std;

class classA

{

private:

int x;

public:

classA()

{

x = 0;

cout << "def cons for classA" << endl;

}

classA(int x1)

{

x = x1;

cout << "par cons for classA" << endl;

}

~classA()

{

cout << "end of classA" << endl;

}

int getx()

{

return x;

}

void setx(int x1)

{

x = x1;

}

};

class classB

{

private:

int l; classA obj,ob1; //composition

public:

classB()

{

l = 0;

cout << "def cons for classB" << endl;

}

classB(int l1)

{

l = l1;

cout << "par1 cons for classB" << endl;

}

~classB()

{

cout << "end of classb" << endl;

}

int getl()

{

cout << "classA ob.x=" << obj.getx() << endl;

return l;

}

void setl(int l1)

{

l = l1;

obj.setx(l1);

}

};

int main()

{

classB r(30);

r.setl(30);

cout << r.getl() << endl;

return 0;

}

What is the output:

|  |
| --- |
|  |

The above example will call the constructor for default constructor ClassA and the parametrized constructor for classB. To call the parametrized constructor for classA, this can be done in different way. This can be done by member initializer list

The parametrized constructor for classA (base class) can be called only by the parametrized constructor for classB (containing class). As in the following example:

Update classB by adding the following constructor

classB(int l1, int x1) :obj(x1) // member initializer list

{

l = l1;

cout << "par2 cons for classB" << endl;

}

And in the main function

classB r(10,20);

If you have more than one member object, you can update the member

initializer list as following

classB(int l1, int x1) :obj(x1),ob1(x1),ob2(25)

{

l = l1;

cout << "par2 cons for classB" << endl;

}

Remember, using composition you can member functions for the base class (classA) in the main function.

Part2: Inheritance

Inheritance is the concept by which the properties of one entity are made available to another. It allows new classes to be built from older and less specialized classes instead of being rewritten from scratch. The class that inherits properties and functions is called the *subclass* or the *derived class* and the class from which they are inherited is called the *superclass*or the *base class*. The derived class inherits all the properties of the base class and can add properties and refinements of its own. The base class remains unchanged.

Example:

#include <iostream>

using namespace std;

class classA

{

private:

int x;

public:

classA()

{

x = 0;

cout << "def cons for classA" << endl;

}

classA(int x1)

{

x = x1;

cout << "par cons for classA" << endl;

}

int getx()

{

return x;

}

void setx(int x1)

{

x = x1;

}

};

class classB: public classA //inheritance

{

private:

int l;

public:

classB()

{

l = 0;

cout << "def cons for classB" << endl;

}

classB(int l1)

{

l = l1;

cout << "par1 cons for classB" << endl;

}

int getl()

{

return l;

}

void setl(int l1)

{

l = l1;

}

};

int main()

{

classB r(20);

r.setx(25);

r.setl(35);

cout<<r.getl() << endl;

cout << r.getx() << endl;

return 0;

}

What is the output:

|  |
| --- |
|  |

Now, update setl function by adding the following line:

x=10;

* Have you got an error? Why;

We need to get member data to be protected instead of private. Retry your new version of setl method.

Exercises:

1. Create two classes, A and B, with default constructors that announce themselves. Inherit a new class called C from A, and create a member object of B in C, but do not create a constructor for C. Create an object of class C and observe the results.
2. Update both classA and classB by adding print() method in both classes, use overloading concept so you will not get any errors.

That is the end of labsheet.. Good Luck