**Introduce “Inheritance”**

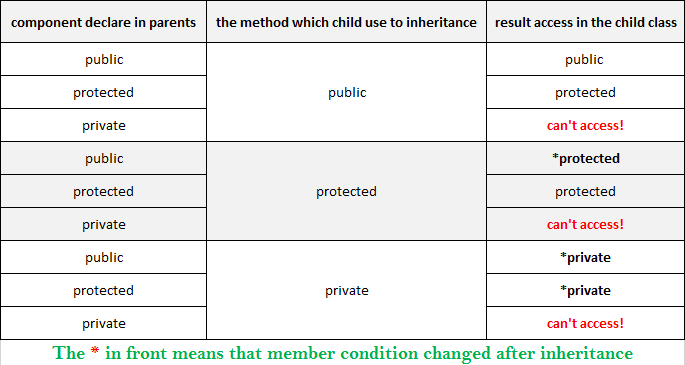
In OOP, you can define class to create objects. Sometimes you define classes very similar to the class you define before. Parts of the members ( variables, functions ) are the same so you can just share the variables and functions by “inheritance”.

**So, the purpose to apply inheritance is to save your time ! No more duplicate codes !**

Before we start, let’s check the types of member in class. You already know public and private member, but actually private member can never be inherited. It’s the design by inventor so you can’t access private member from parent class. But you don’t want to create all the variable in public because the high risk. Public member can be change by anything outside the class. How you access public member is just by a dot. (e.g: object\_name.variable\_name). So we have the third type of member which called “protected” member. Protected member can not be access by the dot, it just like a private member in a single class.

**The different between private and** **protected is protected member can be inherited by children class**.

In some case the member type will be changed after inheritance, here is the chart:



Example:

#include <iostream>

using namespace std;

class Shape

{

public:

void setWidth(int w)

{

width = w;

}

void setHeight(int h)

{

height = h;

}

protected:

int width;

int height;

};

/\* notice the “public” on this line indicate this inheritance is using **public** inheritance, check on the chart below to see the rule of public inheritance.

The public members in the base class will be inherit to child class as public members.

The protected members in the base class will be inherit to child class as protected members.

Remember the private members in the base class can never be inherited ! \*/

class Rectangle : public Shape {

public:

int getArea\_r()

{

return (width \* height);

}

};

int main()

{

Rectangle Rect;

/\*Public function in class shape has been inherited to child class

So we can use Rect.setWidth() to directly call this public member function.(Rect has inherited shape class\*/

Rect.setWidth(5);

Rect.setHeight(7);

cout << "Total area: " << Rect.getArea\_r() << endl;

system("pause");

return 0;

}

**Result:**



class Rectangle can inherit the “**public**” functions and “**protected**” variables from class Shape

The variables width and height can be inherited because they were defined by protected type variables.

If you define them in private like this:

class Shape

{

public:

void setWidth(int w)

{

width = w;

}

void setHeight(int h)

{

height = h;

}

private:

int width;

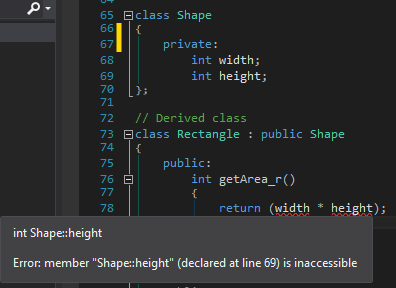
int height;

};

Then child class Rectangle will fail to inherit width and height from Shape class (base class).

* Remember private members can never be inherited !

You can see the error message from compiler like this:



So if you want to design a class which can be inherited by other classes, remember to define the variables or functions by protected or public type. Usually functions in public will be more useful because you can directly use the function which inherited from public type.

From the example above, you can see this:

Rect.setWidth(5);

Rect is a Rectangle type class which inherit the public function setWidth from class Shape:

void setWidth(int w)

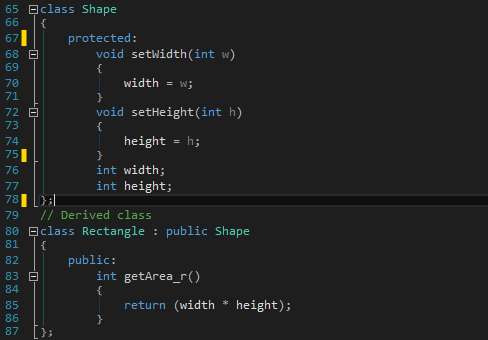
{

width = w;

}

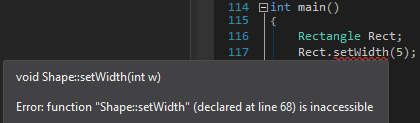
Since it is a public function, So you can directly use it just like any puclic function you define in Rectangle class. Just put a dot right after the object name(Rect). If you define setWidth in protected type member, it will still be a proteted member in the child class(Rectangle), Then you can’t directly use it by the dot. (you can only access the public members(variables or functions) by using the dot).

Let’s see what will happen if you declare the function setWidth as protected member in the base class(Shape), then inherit it by the child class (Rectangle)



Now you want to use the function setWidth after you declare the Rectangle type object, the system will show you error message because setWidth function is defined as a protected member in Shape, it’s also a protected member after inherited by Rectangle class. Since it is a protected member not a public member, you can’t directly use it by :

Rect.setWidth(5);



For easier using functions which inherited from base class(parent class), I recommend you to define functions in public, and the variables in protected. So they can both be inherited to the child class. And the functions can be directly use because they are public members, you can use the public function to setting the value of protected variables.

.