You already know the constructor and destructor can automatically run when you declare an object.

Inheritance can also inherit the constructor and destructor from base class, you can simply print message to prove it.

Example:

// Base class

class Shape

{

protected:

int width;

int height;

public:

Shape()

{

cout << " Hello from constructor: Shape" << endl;

system("pause");

}

~Shape()

{

cout << " Hello from destructor: Shape" << endl;

system("pause");

}

};

// Derived class

class Rectangle : public Shape

{

public:

Rectangle()

{

cout << " Hello from constructor: Rectangle" << endl;

system("pause");

}

~Rectangle()

{

cout << " Hello from destructor: Rectangle" << endl;

system("pause");

}

};

int main()

{

Rectangle Rect;

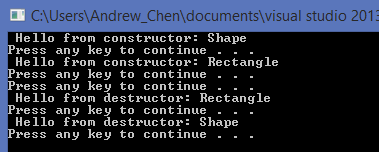
system("pause");

return 0;

}

In this case, I add a cout message in the constructor and destructor and the message will tell you which constructor is currently running ( parent or child ).

Let’s see the result:



I just declare a Rectangle class called Rect. You can see once I declare the object “Rect”, the constructor is automatically activated( both of the constructor in parent class and child class ). The order is firstly running the constructor from parent class(Shape), then running the child constructer(Rectangle).

After the program terminate ( the destructor runs when your program terminate by return 0 ), the object “Rect” also runs both the destructor in parent class and child class. But now the order changes. It firstly runs destructor from child class(Rectangle), then runs the destructor from parent class(Shape).

**\*It obviously show you the constructor and destructor be inherited automatically.**

The example shows you the constructor and destructor are automatically inherited to the child class, but it can only inherit the default constructor ( no parameter ), how about if you have another constructor with parameters( overloading constructor function ) ?

You have to declare the overloading constructor (with parameters) in the child class and use a “ : “ to indicate that you want to inherit the different constructor.

**Let’s see the example on next page:**

I also put a cout message in the second constructor( with parameters ) to show you which constructor is running.

The order is the same as the previous case:

**Parent constructor** **=>** **child constructor** **=>** **child destructor** **=>** **parent destructor**

// Base class

class Shape

{

protected:

int width;

int height;

public:

Shape()

{

cout << " Hello from constructor: Shape" << endl;

system("pause");

}

Shape(int w, int h)

{

cout << " Hello from constructor (with 2 parameters): Shape" << endl;

width = w;

height = h;

system("pause");

}

~Shape()

{

cout << " Hello from destructor: Shape" << endl;

system("pause");

}

};

// Derived class

class Rectangle : public Shape

{

public:

Rectangle()

{

cout << " Hello from constructor: Rectangle" << endl;

system("pause");

}

Rectangle(int w, int h) :Shape(w, h)

{

cout << " Hello from constructor: Rectangle (with 2 parameters)" << endl;

system("pause");

}

~Rectangle()

{

cout << " Hello from destructor: Rectangle" << endl;

system("pause");

}

};

int main()

{

Rectangle Rect(5, 10);

system("pause");

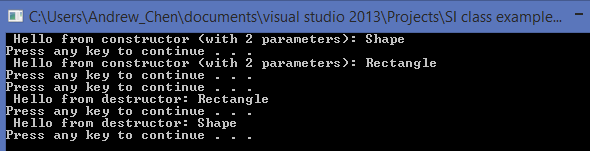
return 0;

}

Since I want to call the second constructor, I have to satisfy the two parameters when I declare the object “Rect”. The second constructor defined with two integer parameters, so the instruction to declare the object in the main function:

Rectangle Rect(5, 10);

Let’s see the result:



You can reads the code to understand the concept and learn how to inherit different constructor. If there is any questions, feel free to ask in any SI sessions!

**Pointer and class are the most important concept in 121 class, and you will always play with them in 131 data structure class. So have fun and make sure you get the concept completely ~**