Lesson 30 构造函数和析构函数

1、构造函数和析构函数的作用

构造函数和析构函数都是类的成员函数，但不能被主动调用执行。当对象创建（定义）时，构造函数会被自动执行一次；当对象被消灭（回收内存）时，析构函数会被自动执行一次。同一个对象的构造函数最多只能运行一次，析构函数也最多只能运行一次。

构造函数的作用通常用于初始化对象，例如为成员变量赋值。析构函数的作用通常用于清理对象占用的内存空间（new/delete）。

在定义类时，可以不定义构造函数和析构函数。

2、构造函数的定义和使用

构造函数必须定义成类的公有成员函数，负责初始化对象。构造函数具有以下特点：

（1）构造函数是公有成员函数；

（2）构造函数的名字和类的名字相同；

（3）构造函数不能写返回值数据类型，所以构造函数里不能写 return 语句；

（4）构造函数可以有参数，所以可以重载；

（5）构造函数在对象创建时自动执行，不可主动调用。

【例 1】猫类（无参数的构造函数）

```
#include<iostream>
#include<string>
using namespace std;
class Cat
{
private:
    string name;
public:
    Cat()
    {
        name = "Tom";
    }
    void cry()
    {
        cout << name << ": Miaow" << endl;
    }
};
int main()
{
    Cat cat1;
    cat1.cry();
    return 0;
}
```

运行结果：

```
Tom: Miaow
```

说明：当主函数执行 Cat cat1 时，自动触发 cat1 构造函数的运行。构造函数 Cat()负责为 cat1 的成员变量 name 赋值。

【例 2】猫类（有参数的构造函数、构造函数重载）

```
#include<iostream>
#include<string>
using namespace std;
class Cat
{
private:
	string name;
public:
	Cat()
	{
		name = "Tom";
	}
	Cat(string n)
	{
		name = n;
	}
	void cry()
	{
		cout << name << ": Miaow" << endl;
	}
};
int main()
{
	Cat cat1;
	cat1.cry();
	Cat cat2("Garfield");
	cat2.cry();
	return 0;
}
```

运行结果：

```
Tom: Miaow
Garfield: Miaow
```

说明：

（1）主函数中 Cat cat1;自动触发无参数的构造函数 Cat()，因而 cat1.name 被赋值为"Tom"。

（2）主函数中 Cat cat2("Garfield");自动触发有参数的构造函数 Cat(string n)，实参"Garfield"传递给了形参 n，因而 cat2.name 被赋值为"Garfield"。

（3）Cat cat1;和 Cat cat2("Garfield");可以合并写成 Cat cat1, cat2("Garfield");

（4）Cat 类里，有参数的构造函数 Cat(string n)还可以有如下三种相同功能的写法：

写法 1：

```
Cat(string n) :name(n) { }
```

写法 2：

```
Cat(string name) :name(name) { }
```

写法 3:

```
Cat(string name)
{
    this->name = name;
}
```

说明：在成员函数（包括构造函数）的定义中，如果参数和成员变量同名，那么可以用 this->标注成员变量，没有使用 this->的是参数。

注意：写法 1 和写法 2 只能用在构造函数，不可以用在其他成员函数和普通函数。

【例 3】猫类（拷贝构造函数）

当使用一个对象构造同类型的另一个对象时，触发拷贝构造函数。拷贝构造函数固定一个参数，参数类型是对自己类型的引用调用。例如 Cat 类拷贝构造函数声明应该是 Cat(Cat&)。

```
#include<iostream>
#include<string>
using namespace std;
class Cat
{
private:
    string name;
    int age;
public:
    Cat(){ }
    Cat(string name, int age) :name(name), age(age) { }
    Cat(Cat& c)
    {
        name = c.name + "(copy)";
        age = c.age / 2;
    }
    void display()
    {
        cout << name << " " << age << endl;
    }
};
int main()
{
    Cat cat1("Tom", 20);
    Cat cat2(cat1); //使用 cat1 构造 cat2，触发拷贝构造函数
    cat1.display();
    cat2.display();
    return 0;
}
```

运行结果：

```
Tom 20
Tom(copy) 10
```

说明：

（1）当类中没有定义拷贝构造函数时，默认有一个拷贝构造函数，根据参数值复制一个完全相同的对象；

（2）当类中定义了拷贝构造函数时，则默认的拷贝构造函数不再存在。

（3）Cat cat2(cat1);还可以写成 Cat cat2 = cat1; 一样触发 Cat 类里的拷贝构造函数。但是 Cat cat2; cat2 = cat1; 则不触发拷贝构造函数，而是触发无参数的构造函数，因为定义 cat2 的语句是 Cat cat2; ，此时 cat2 = cat1 进行了逐字节的复制，所以 cat2.name 是"Tom"，cat2.age 是 20。

【例 4】构造函数使用中常见的错误

```
#include<iostream>
#include<string>
using namespace std;
class Cat
{
private:
    string name;
public:
    Cat(string name)
    {
        this->name = name;
    }
    void cry()
    {
        cout << name << ": Miaow" << endl;
    }
};
int main()
{
    Cat cat1;//报错，因为没有无参数的构造函数
    Cat cat2("Garfield", 20);//报错，因为没有定义两个参数的构造函数
    Cat cat3(20);//报错，因为与定义的构造函数参数类型不一致（int 和 string 也不兼容）
    return 0;
}
```

说明：

（1）当类中没有定义构造函数时，默认类中有一个无参数的构造函数；

（2）当类中定义了构造函数，则默认的无参数构造函数不再存在。

3、析构函数的定义和使用

析构函数必须定义成类的公有成员函数，负责清理对象占用的内存空间。析构函数具有以下特点：

（1）析构函数是公有成员函数；

（2）析构函数的名字与类的名字相比，多了个~；

（3）析构函数不能写返回值数据类型，所以析构函数里不能写 return 语句；

（4）析构函数不能有参数，所以不能重载；

（5）析构函数在对象消灭时自动执行，不可主动调用。

【例 5】学生类：存放姓名、3 门课程的名称和成绩，输出每门课成绩和总成绩。
分析：构造函数负责初始化姓名和各门课程成绩，暂时不需要析构函数。

```
#include<iostream>
#include<string>
using namespace std;
class Student
{
private:
	string name;
	string subject[3];//课程名称
	int score[3];//成绩
public:
	Student(string n, string sub[3], int s[3])
	{
		name = n;
		for (int i = 0; i < 3; i++)
		{
			subject[i] = sub[i];
			score[i] = s[i];
		}
	}
	void display()
	{
		int sum = 0;
		cout << name << endl;
		for (int i = 0; i < 3; i++)
		{
			cout << subject[i] << ": " << score[i] << "\t";
			sum += score[i];
		}
		cout << endl << "Total = " << sum << endl;
	}
};
int main()
{
	string sub1[3] = { "Maths","English","Chinese" };
	int s1[3] = { 100,90,80 };
	string sub2[3] = { "Maths","English","CPP" };
	int s2[3] = { 90,85,95 };
	Student stu1("Tom", sub1, s1), stu2("Jerry", sub2, s2);
	stu1.display();
	stu2.display();
	return 0;
}
```

运行结果：

```
Tom
Maths: 100          English: 90         Chinese: 80
Total = 270
Jerry
Maths: 90           English: 85         CPP: 95
Total = 270
```

【例 6】学生类：存放姓名、若干门课程的名称和成绩，输出每门课成绩和总成绩。

分析：无法在类里定义固定长度的数组，可以使用 new 方法根据课程数量动态创建数组。定义析构函数负责回收 new 的空间。

```
#include<iostream>
#include<string>
using namespace std;
class Student
{
private:
    string name;
    int cnt;//课程数量
    string* subject;//课程名称
    int* score;//成绩
public:
    Student(string n, int c, string* sub, int* s)
    {
        name = n;
        cnt = c;
        subject = new string[c];
        score = new int[c];
        for (int i = 0; i < c; i++)
        {
            subject[i] = sub[i];
            score[i] = s[i];
        }
    }
    ~Student()
    {
        delete[ ]subject;
        delete[ ]score;
    }
    void display()
    {
        int sum = 0;
        cout << name << endl;
        for (int i = 0; i < cnt; i++)
        {
```

```
            cout << subject[i] << ": " << score[i] << "\t";
            sum += score[i];
        }
        cout << endl << "Total = " << sum << endl;
    }
};
int main()
{
    string sub1[3] = { "Maths","English","Chinese" };
    int s1[3] = { 100,90,80 };
    string sub2[4] = { "Maths","English","Physics","CPP" };
    int s2[4] = { 90,85,90,95 };
    Student stu1("Tom", 3, sub1, s1), stu2("Jerry", 4, sub2, s2);
    stu1.display();
    stu2.display();
    return 0;
}
```

运行结果：

```
Tom
Maths: 100      English: 90      Chinese: 80
Total = 270
Jerry
Maths: 90       English: 85      Physics: 90      CPP: 95
Total = 360
```

说明：

（1）当类中没有定义析构函数时，默认有一个析构函数；

（2）当类中定义了析构函数，则默认的析构函数不再存在。

4、构造函数和析构函数的执行顺序

（1）根据对象定义的顺序触发构造函数。当创建对象数组时，下标小的元素先构造。定义对象指针不会触发构造函数，因为此时在内存中仅创建了一个 4（或者 8）字节的指针变量，并没有建立对象。

（2）当多个对象生命周期结束时，栈空间对象析构顺序根据内存地址从小到大顺序，所以先创建的对象后析构。对于对象数组，下标小的元素先析构。堆空间对象（new）由 delete 语句负责触发析构函数的执行。如果 delete 的是对象数组，下标小的元素先析构。

【例 7】构造函数和析构函数的执行顺序

```
#include<iostream>
#include<string>
using namespace std;
class Cat
{
public:
    string name;
    int age;
    Cat()
```

```
    {
        cout << "触发???的无参构造函数" << endl;
    }
    Cat(string name, int age) :name(name), age(age)
    {
        cout << "触发" << name << "的有参构造函数" << endl;
    }
    Cat(Cat& c)
    {
        name = c.name + "(copy)";
        age = c.age / 2;
        cout << "触发" << name << "的拷贝构造函数" << endl;
    }
    ~Cat()
    {
        cout << "触发" << name << "的析构函数" << endl;
    }
    void display()
    {
        cout << name << " " << age << endl;
    }
};
int main()
{
    Cat* cat1 = new Cat();
    cat1->name = "Garfield";    cat1->age = 15;
    Cat cat2("Tom", 20);
    Cat cat3(cat2);
    cat1->display();
    cat2.display();
    delete cat1;
    cat3.display();
    return 0;
}
```

运行结果：

```
触发???的无参构造函数
触发 Tom 的有参构造函数
触发 Tom(copy)的拷贝构造函数
Garfield 15
Tom 20
触发 Garfield 的析构函数
Tom(copy) 10
触发 Tom(copy)的析构函数
触发 Tom 的析构函数
```