

Лекция 5. Конструкторы.

2 семестр

Лектор: ст.пр. Бельченко Ф.М.

Конструктор

Конструктор — специальная функция, которая выполняет начальную инициализацию элементов данных, причём имя конструктора обязательно должно совпадать с именем класса. Важным отличием конструктора от остальных функций является то, что он не возвращает значений вообще никаких, в том числе и `void`.

В любом классе должен быть конструктор. Даже если явным образом конструктор не объявлен (как в предыдущем классе), то компилятор предоставляет конструктор по умолчанию, без параметров.

Порядок «строительства»

Конструктор выполняет свою работу в следующем порядке:

1. Вызывает конструкторы базовых классов и членов в порядке объявления.
2. Если класс является производным от виртуальных базовых классов, конструктор инициализирует указатели виртуальных базовых классов объекта.
3. Если класс имеет или наследует виртуальные функции, конструктор инициализирует указатели виртуальных функций объекта. Указатели виртуальных функций указывают на таблицу виртуальных функций класса, чтобы обеспечить правильную привязку вызовов виртуальных функций к коду.
4. Выполняет весь код в теле функции.

Пример с вызовом конструкторов

```
#include <iostream>

using namespace std;

class Contained1 {
public:
    Contained1() { cout << "Contained1 ctor\n"; }
};

class Contained2 {
public:
    Contained2() { cout << "Contained2 ctor\n"; }
};

class Contained3 {
public:
    Contained3() { cout << "Contained3 ctor\n"; }
};

class BaseContainer {
public:
    BaseContainer() { cout << "BaseContainer ctor\n"; }
private:
    Contained1 c1;
    Contained2 c2;
};

class DerivedContainer : public BaseContainer {
public:
    DerivedContainer() : BaseContainer() { cout << "DerivedContainer ctor\n"; }
private:
    Contained3 c3;
};

int main() {
    DerivedContainer dc;
}
```

Консоль отладки Microsoft Visual Studio

Contained1 ctor
Contained2 ctor
BaseContainer ctor
Contained3 ctor
DerivedContainer ctor

C:\Users\phile\source\repos\ClassLec\x64\Debug\ClassLec.exe
(процесс 20280) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:

Конструкторы и наследование

Конструктор производного класса всегда вызывает конструктор базового класса, чтобы перед выполнением любых дополнительных операций иметь в своем распоряжении полностью созданные базовые классы. Конструкторы базового класса вызываются в порядке наследования — например, если ClassA является производным от ClassB, который является производным от ClassC, сначала вызывается конструктор ClassC, затем конструктор ClassB и последним конструктор ClassA.

И тут стоит учитывать, что конструкторы при наследовании **не наследуются**. И если базовый класс содержит только конструкторы с параметрами, то производный класс должен вызывать в своем конструкторе один из конструкторов базового класса.

Пример конструкторов и наследования

```
class Box {
public:
    Box(int width, int length, int height){
        m_width = width;
        m_length = length;
        m_height = height;
    }

private:
    int m_width;
    int m_length;
    int m_height;
};

class StorageBox : public Box {
public:
    StorageBox(int width, int length, int height, const string label&) : Box(width, length, height){
        m_label = label;
    }

private:
    string m_label;
};

int main(){

    const string aLabel = "aLabel";
    StorageBox sb(1, 2, 3, aLabel);
}
```

Пример конструкторов и множественного наследования

Если класс является производным от нескольких базовых классов, конструкторы базового класса вызываются в порядке, в котором они перечислены в объявлении производного класса:

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

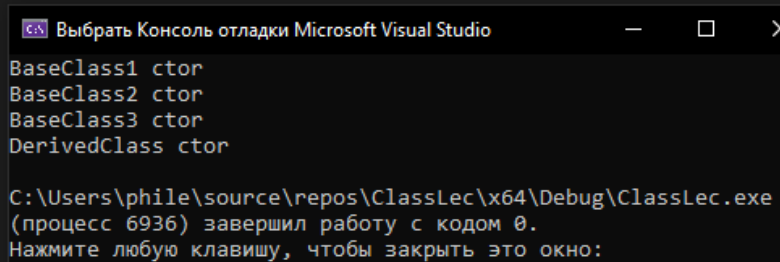
class BaseClass1 {
public:
    BaseClass1() { cout << "BaseClass1 ctor\n"; }
};

class BaseClass2 {
public:
    BaseClass2() { cout << "BaseClass2 ctor\n"; }
};

class BaseClass3 {
public:
    BaseClass3() { cout << "BaseClass3 ctor\n"; }
};

class DerivedClass : public BaseClass1,
                    public BaseClass2,
                    public BaseClass3
{
public:
    DerivedClass() { cout << "DerivedClass ctor\n"; }
};

int main() {
    DerivedClass dc;
}
```



```
Выбрать Консоль отладки Microsoft Visual Studio
BaseClass1 ctor
BaseClass2 ctor
BaseClass3 ctor
DerivedClass ctor

C:\Users\phile\source\repos\ClassLec\x64\Debug\ClassLec.exe
(процесс 6936) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

Пример наследования конструкторов через using

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

class Base
{
public:
    Base() { cout << "Base()" << endl; }
    Base(const Base& other) { cout << "Base(Base&)" << endl; }
    explicit Base(int i) : num(i) { cout << "Base(int)" << endl; }
    explicit Base(char c) : letter(c) { cout << "Base(char)" << endl; }

private:
    int num;
    char letter;
};

class Derived : Base
{
public:
    // Inherit all constructors from Base
    using Base::Base;

private:
    // Can't initialize newMember from Base constructors.
    int newMember{ 0 };
};

int main()
{
    cout << "Derived d1(5) calls: ";
    Derived d1(5);
    cout << "Derived d1('c') calls: ";
    Derived d2('c');
    cout << "Derived d3 = d2 calls: ";
    Derived d3 = d2;
    cout << "Derived d4 calls: ";
    Derived d4;
}
```

Консоль отладки Microsoft Visual Studio

Derived d1(5) calls: Base(int)
Derived d1('c') calls: Base(char)
Derived d3 = d2 calls: Base(Base&)
Derived d4 calls: Base()

C:\Users\phile\source\repos\ClassLec\x64\Debug\ClassLec.exe
(процесс 16844) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно: