28. Віртуальні деструктори.

**При работе с наследованием ваши деструкторы всегда должны быть виртуальными**. Рассмотрим следующий пример:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | #include <iostream>    class Parent  {  public:      ~Parent() // примечание: Деструктор не виртуальный      {          std::cout << "Calling ~Parent()" << std::endl;      }  };    class Child: public Parent  {  private:      int\* m\_array;    public:      Child(int length)      {          m\_array = new int[length];      }        ~Child() // примечание: Деструктор не виртуальный      {          std::cout << "Calling ~Child()" << std::endl;          delete[] m\_array;      }  };    int main()  {      Child \*child = new Child(7);      Parent \*parent = child;      delete parent;        return 0;  } |

Поскольку parent является [**указателем**](https://ravesli.com/urok-80-ukazateli-vvedenie/) класса Parent, то при его уничтожении компилятор будет смотреть, является ли деструктор класса Parent виртуальным. Поскольку это не так, то компилятор вызовет только деструктор класса Parent.

Результат выполнения программы:

Calling ~Parent()

Тем не менее, нам нужно, чтобы [**delete**](https://ravesli.com/urok-143-konstruktory-preobrazovaniya-klyuchevye-slova-explicit-i-delete/) вызывал деструктор класса Child (который, в свою очередь, будет вызывать деструктор класса Parent), иначе m\_array не будет удален. Это можно выполнить, сделав деструктор класса Parent виртуальным:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | #include <iostream>    class Parent  {  public:      virtual ~Parent() // примечание: Деструктор виртуальный      {          std::cout << "Calling ~Parent()" << std::endl;      }  };    class Child: public Parent  {  private:      int\* m\_array;    public:      Child(int length)      {          m\_array = new int[length];      }        virtual ~Child() // примечание: Деструктор виртуальный      {          std::cout << "Calling ~Child()" << std::endl;          delete[] m\_array;      }  };    int main()  {      Child \*child = new Child(7);      Parent \*parent = child;      delete parent;        return 0;  } |

Результат выполнения программы:

Calling ~Child()  
Calling ~Parent()

**Правило: При работе с наследованием ваши деструкторы должны быть виртуальными.**