Конструкторы и деструкторы

Конструкторы

- Конструктор – особая функция, являющаяся членом класса и позволяющая инициализировать объекты в момент их создания
- Это означает, что конструктор автоматически вызывается в момент создания объекта, т.е. при его объявлении. При этом:
  - при объявлении локальных объектов конструкторы вызываются каждый раз при входе в соответствующий блок
  - для глобальных и статистических локальных объектов конструкторы вызываются лишь однажды

Объявление конструктора

- Как метод класса.
- Правила синтаксиса:
  - Имя конструктора должно совпадать с именем класса
  - В объявлении конструктора не указывается тип возвращаемого значения, так как они не могут возвращать значения

Определение конструктора

- Внутри класса:

  имя конструктора (параметры) { тело конструктора}
- Вне класса:

  имя класса :: имя конструктора (параметры) {тело конструктора}

Вызов конструктора

В явном виде не вызывается, автоматически запускается при объявлении объекта

Виды конструкторов

- Конструктор без параметров
- Конструктор с параметрами:
  - конструктор с одним параметром
  - конструктор с несколькими параметрами

Конструктор без параметров

```
class church {
        char *name;
        char school;
        unsigned int count;
        float square;
public:
church (); // объявление конструктора
void show(void);
};
church :: church () // определение конструктора
{
        name = "по умолчанию";
        school = 'a';
        unsigned int _count = 0;
        float square = 0;
} конструктора
```

Конструктор автоматически вызывается при объявлении (создании) объекта

То есть объявление объекта в C++ - это не пассивная запись оператора, а активный процесс

```
church obj; // автоматический вызов запуск
```

Конструкторы с одним параметром

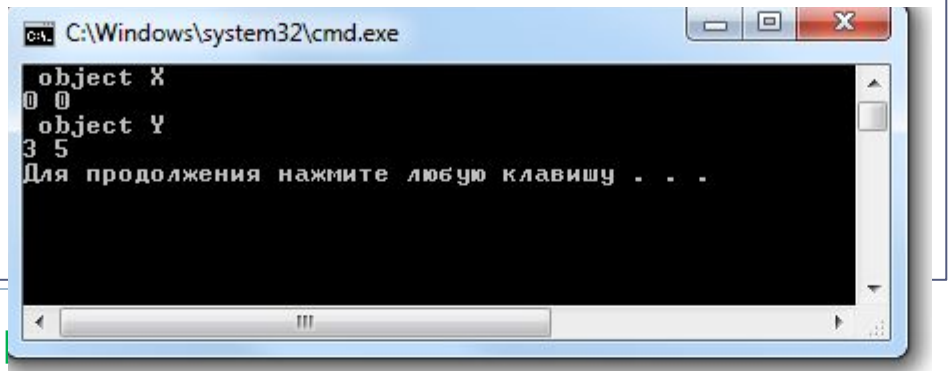
Конструкторам можно передавать аргументы, предназначенные для инициализации объекта

```
#include <iostream>
using namespace std;
class myclass {
  int a;
public:
  myclass (int i) {      a = i;     }
  int geta ()   { return a; }
};
int main()
{
  myclass x = 3; // передает параметру i значение 3
  cout << x.geta(); // выводит на экран 3
  return 0;
}
```

Конструкторы с параметрами

Конструкторам можно передавать аргументы, предназначенные для инициализации объекта

```
#include <iostream>
using namespace std;
class myclass {
  int a;
  int b;
 public:
  myclass (int i =0 , int j=0) {
    a = i;       b = j;
  }
  void show() {
   cout << a << " " << b << "\n"; };
int main(){
```



- **myclass x;** // без аргументо

```
class church {
      char *name;
      char school;
      unsigned int count;
      float square;
public:
church (char* _name = "по умолчанию", char _school = 'a', unsigned int _count = 0, float _square =
0.); //конструктор
~church ();//прототип деструктора
void show(void);
};//конец класса
church::~church () // определение деструктора
{
cout<<"\n работает деструктор - объект уничтожен\n" ;
}
church::church (char* _name, char _school, unsigned int _count, float _square)
{
cout << "работает конструктор\n" ;
name=new char[]; strcpy(name, _name);
school = _school;
count = _count;
square = _square;
}
void church::show(void)
{
cout<<name<<" "; cout<<school<<" "; cout<<count<<" "; cout<<square<<" ";
}

int main() {
church obj; cout<<"результат работы конструктора по умолчанию: \n" ;
obj.show();
cout<<"\n";
obj.~church();

cout<<"\n результат работы конструктора с указанием параметров: \n" ;
church obj1("name", 'r', 10, 10.657);
obj1.show();
obj1.~church();
}
```

Деструкторы

- Деструктор – антипод конструктора, который вызывается автоматически при разрушении объекта
- Имя деструктора совпадает с именем конструктора, но перед ним ставится знак ~ (тильда)

Объявление деструктора

Деструкторы по умолчанию являются открытыми

При объявлении деструкторов действуют несколько правил:

- Не могут иметь аргументов
- Не могут иметь возвращаемого типа (включая **void**)
- Не могут возвращать значение с помощью оператора **return**
- Не могут объявляться как **const**, **volatile** или **static**. Однако их можно вызывать для уничтожения объектов, объявленных как **const**, **volatile** или **static**

Использование деструкторов

Деструкторы вызываются, когда происходит одно из следующих событий:

- Объект, предоставленный с использованием оператора **new**, можно явно освободить с использованием оператора **delete**
- Локальный (автоматический) объект с областью видимости "блок" выходит за пределы области видимости
- Время существования временного объекта заканчивается
- Программа заканчивается, глобальные или статические объекты продолжают существовать
- Деструктор можно явно вызывать

  имя объекта. ~ имя деструктора ()

Ограничения на использование деструкторов:

- Невозможно взять адрес деструктора
- Производные классы не наследуют деструкторы базового класса

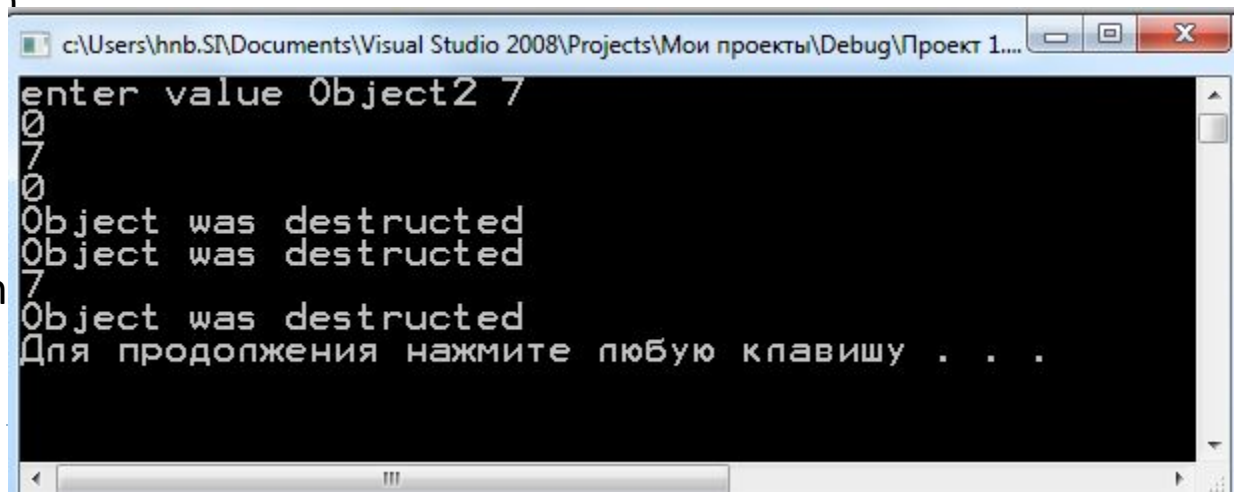
Задание

- К классу Int, имитирующий стандартный тип int добавьте:
  - Конструктор без параметра, инициализирующий поле 0
  - Конструктор с параметром, инициализирующий поле целым числом или 0 по умолчанию
  - Деструктор
- В программе должно быть создано три объекта класса Int.
- Первый и третий объекты должны быть инициализированы конструктором без параметра
- Второй должен быть инициализирован конструктором целым числом, введенным пользователем с клавиатуры
- Сложите два инициализированных объекта и присвойте результат третьему, а затем отобразите результат на экране
- Выполните явный вызов деструктора

```
#include "iostream"
#include "stdlib.h"
using namespace std;
class Int {
    int x;
public:
    Int(int y=0){x=y;}
   ~Int(){
     cout << "Object was destructed" << "\n";};
    void setX(int a);
    void show();
    int summa(Int obj1, Int obj2) {
      return obj1.x+obj2.x;}
};


void Int::show(){cout << x << "\n
void Int::setX(int a){x=a;}


void main()
```



Контрольные вопросы

- Как можно объявить конструктор?
- Как можно определить конструктор?
- Как можно вызвать конструктор?
- Как можно объявить деструктор?
- Как можно определить деструктор?
- Как можно вызвать деструктор?
- Где в приведенном примере срабатывает конструктор и какой конкретно оператор конструктора.
- Объясните, какие события в приведенном примере приводят к вызову деструктора.

Задание на самостоятельную работу

Постановка задачи «Кошелек студента». Владелец кошелька может выполнить следующие действия с кошельком: добавить деньги в кошелек, взять деньги, пересчитать, посмотреть, дать деньги в долг. Источниками пополнения кошелька могут быть родители, также это может быть зарплата или стипендия.

Задание:

- Добавить в разработанные классы задачи «Кошелек студента» необходимые конструкторы и деструкторы