Конструкторы и деструкторы в C++

Тема 2.6 Конструкторы и деструкторы

Конструкторы

- **Конструктор** — **это** особая функция, являющаяся членом класса и инициализирующая экземпляр своего класса.

Конструкторы - продолжение

- Конструктор не возвращает значение, даже типа void. Нельзя получить указатель на конструктор.
- Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации.
- Конструктор, вызываемый без параметров, называется конструктором по умолчанию.
- Параметры конструктора могут иметь любой тип. Можно задавать значения параметров по умолчанию.

Пример создания конструктора:

```
class Account {
private:
   char *_name;
   unsigned int _acct_nmbr;
   double _balance;
public:
   Account(); // конструктор
};
```

Пример создания конструктора с аргументами:

```
public:
    // конструктор
   Account(char *name, int acct, double balance =
   10000.00)
   {
       strcpy(_name, name);
       _acct_nmbr = acct;
       if (balance < 50000.0)
       _balance = balance;
    };
```

Вызов конструктора

Вызов конструктора выполняется, если в программе встретилась одна из конструкций:

```
имя_класса имя_объекта [(список параметров)];
имя_класса (список параметров);
имя_класса имя_объекта = выражение;
```

```
monstr Super(200, 300), Vasia(50), Z;
monstr X = monstr(1000);
monstr Y = 500;
```

Конструктор копирования

```
T::T(T&) {  /* Тело конструктора */ }
```

- при описании нового объекта с инициализацией другим объектом;
- при передаче объекта в функцию по значению;
- при возврате объекта из функции.
- при обработке исключений.

Пример конструктора копирования:

```
monstr::monstr(monstr &M){
 if (M.name){
   name = new char [strlen(M.name) + 1];
   strcpy(name, M.name);} else name = 0;
 health = M.health; ammo = M.ammo;
 skin = M.skin;
}
```

```
monstr Vasia (blue);
monstr Super = Vasia;
monstr *m = new monstr ("Ork");
monstr Green = *m;
```

Деструкторы

Деструктор вызывается автоматически, когда объект выходит из области видимости:

- для *локальных* объектов — при выходе из блока, в котором они объявлены;
- для *глобальных* — как часть процедуры выхода из main;
- для *объектов, заданных через указатели*, деструктор вызывается неявно при использовании операции delete.

Деструкторы - продолжение

- не имеет аргументов и возвращаемого значения;
- не может быть объявлен как const или static;
- не наследуется;
- может быть виртуальным
- Если деструктор явным образом не определен, компилятор автоматически создает пустой деструктор.

Пример деструктора:

```
monstr::~monstr() {/*Тело деструктора*/;}
```

Деструктор можно вызвать явным образом путем указания полностью уточненного имени, например:

```
monstr *m;
m -> ~monstr();
```