ДЕСТРУКТОР

доц. д-р Нора Ангелова

Жизнен цикъл на обект

- Създаване на обект в даден скоуп(област) – заделя се памет и член-данните могат да се инициализират
- Достига се до края на скоупа(област)
- **Обектът и паметта, която е асоциирана с него се разрушава**

Жизнен цикъл на обект

- **Обектът и паметта, която е асоциирана с него се разрушава**
  - Кой ще освободи динамично заделената памет в конструкторите?!?
  - За всяко new трябва да има и съответен delete

ДЕСТРУКТОР

- Носи името на класа
- Няма тип на връщания резултат
- Няма параметри
- Пред името стои знакът ~
- Извиква се при разрушаване на обекти
  - Разрушаване на обект чрез оператора delete
  - Излизане от блок, в който е бил създаден обект на класа
- Извикват се в обратен ред на конструкторите

```
~Point2D() {
  //…
}
```

Ако конструкторът или някоя член-функция реализира динамично заделяне на памет за член-данните, използването на деструктор е задължително, тъй като в този случай той трябва да освободи заетата памет.

ОПЕРАТОР NEW/DELETE

- Създаване и разрушаване на динамично заделен масив от обект/и.

Пример:

```
Point2D * dynamicPointsArr = new Point2D[10];

delete [] dynamicPointsArr;
```

или

```
delete [10] dynamicPointsArr;
```

Как ще се разрушат обектите?

- Деструкторът ще се извика 10 пъти.

* delete [size]- they provide an optimization point for custom implementations:
they are called with the same *size* argument used in the call to the corresponding operator new.
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3536.html

ДЕСТРУКТОР

- Излизане от блок, в който е бил създаден обект на класа

```
class TestClass {
  private:
    int field1;
  public:
    TestClass(int fieldData = 1);
    ~TestClass();
};

TestClass::TestClass(int fieldData) {
    field1 = fieldData;
    std::cout << "TestClass(" << field1 << ")" << std::endl;
}
// Деструктор без освобождаване на памет е излишен в стандартна задача. Тук целта му е да
// демонстрира кога се извика в следната програма
TestClass::~TestClass() {
    std::cout << "~TestClass" << field1 << std::endl;
}

void print() {
   TestClass objFromTestClass;
}

int main() {
    print();
    return 0;
}
```

Резултат:
TestClass(1)
~TestClass1

ДЕСТРУКТОР

- Разрушаване на обект чрез оператора delete

```
class TestClass {
  private:
    int field1;
  public:
    TestClass(int fieldData = 1);
    ~TestClass()
};

TestClass::TestClass(int fieldData) {
    field1 = fieldData;
    std::cout << "TestClass(" << field1 << ")" << std::endl;
}
// Деструктор без освобождаване на памет е излишен в стандартна задача. Тук целта му е да
// демонстрира кога се извика в следната програма
TestClass::~TestClass() {
    std::cout << "~TestClass" << field1 << std::endl;
}

int main() {
  TestClass * ptr = new TestClass;
  delete ptr;

  return 0;
}
```

Резултат:
TestClass(1)
~TestClass1

ВРЕМЕ ЗА ВАШИТЕ ВЪПРОСИ