**38.Виртуални деструктори**

Деструктурът е функция, която се извиква при унищожаването на даден обект. Името

на деструктура на един клас е името на класа, предшествано от ~. Локалните обекти се

унищожават, когато излязат извън областта си на видимост, а глобалните – когато

завърши изпълнението на програмата. Ако е заделен ръчно, той бива освободен при

изход на програмата или чрез delete.

При наследяването на класове дестурктурът на един производен клас трябва да се

изпълни преди деструктура на базовия клас. Ако деструкторът на базовия клас се

изпълняваше пръв, това би довело до унищожаването на производния клас.

Пример:

class Base {

public:

~Base(){ ... }

};

class Derived: public Base {

public:

~Derived(){ ... }

};

void main() {

Base \*Var = new Derived();

delete Var;

}

Създава се указател от типа на базовия клас, сочещ към нов обект от типа на

производния. При изтриване на указателя от типа на базовия клас директно ще се

извика деструктора на базовия клас. Когато се извика деструктора на базовия клас,

полетата липсващи в базовия клас Base специфични за наследения клас няма да бъдат

унищожени => разрушаването на производния клас ще остане незавършено и

специфичните полета няма да бъдат освободени от паметта т.е. програмата ще стане

следното:

Constructor: Base

Constructor: Derived

Destructor : Base

Проблемът се решава като се обяви деструкторът на базовия клас за виртуален.

Обявяването на деструктора на един клас за виртуален води до това, че всички

деструктори на класове по-надолу от него в йерархията се разглеждат като виртуални и

всичките се извикват в правилния ред. Т.е. ако сложиме:

virtual ~Base(){ ... }

В този случай като изтрием указателя от базовия клас, компилаторът ще провери кой

подходящ конструктор да извика, а именно този на наследеният.