КОНСТРУКТОРИ И ДЕСТРУКТОРИ

1. Цел на упражнението

Усвояване на умения за работа с управляващи методи на класовете - конструктори и деструктори.

2. Предназначение, синтаксис и използване на конструктори и деструктори

В C++ са предвидени специални езикови конструкции за реализиране на предварителни и завършващи действия. Синтактичната конструкция за предварителни действия се нарича конструктор, а тази за заключителните действия - деструктор. Тези наименования се обясняват с това, че конструкторът реализира някои езикови конструкции, а деструкторът, обратно, освобождава от действието на тези конструкции.

Конструкторът и деструкторът са методи на съответния клас. Конструкторът е метод, който се извиква автоматично при дефинирането на обект, след резервирането на памет за неговите компоненти. Деструкторът се изпълнява автоматично непосредствено преди освобождаването на паметта, която е заета от обекта. Като методи конструкторът и деструкторът се подчиняват на всичките изисквания за членове-функции: декларират се в секция **`public`** на класа, могат да се дефинират в тялото на класа или извън него.

Конструкторите и деструкторите имат следните основни характеристики:

⇒ ***Имената на конструкторите на даден клас съвпадат с името на този клас.*** Когато един конструктор се дефинира, заглавието ще бъде (в случай, че дефиницията е извън класа)

```
<име_клас> :: <име_клас> ( <формални_параметри> )
{
     <тяло на конструктор>
}
```

Тип на връщания резултат не се посочва.

⇒ ***Конструкторът може да няма параметри.*** Тогава той се нарича конструктор по подразбиране. Ако в даден случай не е указано кой конструктор трябва да се изпълни, а ситуацията изисква такова изпълнение, ще се изпълни конструкторът по подразбиране. Конструкторът по подразбиране се оформя като функция без параметри:

```
<име_клас> :: <име_клас> ( void )
{
     <тяло на конструктор>
}
```

⇒ ***Деструкторът има име, съвпадащо с това на класа с префикс тилда “~”.*** По аналогия с математиката символът ~ означава “допълнението

към конструктора” (извършва действия, “обратни” на тези на конструктора). Деструкторът няма параметри. В дефиницията му не се посочва връщан резултат.

⇒ ***Един клас може да има само един деструктор.***

⇒ ***Един клас може да няма конструктори или дерструктор.***

⇒ Няма значение как са подредени конструкторите и деструкторът в тялото на класа. Общоприето е конструкторите да са първите методи, а деструкторът да е последният метод в тялото на класа.

⇒ Конструктор и деструктор се дефинират за класа, но те могат да се използват от всеки обект от този клас.

⇒ Типовете на върнатите стойности на конструкторите и деструкторите са съответно указател към новосъздаден обект (указателят this) и void. Тези типове не могат да бъдат променяни, поради което в декларациите, респективно дефинициите на конструкторите и деструкторите, не се задава тип на върнатата стойност.

⇒ ***Конструкторите не могат да се извикват явно.*** Деструкторите могат да бъдат извиквани явно чрез указване на тяхното пълно име, състоящо се от името на класа, оператора :: и името на деструктора.

Всяка декларация на обект поражда неявно обръщение към конструктор на класа, при което се прави пълен контрол на типовете. Съществува явен и съкратен начин за предаване на аргументи на конструктор:

- явен
```
String searchWord = String(“rose”);
```
- съкратен
```
String commonWord(“the”);
String inBuf = 1024;
```
- използването на new изисква явно предаване
```
String *ptrBuf = new String (1024);
```

Един и същ клас може да има няколко конструктора. Всички те трябва да имат едно и също име (името на класа), но трябва да се различават по брой и/или тип на параметрите си. Такива конструктори се наричат предефинирани (overloading constructors). При създаването на обекти се изпълнява само един от предефинираните конструктори в зависимост от броя и/или типа на предаваните фактически параметри. Предефинираните конструктори дават възможност различните обекти на един и същ клас да бъдат инициализирани по различен начин.

3. Задачи за изпълнение

Задача 1:

Да се добави метод към класа **OneData**, който ще връща квадрата на `dataStore`. Да се добавят в главната програма оператори за извеждане на изчислената стойност.

Декларацията и дефинициите на методите на класа **OneData** е представена във файла `H:\SKELET\EX4_1.CPP`.

Да се създаде проект с име `EX41`.

Файлът `EX4_1.CPP` да се добави към създадения проект (виж "Добавяне на .cpp файл към съществуващ проект").

Проектът да се компилира и изпълни.

Да се анализират получените резултати.

Задача 2:

Към класа **OneData** да се добави конструктор за инициализиране със стойност по подразбиране. В главната програма да се добавят оператори за извеждане на инициализираната стойност.

Задача 3:

Да се замени функцията `setData` на класа **Person** с конструктор по подразбиране, да се добавят инициализиращ и копиращ конструктори. Да се добави деструктор на същия клас. Да се добавят оператори за извеждане на съобщения в конструктора и в деструктора на класа **Person**, които да сигнализират входа и изхода в/от областта на действие на класа. Функцията `main` да се преработи, като се предвидят подходящи оператори за тестване на създадените методи.

Декларацията и дефинициите на методите на класа **Person** е представена във файла `H:\SKELET\EX4_3.CPP`.

Да се създаде проект с име `EX43`.

Файлът `EX4_3.CPP` да се добави към създадения проект.

Задача 4:

Да се замени функцията `inputComplex` на класа **ComplexNum** с конструктор по подразбиране, да се добавят инициализиращ и копиращ конструктори. Да се добави деструктор на същия клас. Функцията `main` да се преработи, като се предвидят подходящи оператори за тестване на създадените методи.

Декларацията и дефинициите на методите на класа **Person** е представена във файла `H:\SKELET\EX4_4.CPP`.

Да се създаде проект с име `EX44`.

Файлът `EX4_4.CPP` да се добави към създадения проект.

Задача 5:

В декларацията на класа **Time** да се добави инициализиращ конструктор с подразбиращи се стойности, който да извиква функцията `setTime`. Функцията `main` да се преработи, като се предвидят подходящи оператори за тестване на създадените методи.

Декларацията и дефинициите на методите на класа **Time** е представена във файла `H:\SKELET\EX4_5.CPP`.

Да се създаде проект с име `EX45`.

Файлът `EX4_5.CPP` да се добави към създадения проект.

Задача 6:

Към класа **IntArray** да се дефинира конструктор с аргумент размера на масива, който да заделя памет за масива и да инициализира неговите елементи с определена стойност. Да се дефинира конструктор с аргумент обект от клас **IntArray**, който да копира елементите на аргумента в елементите на текущия обект. Да се дефинира деструктор, който да освобождава заетата от масива памет.

Декларацията и дефинициите на методите на класа **IntArray** е представена във файла `H:\SKELET\EX4_6.CPP`.

Да се създаде проект с име `EX46`.

Файлът `EX4_6.CPP` да се добави към създадения проект.

Задача 7:

Към класа **IntArray** да се дефинират следните методи: за задаване стойности на елементите на масива; за извеждане стойностите на елементите на масива; за търсене на минимален елемент и за сортиране елементите на масива по метода на мехурчето.

Задача 8:

Да се декларира клас, който съхранява лично, бащино и фамилно име и връща пълното име на лицето в един от следните формати:

Иван Павлов Димитров

И.П.Димитров

Димитров, Иван П.

или какъв да е друг формат по избор.