# НАСЛЕДЯВАНЕ. ПРОИЗВОДНИ КЛАСОВЕ

### *1. Цел на упражнението*

Усвояване на практически умения при приложение на механизма на наследяване и работа с производни класове.

### *2. Механизъм на наследяване*

В обектно-ориентираното програмиране се използва един мощен механизъм за построяване на отношения между класове, наречен *механизъм на наследяване*. Този механизъм позволява на един клас да наследи нещо от друг клас. Наследяването се осъществява чрез използване на ***производни класове*** (derived classes). Класът, който наследява, се нарича ***наследник***, а този, от който се наследява - ***предшественик***.

Класовете, които наследяват от друг клас, се наричат ***производни*** класове. Класът, от който се наследява, се нарича ***базов***. Процесът на наследяване по отношение на производните класове се изразява в следното:

- наследява се структурата на данните на базовия клас и неговите методи;
- получава се достъп до наследените членове на базовия клас;
- производният клас познава реализацията само на базовия клас, от който произлиза;
- производен клас може да изпълнява функцията на *базов* за други класове; това правило определя създаването на йерархична структура на програмата.

Тъй като наследяването се разглежда и като еволюционен процес, производният клас може да дефинира допълнително:

- свои собствени данни;
- методи, аналогични на тези на базовия клас, но видоизменени;
- нови методи.

### *3. Деклариране на производни класове (просто наследяване)*

Под ***просто наследяване*** се разбира, че производният клас наследява от единствен базов клас. Декларацията на производния клас е следната:

```
class<име_производен_клас>:<атрибут_за_област><име_базов_клас>
{
     <тяло на производния клас>
};
```

където ***<атрибут_за_област>*** може да бъде една от ключовите думи ***public*** или ***private***. Атрибутът определя каква област на действие ще имат наследените членове.

Производният клас автоматично (след декларирането му) наследява като дефиниции всичките членове-данни на базовия клас. С други думи, в тялото на производния клас автоматично се копират моделите на локалната секция ***private*** и глобалната секция ***public*** на базовия клас. Освен това, производният клас получава възможността да използва всичките нелокални методи на базовия клас. Методите могат да се извикват чрез обекти и на базовия, и на производния клас - използва се общ код на функциите. Производен клас

може да се декларира, след като вече е деклариран базовият клас, от който той произлиза.

Секция от вида ***protected***, дефинирана в базов клас, се наследява като такава от производния клас. За базовия клас защитената секция е като локалната. До нея имат директен достъп само методи и приятелски функции на класа. Ако от клас със защитена секция се получи производен клас, то неговите методи (ненаследени) и приятелските му функции имат директен достъп до членовете на наследената защитена секция. Следователно, защитената секция се дефинира обикновено в случаите, когато се очаква да се дефинират производни класове.

Достъпът до наследените компоненти на производните класове се определя от два фактора:

- от начина на деклариране на компонентите в базовите класове (***public, protected, private***);
- от начина на деклариране на базовите класове в декларациите на производните класове (***public, private***).

Пред всяко от имената на базовите класове в декларацията на производните класове може да се постави една от ключовите думи ***private*** или ***public***, като ***private*** е по подразбиране. Ако пред името на един базов клас в декларацията на производен клас се постави ключовата дума ***public***, компонентите, които се наследяват от базовия клас, ще бъдат наследени заедно с техните декларации (***public, protected, private***), указани в базовия клас.

Ако вместо ***public***, пред името на базовия клас се постави спецификаторът ***private***, наследените компоненти, които са декларирани като ***public*** или ***protected*** в базовия клас, ще придобият статут на ***private*** в производния клас.

### *4. Конструктори и деструктори на производни класове*

Принципен е въпросът как да се инициализира наследената част на производния клас. Най-естествено е това да се възложи на конструктора на производния клас. Но ако е така, конструкторът на производния клас ще има достъп до наследените локални членове от базовия клас. Това противоречи на принципа за скриване на информацията. Затова функциите по инициализирането са разделени между конструкторите на производния и на базовия клас.

**Конструкторът на производния клас инициализира само новите собствени членове на този клас**. Наследените членове на производния клас се инициализират от конструктор на базовия клас. Как се активира конструкторът на базовия клас? Това става в дефиницията на конструктора на производния клас:

```
<име_произв._клас>::<име_произв._клас>(<параметри>):<иниц._списък>
     { <тяло_на_конструктора> }
```

където ***<инициализиращ_списък>*** е обръщение към конструктор на базовия клас.

**Ред на активиране на конструкторите**. Най-напред се извиква конструкторът на базовия клас, а след това този на производния. По-точно процедурата е следната:

- при обръщение към конструктор на производен клас стандартно се заменят формалните параметри с фактическите;
- изпълнява се обръщение към конструктора на базовия клас и изпълнение на операторите в тялото му;
- изпълняват се операторите в тялото на конструктора на производния клас.

Базовият клас може да няма конструктор, да има конструктор без параметри или такъв неизискащ параметри (параметрите му са подразбиращи се). Обръщения към такива конструктори не се включват в инициализиращия списък.

**Ако производният клас има членове-обекти**, то техните конструктори се извикват преди конструктора на производния клас. Конструкторите на обектите се извикват по реда на тяхното деклариране в тялото на производния клас.

Деструкторите се извикват по ред, обратен на този на активиране на конструкторите.

### *5. Множествено наследяване*

Когато един призводен клас има няколко базови класа, се осъществява ***множествено наследяване*** (multiple inheritance). В резултат на това, производният клас наследява компонентите на всички базови класове.

По принцип няма значение как се изброяват базовите класове в списъка на производен клас. Но веднъж фиксирана, последователността на базовите класове в този списък се използва в редица случаи:

- наследените части на производния клас се разполагат в паметта в съответствие с последователността на базовите класове в декларацията му;
- неявното извикване на конструкторите (деструкторите) на базовите класове се извършва съгласно тази последователност;
- няма значение в какъв ред са записани явно конструкторите на базовите класове в инициализиращия списък на конструктора на производния клас - те ще се активират съгласно последователността на базовите класове в декларацията на производния клас.

Когато се използва множествено наследяване, трябва да се спазва следното правило: изискванията към базовите класове и съгласуването им с производния клас са такива, както в случай на просто наследяване.

### *6. Задачи за изпълнение*

**Задача 1:**

Декларациите и дефинициите на методите на класовете **`Vehicle, Car`** и **`Truck`** са представени във файла `H:\SKELET\EX6_1.CPP`.

Да се създаде проект с име `EX61`.

Файлът `EX6_1.CPP` да се добави към създадения проект.

Проектът да се компилира и изпълни.

Да се анализират получените резултати.

**Задача 2:**

Към декларациите от **Задача 1** да се добави друг обект от клас **`Vehicle`**, наречен велосипед (**`bicycle`**) и да се тестват възможните за него методи. Да се добави обект **`ford`** от клас **`Car`** и се извикат всички достъпни за обекта методи.

**Задача 3:**

Към йерархията **Задача 1** да се добавят два нови класа, луксозен автомобил (**`LuxCar`**) и спортен автомобил (**`SportsCar`**), наследяващи класа **`Car`**. Да се добавят нови признаци, характеризиращи класовете и съответни методи за използването им.

**Задача 4:**

Декларациите на класа **`Point`** и на класа **`Circle`** са представени във файла `H:\SKELET\EX6_2.CPP`.

Да се създаде проект с име `EX62`.

Файлът `EX6_2.CPP` да се добави към създадения проект.

Да се дефинират следните методи, декларирани в класа **`Point`**:

- ***setX*** – установява нова стойност на X;
- ***getX*** – връща текущата стойност на X;
- ***setY*** – установява нова стойност на Y;
- ***getY*** – връща текущата стойност на Y;
- ***show*** – визуализира текущите координати на точката.

Да се дефинират следните методи, декларирани в класа **`Circle`**:

- ***setRadius*** – задава нова стойност на радиуса. Ако новата стойност е <0, на **`radius`** се присвоява 0, в противен случай радиусът приема зададената стойност;
- ***getRadius*** – връща текущата стойност на радиуса;
- ***getDiameter*** – връща диаметъра на окръжността;
- ***getCircumference*** – връща дължината на окръжността;
- ***getArea*** – връща лицето на кръга;
- ***show*** – визуализира координатите на центъра на окръжността и дължината на радиуса.

Функцията **`main`** да се преработи, като се предвидят подходящи оператори за тестване на създадените методи.

**Задача 5:**

Декларациите на класовете **Point** и **Circle**, както и дефинициите на техните методи са представени във файла `H:\SKELET\EX6_3.CPP`.

Да се създаде проект с име `EX63`.

Файлът `EX6_3.CPP` да се добави към създадения проект.

Да се декларира клас **Cylinder**, който наследява от класа **Circle**. Към класа **Cylinder** да се добави атрибут, отразяващ височината на цилиндъра.

Да се декларират и дефинират следните методи за класа **Cylinder**:

- ***setHeidht*** – задава нова стойност на височината на цилиндъра. Ако новата стойност е <0, на атрибута „височина" се присвоява 0, в противен случай височината приема зададената стойност;
- ***getHeight*** – връща височината на цилиндъра;
- ***getArea*** – връща околната повърхнина на цилиндъра;
- ***getVolume*** – връща обема на цилиндъра;
- ***show*** – визуализира координатите на центъра на основата, дължината на радиуса и височината на цилиндъра.

Функцията **main** да се преработи, като се предвидят подходящи оператори за тестване на създадените методи.

**Задача 6:**

Да се декларира клас **Rectangle**, наследяващ класа **Point** (X и Y – координати на горния ляв ъгъл) и членове-данни "дължина" и "ширина". Да се дефинират методи, аналогични на тези от клас **Circle**. Направените дефиниции да се тестват за различни обекти.

**Задача 7:**

Да се декларира клас **Paral (паралелепипед)**, наследяващ класа **Rectangle** и членове-данни "височина". Да се дефинират методи, аналогични на тези от клас **Cylinder**. Направените дефиниции да се тестват за различни обекти.