ВИРТУАЛНИ ФУНКЦИИ И ПОЛИМОРФИЗЪМ

1. Цел на упражнението

Усвояване работата с виртуални функции.

2. Механизъм на динамичното свързване

Метод на клас може да бъде активиран чрез указател към обект от този клас или чрез алтернативно име на такъв обект. Следователно, какъв метод ще бъде активиран зависи от това, ***каква стойност има указателят (алтернативното име), чрез който се извиква методът***. Тази стойност не е известна след завършване на процеса на компилация. Тя става известна по време на изпълнение на програмата. Ето защо по аналогия със статичното свързване се казва, че в случая има ***динамично свързване***. Този механизъм на свързване може да бъде обобщен при използване на базов клас и на производни класове. *Трябва да се има предвид, че на указател към обект от базов клас може да се присвои указател към обект от производен клас и също така формален параметър на функция, който е указател към обект от базов клас, може да се замени с фактически параметър, който е указател към обект от производен клас.* Схемата на разглеждания механизъм е следната:

- Определят се методите, които ще се извикват чрез механизма. За да бъдат посочени тези методи, се използва ключовата дума ***`virtual`***. Такива методи се наричат ***виртуални***.
- *Виртуалните функции се извикват чрез указател или алтернативно име към обект от базовия клас*.
- Указателят, чрез който се активира виртуална функция, може да приема стойности на указатели към обекти на производни класове.

Предимството на виртуалните функции е, че детайлите на реализацията са скрити за програмиста. При промени в наследствената йерархия е необходимо само да се добавят декларациите за новия клас. Класът ще бъде включен автоматично в механизма на динамичното свързване при компилиране на програмата.

3. Дефиниране на виртуални функции

Виртуалните методи имат следните особености:

- Виртуален метод се декларира чрез ключовата дума ***`virtual`***, поставена пред неговото име;
- Статичните методи не могат да бъдат виртуални;
- Виртуални методи не могат да се декларират като приятелски на други класове;
- Ако в даден клас бъде дефиниран виртуален метод и след това в производен на дадения клас бъде дефиниран метод със същия прототип, то методът в производния клас ще бъде също виртуален, дори ако модификаторът ***`virtual`*** бъде пропуснат.

Виртуална функция може да бъде само член-функция на клас. Модификаторът ***`virtual`*** стои на първо място в декларацията на метода.

Виртуалните функции се извикват чрез алтернативно име или указател към обект от базовия клас. Базовият клас трябва да бъде от вида **`public`**. Виртуалната функция може да се дефинира в тялото на класа или извън него. В дефиницията на функцията извън класа не трябва да се използва ключовата дума ***`virtual`***.

Чрез използване на виртуални методи се постигат два много полезни ефекта:

- Ако даден виртуален метод не е дефиниран в един производен клас, но бъде извикан, то автоматично се извиква виртуалният метод от неговия базов клас. Ако в базовия клас също не е дефиниран такъв метод, търсенето продължава в следващото ниво на йерархията на класовете, докато бъде намерен извиканият метод или докато се изчерпят всички класове в йерархята.
- На указател към даден клас могат да се присвояват адресите на обекти, чиито класове са производни на дадения. Извикването на виртуален метод чрез указателя води до изпълнение на този метод, който е дефиниран за обекта (по-точно за класа на обекта), към който сочи указателят в момента на извикването, а не до изпълнение на метода, дефиниран за класа, който е тип на указателя. По този начин чрез една и съща синтактична конструкция (извикване на виртуален метод чрез указател) могат да се извикват различни методи в зависимост от обекта, към който е насочен указателят.

Целият виртуален механизъм се управлява неявно от компилатора. Ако името, сигнатурата или типът на резултата на предекларираната виртуална функция не съответстват точно на тези в първата дефиниция на функцията, тя няма да е виртуална за съответния производен клас.

4. Виртуални деструктори

Когато деструкторът на базовия клас е обявен за виртуален, деструкторите на производните класове в наследствената йерархия също се разглеждат като виртуални.

Ако класът има виртуална функция и деструктор, най-добре е да се обяви деструкторът за виртуален. Това спестява проблемите по определяне на подходящия деструктор.

5. Абстрактни базови класове

В класовете могат да се декларират ***неопределени виртуални методи*** (чисти виртуални функции). Неопределен виртуален метод е този, който има само декларация, но няма дефиниция. За да се укаже, че един метод е неопределен, се използва присвояване на стойност 0 в декларацията на метода. **Клас, в който е деклариран поне един неопределен виртуален метод, се нарича абстрактен клас**. Обикновените класове (които нямат чисти виртуални функции) се наричат реални.

Абстрактните класове имат следните особености в сравнение с реалните класове:

- Обекти на абстрактни класове не могат да бъдат създавани;
- Абстрактните класове могат да се използват само като базови за други класове;

- Абстрактните класове не могат да се използват като тип на параметри и тип на върнатата стойност на функциите;
- Методите на абстрактните класове могат да се извикват от техните конструктори, но извикване на чиста виртуална функция не е възможно и води до грешка по време на изпълнение на програмите;
- Допустимо е дефиниране и използване на указатели към абстрактни класове и псевдоними на абстрактни класове.

Едно характерно приложение на абстрактните класове и виртуалните методи е свързано със създаване на хетерогенни структури от данни (списъци, графи, дървета). Техните елементи могат да бъдат обекти от различни класове.

6. Полиморфизъм

Полиморфизмът е една от отличителните характеристики на ОО езици. Той се изразява в това, че едни и същи действия (в общ смисъл) се реализират по различен начин в зависимост от обектите, върху които се прилагат, т.е. ***действията са полиморфични***. В C++ полиморфизмът се реализира чрез виртуални методи.

За да се реализира едно полиморфично действие, класовете на обектите, върху които то ще се прилага, трябва да имат общ корен, т.е. да бъдат производни на един и същ клас. В този клас трябва да бъде дефиниран виртуален метод, съответстващ на полиморфичното действие. Във всеки от производните класове този метод може да бъде предефиниран съобразно особеностите на конкретния клас. Активирането на полиморфичното действие трябва да става чрез указател към базовия клас, на който могат да се присвояват адресите на обекти на който и да е клас от йерархията. Съгласно механизма на извикване на виртуалните методи, ще бъде изпълняван методът на съответния обект, т.е. в зависимост от обекта към който сочи указателят ще бъде изпълняван един или друг метод.

Ако класовете, в които трябва да се дефинират виртуалните методи нямат общ базов клас, то такъв може да бъде създаден изкуствено чрез използване на един абстрактен клас. Този клас ще съдържа само ***неопределени виртуални методи*** (чисти виртуални функции), които трябва да се дефинират в производните класове.

7. Задачи за изпълнение

Задача 1:

Декларациите и дефинициите на методите на класовете **`Parent, Child`** и **`GrandChild`** са представени във файла `H:\SKELET\EX7_1.CPP`.

Да се създаде проект с име `EX71`.

Файлът `EX7_1.CPP` да се добави към създадения проект.

Проектът да се компилира и изпълни.

Да се анализират получените резултати.

Задача 2:

Към декларациите на класовете от **Задача 1** да се добави нов виртуален метод, наречен **`getClassName`**, който връща като символен низ името на съответния клас.

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

Задача 3:

Към декларациите от **Задача 2** да се добавят нови членове-данни и нов виртуален метод, наречен **birthPlace**, който извежда родното място: държавата за класа **Parent**, града за класа **Child** и адреса за класа **GrandChild**.

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

Задача 4:

Декларациите на класовете **Point, Circle** и **Cylinder**, както и дефинициите на техните методи са представени във файла `H:\SKELET\EX7_2.CPP`.

Да се създаде проект с име `EX72`.

Файлът `EX7_2.CPP` да се добави към създадения проект.

Към декларациите на класовете да се добави нов виртуален метод, наречен **getName**, който връща като символен низ името на съответния клас.

Във функцията **main** да се създаде масив от указатели от типа на базовия клас, към който да се добавят елементи (указатели), които да сочат към обекти от класовете на йерархията. Да се предвиди виртуално извикване на метода **getName**.

Да се реализират две версии на функцията **main**. В едната виртуалният метод да се извика чрез указател, а в другата – чрез алтернативно име.

Задача 5:

Кои други функции могат да бъдат декларирани като виртуални за представената в **Задача 4** йерархия от класове **Point, Circle** и **Cylinder**? Това трябва да бъдат функции, които имат полиморфично действие върху цялата наследствена структура.

Да се направят необходимите промени в декларациите на тези методи.

Към йерархията от класове **Point, Circle** и **Cylinder** да се добави абстрактен базов клас **Shape**, в който методът **getName** да бъде чисто виртуална функция. Не е необходимо класът **Shape** да има членове-данни. Класът **Point** наследява от **Shape**. Декларациите на виртуалните методи да се добавят в класа **Shape**.

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