Лабораторна робота №5. Поліморфізм та віртуальні функції

МЕТА РОБОТИ: освоїти механізм пізнього зв’язування через використання віртуальних функції при простому й множинному успадкуванні в класах.

5.1. Програма роботи

5.1.1. Отримати завдання. Ознайомитись з правилами оголошення та використання віртуальних функцій.

5.1.2. Написати програми відповідних класів, основну та відповідні допоміжні функції, згідно з вказівками до виконання роботи.

5.1.3. Підготувати власні коректні вхідні дані (вказати їх формат і значення) і проаналізувати їх.

5.1.4. Оформити електронний звіт про роботу та захистити її.

5.2. Вказівки до виконання роботи

5.2.1. Студент, згідно з індивідуальним номером, вибирає своє завдання з розд. 5.4 і записує його до звіту.

5.2.2. Оголошення класу (структури), основну та відповідні допоміжні функції необхідно запрограмувати так, як це показано у розд. 7.4.

5.2.3. Власних вхідних даних необхідно підготувати не менше двох комплектів. Їхні значення мають бути коректними, знаходитися в розумних межах і відповідати тим умовам, які стосуються індивідуального завдання.

5.2.4. Звіт має містити такі розділи:

* мету роботи та завдання з записаною умовою задачі;
* коди всіх використовуваних .h і .ccp файлів, а також пояснення до них;
* результати реалізації програми, які виведені в вікно форми;
* діаграму класів та варіантів використання з поясненням;
* висновки, в яких наводиться призначення програми, обмеження на її застосування і можливі варіанти удосконалення, якщо такі є.

5.3. Теоретичні відомості

**Поліморфізм**

***Поліморфізм*** в об’єктно – орієнтованому проектуванні – це властивість вказівника (або посилання) на базовий клас, проініціалізованого адресою (або іменем – для посилання) об’єкта похідного класу, викликати віртуальні методи похідного класу у межах дії поліморфічного кластера.

Сукупність класів, у яких оголошується, визначається та перевизначається віртуальний метод, називається ***поліморфічним кластером***. Поліморфічний кластер реалізується тільки для ***public – успадкувань класів***. В одній ієрархії успадкування класів може бути визначено декілька різних поліморфічних кластерів.

У С++ є такі види поліморфізму:

* ***статичний*** – перевантаження і перевизначення функцій та операцій; шаблони функцій та класів;
* ***динамічний*** – перевизначення віртуальних функцій.

Статичний поліморфізм забезпечується під час компіляції програми, а динамічний – під час її виконання.

Поліморфізм перевантаження функцій (overload) полягає у можливості вибору компілятором різних реалізацій однієї і тієї самої функції залежно від кількості або типів її аргументів. Наприклад:

#include <iostream>

using namespace std;

class A {

public:

void f ( ) { cout<< “A::f ( )”<<endl; }

void f (int x) { cout<< “A::f (int)”<<endl; }

void f (double x) { cout<< “A::f (double)”<<endl; }

void f (int, int) { cout<< “A::f (int, int)”<<endl; }

void f (int, int, int) { cout<< “A::f (int, int, int)”<<endl; }

};

void main ( ) {

A a;

a.f ( ); // A::f ( )

a.f (1); // A::f (int)

a.f (3.14); // A::f (double)

a.f (2, 3) // A::f (int, int)

a.f (4, 5, 6) // A::f (int, int, int)

}

У цій програмі здійснюються виклики перевантажених методів A::f ( ) класу A. Перевантажені методи мають одинакове ім’я, але можуть відрізнятись їх внутрішньою реалізацією.

У контексті успадкування класів поліморфізм полягає у перевизначенні (override) нащадком методів базового класу. Тоді для роботи з об’єктами похідних класів використовується інтерфейс їх базового класу, наприклад:

#include <iostream>

using namespace std;

class A {

public:

void output ( ) const { cout<< “A::output ( )”<<endl; }

};

class B : public A {

public:

void output ( ) const { cout<< “B::output ( )”<<endl; }

};

class C : public A {

public:

void output ( ) const { cout<< “C::output ( )”<<endl; }

};

void main ( ) {

C \*p1 = new C;

p1 –>output ( ); // C::output ( )

A \*p2 = p1; // p1 == p2

p2 –>output ( ); // A::output ( )

B \*p3 = new B;

p3 –>output ( ); // B::output ( )

A \*p4 = p3; // p3 == p4

p4 –>output ( ); // A::output ( )

}

У базовому класі А визначено метод output ( ), який перевизначається у похідних класах В та С. У функції main ( ) вказівник р2 на клас А проініціалізовано вказівником р1 на клас С. Така ініціалізація допускається без явного перетворення типу, оскільки клас С є похідним від класу А. Проініціалізовані вказівники використовуються для виклику методу output ( ). Незважаючи на те, що вказівники р1 та р2 набувають однакових значень, за допомогою р1 буде викликано метол С::output ( ), а за допомогою р2 – метод A::output ( ). Аналогічні дії відбудуться з вказівниками р3 та р4 на класи В та А. Те, який метод буде викликано за допомогою того чи іншого вказівника, визначатиметься під час компіляції програми (раннє зв’язування).

Розглянуті вище види поліморфізму є ***статичними***. На відміну від них, при ***динамічному***поліморфізмі вибір того або іншого методу здійснюється під час виконання програми застосуванням механізму пізнього зв’язування. Динамічний поліморфізм реалізується на основі віртуальних методів. Наприклад:

#include <iostream>

using nsmespace std;

class A {

public:

virtual void output ( ) const { cout<< “A::output ( )”<<endl; }

};

class B : public A {

public:

virtual void output ( ) const { cout<< “B::output ( )”<<endl; }

};

class C : public A {

public:

virtual void output ( ) const { cout<< “C::output ( )”<<endl; }

};

void main ( ) {

C \*p1 = new C;

p1 –> output ( ); // C::output ( )

A \*p2 = p1; // p1 == p2

p2 –> output ( ); // C::output

B \*p3 = new B;

p3 –> output ( ); //B::output ( )

A \*p4 = p3; // p3 == p4

p4 –> output ( ); // B:: output ( )

}

Тепер за допомогою вказівників p2 та p4 будуть викликані віртуальні методи output ( ) з похідних класів С та В відповідно.

**Віртуальні методи класу. Раннє та пізнє зв’язування**

Термін “зв’язування” використовується для визначення особливостей виклику методів класів за допомогою їх об’єктів. Розрізняють раннє та пізнє зв’язування.

За ***раннього зв’язування*** поєднання методів з об’єктами класу здійснюється на етапі компіляції програми. Це означає, що за допомогою об’єкта класу, посилання на вказівника на клас можна викликати тільки методи цього самого класу (залежно від прав доступу). Якщо об’єкт, посилання або вказівник на базовий клас проініціалізовані об’єктом (або його адресою – для вказівника) похідного класу, то за їх допомогою не можна забезпечити виклик методів похідного класу.

***Пізнє зв’язування*** реалізується за допомогою віртуальних методів (віртуальних функцій). ***Віртуальний метод*** – це сукупність перевизначених функцій – членів (з одинаковим іменем, списком параметрів та типом результату) в ієрархії успадкування класів, оголошених зі словом *virtual*. При пізньому зв’язуванні поєднання віртуальних методів з об’єктами здійснюється під час виконання програми. Це означає, що за допомогою вказівника або посилання на базовий клас, проініціалізованих відповідно адресою або іменем об’єкта похідного класу, можна викликати віртуальні методи похідного класу.

Пізнє зв’язування забезпечує динамічний поліморфізм, коли для роботи з похідним класом використовується інтерфейс (набір віртуальних методів) базового класу. Динамічний поліморфізм ґрунтується на перевизначенні віртуальних методів, які можуть бути звичайними (неіндексованими) або динамічними (індексованими). Пізнє зв’язування не підтримується, якщо віртуальні методи викликаються з конструкторів або деструктора класу.

***Суто віртуальна функція*** – це віртуальна функція, яка немає визначення в базовому класі. Тому будь – який похідний тип повинен визначити власну версію цієї функції, адже у нього просто немає ніякої можливості використовувати версію з базового класу (через її відсутність). Щоб оголосити суто віртуальну функцію, використовують такий загальний формат:

**virtual** *тип ім'я\_функції* (*список\_параметрів*) = 0;

Оголосивши функцію суто віртуальною, програміст створює умови, при яких похідний клас просто вимушений мати визначення власної її реалізації. Без цього компілятор видасть повідомлення про помилку.

Якщо клас містить хоч би одну суто віртуальну функцію, то він називається ***абстрактним***. Абстрактний клас характеризує одна важлива особливість: у такого класу не може бути об'єктів. Абстрактний клас можна використовувати тільки як базовий, з якого виводитимуться інші класи.

***Особливості віртуальних методів:***

1. віртуальними можуть бути тільки функції – члени класу, друзі не можуть бути віртуальними;
2. шаблонні методи (з оголошенням *template*) не можуть бути віртуальними;
3. віртуальні методи можуть бути оголошені у класі або структурі, об’єднання не може мати віртуальних методів;
4. слово *virtual* достатньо записати тільки у першому оголошенні віртуальних методів;
5. віртуальні методи не можуть бути статичними (*static*);
6. віртуальний метод повинен бути визначений у класі або оголошений чистим;
7. для забезпечення пізнього зв’язування віртуальні методи повинні викликатись за допомогою вказівників або посилань;
8. у межах поліморфічного кластера віртуальний метод не повинен змінювати свій тип, кількість або типи параметрів. Якщо віртуальний метод не перевизначається у похідному класі, то діє віртуальний метод попереднього в ієрархії успадкування класу;
9. якщо на деякому рівні успадкування класів оголошується метод, ім’я якого збігається з попередньо оголошеним віртуальним методом, але метод має змінений список параметрів, то за наявності в оголошенні слова *virtual* утворюється новий поліморфічний кластер. Якщо слово *virtual* не використовується, то для цього класу віртуальний метод не перевизначається (діє визначення з попереднього класу). Новий поліморфічний кластер не утворюється, а попередньо визначений кластер не переривається. Якщо оголошення методу відрізняється тільки його типом порівяно з однойменним віртуальним методом, то виникає конфлікт і виводиться повідомлення про помилку;
10. у класі допускається перевантаження віртуальних методів. Кожний з таких методів започатковує власний поліморфічний кластер;
11. віртуальний метод може бути оголошений дружнім (*friend*) до іншого класу;
12. пізнє зв’язування реалізується в межах поліморфічного кластера за допомогою вказівників або посилань. Вказівник на клас – предок можна проініціалізувати адресою об’єкта – нащадка. Посилання на клас – предок можна проініціалізувати об’єктом класу – нащадка.

**Приклад поліморфізму віртуальних методів**

Нехай клас В успадковує клас А. У класі А визначений, а у класі В перевизначений віртуальний метод f ( ), який виводить повідомлення про належність до класу.

Якщо віртуальний метод викликається з об’єкта класу, то діє раннє зв’язування. Це означає, що тип об’єкта визначатиме клас, з якого буде викликаний віртуальний метод.

Якщо для виклику віртуального методу використовуються вказівник (або посилання), проініціалізований адресою (або іменем – для посилання) об’єкта похідного класу, то діє пізнє зв’язуваня. У результаті буде викликаний віртуальний метод з похідного класу. Наприклад:

#include <iostream>

using namespace std;

class A {

public:

virtual void f ( );

};

void A::f (void) { cout<< “A::f ( )”<<endl; }

class B : public A { // всі успадкування в поліморфічному кластері

// повинні мати режим *public*

public:

virtual void f ( );

};

void B::f (void) { cout<< “B::f ( )”<<endl; }

void main ( ) {

A a;

a.f ( ); // раннє зв’язування, викликається A::f ( )

B b;

b.f ( ); // раннє зв’язування, викликається B::f ( )

a = b;

a.f ( ); // раннє зв’язування, викликається A::f ( )

A \*p = &b;

p –>f ( ); // пізнє зв’язування, викликається B::f ( )

A &x = b;

x.f ( ); // пізнє зв’язування, викликається B::f ( )

}

У межах поліморфічного кластера віртуальні методи мають однакову адресу. Для перевірки цього рекомендується внести таке доповнення до попередньої програми:

void (A::\*pf1) (void) = &A::f;

void (B::\*pf2) (void) = &B::f;

printf (“%Fp %Fp\n”, pf1, pf2);

Слід зазначити, що поліморфізм стосується лише віртуальних методів класу. Застосування вказівника (або посилання) на базовий клас, проініціалізованого адресою (або іменем) об’єкта похідного класу, для виклику невіртуального методу похідного класу або для звернення до даних похідного класу призведе до помилок. Наприклад:

#include <iostream>

using namespace std;

class A {

public:

int x;

A (int x = 0) { this –> x = x; }

virtual void print ( ) { cout<<x<<endl; }

};

class B : public A {

public:

int y;

B (int x = 0, int y = 0) : A (x) { this –> y = y; }

virtual void print ( ) { cout<<x<< ‘ ’<<y<<endl; }

void output ( ) { cout<<x<< ‘ ’<<y<<endl; }

};

void main ( ) {

A \*p = new B (1, 2);

// виклик віртуального методу

p –>print ( ); // викликається віртуальний метод B::print ( )

// виклик невіртуального методу В

// p –> output ( ); // помилка: output не є членом класу А

//звернення до даних

cout<<p –>x<<endl;

// cout<<p –>y<<endl; // помилка: у не належить до класу А

delete p;

}

5.4. Індивідуальні завдання

1. Створити абстрактний клас ВАЛЮТА для роботи з грошовими сумами. Створити похідні класи, у яких визначити віртуальні методи переведення гривні в євро та долар.
2. Створити базовий клас – ПРАЦІВНИК і похідні класи – СЛУЖБОВЕЦЬ З ПОГОДИННОЮ ОПЛАТОЮ, СЛУЖБОВЕЦЬ З ОКЛАДОМ. Визначити віртуальну функцію нарахування зарплати.
3. Створити ієрархію класів ОСОБА і СТУДЕНТ. У класі ОСОБА визначити віртуальну операторну функцію для зміни віку особи. Перевизначити цю функцію у класі СТУДЕНТ для переходу на наступний курс. Використати операторну віртуальну функцію для ілюстрації дії механізму пізнього зв’язування.
4. Створити базовий клас – БАТЬКО, у якого є ім’я. Визначити віртуальну функцію виведення імені на екран. Створити похідний клас ДИТИНА, у якої є ім’я та успадковане поле по батькові.
5. Створити абстрактний БАЗОВИЙ клас з віртуальною функцією – площа. Створити похідні класи: ПРЯМОКУТНИК, КОЛО, ПРЯМОКУТНИЙ ТРИКУТНИК, ТРАПЕЦІЯ зі своїми функціями площі. Площа прямокутника: (a, b – сторони); площа круга: (r – радіус); площа прямокутного трикутника: (a, b – катети); площа трапеції: (a, b – основи трапеції, h – висота).
6. Створити абстрактний БАЗОВИЙ клас з віртуальною функцією – сума прогресії. Створити похідні класи: АРИФМЕТИЧНА ПРОГРЕСІЯ та ГЕОМЕТРИЧНА ПРОГРЕСІЯ. Кожний клас має два поля типу double. Перше – перший елемент прогресії, друге – постійна різниця для арифметичної і постійне відношення для геометричної прогресії. Визначити функцію обчислення суми, де параметром є кількість елементів прогресії. Арифметична прогресія: Сума арифметичної прогресії: . Геометрична прогресія: Сума геометричної прогресії:
7. Створити абстрактний клас – ССАВЦІ. Визначити похідні класи – ТВАРИНИ і ЛЮДИ. У тварин визначити похідні класи КОНЕЙ і КОРІВ. Визначити віртуальні функції опису людини, коня і корови.
8. Створити абстрактний БАЗОВИЙ клас з віртуальною функцією – об’єм. Створити похідні класи: ПАРАЛЕЛЕПІПЕД, ПІРАМІДА, КУЛЯ зі своїми функціями об’єму. Об’єм прямокутного паралелепіпеда: (a, b, c – ребра), правильної піраміди: (a, b – сторони прямокутної основи, h – висота), тетраедра: (a – довжина ребра), кулі: (r – радіус).
9. Створити абстрактний клас КАЛЬКУЛЯТОР з чистими віртуальними методами для виконання арифметичних операцій. Визначити похідні класи, у яких реалізувати віртуальні методи виконання арифметичних операцій над числами.
10. Створити абстрактний клас геометричне ТІЛО з віртуальними методами обчислення площі поверхні та об’єму. Створити похідні класи ПІРАМІДА та КУЛЯ зі своїми віртуальними методами обчислення площі поверхні та об’єму.
11. Побудувати ієрархію класів КОЛО, КРУГ та ЦИЛІНДР. Визначити віртуальні методи визначення довжини кола, площу круга, об’єм циліндра. Продемонструвати дію механізму пізнього зв’язування.
12. Створити абстрактний клас з віртуальними методами для відкривання файла, записування даних у файл та зчитування даних з файла. Визначити похідні класи, у яких реалізовано ці віртуальні методи для роботи з текстовими файлами. Створити та вивести на екран текстовий файл.
13. Створити абстрактний БАЗОВИЙ клас з віртуальною функцією – корені рівняння. Створити похідні класи: клас ЛІНІЙНИХ РІВНЯНЬ і клас КВАДРАТНИХ РІВНЯНЬ. Визначити функцію обчислення коренів рівнянь.
14. Створити абстрактний клас з віртуальним методом для обчислення характеристик одновимірного масиву цілих чисел. Визначити похідні класи, у яких реалізовано віртуальний метод для знаходження суми та добутку елементів масиву, найменший та найбільший елементи, кількість додатних, від’ємних та нульових елементів. Ввести масив з клавіатури та знайти його характеристики.
15. Створити абстрактний клас ТРИКУТНИК з віртуальними методами обчислення площі. Трикутник задається двома сторонами та кутом між ними. Визначити похідні класи ПРЯМОКУТНИЙ, РІВНОБЕДРЕНИЙ та РІВНОСТОРОННІЙ трикутник зі своїми віртуальними методами обчислення площі.

5.5. Контрольні запитання

1. Чи проявляється поліморфізм при перезавантаженні операторів і функцій?
2. Чи наслідується конструктор у похідному класі?
3. Для чого використовуються віртуальні функції?
4. Чи можуть функції похідного класу використовувати закриті члени базового класу?
5. Чи можливе повторне визначення віртуальних функцій у похідному класі?
6. Що таке повністю віртуальна функція?
7. Які властивості мають дружні функції та дружні класи?
8. Що таке поліморфізм в ООП?
9. Чи може виклик однієї і тієї ж функції з однаковими параметрами при використанні принципів ООП дати різні результати?
10. В якому випадку множинного наслідування базовий клас та його функції описуються віртуальними?
11. Поясніть коментарі в наведеному прикладі програми.
12. Поясніть суть правила домінування імен при множинному наслідуванні.