**Лабораторна робота №15. Розробка програм із застосуванням абстрактних класів**

**Мета:** Засвоєння навичок розробки програм з використанням віртуальних функцій та поліморфізму

**Порядок виконання роботи**

1. Ознайомитися з теоретичними основами розробки віртуальних функцій та застосуванням поліморфізму.
2. Створити абстрактний базовий клас Shape (фігура) та похідні від нього Circle (коло) з полем радіус, Triangle (трикутник) з полями 3 сторони, Rectangle (прямокутник) з полями ширина та висота. Похідні класи мають метод draw. Створити екземпляри об‘єктів, розмістивши їх у поліморфному контейнері. Викликати для всіх об‘єктів метод draw. Оформити однією програмою, до якої включити опис класу.
3. Продовжити працювати над програмою лабораторної роботи №11, закінчити попередні завдання з урахуванням наданих зауважень (для тих, хто не здав). Результати надсилати разом зі сформованими файлами.
4. В першому рядку програми та заголовкового файлу повинні бути записаними в коментарі номер групи та прізвище, а також номер ЛР (через кому до попередньої).
5. Результати надсилати на електронну адресу викладача

[**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді заголовкового файлу та cpp-файлу з іменем у форматі

Для заголовкового файлу

**<Прізвище англійською>.h**

Наприклад, Ivanov.h

Для cpp-файлу:

**<Номер групи> <Номер лабораторної><Прізвище англійською>.cpp**

Наприклад, 31-01Ivanov.cpp.

Тему в заголовку листа записати

**ООП<Номер групи>-ЛР <Номер лабораторної>-<Прізвищеанглійською>**

**Строк відсилки ЛР ІПЗ-31 07.11.2022**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, тему в заголовку листа записати

**ООП<Номер групи>-Запитання-<Прізвище англійською>**.

**Контрольні запитання**

1. Назвіть основні парадигми OOП.
2. Що таке інкапсуляція
3. За допомогою чого реалізується поліморфізм?
4. Чи можна покажчику на базовий клас присвоїти значення адреси об'єкта похідного класу?
5. Що необхідно для того, щоб реалізувати поліморфізм в програмі?
6. Що таке абстрактний клас?
7. Чим абстрактний клас відрізняється від інтерфейсу?
8. Що таке інтерфейс?
9. Дайте визначення абстрактного класу.
10. Чи можна створювати об’єкти абстрактного класу?
11. Виявіть відмінності між віртуальними функціями і чистими віртуальними функціями.
12. Чим абстрактний клас відрізняється від конкретного?
13. Які члени можуть бути абстрактними?
14. Чи вірно, що всі віртуальні функції в абстрактному базовому класі повинні бути оголошені як чисті віртуальні функції?
15. Чи можна створювати функцію, параметром якої є покажчик на абстрактний клас?
16. Як розміщуються об’єкти у пам’яті? Чи має кожен об’єкт свої дані? Чи має кожен об’єкт свої методи?
17. Що означають статичні дані класу? Що означають статичні методи класу?
18. Що означають вбудовані методи класу? Коли вони застосовуються? Чи є вони обов’язковими?
19. Охарактеризуйте статичне і динамічне зв’язування.
20. Що таке віртуальні функції та віртуальні деструктори?
21. Який механізм дозволяє створювати ієрархії класів?
22. Які базові класи називаються віртуальними?
23. Що таке поліморфічний кластер?
24. Які класи називаються абстрактними?
25. Що таке таблиця віртуальних функцій?
26. Коли застосовується віртуальний деструктор
27. Опишіть оператор dynamic\_cast.
28. Опишіть три види приведення за допомогою оператора dynamic\_cast в ієрархії класів.
29. Опишіть оператор typeid.

**Теоретичні відомості.**

*Суто віртуальна функція –* це віртуальна функція, яку оголошено в базовому класі, але вона не має у ньому ніякого визначення. 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* (*перелік\_параметрів*) = 0;

У цьому записі під елементом *тип* маємо на увазі тип значення, що повертається функцією, а елемент *ім'я\_функції –* використовуване у програмі її ім'я. Позначення = 0 є ознакою того, що функція тут оголошується як *суто віртуальна*. Наприклад, в наступній версії визначення базового класу figure функція Show()представлена як суто віртуальна:

**class** figure {

**double** x, y;

**public:**

**void** Set(**double** \_x, **double** \_y=0) { x = \_x; y = \_y; }

**virtual void** Show() = 0; // Суто віртуальна функція

};

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. Наприклад, спробуйте скомпілювати цю модифіковану версію програми обчислення площ геометричних фігур, у якій з класу circle видалено визначення функції Show().

**Приклад. Демонстрація не коректної програми, яка у класі circle немає перевизначення функції Show()**

**#include** <**vcl**>

**#include** <**conio**>

**#include** <**iostream**> // Для потокового введення-виведення

**using namespace std**; // Використання стандартного простору імен

**class** figure {

**protected**:

**double** x, y;

**public**:

**void** Set(**double** \_x, **double** \_y) { x = \_x; y = \_y; }

**virtual void** Show() = 0; // Суто вiртуальна функцiя

};

**class** triangle : **public** figure {

**public**:

**void** Show()

{

**cout** << Cyr("Трикутник з висотою ") << x; **cout** << Cyr(" i основою ") << y;

**cout** << Cyr(" має площу ") << x \* 0.5 \* y ; **cout** << Cyr(" кв. од.") << **endl**;

}

};

**class** rectangle : **public** figure {

**public**:

**void** Show()

{

**cout** << Cyr("Прямокутник розмiрами ") << x << " x " << y;

**cout** << Cyr(" має площу ") << x \* y ; **cout** << Cyr(" кв. од.") << **endl**;

}

};

**class** circle : **public** figure {

// Вiдсутнiсть визначення функцiї Show()

// Викличе повiдомлення про помилку.

};

**int main**()

{

figure \*p; // Створення покажчика на об'єкт базового типу

triangle ObjT; // Створення об'єкта похідного типу

rectangle ObjR; // Створення об'єкта похідного типу

circle ObjC; // Помилка: створення цього об'єкта є неможливим!

p = &ObjT; // Присвоєння покажчику адреси об'єкта похідного класу

p->Set(10.3, 5.5);

p->Show();

p = &ObjR; // Присвоєння покажчику адреси об'єкта похідного класу

p->Set(10.3, 5.5);

p->Show();

**getch**(); **return** 0;

}

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 похідні класи. Причина того, що абстрактний клас не можна використовувати для побудови об'єктів, полягає, безумовно, у тому, що його одна або декілька функцій не мають визначення. Але навіть якщо базовий клас є абстрактним, то його все одно можна використовувати для оголошення покажчиків і посилань, які необхідні для підтримки динамічного поліморфізму.