1. **Що таке віртуальні (virtual) елементи класу? Які елементи класу можуть бути віртуальними? Як віртуальні елементи пов'язані з інкапсуляцією, успадкуванням, поліморфізмом? (Навести приклади з власного коду)**

Віртуальні (virtual) елементи класу, а саме методи — це особливий тип методів, при виклику яких виконується «найдочірніший», який існує між батьківським і дочірніми класами. Саме за допомогою віртуальних методів досягається динамічний поліморфізм. Дочірній метод викликається тоді, коли збігається сигнатура (ім’я, типи параметрів) і тип повернення методів у дочірньому та батьківському класах. Такі методи називаються перевизначеннями (або «перевизначеними методами»).

Віртуальні методи застосовуються для досягнення трьох основних концепцій ООП: інкапсуляції, успадкування та поліморфізму.

* Віртуальні методи дозволяють приховувати реалізацію функцій у батьківському класі від користувача. (інкапсуляція)
* Віртуальні методи дають можливість у дочірньому класі перевизначити віртуальні методи батьківського класу. Це дозволяє створювати спеціалізовану логіку для конкретного похідного класу. (успадкування)
* Віртуальні методи грають ключову роль у поліморфізмі. Об'єкти батьківського класу можуть бути використані для роботи з об'єктами дочірніх класів завдяки тому, як вже писалося вище, що виконується реалізація саме дочірнього класу. Це дозволяє динамічно визначати поведінку об'єктів на етапі виконання програми.

На жаль, у моєму проєкті не було застосовано віртуальних функцій через відсутність дочірніх класів, а перша лабораторна була написана була написана на Python, де концепція віртуальних таких функцій відсутня. Тому як приклад я спробую показати як могла виглядати реалізація коду з першої лабораторної на C++ з використанням віртуальних методів:

struct Coords

{

int x1 = 0;

int y1 = 0;

int x2 = 0;

int y2 = 0;

}

class Sprite

{

protected:

Coords coordinates {0, 0, 0, 0};

Sprite(Coords value) : value(coordinates);

…

public:

virtual const Coords coords()

{

return coordinates;

}

}

class PlatformSprite : public Sprite

{

protected:

…

Coords coordinates {x, y, x + width, y + height};

public:

PlatformSprite(Coords coordinates) : Sprite(coordinates);

}

class MovingPlatformSprite : public PlatformSprite

{

public:

MovingPlatformSprite(Coords coordinates) : PlatformSprite(coordinates);

virtual const Coords coords()

{

something doing with coordinates

…

return coordinates;

}

}

int main()

{

MovingPlatformSprite movingplatform({1, 1, 1, 1});

std::cout << platform.coords().y2 << ‘\n’;

std::cout << movingplatform.coords().y2 << ‘\n’;

}

У висновку для platform виведеться y + height, а для movingplatform якесь значення після обробки у функції cords. Тобто компілятор бачить, що метод cords є віртуальним і обирає реалізацію найбільш дочірнього класу.

Джерела:

<https://acode.com.ua/urok-171-virtualni-funktsiyi-i-polimorfizm/>

<https://github.com/Ke11nyk/Lab1/blob/master/main.py>

P. S. Можливі певні неточності у коді через те, що він переписувався на ходу.

1. **Як перевіряти правильність даних, які вводить користувач? (Навести приклади з власного коду)**

Перевірка є важливою складовою для забезпечення безпеки та коректності програм. Існує кілька загальних підходів, які можна використовувати для перевірки введених даних:

* *Використання умовних операторів*
* *Використання функції getline для рядкових вводів*
* *Обробка винятків (Exceptions)*
* *Використання вбудованих функцій мови програмування:*

Багато мов програмування мають вбудовані функції для перевірки типів даних, довжини рядків, числових діапазонів та інші. Наприклад, у Python можна використовувати isinstance(), len(), int(), float(), тощо.

Зважаючи на те, що мій проєкт оформлений у формі гри, явного на прикладу на це питання теж немає. У рамках мого проєкту можна було б реалізувати можливість користувача вводити нікнейм та перевіряти його на коректність по таким параметрам як:

* словник символів
* довжина
* будь-які додаткові параметри

Після того як би користувач ввів свій нікнейм, певна функція б зчитувала рядок і за допомогою, наприклад, умовних операторів перевіряла б коректність такого вводу, у висновку підтверджуючи коректність вводу, або ж виводячи повідомлення про те, що ввід некоректний.