----------------------------

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ

за курсом «Об’єктно-орієнтоване програмування»

студента групи ПЗ-22-3

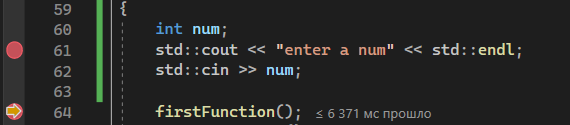
Казанікова Іллі Івановича

кафедра математичного забезпечення ЕОМ, ДНУ

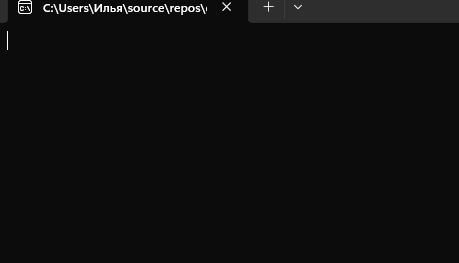
2023/2024 навч.р.

1.**Продемонструвати можливості налагодження, а саме: запуск програми у покроковому режимі, різновиди точок зупинки, умовні точки зупинки, перегляд значень змінних**

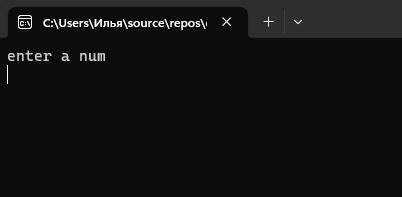
**точки зупинки:**

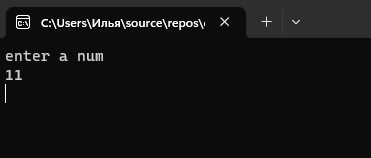


виглядає ось так:



після продовження:





**Відстежування значення зміни:**

зміна ініціалізована:



значення зміни заповнено:

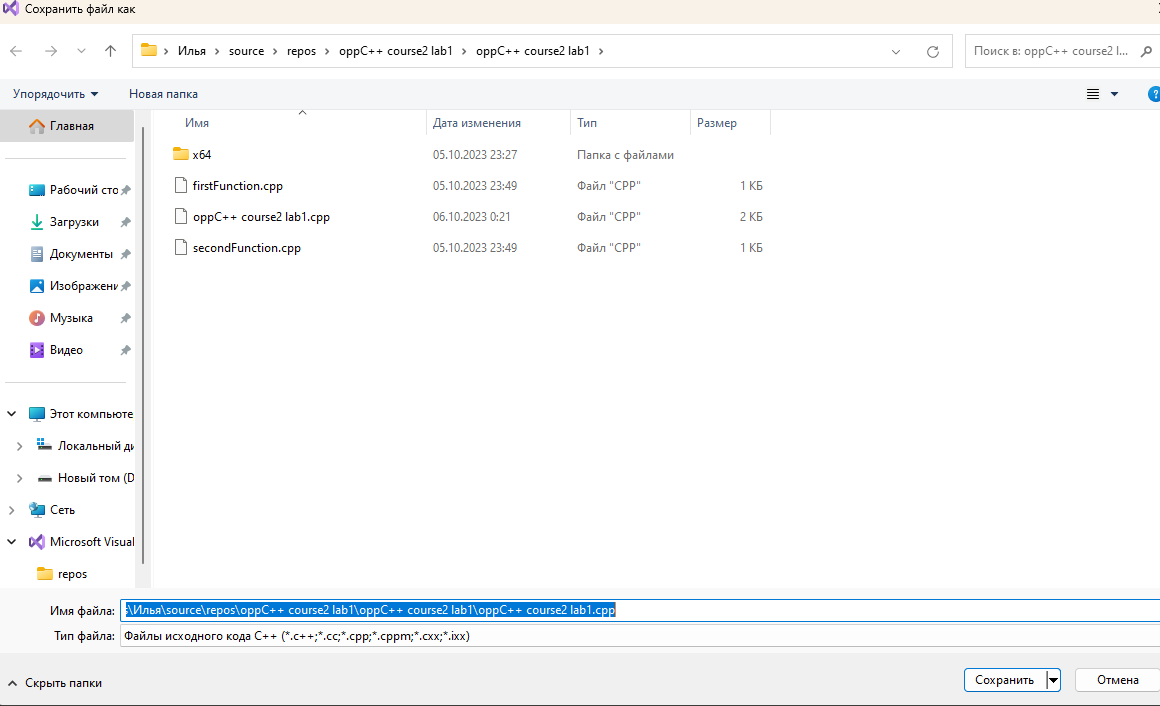


2. **Настроювання інтегрованого середовища. Робота з проєктами.**

**Збереження проєкту у бажану папку. Перенос проєкту з комп’ютера на комп’ютер. Зміна параметрів середовища (кольори, клавіатурні комбінації, тощо). Зміна параметрів проєкту (ключі компілятора, параметри командного рядка, тощо).**

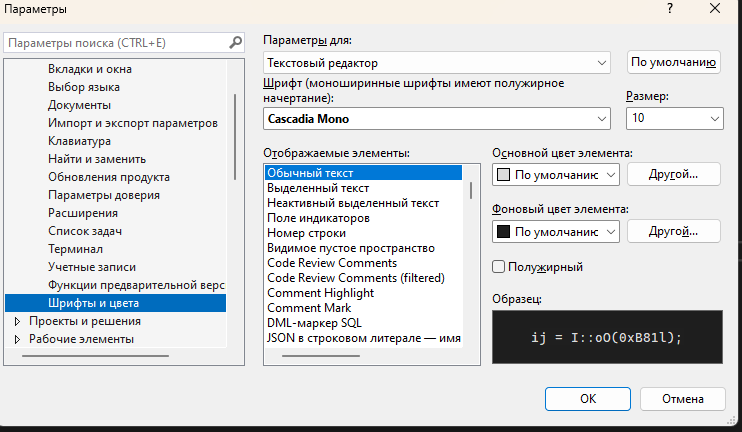
**Щоб зберегти проєкт у бажану папку треба**: Файл - зберегти як - обрати папку





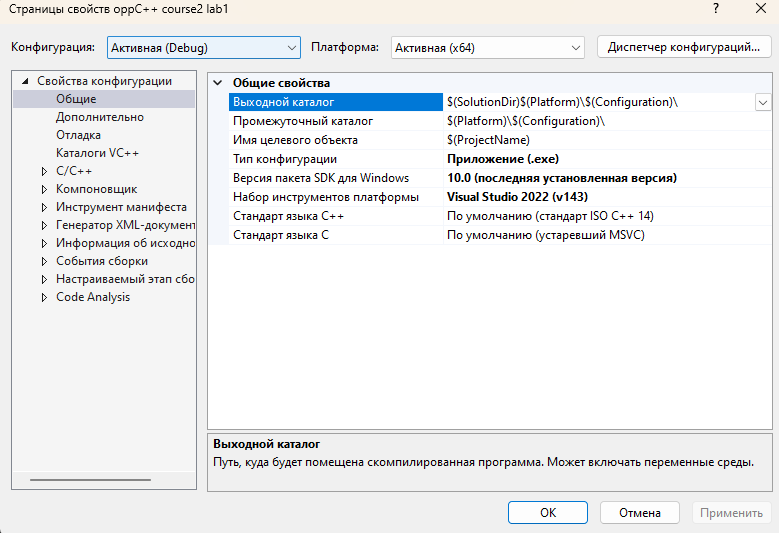
**Перенос проекту з комп'ютера на комп'ютер**: Спочатку потрібно скопіювати всю папку проекту на інший комп'ютер (або використати засоби обміну файлами, такі як USB-накопичувачі, мережу або хмарні сервіси). Відкрити Visual Studio на новому комп'ютері. Виберіть “Файл” > “Открыть” > “Проект/решение” і вибрати файл проекту (.sln), який ми скопіювали.

**Зміна параметрів середовища**: змінити параметри середовища у Visual Studio, такі як кольори, клавіатурні комбінації та інші налаштування, можна таким чином: зайти в “Средства” > “параметры”. Вибрати "Окружение" або "Шрифты и цвета", і налаштувати параметри



**Зміна параметрів проекту:**

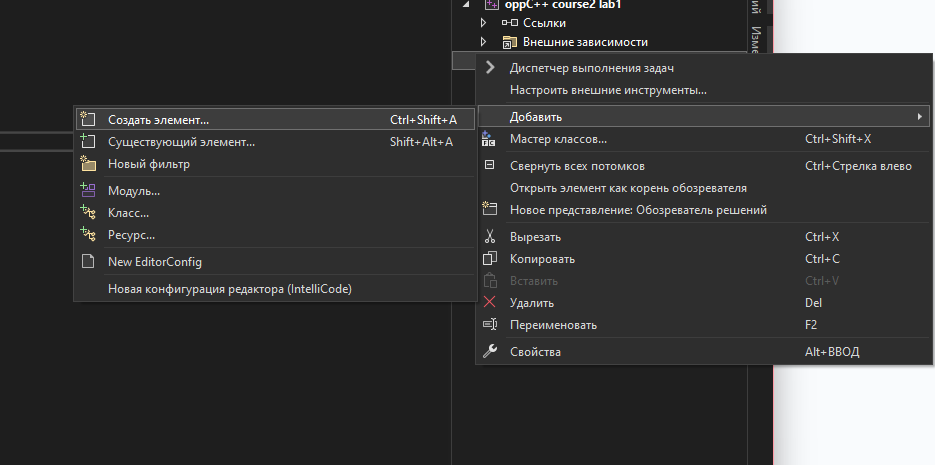
Треба обрати "Проект" в меню і вибрати "Свойства" для вибраного проєкту.

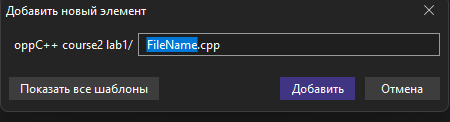


3. **Багатомодульні програми.**

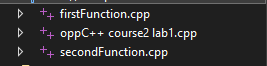
**Скласти програму, яка викликає функції, визначені в інших (які окремо транслюються) файлах. Вміти відкривати/закривати проєкт, вміти додавати новий файл до існуючого проєкту.**

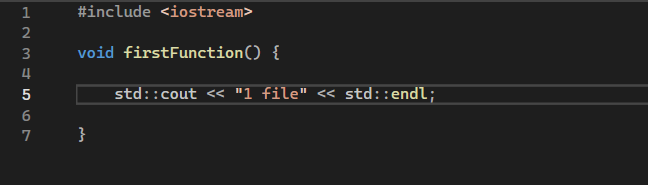
**Щоб створити новий файл треба: ПКМ на “Исходные файлы” - “Добавить” - “Создать элемент”**

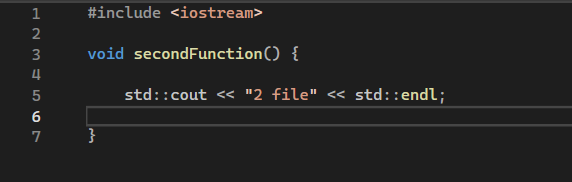
****

****

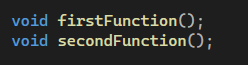
В нас є 3 файли:

****

firstFunction.cpp: ****

secondFunction.cpp: 

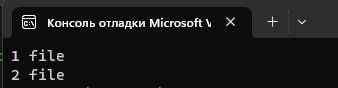
Щоб викликти функції, які визначені в інших файлах, треба в main файлі прописати зовнішні декларації функцій з інших файлів:



та потім викликати їх у main():



результат:



**4.Проєктування ієрархії та структури класів.**

**Спираючись на будь-яку предметну галузь за власним вибором, створити ієрархію класів, що відображатиме сутності, представлені в цій галузі. Описати**

**спроєктовані класи словесно (згадуючи при цьому всі методи та поля спроєктованих класів, їхню область видимості та призначення) або за допомогою діаграми класів.**

**Продемонструвати 3 складові ООП: інкапсуляція, наслідування та поліморфізм (перевантаження та перевизначення).**

**Всі пункти завдання можна оформлювати окремими проєктами або спільним за власним бажанням.**

Клас Character:

Поля:

name (строка) - ім'я персонажа.

level (ціле число) - рівень персонажа.

Конструктор:

Конструктор з параметрами, який приймає ім'я та рівень персонажа.

Методи:

attack() - віртуальний метод, що виводить повідомлення про атаку персонажа.

useAbility() - віртуальний метод, що виводить повідомлення про використання спеціальної здібності персонажа.

Клас Hero (похідний від Character):

Додаткове поле:

specialAbility (строка) - опис спеціальної здібності героя.

Конструктор:

Конструктор з параметрами, який спеціалізується на створенні героя та приймає ім'я, рівень і опис спеціальної здібності героя.

Методи:

attack() - перевизначений метод, що виводить повідомлення про атаку героя.

useAbility() - перевизначений метод, що виводить повідомлення про використання спеціальної здібності героя.

levelUp() - метод, який підвищує рівень героя та виводить повідомлення про це.

Клас Monster (похідний від Character):

Конструктор:

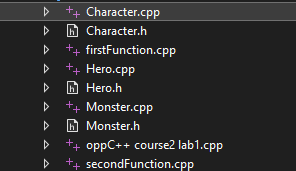
Конструктор з параметрами, який спеціалізується на створенні монстра та приймає ім'я та рівень монстра.

Методи:

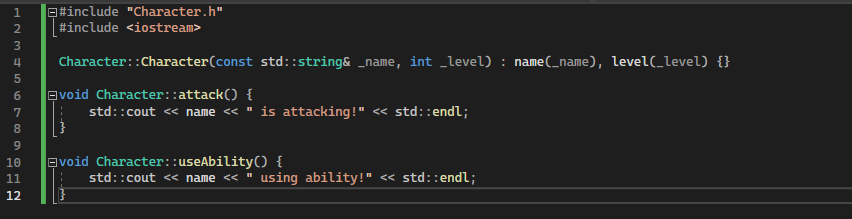
attack() - перевизначений метод, що виводить повідомлення про атаку монстра.

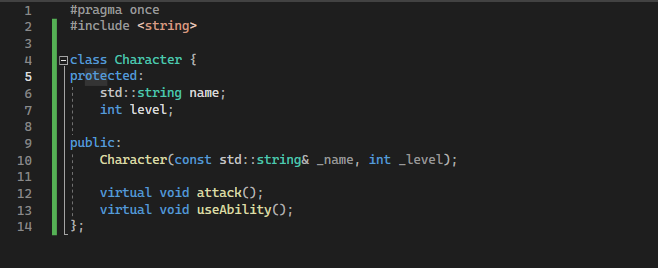
Hero та Monster є похідними від базового класу Character, використовуючи наслідування. Поліморфізм реалізований через віртуальні методи attack() та useAbility(), які перевизначаються в похідних класах Hero та Monster. Також в класі Hero додано додатковий метод levelUp(), який розширює функціональність базового класу.

класи реалізовані за допомогою хедерів та cpp файлів:

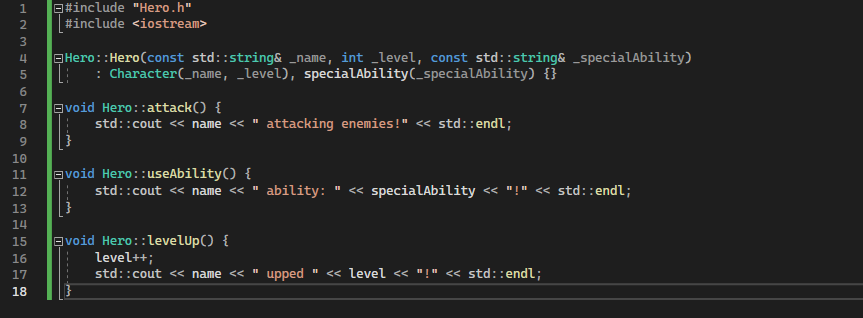


Character.cpp:

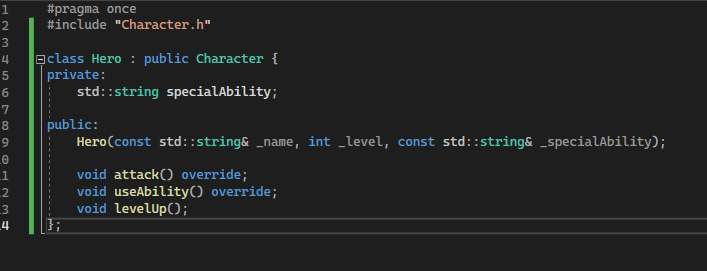


Character.h: 

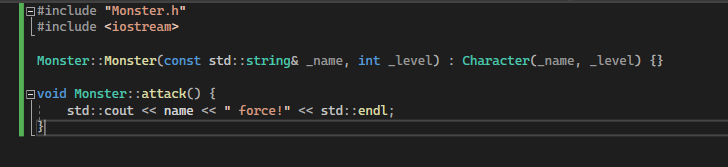
Hero.cpp:



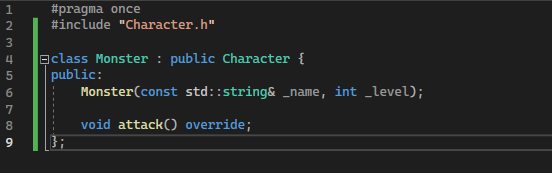
Hero.h:



Monstr.cpp:



Monstr.h:



хедери підключені таким чином в main.cpp:

