Лабораторна робота №9

Конструювання класів з використанням дружніх функцій

Мета: Створити, відлагодити та протестувати програму, у якій створити базовий та похідні класи для обробки даних.

Вказівки до роботи

Відповідно до цієї інструкції вам потрібно у середовищі Visual Studio Code з використанням набору компіляторів GCC створити програму мовою C++ з базовим (батьківським), похідними (дочірніми) та дружнім класом для обробки даних, також реалізувати відповідні конструктори, деструктори, методи. Використайте динамічний масив (вектор) структур для зберігання введених даних. Потрібно реалізувати багатофайловий проєкт у середовищі Visual Studio Code з описом кожного класу у відповідних парах заголовного та виконуваного файлів.

1. Ознайомитися із властивостями базового (батьківського) та похідного (дочірнього) класів. Обдумати способи їх використання для створення програм.
2. Вибрати завдання згідно свого варіанту у **ДОДАТКУ**.
3. Розробити блок-схему алгоритму програми.
4. Реалізувати програму для вводу і виводу даних полів (членів) структур вкладених у клас.

Короткі відомості

Ви вже знаєте про те, що дані вашого класу повинні бути private. Однак може виникнути ситуація, коли у вас є клас і функція, яка працює з цим класом, але не знаходиться у його тілі. Наприклад, є клас, в якому зберігаються дані, і функція (або інший клас), яка виводить ці дані на екран. Хоча код класу і код функції виводу розділені (для спрощення підтримки коду), код функції виводу тісно пов’язаний з даними класу. Отже, зробивши члени класу private, ми бажаного ефекту не отримаємо. В таких ситуаціях є два варіанти:

• Зробити відкритими методи класу і через них функція взаємодіятиме з класом. Однак тут є кілька нюансів. По-перше, ці відкриті методи необхідно буде

визначити, на що знадобиться виділити час, і вони будуть захаращувати інтерфейс класу. По-друге, в класі потрібно буде відкрити методи, які не завжди повинні бути відкритими і надавати доступ ззовні.

• Використовувати дружні класи і дружні функції, за допомогою яких можна буде надати функції виводу доступ до закритих даних класу. Це дозволить функції виводу безпосередньо звертатися до всіх закритих змінних-членів і методів класу, зберігаючи при цьому закритий доступ до даних класу для всіх інших функцій поза тілом класу.

Дружня функція — це функція, яка має доступ до закритих членів класу, наче вона сама є членом цього класу. У всіх інших аспектах дружня функція є звичайною функцією. Нею може бути, як звичайна функція, так і метод іншого класу. Для оголошення дружньої функції використовується ключове слово friend перед прототипом функції, яку ви хочете зробити дружньою класу. Неважливо, оголошуєте ви її в public- чи в private-зоні класу. Наприклад:

```
class Anything
{
private:
    int m_value;
public:
    Anything() { m_value = 0; }
    void add(int value) { m_value += value; }

    // Робимо функцію reset() дружньою до класу Anything
    friend void reset(Anything &anything);
};

// Функція reset() тепер є другом класу Anything
void reset(Anything &anything)
{
// І ми маємо доступ до закритих членів об'єктів класу Anything
    anything.m_value = 0;
}

int main()
{
    Anything one;
    one.add(4); // додаємо 4 до m_value
    reset(one); // скидаємо значення m_value в 0
```

```
    return 0;
}
```

Тут ми оголосили функцію reset(), яка приймає об’єкт класу Anything і встановлює m_value значення 0. Оскільки reset() не є членом класу Anything, то в звичайній ситуації функція reset() не мала б доступу до закритих членів Anything. Однак, оскільки ця функція є дружньою класу Anything, вона має доступ до закритих членів Anything.

Зверніть увагу, ми повинні передавати об’єкт Anything в функцію reset() в якості параметра. Це пов’язано з тим, що функція reset() не є методом класу. Вона не має прихованого константного вказівника *this, що містить адресу об’єкта, який викликає метод класу.

Дружні класи - один клас може бути дружнім іншому класу. Це відкриє всім членам першого класу доступ до закритих членів другого класу. Оскільки клас Display є другом класу Values, то будь-який з членів Display має доступ до private-членів Values.

```
class Values
{
private:
    int m_intValue;
    double m_dValue;
public:
    Values(int intValue, double dValue)
    {
        m_intValue = intValue;
        m_dValue = dValue;
    }

    // Робимо клас Display другом класу Values
    friend class Display;
};

class Display
{
private:
    bool m_displayIntFirst;

public:
```

```
        Display(bool displayIntFirst) { m_displayIntFirst =
displayIntFirst; }

        void displayItem(Values &value)
        {
            if (m_displayIntFirst)
                std::cout << value.m_intValue << " " <<
value.m_dValue << '\n';
            else // або спочатку виводимо double
                std::cout << value.m_dValue << " " <<
value.m_intValue << '\n';
        }
    };

    int main()
    {
        Values value(7, 8.4);
        Display display(false);

        display.displayItem(value);

        return 0;
    }
```

Замість того, щоб робити дружнім цілий клас, ми можемо зробити дружніми тільки певні методи класу. Їх оголошення аналогічні оголошенням звичайних дружніх функцій, за винятком імені методу з префіксом `ім'яКласу::` на початку (наприклад, `Display::displayItem()`).

Дружня функція/клас — це функція/клас, яка має доступ до закритих членів іншого класу, наче вона сама є членом цього класу. Це дозволяє функції/класу працювати в тісному контакті з іншим класом, не змушуючи інший клас робити відкритими свої закриті члени.

Щоб об'єднати водній програмі батьківський клас, дочірній клас та дружній клас, ви можете використовувати успадкування разом з ключовим словом friend. Ось приклад, який демонструє, як це зробити:

```
    #include <iostream>

    class Parent {
    protected:
    int id;
```

```
    public:
    Parent() : id(0) {}
    };

    class Child : public Parent {
    protected:
    int value;

    public:
    Child() {
    id = 1; // Доступ до захищеного члена базового класу
    value = 12;
    }

    // Оголошення дружного класу FriendClass
    friend class FriendClass;
    };

    class FriendClass {
    public:
    void showId(Child& c) {
    // Оскільки клас FriendClass is a дружнім до класу Child, він
може отримати доступ до захищених членів Parent через Child
    std::cout << "id from Parent through Child: " << c.id <<
std::endl;
    }
    void showValue(Child& c) {
    // Оскільки клас FriendClass is a дружнім до класу Child, він
може отримати доступ до захищених членів Child
    std::cout << "Value from Child: " << c.value << std::endl;
    }
    };

    int main() {
    Child childObj;
    FriendClass friendObj;

    friendObj.showId(childObj);
    friendObj.showValue(childObj);

    return 0;
    }
```

Хід роботи:

Рекомендації до роботи:

1. продумайте, які типи змінних вам потрібно використати для виконання завдання.
2. створіть проєкт із кількома файлами, тобто крім основного файлу, наприклад main.cpp, додайте для кожного класу заголовний файл <class>.h та виконуваний файл <class>.cpp. Також модифікуйте ***..\.vscode\\tasks.json***, щоб вказати йому шлях до всіх файлів проекту.
3. оголосіть базовий (батьківський) та похідний (дочірній) класи, а також відповідні структури для збереження і обробки даних, конструктори та деструктори.
4. у заголовному файлі дружнього класу оголосіть конструктори та деструктори, методи для введення, виведення та обробки даних із батьківського та дочірнього класів.
5. у виконувану файлі дружнього класу реалізуйте для введення, виведення та обробки даних із батьківського та дочірнього класі.
6. можете збільшити кількість членів класу додатковими полями чи методами.
7. обчислити результат відповідно заданого варіанту, наприклад загальну вартість, тривалість, кількість, максимальне чи мінімальне значення тощо.
8. програма має вивести на консоль (термінал) вхідні дані та результат виконання.

Оформлення звіту

У звіті студент має вказати свій варіант індивідуального завдання та представити:

1. Блок-схему алгоритму програми.
2. Код (лістинг) програми.
3. Результати тестування для різних вхідних умов.
4. Конструктивний висновок — що ви дізналися нового, чого навчилися, якими способами та інструментами досягнули мети.

ДОДАТОК

Варіанти завдань

№	Батьківський клас	Дочірні класи
1	Працівник	Касир, вантажник
2	Перевезення	Вантажні, пасажирські
3	Місто	Промисловий центр, культурний центр
4	Літак	Пасажирський, транспортний
5	Корабель	Авіаносець, крейсер
6	Пасажири	Поїзд, автобус
7	Транспортний засіб	Вантажівка, автобус
8	Команда	Футбол, баскетбол
9	Компанія	Будівництво, логістика
10	Морські тварини	Риби, ссавці
11	Сухопутні тварини	Хижаки, травоїдні
12	Реклама	Телебачення, вулична
13	Депозит	Прості відсотки, Складні відсотки
14	Митець	Художник, письменник
15	Друковане видання	Книга, журнал
16	Екскурсія	Музей, театр
17	Подорож	Пішохідна, автобусна
18	Геометрична фігура	Коло, квадрат
19	Упакування	Фасування у коробках, рідини у бочках
20	Змагання	Теніс, шахи
21	Тариф	З абонплатою, з післяоплатою
22	Квиток	Поїзд, літак
23	Автомобіль	Легковик, вантажівка
24	Перегони	Формула 1, Мото Гран-Прі
25	Судно	Контейнеровоз, танкер
26	Комп’ютер	Ноутбук, стаціонарний
27	Будівельна техніка	Бульдозер, екскаватор
28	Летючі тварини	Птахи, кажани
29	Сільськогосподарська техніка	Трактор, комбайн
30	Водний спорт	Плавання, стрибки з трампліна