Лабораторна робота №5

Розробка багатовіконного інтерфейсу користувача для графічного редактора об'єктів

Мета: Мета роботи – отримати вміння та навички програмувати багатовіконний інтерфейс програми на С++ в об'єктно-орієнтованому стилі.

Завдання:

1. Створити у середовищі MS Visual Studio C++ проект Win32 з ім'ям **Lab5**.
2. Написати вихідний текст програми згідно варіанту завдання.
3. Скомпілювати вихідний текст і отримати виконуваний файл програми.
4. Перевірити роботу програми. Налагодити програму.
5. Проаналізувати та прокоментувати результати та вихідний текст програми.
6. Оформити звіт.

Методичні рекомендації

Використати матеріал лекції "Особливості програмування для Windows" – відомості по немодальним діалоговим вікнам, а також результати виконання попередньої лабораторної роботи №4

Загальна структура програми

Програма складається, як мінімум, з двох частин:

- частини підтримки головного вікна, зокрема, для вводу та відображення об'єктів геометричних форм;
- частина підтримки окремого **немодального діалогового вікна**, у якому відображаються табличний опис множини об'єктів. Таблиця має бути описана окремим класом **MyTable.** Ця частина повинна бути запрограмована у окремому модулі (файли *.cpp, *.h, *.rc)

Програма повинна бути модульною. Вихідний текст складається з головного файлу **Lab5.cpp** та інших модулів і файлів проекту Win32. Усі модулі повинні роздільно компілюватися у проекті.

Вікно таблиці

Множину об'єктів, які вводяться та відображаються у головному вікні, утворюючи складний малюнок з багатьох форм, потрібно відображати у таблиці, яка представляється у окремому немодальному вікні діалогу.
Як запрограмувати немодальне діалогове вікно? Це було розглянуто на лекціях. Використайте матеріал лекції по немодальним діалоговим вікнам.
Кожний рядок таблиці – це опис відповідного геометричного об'єкту, який уведено у масив форм та відображається у головному вікні. Рядки таблиці заповнюються у порядку відповідно вводу об'єктів. У таблиці повинен бути скролінг по вертикалі, щоб мати можливість переглядати список з багатьох об'єктів – геометричних форм.
Стовпчиків таблиці п'ять – назва об'єкту та координати розташування.
Наприклад:

Назва	x1	y1	x2	y2
Точка	10	130	10	130
Лінія	45	23	400	327
Еліпс	123	304	257	412

Впродовж роботи редактора при кожному додаванні нового об'єкта у головному вікні, інформація про цей об'єкт повинна автоматично з'являтися у вікні таблиці у новому нижньому рядку
Вікно таблиці можна запрограмувати на основі стандартного вікна списка. Інший варіант – щоб програма сама малювала на поверхні діалогового вікна лінії вкликом функцій Windows API: кожна лінія малюється MoveToEx( ), LineTo( ), а текст – TextOut( ).

Вимоги щодо класу таблиці

Вікно таблиці програмується у незалежному модулі **my_table**, який складається з файлів **my_table.cpp, h, rc**. У файлі my_table.h записується оголошення класу **MyTable**

```
class MyTable
{
   . . .          //якісь члени класу
public:
   void Add(..);  //функція додавання у таблицю нового рядка з описом об'єкту
   . . .          //інші функції

};
```

Після уводу у головному вікні нового об'єкту обробник повідомлення WM_LBUTTONUP викликає фанкцію MyTable::Add( ) щоб занести у список-таблицю назву та координати нового об'єкту. Після цього у вікні таблиці з'являється новий рядок.

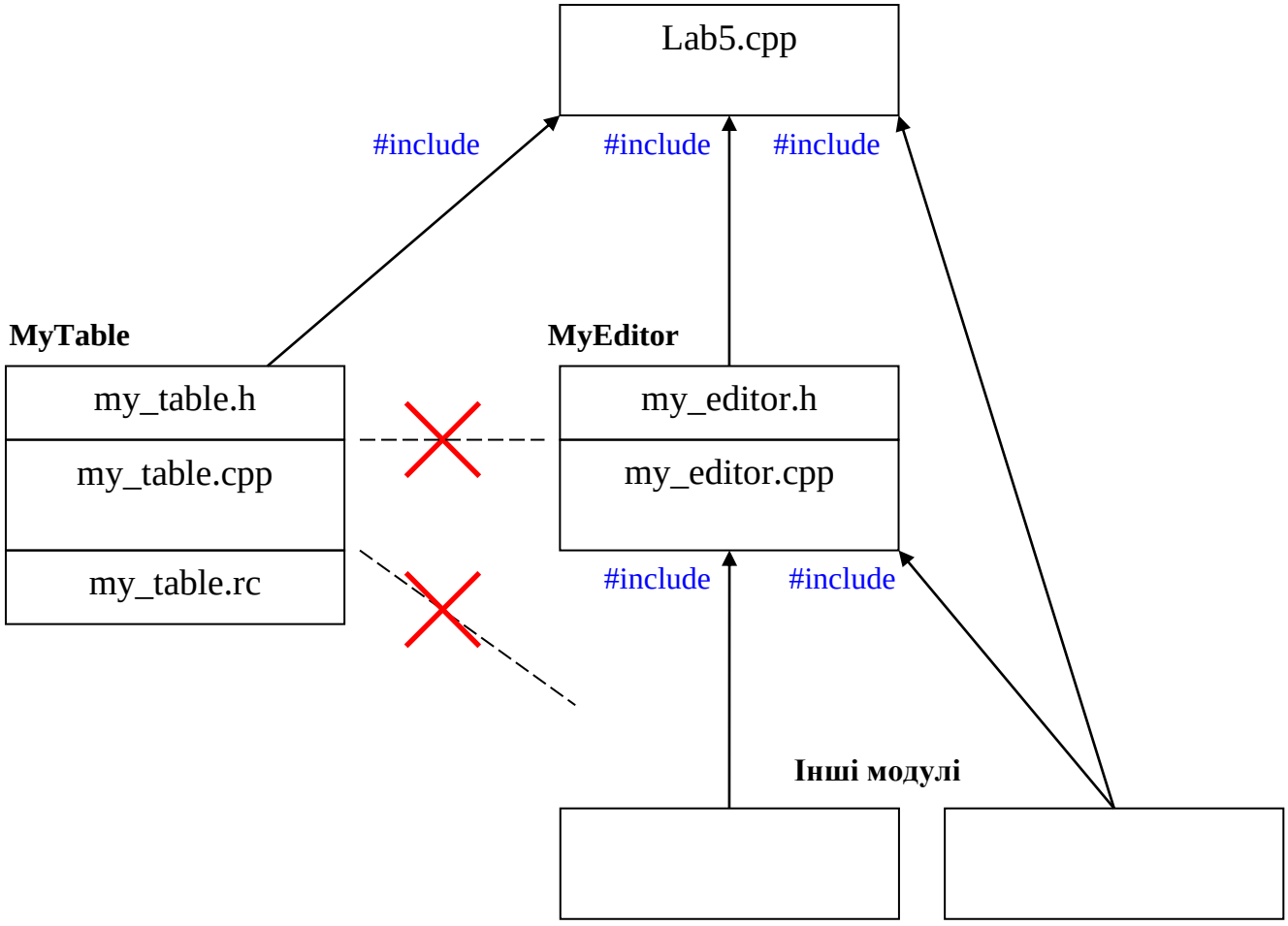
Вимоги щодо головного класу графічного редактора

Назвемо цей клас так само, як у попередньої лаб. №4 – **MyEditor**. Його оголошення записується у файлі-заголовку **my_editor.h**. Цей клас буде інкапсулювати масив об'єктів типу Shape, та містити інтерфейсні public функції-члени, які повинні реалізовувати основні функції графічного редактора

Головний клас повинен бути **Singleton** відповідно варіанту завдання

Залежності модулів

Клієнт – головний файл (Lab6.cpp) використовує інтерфейсні функції модулів **my_editor, my_table** та інших модулів. Ієрархія залежностей модулів відображена нижче



Вимога: модуль **my_table** повинен бути самодостатнім та **незалежним від будь-якого з модулів програми**. Він не може використовувати по #include будь-який файл проекту (окрім хіба що файлу **framework.h**).
Можна сказати, що вікно таблиці є самодостатнім компонентом, який можливо використовувати у інших проектах, не тягнучи за собою залежності від модулів проекту Lab5.

Клас Singleton

Цей клас забезпечує можливість створення тільки одного екземпляру об’єкта

Класична реалізація Singleton

Для блокування створення інших об’єктів контролюється значення статичного члену – вказівника **p_instance.** Об’єкт створюється шляхом **new**.

Оголошення класу – файл singleton.h

```
class Singleton
{
private:
   static Singleton * p_instance;
   Singleton() {}
   Singleton( const Singleton& );
   Singleton& operator=( Singleton& );
public:
   static Singleton * getInstance();
   void Func();   //змістовна функція для клієнта
   . . .          //інші змістовні функції, потрібні для клієнта
};
```

Визначення функцій класу – файл singleton.cpp

```
#include "singleton.h"

Singleton* Singleton::p_instance = 0;

Singleton* Singleton::getInstance()
{
if(!p_instance)
       p_instance = new Singleton();
return p_instance;
}

void Singleton::Func()
{
. . .    //щось виконується

}
```

Створення об’єкта:

```
#include "singleton.h"
. . .

Singleton* s = Singleton::getInstance();    //створюємо екземпляр об'єкта

s->Func();              //викликаємо потрібну змістовну функцію
```

Примітка. Імена файлів **singleton.h,cpp**, вочевидь, можуть бути іншими

Singleton Меєрса

Оголошення класу:

```
class Singleton
{
private:
    Singleton() {}
    Singleton( const Singleton&);
    Singleton& operator=( Singleton& );
public:
    static Singleton& getInstance()
    {
    static Singleton  instance;
    return instance;
    }
   void Func();   //змістовна функція для клієнта
   . . .          //інші змістовні функції, потрібні для клієнта
};
```

Визначення функцій класу

```
void Singleton::Func()
{
. . .    //щось виконується

}
```

Приклад використання класу

```
Singleton& a = a.getInstance();    //створюємо екземпляр об'єкта

a.Func();   //викликаємо потрібну змістовну функцію
```

Примітка. Так само, як і для класичної реалізації, оголошення файлу можна зробити у окремому файлі-заголовку, а визначення функцій та створення-використання – у відповідних файлах *.cpp

Запис об’єктів у файл

Множину об’єктів, які вводяться у головному вікні та відображаються у головному вікні та вікні таблиці, потрібно записувати у файл.

Програма графічний редактор лаб. №5 повинна записувати кожний об’єкт при вводі у текстовий файл, який міститься у поточній директорії редактора.

Інформацію про поточний об’єкт, що введено, записувати у файл у вигляді одного рядка тексту. Цей рядок додається до вже записаного файлу. Рядок файлу містить п’ять полів: назва класу об’єкта та чотири координати: x1, y1, x2, y2. Поля повинні відокремлюватися символами табуляції.

Для програмування запису у файл можна скористатися бібліотечними функціями, наприклад: **fopen_s( )**, **fprintf( )**, **fclose( )** або іншими.
Нижче приклад запису у файл 10 чисел, розділених символами табуляції ’\t’:

```
    FILE* fout;
    fopen_s(&fout, "myfile.txt", "wt");
    if (fout)
    {
        int num = 10;
        fprintf(fout, "%d numbers:\n", num);
        for (int i=1; i<num; i++)
            fprintf(fout, "%d\t", i);
        fprintf(fout, "%d\n", num);     //завершуємо рядок символом \n
        fclose(fout);

    }
```

Результат:

```
10 numbers:
1    2    3    4    5    6    7    8    9    10
```

Приклад коду для читання усіх рядків текстового файлу:

```
    FILE* fin;
    fopen_s(&fin, "myfile.txt", "rt");
    if (fin)
    {
        char buf[256];
        while (fgets(buf, 256, fin))   //читаємо рядок тексту у буфер
        {
            //... щось робимо з рядком тексту
        }
        fclose(fin);
    }
```

Варіанти завдань та основні вимоги

1. Для усіх варіантів завдань необхідно дотримуватися вимог та положень, викладених вище у порядку виконання роботи та методичних рекомендаціях.

2. Номер варіанту завдання дорівнює номеру зі списку студентів у журналі.
Студенти з **непарним** номером (1, 3, 5, . . .) програмують глобальний статичний об'єкт класу MyEditor у вигляді **Singleton Меєрса**.
Студенти з **парним** номером (2, 4, 6, . . .) програмують об’єкт класу MyEditor на основі **класичної реалізації Singleton**.

3. Усі кольори та стилі геометричних форм – як у попередньої лаб. роботі №4.

4. Запрограмувати вікно таблиці. Для його відкриття та закриття передбачити окремий пункт меню. Вікно таблиці повинно автоматично закриватися при виході з програми.

5. Вікно таблиці – немодальне вікно діалогу. Таблиця повинна бути запрограмована як клас у окремому модулі. Інтерфейс модуля у вигляді оголошення класу таблиці

6. Запрограмувати запис файлу множини об’єктів, що вводяться

7. Оголошення класів для усіх типів об'єктів робити у окремих заголовочних файлах *.h, а визначення функцій членів – у окремих файлах *.cpp. Таким чином, програмний код для усіх наявних типів об'єктів розподілюється по множині окремих модулів.

8. Ієрархія класів та побудова модулів повинні бути зручними для можливостей додавання нових типів об'єктів без переписування коду вже існуючих модулів.

9. У звіті повинна бути схема успадкування класів – діаграма класів. Побудувати діаграму класів засобами Visual Studio C++.

10. Документи звіту – тексти, діаграми, схеми тощо оформлювати у електронному форматі так, щоб їх легко було сприймати у надрукованому звіті. Забороняється текст або графіка "світле на світлому фоні" або "темне на темному фоні". Тільки чорний текст та чорні лінії на білому фоні. Оформлення звіту впливатиме на оцінку.

11. **Зверніть увагу на бонуси-заохочення**, які пропонуються нижче і можуть суттєво підвищити оцінку лабораторної роботи

Бонуси - заохочення

Оцінка підвищується за кожний пункт, з наведених нижче:

1. Якщо у вікні таблиці буде передбачено, щоб користувач міг виділити курсором рядок таблиці і відповідний об’єкт буде якось виділятися на зображенні у головному вікні.

2. Якщо у вікні таблиці користувач може виділити курсором рядок таблиці і відповідний об’єкт буде вилучено з масиву об’єктів.

При виконанні бонусів 1 та 2 забороняється робити для цього нові залежності модуля **my_table** від інших .cpp файлів. Тоді як надіслати повідомлення (наприклад, про виділення користувачем якогось рядка таблиці) від вікна таблиці клієнту цього вікна (наприклад, коду головного файлу .cpp)? Підказки ви зможете знайти у матеріалі лекції стосовно технології **Callback** (а також, можливо, патернів Observer, Listener).

3. Якщо програма не тільки записує у файл опис множини об’єктів, а ще й здатна завантажити такий файл і відобразити відповідні об’єкти у головному вікні та вікні таблиці

Контрольні запитання

1. Що таке Singleton?
2. Чим відрізняється класична реалізація Singleton від Singleton Меєрса?
3. Як запрограмувати немодальне діалогове вікно?
4. Як запрограмувати запис у файл об’єктів - геометричних форм?
5. Покажіть у програмі поліморфізм

У ході захисту-прийняття роботи викладач може також запитувати інше, що стосується виконання роботи.

Зміст звіту

1. Титульний аркуш
2. Варіант завдання
3. Вихідний текст головного файлу .cpp (фрагменти, що ілюструють власний код), та усі вихідні тексти власних модулів
4. Схеми, діаграми згідно завданню. Приклад текстового файлу множини об’єктів.
5. Ілюстрації (скріншоти)
6. Висновки