ЛАБОРАТОРНА РОБОТА №14

РОЗРОБКА БАГАТОФАЙЛОВИХ ПРОЕКТІВ

Мета роботи:

- познайомитися із принципами розробки багатофайлових проектів на мові програмування С;

Теоретичний вступ:

Великі програми на мові С++ (особливо ті, які складаються з тисяч, десятків тисяч чи більше рядків тексту), як правило, розбивають на окремі функції, які, в свою чергу, можуть розбиватися на ще дрібніші функції і так далі. Таке структурування (при грамотному розбитті) значно спрощує завдання кодування та відлагодження програми. Адже відлагодження всіх функцій ніколи не виконується одночасно. Крім того, зберігати всі функції програми в одному файлі незручно і недоцільно. Тому системи розробки програм (в тому числі і Microsoft Visual Studio2019) передбачають засоби розробки, в яких окремі функції (чи група функцій) зберігаються в окремих файлах.

Багатофайловий проект складається з таких частин:

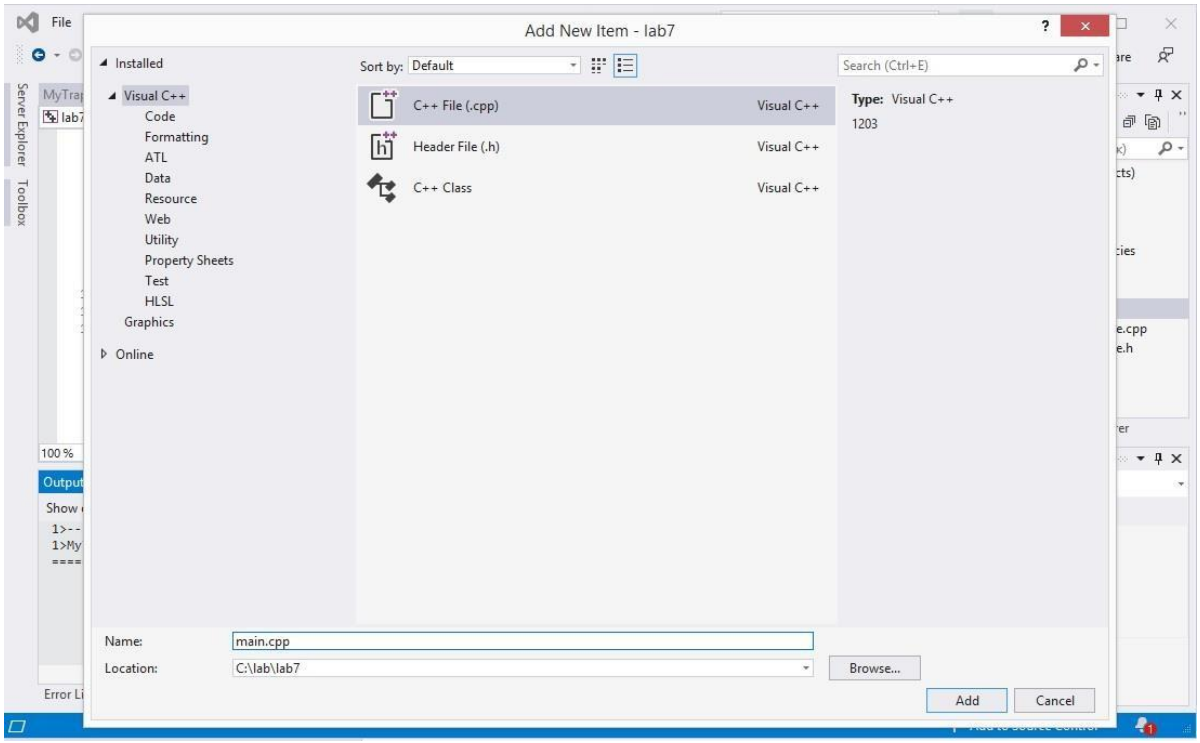
- Файл, в якому міститься визначення функції `main()`;
- Файли, в яких містяться оголошення (прототипи) розроблених функцій;
- Файли, в яких містяться визначення розроблених функцій.

Файли, в яких містяться оголошення, називаються заголовними і мають розширення `.h`, а файли, в яких містяться визначення, мають розширення `.cpp`. Для кожного заголовного файлу створюється `cpp`-файл з реалізацією функцій, оголошених у `h`-файлі. Для зручності такі файли мають однакові імена, але різні розширення.

Розглянемо детальніше створення багатофайлової програми на прикладі задачі обчислення інтегралу за формулою трапецій:

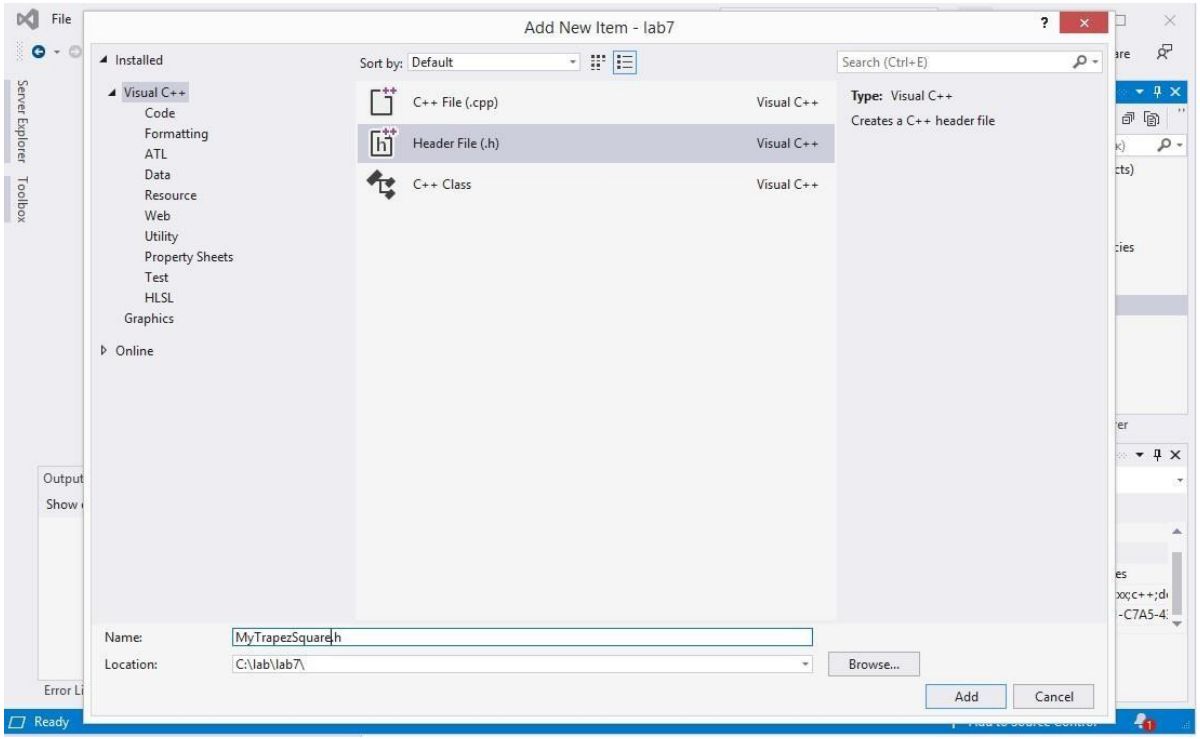
```
/* ----- File main.c ------ */
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "MyTrapezSquare.h"
int main(void)
{
      double a, b, S; int N;
      printf("Enter a, b and N: ");
      scanf("%lf%lf%d", &a, &b, &N);
      S = MyTrapezSquare(a, b, N);
      printf("\nFor a = %f, b = %f, N = %d, S = %f", a, b, N, S);
      return 0;
}

/* File MyTrapezSquare.h */
double MyTrapezSquare(double a, double b, int N);

/* File MyTrapezSquare.cpp */
#include <math.h>
double MyTrapezSquare(double a, double b, int N)
{
      double x, h, S1, S2;
      h = (b - a) / N;
      S1 = (sin(a) + sin(b)) / 2;
      S2 = 0;
      for (x = a + h; x < b; x = x + h)
            S2 = S2 + sin(x);
      return  h * (S1 + S2);
}
```

Спочатку у порожньому проекті додаємо файл `main.cpp`, де буде визначення функції `main()` за допомогою пункту меню *Project –> Add New Item.*



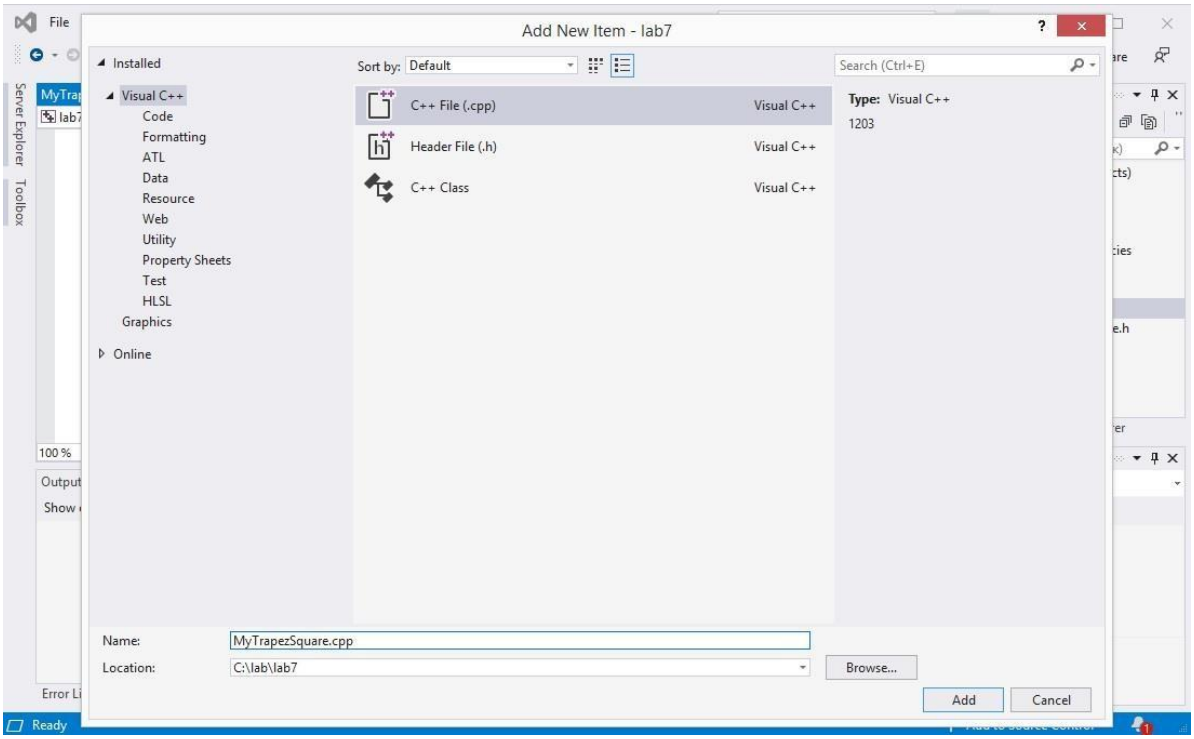
Додавання до проекту файлу `main.cpp`

Далі додаємо аналогічним чином файл `MyTrapezSquare.h`:



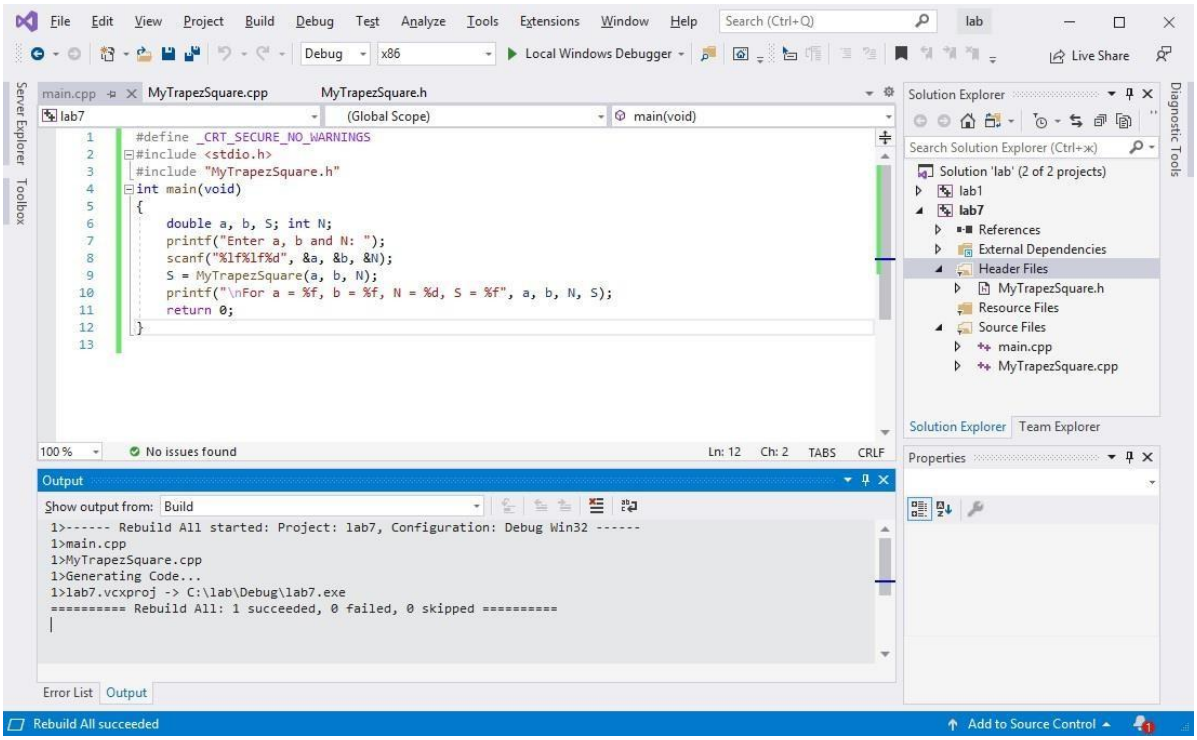
Додавання до проекту файлу `MyTrapezSquare.h`

І файл `MyTrapezSquare.cpp`:



Додавання до проекту файлу `MyTrapezSquare.cpp`

Тепер поміщаємо у створені файли тексти програми. Після цього ми можемо відкомпілювати кожен `cpp`-файл окремо і створити `exe`-файл програми:



Створення виконавчого файлу багатофайлового проекту

Тексти у cpp-файлах є одиницями трансляції (*translation unit*), або програмними одиницями. Кожну одиницю трансляції можна відкомпілювати окремо, результатом будуть об’єктні файли з розширенням `.obj`. Далі за допомогою компоновщика з цих файлів створюється виконавчий код програми.

Якщо тепер внести зміни лише в деякі з файлів багатофайлового проекту, то при новому запуску на компіляцію, компілюватися будуть лише змінені файли, незмінені файли не компілюються, а для компоновки використовуються їх раніше відкомпільовані об'єктні модулі. Такий підхід дає значну економію часу для великих проектів.

Порядок виконання лабораторної роботи:

- Написати програму згідно індивідуального завдання у вигляді багатофайлового проекту.
- Створити багатофайловий проект згідно із індивідуальним завданням, відлагодити програму та отримати результати роботи програм.

Контрольні запитання:

1) Що таке багатофайловий проект?
2) Для чого використовуються багатофайлові проекти?
3) Що таке заголовний файл?
4) Як підключити заголовний файл до програми?
5) Що таке одиниця трансляції?
6) Як додати до проекту новий cpp-файл?
7) Як додати до проекту новий h-файл?

Індивідуальні завдання:

1. **Обчислення площі прямокутника**

Реалізувати окрему функцію обчислення площі прямокутника за заданими сторонами.

2. **Обчислення периметра та площі кола**

Функції для обчислення довжини кола та площі за заданим радіусом.

3. **Обчислення середнього арифметичного масиву**

Функція отримує масив і його розмір та повертає середнє значення.

4. **Пошук максимального елемента масиву**

Реалізувати функцію пошуку максимального елемента одновимірного масиву.

5. **Пошук мінімального елемента масиву**

Реалізувати функцію пошуку мінімального елемента масиву.

6. **Обчислення суми елементів масиву**

Функція повертає суму всіх елементів масиву.

7. **Перевірка числа на парність**

Реалізувати функцію, яка визначає, чи є число парним.

8. **Обчислення факторіалу числа**

Функція обчислює факторіал заданого цілого числа (без рекурсії).

9. **Піднесення числа до степеня**

Реалізувати окрему функцію піднесення числа до цілого степеня.

10. **Обчислення суми цифр числа**

Функція визначає суму цифр заданого цілого числа.

11. **Перевірка числа на простоту**

Реалізувати функцію, що визначає, чи є число простим.

12. **Обчислення площі трикутника за формулою Герона**

Функція отримує довжини сторін трикутника.

13. **Перетворення температури**

Реалізувати функції переведення температури з °C у °F та навпаки.

14. **Підрахунок кількості додатних елементів масиву**

Функція повертає кількість додатних чисел у масиві.

15. **Обчислення середнього геометричного двох чисел**

Реалізувати окрему функцію для обчислення середнього геометричного.