

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

НАВЧАЛЬНЕ ВИДАННЯ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

**з дисципліни
ПРОГРАМУВАННЯ, ЧАСТИНА 1
(ОСНОВИ АЛГОРИТМІЗАЦІЇ І ПРОГРАМУВАННЯ)**

для студентів галузі знань «12 Інформаційні технології»
спеціальності 123 «Комп'ютерна інженерія»

*Рекомендовано Науково-методичною радою
Національного університету "Львівська політехніка"*

Львів 2021

Автор:

Ногаль М. В., ст. викл. каф. ЕОМ.

Рецензенти:

Глухов В. С., д. т. н., проф. каф ЕОМ,
Сало А. М., к. т. н., генеральний директор ТзОВ
«Кіберсофт технології»,
Шаров Б. Г., к. т. н., ТзОВ «ЕЛСИ».

Рекомендовано Науково-методичною радою
Національного університету «Львівська політехніка»
(протокол № 59 від 20 жовтня 2021 року)

Лабораторний практикум з дисципліни «Програмування, частина 1 (Основи алгоритмізації і програмування)» для студентів галузі знань «12 Інформаційні технології» спеціальності 123 «Комп’ютерна інженерія» / Автор. Ногаль М. В.

Практикум містить комплекс з дев’яти методичних вказівок до виконання лабораторних робіт з курсу «Програмування, частина 1 (Основи алгоритмізації і програмування)».

© Ногаль М. В., 2021

© Національний університет

«Львівська політехніка», 2021

ISBN

ВСТУП

Не зважаючи на всю множину сучасних напрямків та мов програмування, основи програмування завжди однакові. Розв'язання будь-якої задачі починається з постановки завдання, продовжується вибором методу її розв'язку, розробкою алгоритму, написанням програми і завершується етапом налагодження. Для успішного вирішення усіх цих етапів потрібні знання з алгоритмізації і основ програмування.

Даний лабораторний практикум містить методичні вказівки до дев'яти лабораторних робіт з дисципліни «Програмування, частина 1 (Основи алгоритмізації і програмування)». До кожної роботи наведено теоретичні відомості, порядок виконання роботи, перелік контрольних запитань для самоперевірки і індивідуальні завдання. Також для кожної роботи є приклади розв'язку задач заданої тематики, що полегшить роботу здобувачу освіти над своїм індивідуальним завданням.

Метою вивчення дисципліни «Програмування, частина 1 (Основи алгоритмізації і програмування)» є формування у здобувачів освіти знань та практичних навичок, необхідних для усіх етапів розв'язку поставленої задачі, таких як розробка алгоритму, написання програми на мові програмування C/C++, налагодження її і виконання.

В якості мови програмування у дисципліні «Програмування, частина 1 (Основи алгоритмізації і програмування)» вибрано мову програмування C/C++. Мова програмування C – це універсальна, процедурно-орієнтована мова загального призначення, яка була розроблена для написання системного програмного забезпечення. C дозволяє забезпечити низькорівневий доступ до оперативної пам'яті, що дозволяє кодові працювати на дуже обмеженому апаратному забезпеченні, такому як вбудовані системи, які сьогодні мають таку високу функціональність саме завдяки використанню мови C. Мова програмування C++ проектувалась як об'єктно-орієнтоване розширення для мови C і детально розглядається у дисципліні «Програмування, частина 2 (Об'єктно-орієнтоване програмування)».

В даній дисципліні розглядаються основи алгоритмізації задач, поняття алгоритму, види алгоритмів – лінійні, з розгалуженням і циклічні алгоритми, запис алгоритмів за допомогою блок-схем. Розглянута мова програмування C/C++: структура програми, типи даних, константи, змінні, операції, оператори, поняття функції, масиви, вказівники, рядки, засоби для роботи з текстовими і бінарними файлами. Розглянуто типові алгоритми пошуку, сортування, обробки масивів і матриць, а також приклади їх реалізації на мові програмування C/C++.

В результаті вивчення дисципліни «Програмування, частина 1 (Основи алгоритмізації і програмування)» здобувач освіти повинен бути здатним продемонструвати такі результати навчання:

1. знати базові структури алгоритмів і основи алгоритмізації задач;
2. знати прийоми програмування на мові програмування C/C++;
3. вміти розробити алгоритм для конкретної задачі;
4. вміти написати на мові програмування C/C++ програму для вирішення конкретної задачі;
5. вміти налагоджувати програму для вирішення конкретної задачі;
6. вміти виконувати програму для вирішення конкретної задачі;
7. мати навички роботи в операційних системах і інтегрованих середовищах розробки програмного забезпечення.

Для успішного вивчення даної дисципліни здобувач освіти протягом семестру має виконати комплекс лабораторних робіт, а даний практикум допоможе йому в цьому.

ЗМІСТ

Лабораторна робота №1	Інтегроване середовище розробки Microsoft Visual Studio 2019.....	6
Лабораторна робота №2	Розв'язування на мові C найпростіших задач з використанням стандартних функцій вводу-виводу	17
Лабораторна робота №3	Розв'язування на мові C задач з використанням умовних операторів та операторів циклу	28
Лабораторна робота №4	Розв'язування на мові C задач, в яких використовується визначення і виклик функцій.....	48
Лабораторна робота №5	Розв'язування на мові C задач, в яких використовуються числові масиви.....	62
Лабораторна робота №6	Розв'язування на мові C задач, в яких використовуються динамічні числові масиви.....	69
Лабораторна робота №7	Розробка на мові C багатофайлових проектів.....	81
Лабораторна робота №8	Розв'язування на мові C задач, в яких використовуються масиви типу char і рядки.....	90
Лабораторна робота №9	Розв'язування на мові C задач, які використовують файли для вводу та виводу даних.....	98
РЕКОМЕНДОВАНІ ДЖЕРЕЛА.....		107

ЛАБОРАТОРНА РОБОТА №1 ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ MICROSOFT VISUAL STUDIO 2019

Мета роботи:

- познайомитися з послідовністю кроків, які необхідно виконати в інтегрованому середовищі Microsoft Visual Studio 2019, при розробці найпростішої програми;
- познайомитись з вбудованим текстовим редактором інтегрованого середовища Microsoft Visual Studio 2019.

Теоретичний вступ

Інтегроване середовище Microsoft Visual Studio 2019 працює під керуванням операційної системи Windows. Воно дозволяє розробляти програми як об'єктно-орієнтованою мовою C++, так і розробленою ще на початку 70-х років процедурно-орієнтованою мовою C.

Інтегроване середовище об'єднує цілий ряд компонент, які необхідні при розробці програм. Найважливішими з них є компілятор, текстовий редактор, відлагоджувач (англійською *debugger*) та ряд інших.

З допомогою інтегрованого середовища Microsoft Visual Studio 2019 можна розробляти різноманітні програмні продукти – консольні аплікації, які працюють лише в текстовому режимі, модулі динамічних бібліотек, модулі статичних бібліотек, програми для роботи з базами даних та ряд інших.

В лабораторних роботах з курсу "Програмування, частина 1 (Основи алгоритмізації та програмування)" будуть розроблятися програми, які будуть працювати у консольному режимі, тобто такі, які працюють в текстовому режимі.

Розробка в інтегрованому середовищі Microsoft Visual Studio 2019 консольної програми включає ряд етапів:

- створення проекту програми;
- створення файлу програми та його редагування;
- компіляція програми;
- виконання програми.

Розглянемо виконання цих етапів детальніше.

Після запуску інтегрованого середовища Microsoft Visual Studio 2019 з'являється наступне вікно:

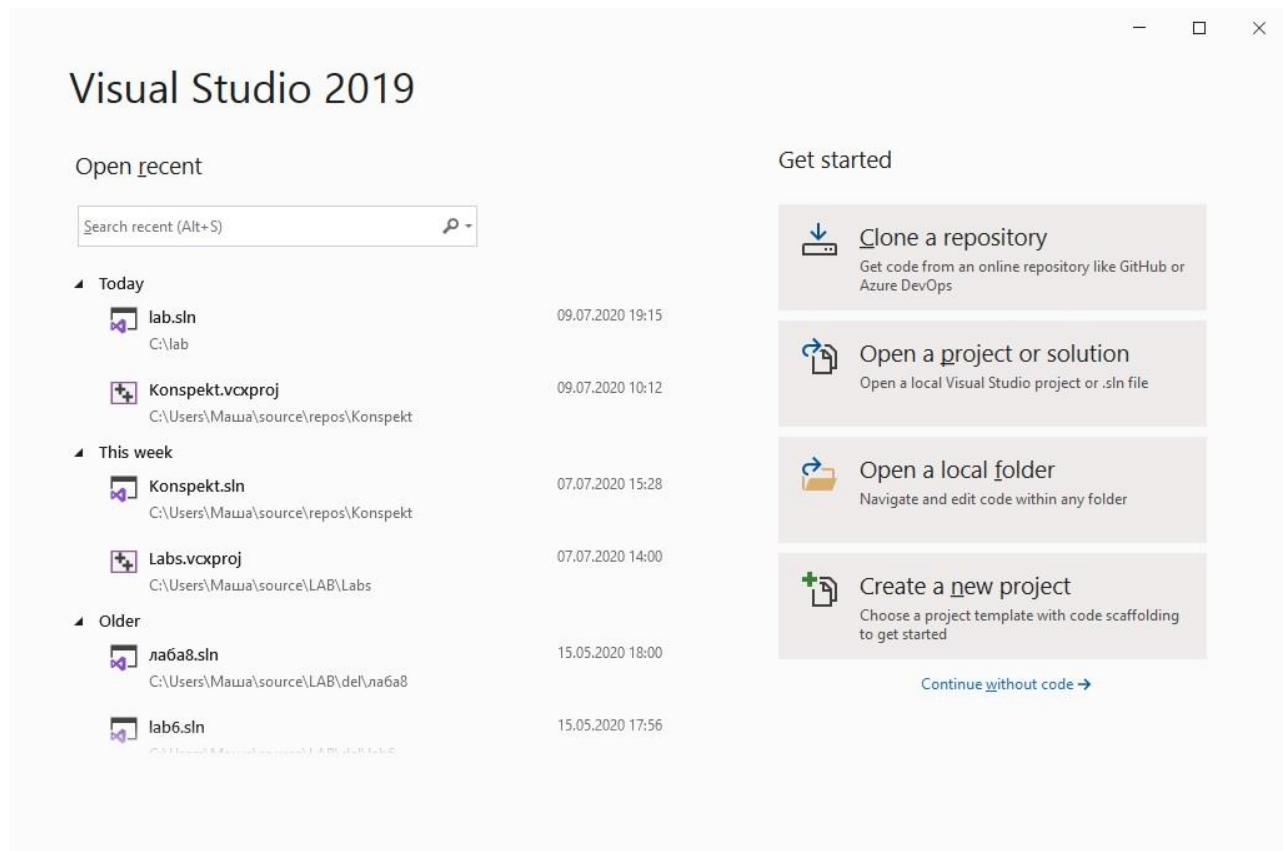


Рисунок 1.1 – Запуск інтегрованого середовища Microsoft Visual Studio 2019

Тепер можна або відкрити існуючий проект, або створити новий проект. Для створення нового проекту треба натиснути *Create a new project*. Якщо середовище вже запущено з іншим проектом, то створити новий можна вибравши меню *File* → *New* – *Project*.

Для відкриття одного з останніх проектів, з яким працювали нещодавно (їх перелік розміщено зліва) потрібно натиснути на його ім'я. Щоб відкрити існуючий проект треба натиснути *Open a project or solution*.

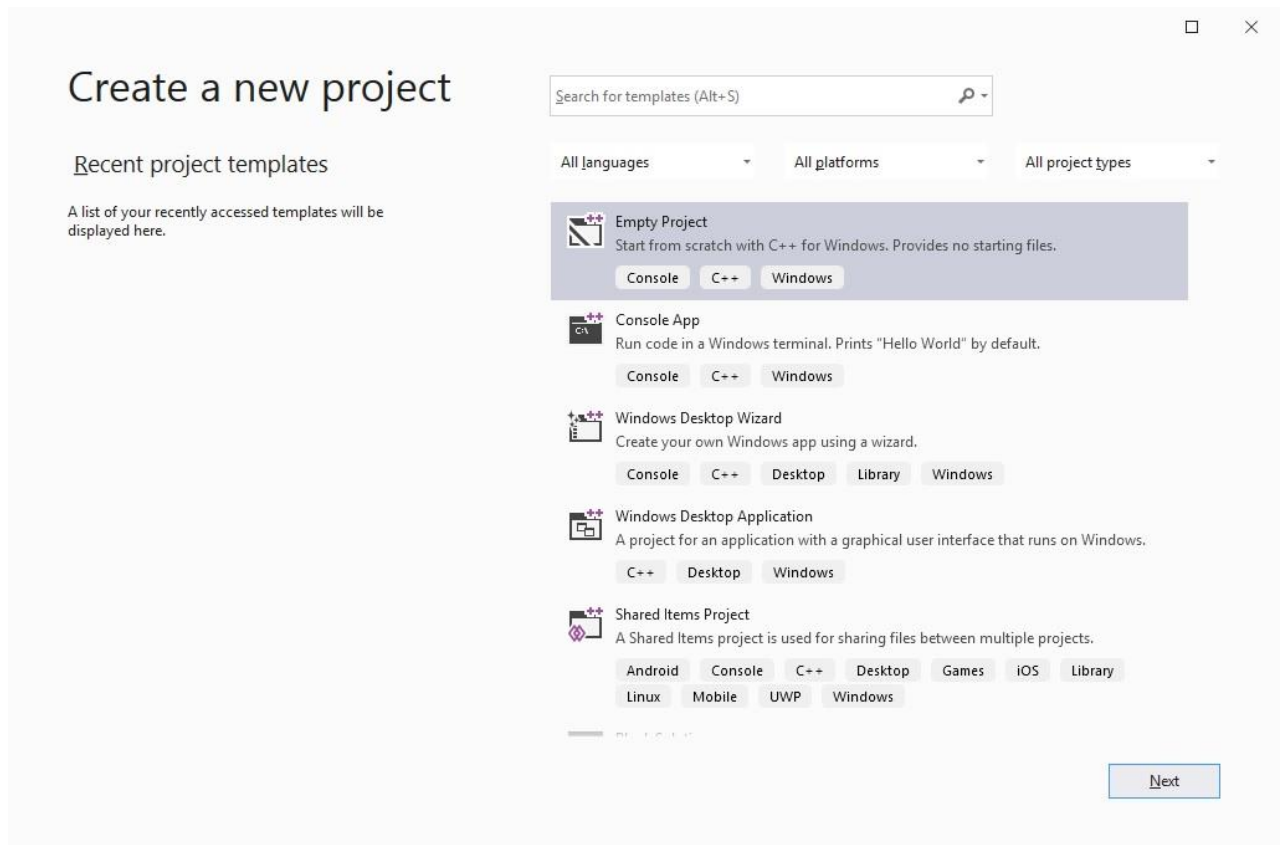


Рисунок 1.2 – Створення нового проекту

Тут необхідно вибрати *Console App* і в наступному вікні вказати ім'я проекту (*Project name*), шлях до його розміщення на диску (*Location*). Також треба вказати ім'я рішення (*Solution* – контейнер для одного або декількох проектів у Visual Studio або поставити галочку «помістити рішення і проект в одну директорію» (*Place solution and project in the same directory*). Натискаємо кнопку *Create*.

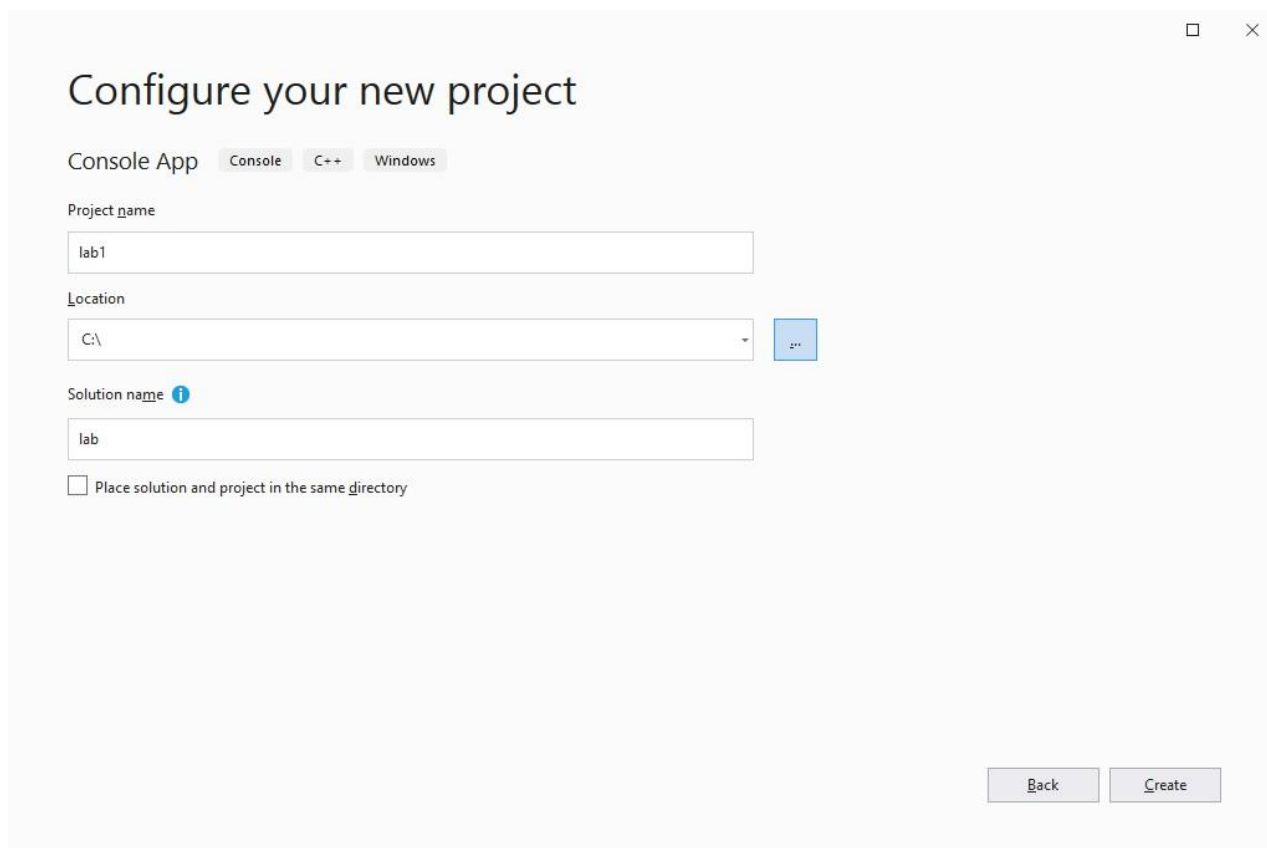


Рисунок 1.3 – Вибір назви і шляху розміщення нового проекту

Після цього з'являється наступне вікно:

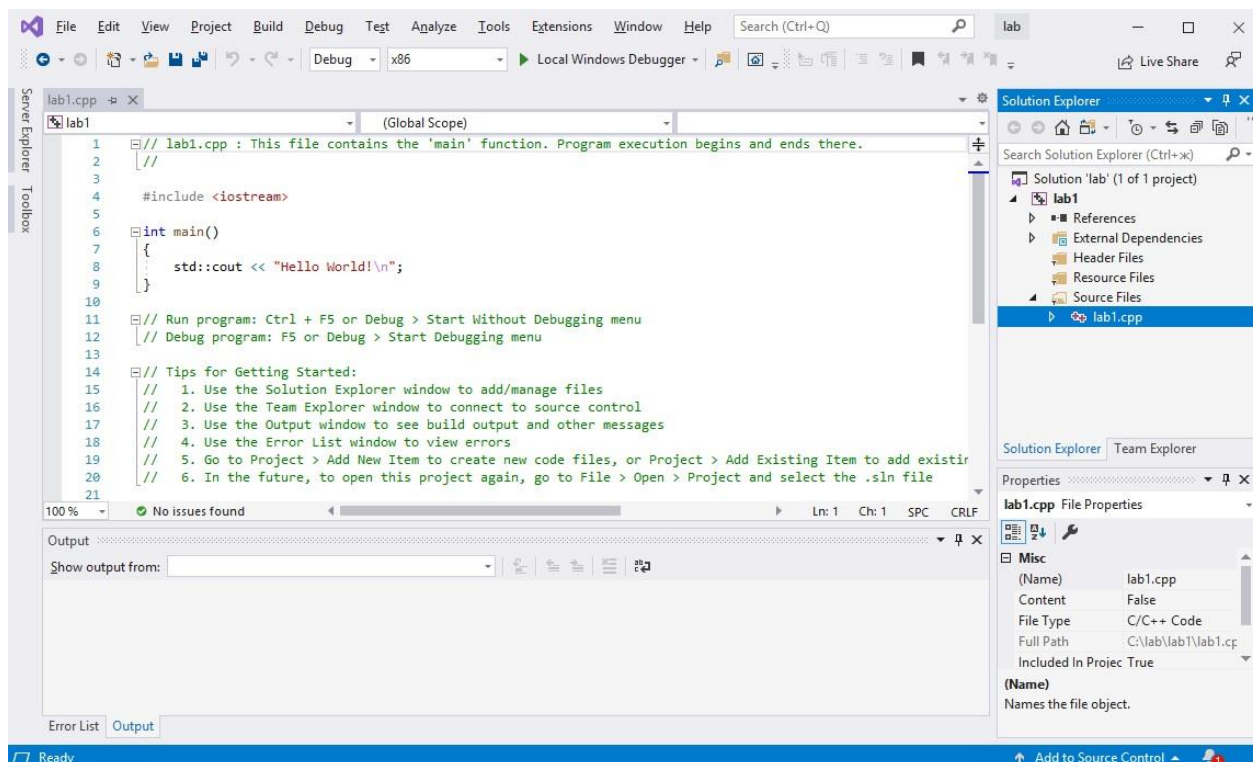


Рисунок 1.4 – Новий проект

У створеному проєкті вже додано файл програми (*lab1.cpp*), вміст якого складається з коментарів (починаються з двох косих ліній *//*) і тексту простої програми, яка виводить на екран повідомлення “Hello world!”. Для написання власної програми вміст файлу *lab1.cpp* можна видалити і на його місці писати свою програму. Вбудований текстовий редактор у середовищі Visual Studio підсвічує синтаксис програми, а також має механізм автодоповнення коду IntelliSense для змінних, функцій, методів, циклів і іншого.

Для того, щоб відкомпілювати програму, необхідно вибрати пункт меню *Build* → *Compile* або натиснути клавіші *Ctrl+F7*.

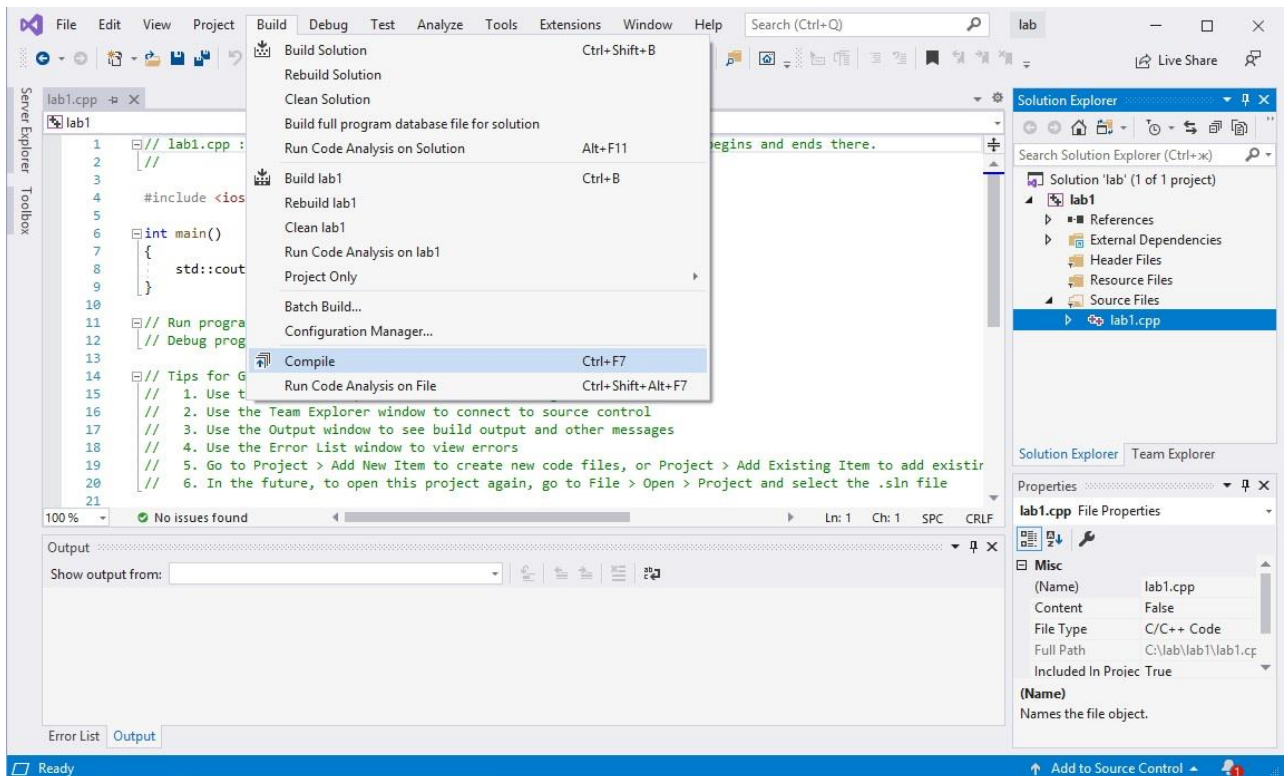


Рисунок 1.5 – Компілювання файлу проєкту

Результати компілювання з’являться у вікні *Output*. Якщо у програмі будуть помилки компілятор про це повідомить.

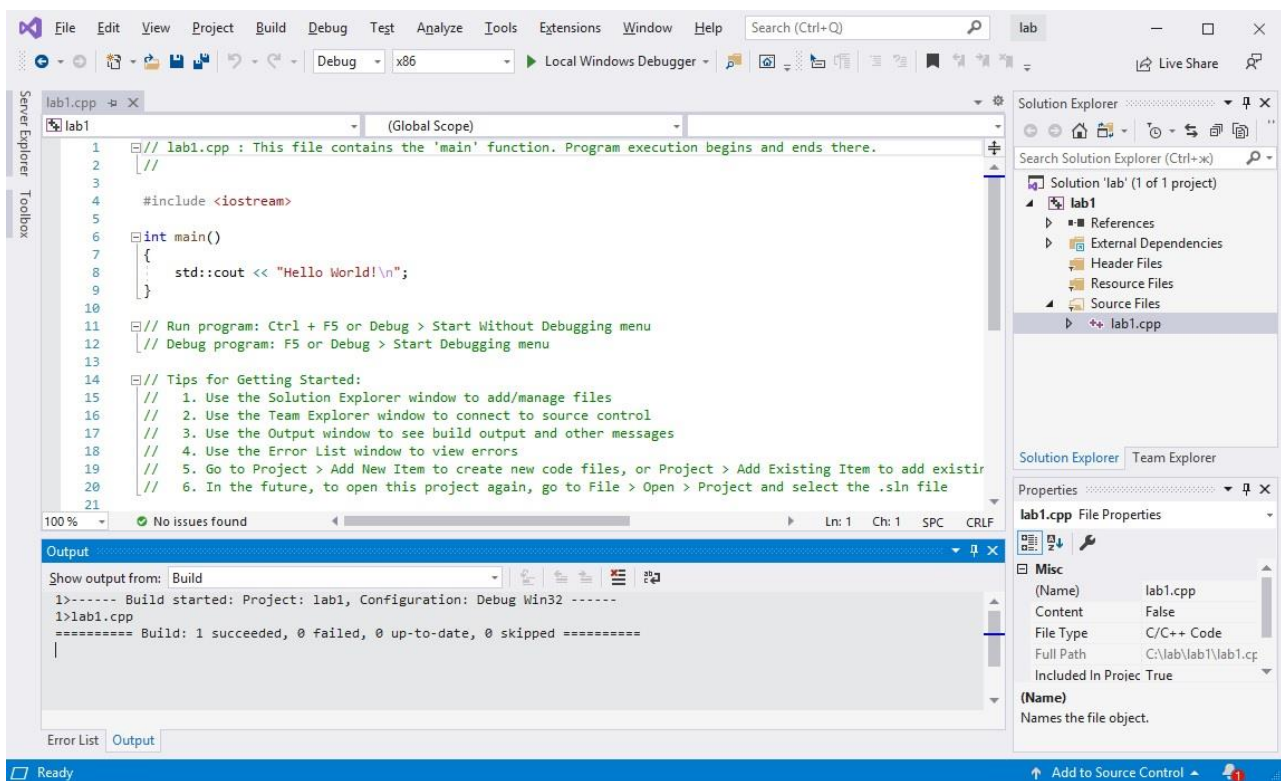


Рисунок 1.6 – Результати компілювання файлу проекту

Для проекту у Visual Studio можна вибрати конфігурацію: відлагодження (*Debug*) і випуск (*Release*). Для лабораторних робіт краще вибирати конфігурацію відлагодження, а конфігурацію випуск вибирати вже для готового відлагодженого проекту. Також можна вибрати платформу, для якої буде створено виконавчий файл: x64 для 64-х розрядних ЕОМ, а x86 для 32-х розрядних.

Для того, щоб запустити програму на виконання з відлагодженням, необхідно натиснути зелену кнопку *Local Windows Debugger* або вибрати пункт меню *Debug* → *Start Debugging* (або натиснути клавішу *F5*). Можна відразу запустити програму вибравши пункт меню *Debug* → *Start Without Debugging* (або натиснути клавіші *Ctrl+F5*).

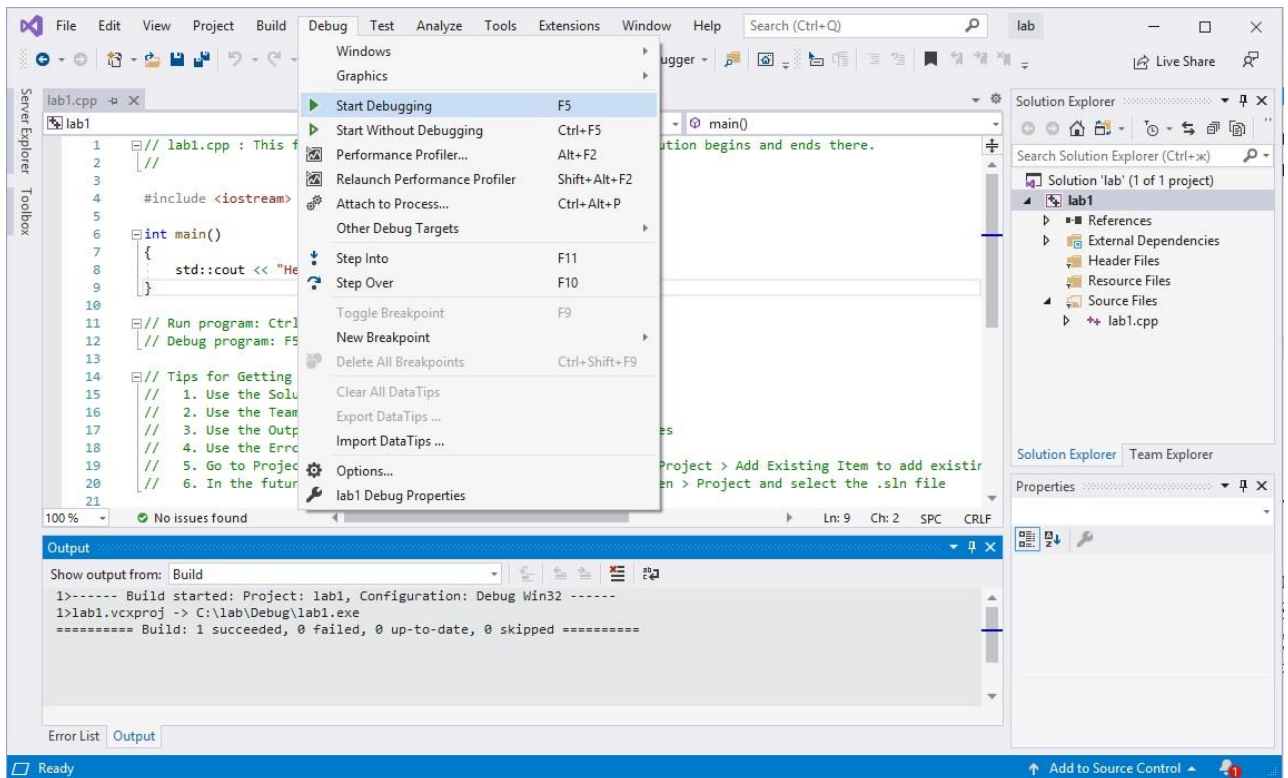


Рисунок 1.7 – Запуск програми на виконання

Під час виконання програми з'явиться чорне вікно – консоль, де будуть виводитись результати програми.

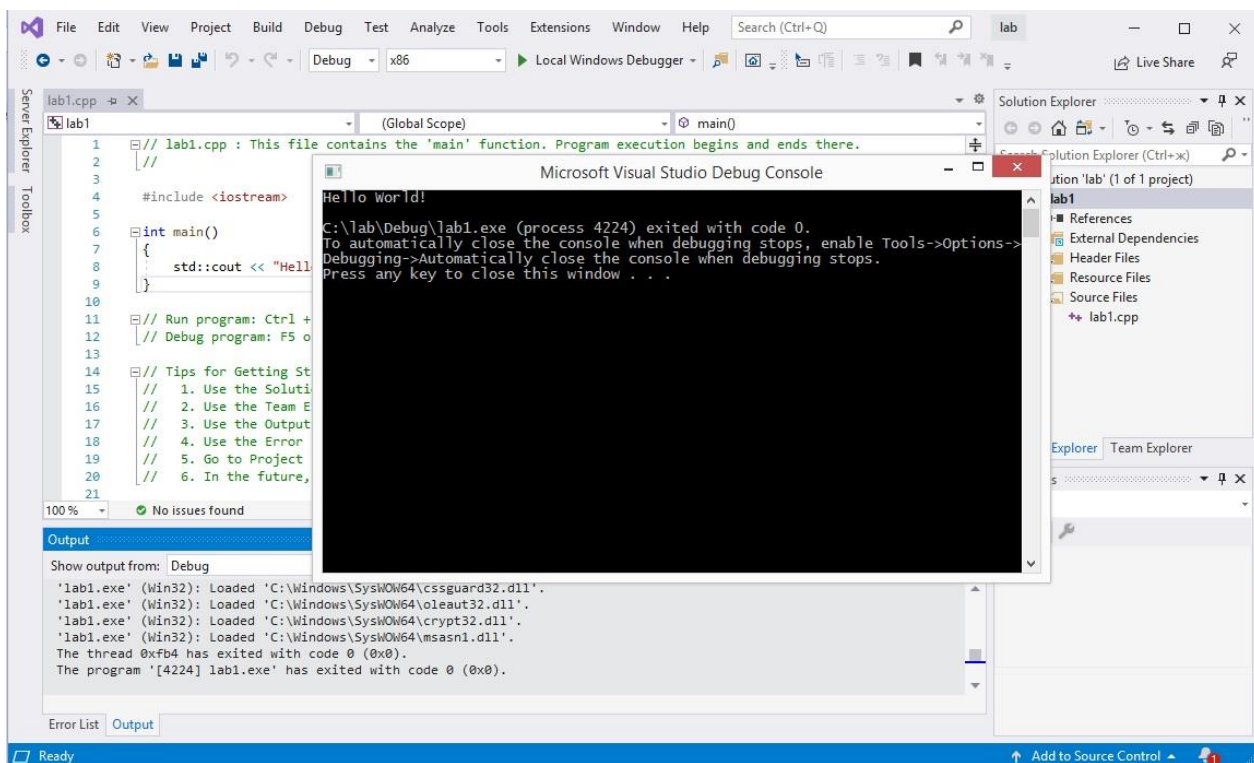


Рисунок 1.8 – Результати роботи програми у консольному вікні

Можна створити виконавчий файл (*lab1.exe*) без запуску програми на виконання, для цього необхідно вибрати пункт меню *Build* → *Build lab1*. В вікні *Output* з'явиться повідомлення з шляхом до створеного файлу.

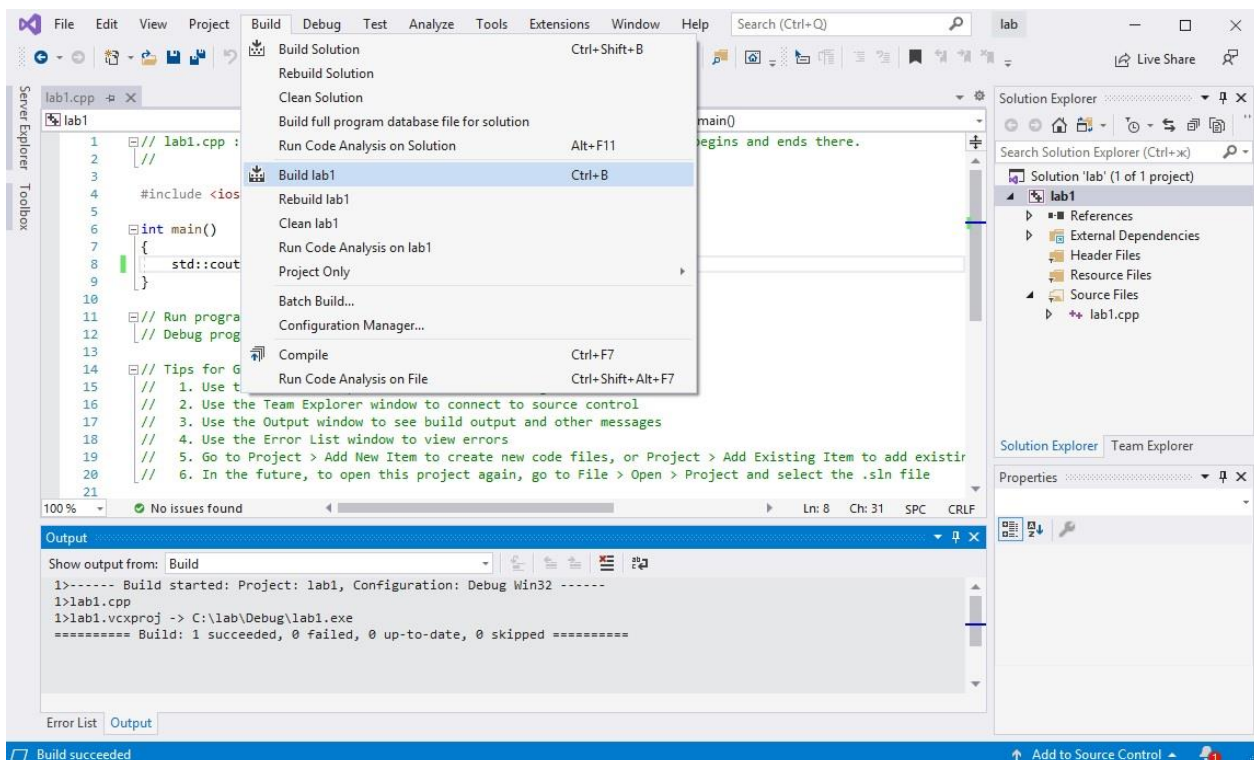


Рисунок 1.9 – Створення виконавчого файлу програми

Якщо є необхідність додати в проект вже готові файли програм (з розширеннями *.h* і *.cpp*), то це можна зробити вибравши пункт меню *Project* → *Add Existing Item*. Далі вибрати шлях і необхідний файл. Також при потреба можна додати нові файли до проекту, за допомогою пункту меню *Project* → *Add New Item*, далі вибрати необхідний тип файлу.

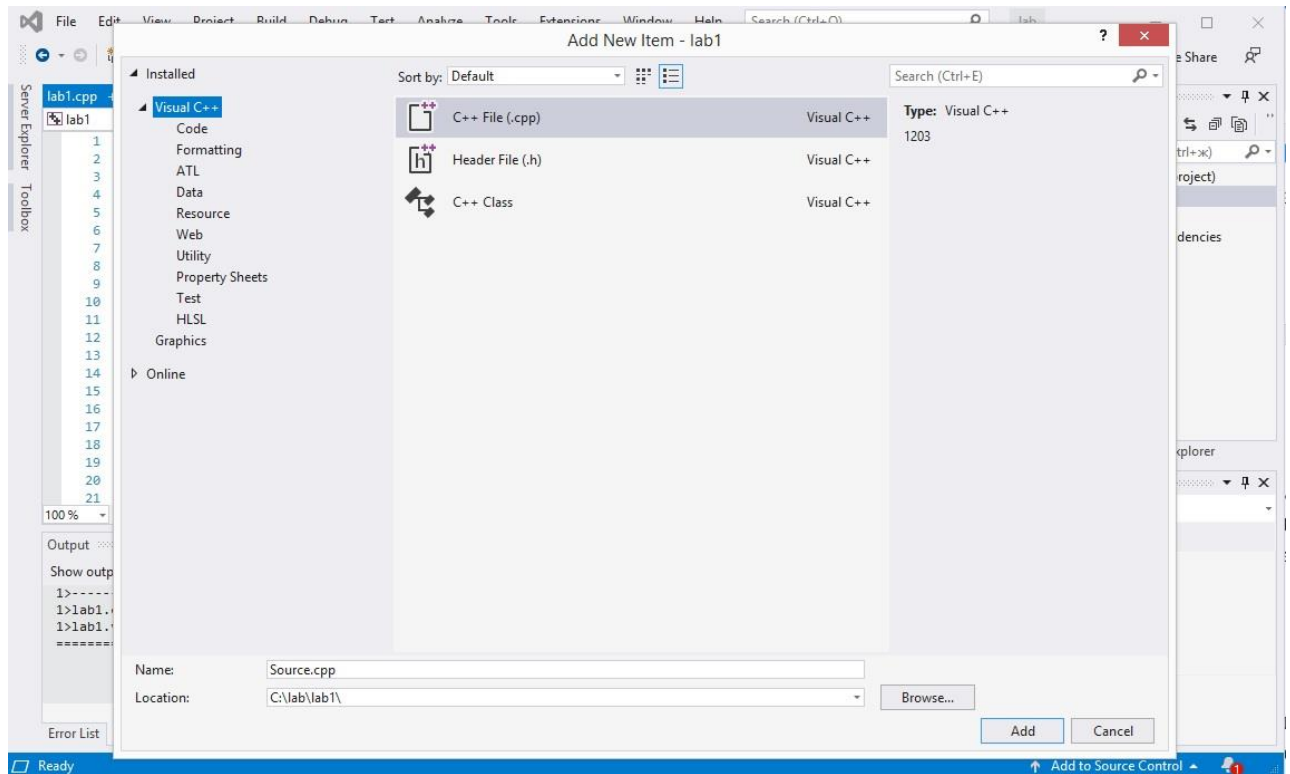


Рисунок 1.10 – Додавання нового файлу до проекту

Порядок виконання лабораторної роботи

- Використовуючи вище наведений опис, познайомитися з інтегрованим середовищем розробки програм Microsoft Visual Studio 2019;
- Запустити на виконання інтегроване середовище розробки програм Microsoft Visual Studio 2019;
- Створити новий проект в середовищі Microsoft Visual Studio 2019;
- З допомогою вбудованого текстового редактора підготувати текст(и) програми(програм), які скаже викладач; записати їх на диск у відповідний каталог, відкомпілювати та запустити на виконання.

Контрольні запитання

- 1) Для чого призначене інтегроване середовище розробки програм Microsoft Visual Studio 2019?
- 2) Які основні компоненти інтегрованого середовища розробки програм Microsoft Visual Studio 2019?
- 3) В якій конфігурації буде використовуватися інтегроване середовище розробки програм Microsoft Visual Studio 2019 у лабораторних роботах з курсу "Програмування, частина 1 (Основи алгоритмізації та програмування)"?
- 4) З яких етапів складається розробка консольних програм в інтегрованому середовищі Microsoft Visual Studio 2019?
- 5) Де при створенні проекту задається ім'я проекту?
- 6) Де при створенні проекту задається місце розміщення проекту?
- 7) Як виглядає вікно інтегрованого середовища після створення нового проекту?
- 8) З допомогою якої клавіші запускається компіляція програми?
- 9) Де розміщається виконавчий файл програми?
- 10) Яким буде ім'я виконавчого файлу?
- 11) З допомогою якої клавіші можна запустити програму на виконання?
- 12) Куди виводяться результати роботи програми?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Ескіз головного вікна керування розробкою програми системи Microsoft Visual Studio 2019 після компіляції програми.
- Текст(и) програми(програм), які набиралися з допомогою вбудованого редактора.
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Приклади програм

Лістинг 1.1: Обчислення середнього арифметичного трьох чисел та вивід результату

```
#include <stdio.h>
int main(void)
{
    double res, x1, x2, x3;
    x1 = 4.5;
    x2 = 5.6;
    x3 = 7.8;
    res = (x1 + x2 + x3) / 3;
    printf("Result is %f\n", res);
    return 0;
}
```

Результати роботи програми:

```
Result is 5.966667
```

Лістинг 1.2: Обчислення кореня квадратного суми квадратів двох чисел

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double res, x1, x2;
    x1 = 4.5;
    x2 = 5.6;
    res = sqrt(x1 * x1 + x2 * x2);
    printf("Square root is %f\n", res);
    return 0;
}
```

Результати роботи програми:

```
Square root is 7.184010
```


ЛАБОРАТОРНА РОБОТА №2 РОЗВ'ЯЗУВАННЯ НА МОВІ C НАЙПРОСТІШИХ ЗАДАЧ З ВИКОРИСТАННЯМ СТАНДАРТНИХ ФУНКЦІЙ ВВОДУ-ВИВОДУ

Мета роботи: познайомитися з виразами і операціями мови C, та стандартними функціями вводу-виводу.

Теоретичний вступ

Операнди, вирази і операції в мові C.

Комбінація знаків операцій і операндів, результатом якої є певне значення, називається *виразом*. Знаки операцій визначають дії, які повинні бути виконані над операндами. Кожен операнд у виразі може бути виразом. Значення виразу залежить від розташування знаків операцій і круглих дужок у виразі, а також від пріоритету виконання операцій. У мові C присвоєння також є виразом, і значенням такого виразу є величина, яка присвоюється.

Операнд - це константа, літерал, ідентифікатор, виклик функції, індексний вираз, вираз вибору елемента або більш складний вираз, сформований комбінацією операндів, знаків операцій і круглих дужок. Будь-який операнд, який має константні значення, називається константним виразом. Кожен операнд має тип.

За кількістю операндів, що беруть участь в операції, операції поділяються на *унарні*, *бінарні* і *тернарні*.

Таблиця 2.1 – Унарні операції в мові C

Знак операції	Операція
-	арифметичне заперечення (заперечення і доповнення)
~	порозрядне (побітове) логічне заперечення (доповнення)
!	логічне заперечення;
*	розадресація (непряма адресація)
&	обчислення адреси
+	унарний плюс
++	збільшення (інкремент)
--	зменшення (декремент)
sizeof	розмір змінної чи типу в байтах

Таблиця 2.2 – Бінарні операції в мові C

Знак операції	Операція	Група операцій
*	Множення	Мультиплікативні
/	Ділення	
%	Залишок від ділення	
+	Додавання	Адитивні
-	Віднімання	
<<	Зсув вліво	Операції зсуву
>>	Зсув вправо	
<	Менше	Операції відношення
<=	Менше або рівно	
>=	Більше або рівно	
==	Рівно	
!=	Не рівно	
&	Порозрядне І	Порозрядні (побітові) операції
	Порозрядне АБО	
^	Порозрядне виключне АБО	
&&	Логічне І	Логічні операції
	Логічне АБО	
,	Послідовне обчислення	Послідовного обчислення
=	Присвоєння	Операції присвоєння
*=	Множення з присвоєнням	
/=	Ділення з присвоєнням	
%=	Залишок від ділення з присвоєнням	
-=	Віднімання з присвоєнням	
+=	Додавання з присвоєнням	
<<=	Зсув вліво з присвоєнням	
>>=	Зсув вправо з присвоєнням	
&=	Порозрядне І з присвоєнням	
=	Порозрядне АБО з присвоєнням	
^=	Порозрядне виключне АБО з присвоєнням	

Таблиця 2.3 – пріоритети операцій

Пріоритет	Знак операції	Типи операцій	Порядок виконання
2	() [] . ->	Вираз	Зліва направо
1	- ~ ! * & ++ -- sizeof приведення типів	Унарні	Справа наліво
3	* / %	Мультиплікативні	Зліва направо
4	+ -	Адитивні	
5	<< >>	Зсуву	
6	< > <= >=	Відношення	
7	== !=	Відношення (рівність)	
8	&	Порозрядне І	
9	^	Порозрядне виключне АБО	
10		Порозрядне АБО	
11	&&	Логічне І	
12		Логічне АБО	
13	? :	Умовна операція	
14	= *= /= %= += -= &= = >>= <<= ^=	Присвоєння	Справа наліво
15	,	Послідовне обчислення	Зліва направо

Приклади виразів на мові програмування C:

```

y = cos(x * x) + pow(sin(x), 2);
y = pow(x, 1./5);
y = (sin(x - 3.14) + 1) / (exp(x) - 2.5 * x);

```

Консольний ввід-вивід з клавіатури.

При запуску програми на C автоматично відкриваються п'ять потоків, основними з яких є наступні:

- Стандартний потік введення `stdin`;
- Стандартний потік виводу `stdout`;
- Стандартний потік виводу повідомлень про помилки `stderr`.

Стандартний потік введення `stdin` за замовчуванням відповідає клавіатурі, а потоки `stdout` і `stderr` - екрану монітора.

Для форматованого виводу інформації в консольному режимі є функція `printf()`. Загальний вигляд функції є такий:

```
int printf( const char* format[, argument]... );
```

Тут `format` задає пояснювальний текст та вигляд значень змінних, імена яких задає параметр `argument` (список змінних). Параметр список змінних не є обов'язковим і являє собою послідовність розділених комами змінних, значення яких виводяться.

Специфікатор формату задає вигляд виведеного результату.

Специфікатор формату для `printf()`:

`%[прапорці] [ширина] [.точність] [{ h | l | L }]тип`

Таблиця 2.4 – Специфікатори формату для функції `printf()`

Модифікатор	Значення										
прапорці	<p>Прапорці :</p> <table><tr><td>–</td><td>Здійснює вирівнювання по лівому краю.</td></tr><tr><td>+</td><td>Дані зі знаком друкуються зі знаком плюс, якщо вони додатні, і зі знаком мінус, якщо вони від'ємні.</td></tr><tr><td>Пробіл (Space)</td><td>Дані зі знаком друкуються з пробілом (без знаку), якщо вони додатні, і зі знаком мінус, якщо вони від'ємні.</td></tr><tr><td>#</td><td>Виводить спочатку 0 для форми <code>%o</code>, та 0x для форми <code>%x</code>. Для форм з плаваючою крапкою гарантує вивід десяткової крапки.</td></tr><tr><td>0</td><td>Для чисельних форм заповнює поле нулями замість пробілів.</td></tr></table>	–	Здійснює вирівнювання по лівому краю.	+	Дані зі знаком друкуються зі знаком плюс, якщо вони додатні, і зі знаком мінус, якщо вони від'ємні.	Пробіл (Space)	Дані зі знаком друкуються з пробілом (без знаку), якщо вони додатні, і зі знаком мінус, якщо вони від'ємні.	#	Виводить спочатку 0 для форми <code>%o</code> , та 0x для форми <code>%x</code> . Для форм з плаваючою крапкою гарантує вивід десяткової крапки.	0	Для чисельних форм заповнює поле нулями замість пробілів.
–	Здійснює вирівнювання по лівому краю.										
+	Дані зі знаком друкуються зі знаком плюс, якщо вони додатні, і зі знаком мінус, якщо вони від'ємні.										
Пробіл (Space)	Дані зі знаком друкуються з пробілом (без знаку), якщо вони додатні, і зі знаком мінус, якщо вони від'ємні.										
#	Виводить спочатку 0 для форми <code>%o</code> , та 0x для форми <code>%x</code> . Для форм з плаваючою крапкою гарантує вивід десяткової крапки.										
0	Для чисельних форм заповнює поле нулями замість пробілів.										
ширина	Мінімальна ширина поля.										

Модифікатор	Значення
.точність	Точність. Для %e, %E, %f, %F задається кількість цифр, які будуть виведені справа від десяткового числа. Для %g, %G задається максимальне число значущих цифр. Для %s задається максимальне число символів, які будуть надруковані. Для цілих чисел задається мінімальне число цифр.
h	Використовується для кодування значень short int та unsigned short int.
l	Використовується для кодування значень long int та unsigned long int.
L	Використовується для кодування значень long double.
тип	Символи перетворення : d – десяткове ціле число; i – десяткове ціле число зі знаком; u – десяткове ціле число без знаку; o – беззнакова вісімкова форма (без лідируючого нуля); x – беззнакова шістнадцяткова форма (без лідируючого 0x); c – окремий символ; s – рядок символів; e – число з рухомою крапкою, експоненціальне представлення; f – число з рухомою крапкою, представлення з поставленою крапкою; g – використовується формат %e або %f , %e використовується, якщо показник експоненти менший ніж – 4 чи більше рівний заданій точності. p – вказівник.

У загальному вигляді функція `scanf()` виглядає так:

```
int scanf( const char* format[, argument]... );
scanf( формат, &змінна );
```

де: `format` – рядок специфікаторів формату у подвійних лапках, `argument` – це ім'я змінної, значення якої вводиться. Перед іменем змінної треба ставити знак `&`, що означає операцію отримання адреси змінної у пам'яті.

Специфікатор формату для `scanf()`:

`%[*] [ширина] [{h | l | L}]тип`

Таблиця 2.5 – Специфікатори формату для функції `scanf()`

Модифікатор	Значення
*	Ігнорує наступний ввід.
ширина	Максимальна ширина поля.
h	Для %hd , %hi значення будуть збережені за допомогою типу <code>short int</code> . Для %ho , %hx , % hu значення будуть збережені за допомогою типу <code>unsigned short int</code> .
l	Для %ld , %li значення будуть збережені за допомогою типу <code>long</code> . Для %lo , %lx , %lu значення будуть збережені за допомогою типу <code>unsigned long</code> . Для %le , %lf , %lg значення будуть збережені за допомогою типу <code>double</code> .
L	Значення будуть збережені за допомогою типу <code>long double</code> .
тип	Символи перетворення : d – інтерпретує ввід як десяткове ціле число зі знаком; i – інтерпретує ввід як десяткове ціле число зі знаком; u – інтерпретує ввід як десяткове ціле число без знаку; o – інтерпретує ввід як вісімкове ціле число зі знаком; x – інтерпретує ввід як шістнадцяткове ціле число зі знаком; c – інтерпретує ввід як окремих символ; s – інтерпретує ввід як рядок символів; ввід починається з першого не службового символу і включає усі символи до наступного службового. e, f, g – інтерпретує ввід як число з рухомою крапкою (<code>float</code>); p – інтерпретує ввід як вказівник (адресу).

Порядок виконання лабораторної роботи:

- Написати програму згідно індивідуального завдання, яка використовує вирази і операції та стандартні функції вводу-виводу мови C.
- Задаючи різні вхідні дані при виконанні програми та встановлюючи у програмі різні значення у специфікаціях формату функцій виводу, дослідити їх вплив на форму виводу даних.

Контрольні запитання:

1. З яких елементів складаються вирази в мові програмування C?
2. Яке призначення функції `printf()` в мові програмування C?
3. Для чого служить форматуючий рядок у функції `printf()`?
4. Що таке специфікатор формату?
5. Яке має бути співвідношення між кількістю специфікаторів формату та кількістю параметрів функції `printf()`?
6. Які типи можуть задаватися специфікатором формату?
7. Яким чином задається ширина поля виводу у специфікаторі формату?
8. Вивід якого типу даних задає специфікатор формату `%d`?
9. Вивід якого типу даних задає специфікатор формату `%f`?
10. Вивід якого типу даних задає специфікатор формату `%s`?
11. Яким чином будуть виведені дані цілого типу, якщо специфікатор формату буде мати значення `%012d`?
12. Яким чином будуть виведені дані типу `double`, якщо специфікатор формату буде мати значення `%12.2f`?
13. Яким чином будуть виведені дані типу `double`, якщо специфікатор формату буде мати значення `%12.0f`?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальне завдання.
- Текст програми та результати її роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальні завдання:

№	Завдання
1	Написати програму, яка для заданого цілого числа a друкує наступну таблицю: $\begin{array}{ccc} a & & \\ a^2 & a^3 & \\ a^3 & a^2 & a \end{array}$
2	Написати програму, яка знаходить добуток цифр заданого цілого тризначного числа.
3	Написати програму, яка визначає число, отримане шляхом запису в зворотному порядку цифр заданого цілого тризначного числа.
4	Написати програму, яка обчислює периметр і площу правильного n -кутника, вписаного в коло заданого радіусу.
5	Написати програму, яка обчислює дробову частину середнього геометричного трьох заданих додатних чисел.
6	Написати програму, яка по заданих довжинах двох сторін деякого трикутника і куту (в градусах) між ними знаходить його площу та довжину третьої сторони.

№	Завдання						
7	Написати програму, яка по заданих координатах трьох вершин деякого трикутника знаходить його площу та периметр.						
8	Написати програму, яка визначає площу кільця, внутрішній радіус якого рівний r_1 , а зовнішній r_2 ($r_1 < r_2$).						
9	Написати програму, яка визначає довжину сторін трикутника, який заданий величиною своїх кутів (в градусах) і радіусом описаного кола.						
10	Написати програму, яка визначає площу рівнобічної трапеції з основами a і b і кутом α при більшій основі a .						
11	Написати програму, яка обчислює відстань від точки $(0, 0)$ до заданої точки (x, y) і до заданої прямої $ax + by + c = 0$.						
12	Дано натуральне тризначне число, у якому всі цифри різні. Написати програму, яка виводить всі числа, утворені при перестановці цифр заданого числа.						
13	Написати програму, яка в заданому натуральному тризначному числі міняє місцями першу і останню цифри.						
14	<p>Написати програму, яка для заданих цілих чисел a і b друкує наступну таблицю:</p> <table> <tr> <td>a</td><td>b</td></tr> <tr> <td>a^2</td><td>b^2</td></tr> <tr> <td>a^3</td><td>b^3</td></tr> </table>	a	b	a^2	b^2	a^3	b^3
a	b						
a^2	b^2						
a^3	b^3						
15	Написати програму, яка знаходить суму цифр заданого цілого тризначного числа.						
16	Написати програму, яка обчислює цілу частину середнього геометричного трьох заданих додатних чисел.						
17	В квадрат заданої площі вписано коло. Написати програму, яка знаходить площу квадрата, вписаного в це коло і у скільки разів площа вписаного квадрата менше площі заданого.						

№	Завдання
18	Трикутник ABC заданий довжинами своїх сторін a , b , c . Написати програму, яка визначає усі його медіани.
19	Трикутник ABC заданий довжинами своїх сторін a , b , c . Написати програму, яка визначає усі його бісектриси.
20	Чотирикутник заданий координатами своїх вершин. Написати програму, яка визначає його периметр.
21	Написати програму, яка знаходить скільки секунд пройшло з початку доби, якщо значення часу задано трьома натуральними числами: h – години, m – хвилини, s – секунди.
22	Трикутник ABC заданий довжинами своїх сторін a , b , c . Написати програму, яка визначає усі його висоти.
23	Написати програму, яка для заданого дійсного числа a друкує наступну таблицю (дійсне число виводити з точністю три знаки після коми): <div style="text-align: center; margin-top: 10px;"> a a^2 a^3 a^3 a^2 a </div>
24	Чотирикутник заданий координатами своїх вершин. Написати програму, яка визначає його площу.
25	Написати програму, яка обчислює і друкує коефіцієнти приведенного квадратного рівняння, коренями якого є два заданих дійсних числа.
26	Написати програму, яка знаходить об'єм і площу поверхні циліндра, якщо задані його радіус основи та висота.
27	Написати програму, яка знаходить об'єм конуса і площу поверхні, якщо задані його радіус основи та висота.
28	Написати програму, яка знаходить об'єм тора і площу поверхні з внутрішнім радіусом r і зовнішнім радіусом R .

№	Завдання
29	Написати програму, яка виводить значення часу у вигляді трьох натуральних чисел: h – години, m – хвилини, s – секунди, якщо задано кількість секунд, що пройшли з початку доби.
30	Написати програму, яка для заданих дійсних чисел a і b друкує наступну таблицю (дійсне число виводити з точністю чотири знаки після коми): <div style="margin-left: 40px;"> a b a^2 b^2 a^3 b^3 </div>

Приклад виконання

Лістинг 2.1: Ввід трьох дійсних чисел, обчислення середнього арифметичного цих чисел та вивід результату

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    double res, x1, x2, x3;
    printf("Enter the first number:");
    scanf("%lf", &x1);
    printf("Enter the second number:");
    scanf("%lf", &x2);
    printf("Enter the third number:");
    scanf("%lf", &x3);
    res = (x1 + x2 + x3) / 3;
    printf("Result is %10.2f\n", res);
    return 0;
}
```

Результати роботи програми:

```
Enter the first number:-5.5
Enter the second number:5.5
Enter the third number:12.783
Result is          4.26
```

ЛАБОРАТОРНА РОБОТА №3 РОЗВ'ЯЗУВАННЯ НА МОВІ С ЗАДАЧ З ВИКОРИСТАННЯМ УМОВНИХ ОПЕРАТОРІВ ТА ОПЕРАТОРІВ ЦИКЛУ

Мета роботи: познайомитися з умовними операторами та операторами циклу мови програмування С.

Теоретичний вступ

Довідкова система інтегрованого середовища Microsoft Visual Studio 2019.

Довідкову систему інтегрованого середовища Microsoft Visual Studio 2019 можна запустити через меню *Help* середовища:

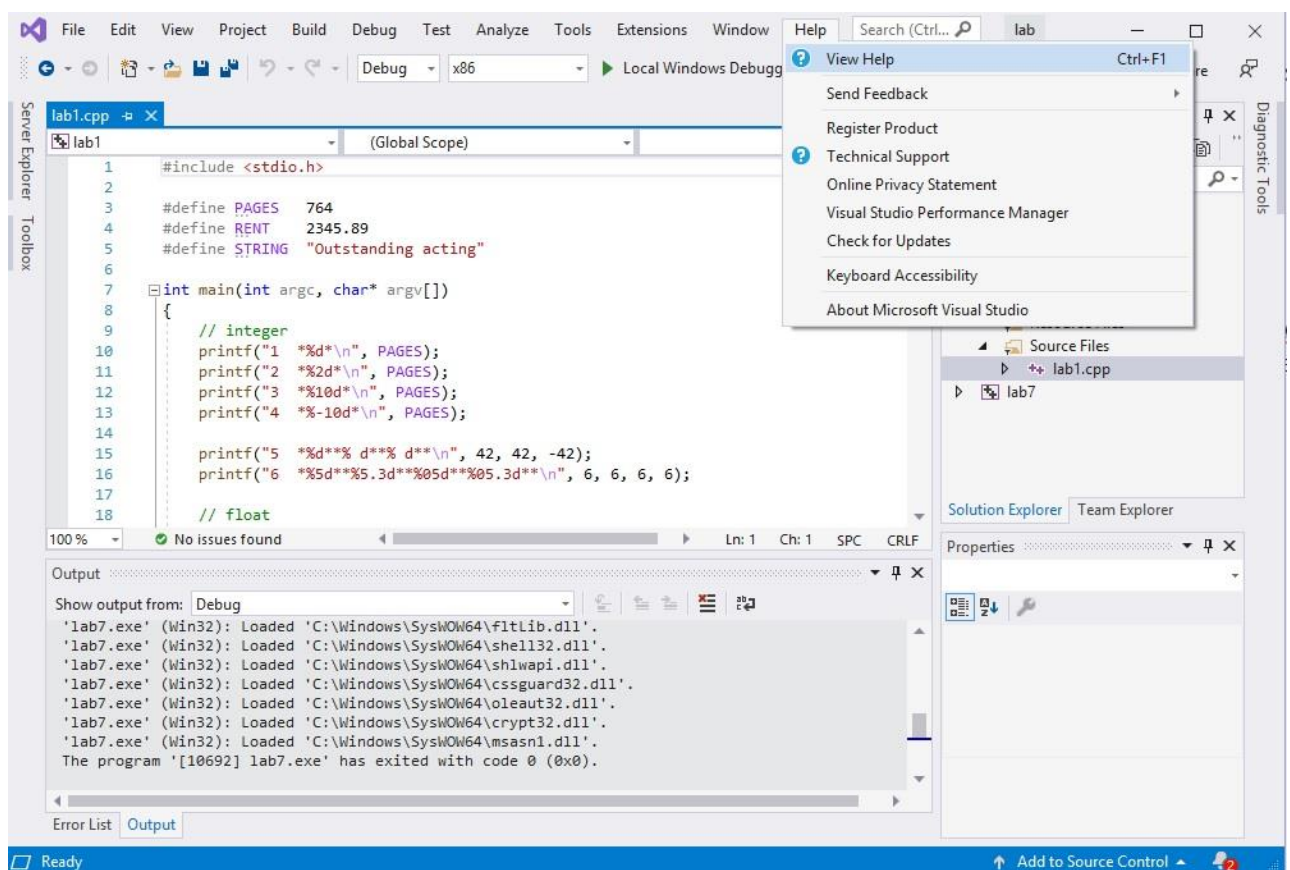


Рисунок 3.1 – Довідкова система інтегрованого середовища Visual Studio 2019

Після натискання кнопки *View Help* в браузері відкривається онлайн довідкова система інтегрованого середовища Microsoft Visual Studio 2019.

Вибравши меню *Languages* → *C++* отримуємо документацію по C++ і С.

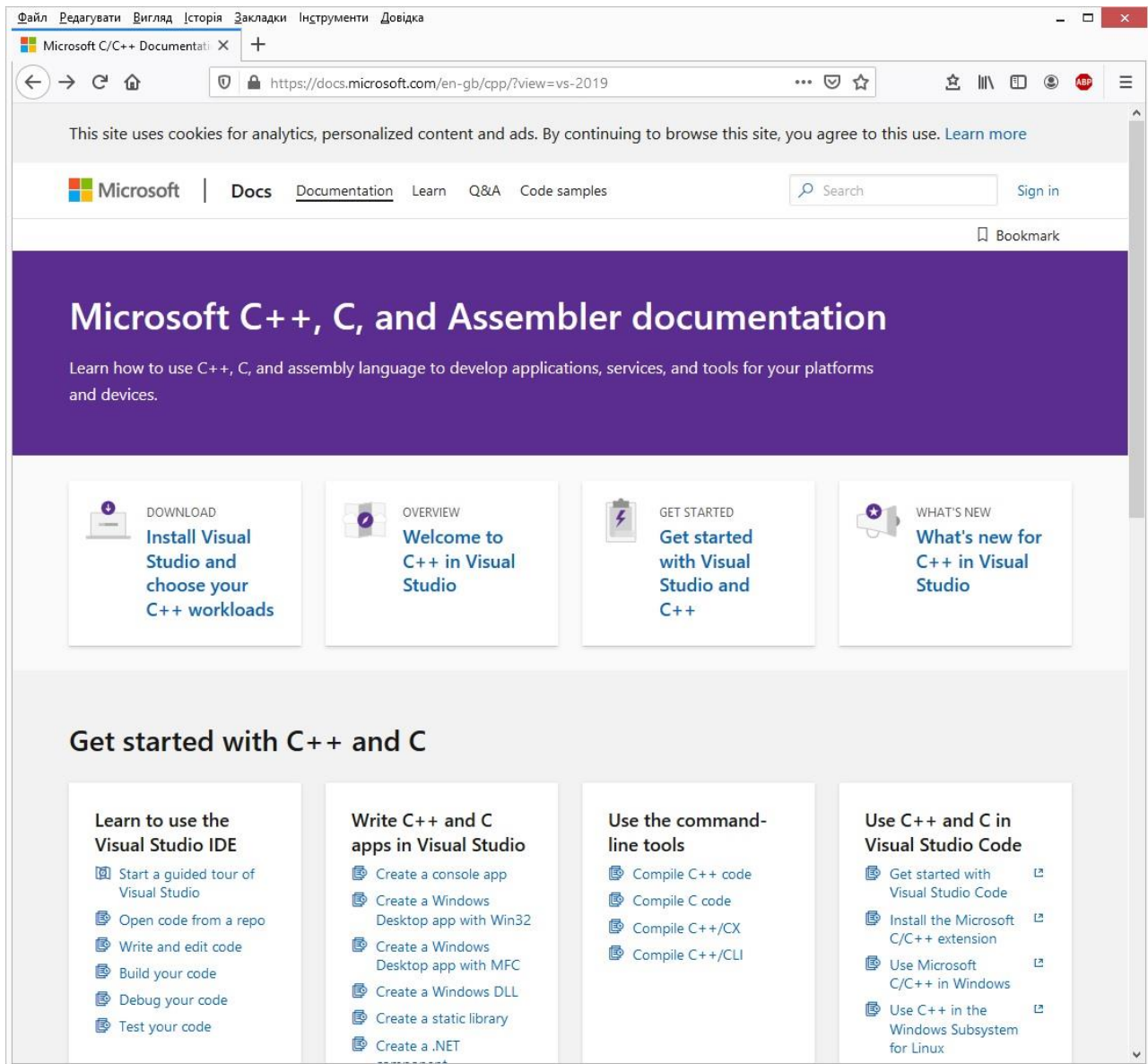


Рисунок 3.2 – довідкова система інтегрованого середовища Visual Studio 2019

Відкрити цю сторінку можна і без вибору меню *Help*, достатньо в браузері перейти за таким посиланням:

<https://docs.microsoft.com/en-gb/cpp/?view=vs-2019>

Документація, яка стосується мови програмування C++ знаходиться за таким посиланням:

<https://docs.microsoft.com/en-gb/cpp/cpp/?view=vs-2019>

Умовний оператор мови C.

Умовний оператор мови C `if` має дві форми: скорочену та повну.

Скорочена форма має вигляд:

`if (умова) оператор;`

Якщо умова є правдива (істинна, ненульове значення, значення *true*), то виконується *оператор* чи група операторів в операторних дужках {}, інакше відбудеться перехід на наступний оператор.



Рисунок 3.3 – Блок-схема скороченої форми умовного оператора `if`

Повна форма цього оператора:

`if (умова) оператор1;
else оператор2;`

Якщо умова є правдива (істинна, ненульове значення, значення *true*), то виконується *оператор1*, інакше – виконується *оператор2*, після чого відбудеться перехід на наступний оператор. Зауважимо, що виконується лише один з операторів, а не обидва.

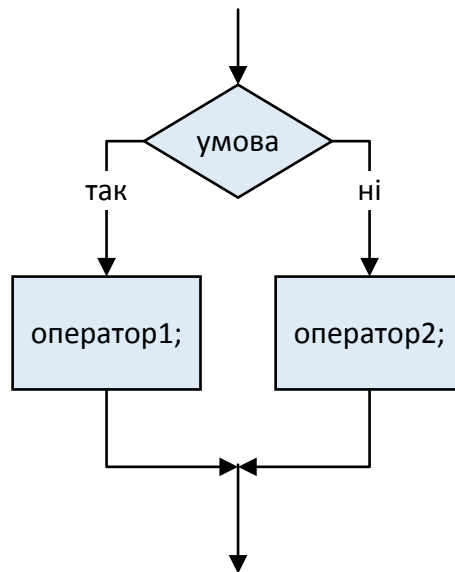


Рисунок 3.4 – Блок-схема повної форми умовного оператора `if`

Оператори циклу в мові С.

В мові С існують три різновиди операторів циклу:

- оператор циклу `for`;
- оператор циклу з передумовою `while`;
- оператор циклу з післяумовою `do-while`.

Синтаксис циклу з передумовою `while`:

```
while (умова)
{
    тіло циклу;
};
```

*Послідовність операторів (тіло циклу) виконується, поки умова є правдива (істинна, *true*, має ненульове значення), а вихід з циклу здійснюється, коли умова стане хибною (*false*, матиме нульове значення). Якщо умова є хибною при входженні до циклу, то послідовність операторів не виконуватиметься жодного разу, а керування передаватиметься до наступного оператора програми.*

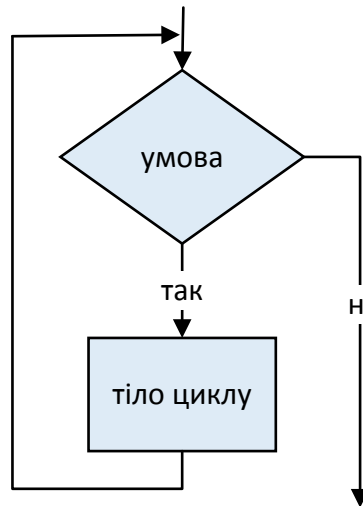


Рисунок 3.5 – Блок-схема циклу `while`

Цикл з післяумовою `do-while`:

```
do
{
    тіло циклу;
}
while (умова);
```

*Послідовність операторів (тіло циклу) виконується один чи кілька разів, поки умова не стане хибною (*false* чи дорівнюватиме нулю). Якщо умова є правдива, то оператори тіла циклу виконуються повторно. Оператор циклу `do-while` використовується в тих випадках, коли є потреба виконати тіло циклу хоча б одноразово, оскільки перевірка умови здійснюється після виконання операторів. Якщо тіло циклу складається з одного оператора, то операторні дужки `{}` не є обов'язкові.*

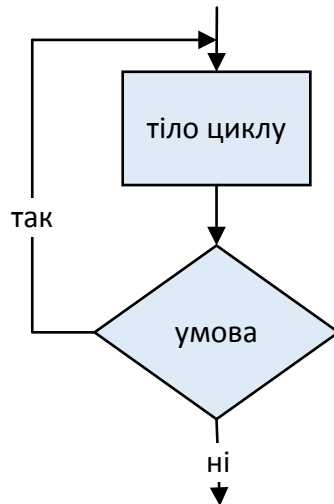


Рисунок 3.6 – Блок-схема циклу do-while

Оператор циклу з параметром **for** зазвичай використовується, коли є заздалегідь відома кількість повторювань, або коли умова продовження виконання циклу записується коротким виразом. За приклади використання даного оператора можуть слугувати обчислення сум заданої кількості доданків, пошук мінімального (максимального) елемента послідовності чисел, сортування елементів масиву за збільшенням (за зменшенням) тощо.

Синтаксис оператора:

```
for (ініціалізація; умова; модифікації)  
    { тіло циклу; }
```

Цей оператор складається з трьох основних блоків, розміщених у круглих дужках і відокремлених один від одного крапкою з комою (;), та операторів (тіла циклу), які мають багаторазово виконуватись у цьому циклі. На початку виконання одноразово у блоці *ініціалізації* задаються початкові значення змінних (параметрів), які керують циклом. Потім перевіряється *умова* і, якщо вона правдива (*true* чи має ненульове значення), то виконується *тіло циклу* (чи група операторів в операторних дужках {}). *Модифікації* змінюють параметри циклу і, в разі правдивості умови, виконання циклу триває. Якщо *умова* хибна (*false* чи дорівнює нулю), відбувається вихід із циклу і керування передається на оператор, який слідує за оператором **for**. Суттєвим є те, що перевірка умови виконується на початку циклу. Це означає, що тіло циклу може не виконатись жодного разу, якщо *умова* спочатку є хибна. Кожне повторення (крок) циклу називається *ітерацією*.

В операторі можливі конструкції, коли є відсутній той чи інший блок: *ініціалізація* може бути відсутня, якщо початкове значення задати попередньо; *умова* – якщо припускається, що умова є завжди правдива, тобто слід неодмінно виконувати тіло циклу, поки не зустрінеться оператор **break**; а *модифікації* – якщо зміна параметра циклу здійснюється в тілі циклу чи взагалі це є непотрібним. Тоді сам вираз блоку пропускається, але крапка з комою (;) неодмінно має залишитись. Можливою є наявність *порожнього* оператора (оператор є відсутній) у тілі циклу.

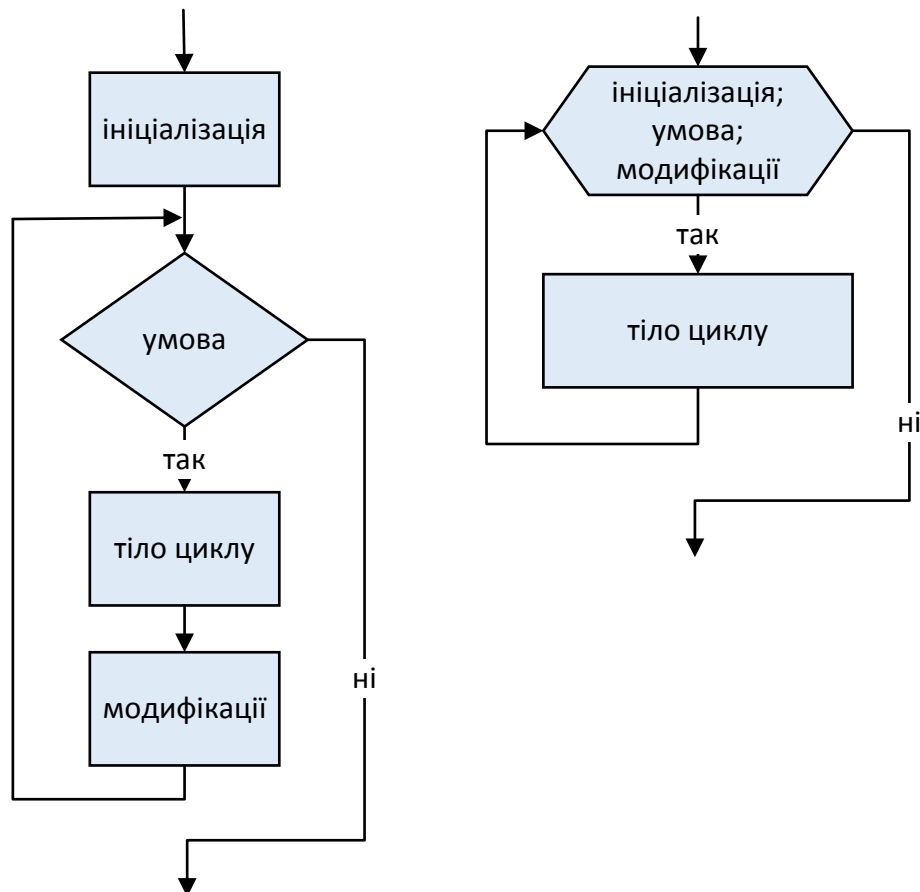


Рисунок 3.7 – Блок-схема циклу **for**

Оператори циклу можуть завчасно завершитись при виконуванні операторів **break**, **goto**, **return** (вихід з поточної функції) усередині тіла циклу.

З допомогою оператора **continue** можна достроково завершувати чергову ітерацію, оминаючи решта операторів циклу.

Порядок виконання лабораторної роботи:

- Познайомитися з умовними операторами та операторами циклу мови C.
- Нарисувати блок-схеми алгоритмів вирішення задач згідно індивідуальних завдань.
- Написати програми згідно індивідуальних завдань, які використовують умовні оператори та оператори циклу, відлагодити їх та отримати результати роботи програм.

Контрольні запитання:

- 1) Як виглядає синтаксис умовного оператора мови програмування C?
- 2) Які дві форми умовного оператора існують у мові програмування C?
- 3) Як виконується умовний оператор мови програмування C?
- 4) Як виглядає синтаксис оператора циклу **while** мови програмування C?
- 5) Як виконується оператор циклу **while** мови програмування C?
- 6) Як виглядає синтаксис оператора циклу **do-while** мови програмування C?
- 7) Як виконується оператор циклу **do-while** мови програмування C?
- 8) Як виглядає синтаксис оператора циклу **for** мови програмування C?
- 9) Як виконується оператор циклу **for** мови програмування C?
- 10) Для чого використовуються в тілі операторів циклу оператори **break**, **return** чи **goto**?
- 11) Для чого використовується в тілі операторів циклу оператор **continue**?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальні завдання.
- Блок-схеми алгоритмів вирішення задач.
- Тексти програм та результати їх роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальне завдання:

Індивідуальне завдання до даної лабораторної роботи складається з **трьох окремих завдань**, до кожного з них необхідно нарисувати блок-схему алгоритму вирішення задачі і написати програму на мові програмування C, яка буде вирішувати поставлене завдання.

Індивідуальне завдання №1 (розгалужений алгоритм).

№	Завдання 1
1	Задано два дійсних числа. Визначити, що більше, сума квадратів або квадрат суми цих чисел.
2	Дві точки задані своїми координатами: $A(x_0, y_0)$ і $B(x_1, y_1)$. Визначити, яка з точок А чи В, найбільш віддалена від початку координат.
3	Задані дійсні додатні числа a, b, c , які є сторонами трикутника. Визначити чи він прямокутний.
4	Задано квадрат з стороною a і коло з радіусом R . Визначити площа якої з фігур буде більшою.
5	Визначити, чи є число a дільником для числа b і навпаки.

№	Завдання 1
6	Дано двозначне число і цифра a . Визначити чи входить в це число цифра a .
7	Дано двозначне число. Перевірити чи однакові в нього цифри.
8	Написати програму, яка обчислює найменше з трьох дійсних чисел x, y, z .
9	Точка площини задана декартовими координатами (x, y) . Перевірити, чи належить вона другому координатному квадранту.
10	Точка на площині задана декартовими координатами (x, y) . Перевірити, чи належить вона колу з радіусом R і центром в початку координат.
11	Задані дійсні додатні числа a, b, c , які є сторонами трикутника. Визначити чи він рівностороннім
12	Дано двозначне число. Перевірити чи різні в нього цифри.
13	Точка площини задана декартовими координатами (x, y) . Перевірити, чи належить вона четвертому координатному квадранту.
14	Точка площини задана декартовими координатами (x, y) . Перевірити, чи належить вона третьому координатному квадранту.
15	Задані дійсні числа a, b, c . Перевірити, чи виконується нерівність $a < b > c$.
16	Задано квадрат з стороною a і прямокутник з сторонами b і c . Визначити периметр якої з фігур буде більшим.
17	Обчислити більше з чисел $(x+y*z), (x*y+z)$. Де x, y, z задані дійсні числа.
18	Задані дійсні числа a, b, c . Перевірити, чи вони усі різні.
19	Точка простору задана декартовими координатами (x, y, z) . Перевірити, чи належить вона кулі з радіусом R і центром у початку координат.
20	Створити програму, яка перевіряє, чи належить початок координат трикутнику з вершинами $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3)$.

№	Завдання 1
21	Точка площини задана декартовими координатами (x, y) . Перевірити, чи належить вона кільцю, з центром у початку координат, внутрішнім радіусом 1 і зовнішнім 2.
22	Обчислити більше з чисел $(x+y+z)$, $(x*y*z)$. Де x, y, z задані дійсні числа.
23	Точка площини задана декартовими координатами (x, y) . Перевірити, чи належить вона першому координатному квадранту.
24	Задані дійсні числа a, b, c . Перевірити, чи виконується нерівність $a=b=c$.
25	Задані дійсні числа a, b, c, d . Вияснити, чи можна прямокутник зі сторонами a, b помістити всередині прямокутника зі сторонами c, d таким чином, щоб кожна зі сторін одного прямокутника була паралельна або перпендикулярна кожній стороні іншого прямокутника.
26	Задані дійсні додатні числа a, b, c . Визначити, чи існує трикутник з довжинами сторін a, b, c .
27	Задано дійсне число. Піднести його до квадрату, якщо воно невід'ємне.
28	Задано дійсне число. Піднести його до квадрату, якщо воно належить інтервалу $(1, 3)$.
29	Задані дійсні числа a, b, c . Перевірити, чи виконується нерівність $a < b < c$.
30	Написати програму, яка обчислює найбільше з трьох дійсних чисел x, y, z .

Індивідуальне завдання №2 (цикли з відомим числом ітерацій)

№	Завдання 2
1	Задане натуральне число n . Обчислити: $\left(1 + \frac{1}{1^2}\right) \left(1 + \frac{1}{2^2}\right) \dots \left(1 + \frac{1}{n^2}\right)$
2	Задане натуральне число n . Обчислити:

№	Завдання 2
	$\frac{1}{\sin 1} + \frac{1}{\sin 1 + \sin 2} + \dots + \frac{1}{\sin 1 + \sin 2 + \dots + \sin n}$
3	<p>Задане натуральне число n. Обчислити:</p> $\sqrt{2 + \sqrt{2 + \dots + \sqrt{2}}}$ <p>n коренів</p>
4	<p>Задане натуральне число n. Обчислити:</p> $\frac{\cos 1}{\sin 1} \times \frac{\cos 1 + \cos 2}{\sin 1 + \sin 2} \times \dots \times \frac{\cos 1 + \cos 2 + \dots + \cos n}{\sin 1 + \sin 2 + \dots + \sin n}$
5	<p>Задане натуральне число n. Обчислити:</p> $\sqrt{3 + \sqrt{6 + \dots + \sqrt{3(n-1) + \sqrt{3n}}}}$
6	<p>Задані дійсне число a, натуральне число n. Обчислити:</p> $\frac{1}{a} + \frac{1}{a(a+1)} + \dots + \frac{1}{a(a+1)\dots(a+n)}$
7	<p>Задані дійсне число a, натуральне число n. Обчислити:</p> $\frac{1}{a} + \frac{1}{a^2} + \frac{1}{a^4} + \dots + \frac{1}{a^{2^n}}$
8	<p>Задані дійсне число a, натуральне число n. Обчислити:</p> $a(a-n)(a-2n)\dots(a-n^2)$
9	<p>Задане дійсне число x. Обчислити:</p> $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \frac{x^{13}}{13!}$
10	<p>Задані дійсні числа x, a, натуральне число n. Обчислити:</p> $\left(\left(\dots \left((x+a)^2 + a \right)^2 + \dots + a \right)^2 \right)^2 + a$

№	Завдання 2
	n дужок
11	Задане дійсне число x . Обчислити: $x^{10} + 2x^9 + 3x^8 + \dots + 10x + 11$
12	Задане дійсне число x . Обчислити: $11x^{10} + 10x^9 + 9x^8 + \dots + 2x + 1$
13	Задане дійсне число x . Обчислити: $\sin x + \sin \sin x + \dots + \sin \sin \dots \sin x$ <p style="text-align: center;">n синусів</p>
14	Обчислити: $(1 + \sin 0.1)(1 + \sin 0.2) \dots (1 + \sin 10)$
15	Задане дійсне число x . Обчислити: $\frac{(x-2)(x-4)(x-8) \dots (x-64)}{(x-1)(x-3)(x-7) \dots (x-63)}$
16	Задане натуральне число n . Обчислити: $\cos \pi + \cos \frac{\pi}{2} + \cos \frac{\pi}{4} + \dots + \cos \frac{\pi}{2^n}$
17	Задане натуральне число n і дійсне число x . Обчислити: $1 + (x-1) + (x-1)^2 + \dots + (x-1)^n$
18	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Знайти кількість додатних чисел у цій послідовності. (Додаткова вимога: масивів не використовувати).
19	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Знайти суму тих додатних чисел цієї послідовності, які розміщені після першого від'ємного числа. (Додаткова вимога: масивів не використовувати).

№	Завдання 2
20	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Знайти суму додатних чисел цієї послідовності. (Додаткова вимога: масивів не використовувати).
21	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Знайти порядковий номер найбільшого числа цієї послідовності. (Додаткова вимога: масивів не використовувати).
22	Задано 10 дійсних чисел. Визначити чи вони утворюють зростаючу послідовність. (Додаткова вимога: масивів не використовувати).
23	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Знайти величину найменшого серед чисел цієї послідовності. (Додаткова вимога: масивів не використовувати).
24	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Знайти суму від'ємних чисел цієї послідовності, які розміщені після першого додатного. (Додаткова вимога: масивів не використовувати).
25	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Визначити скільки раз у цій послідовності змінюється знак. Наприклад, у послідовності 1, -34, 8, 14, -5 знак змінюється 3 рази. (Додаткова вимога: масивів не використовувати).
26	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Знайти кількість від'ємних чисел у цій послідовності (Додаткова вимога: масивів не використовувати).
27	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Обчислити суму, яка рівна сумі першого, третього і так далі чисел послідовності (Додаткова вимога: масивів не використовувати).
28	Задане ціле $n > 0$ та послідовність з n цілих чисел. Знайти кількість парних і непарних чисел послідовності. (Додаткова вимога: масивів не використовувати).
29	Задане ціле $n > 0$ та послідовність з n дійсних чисел. Обчислити суму, яка рівна сумі другого, четвертого і так далі чисел послідовності (Додаткова вимога: масивів не використовувати).

№	Завдання 2
30	Задано 10 дійсних чисел. Визначити чи вони утворюють спадаючу послідовність. (Додаткова вимога: масивів не використовувати).

Індивідуальне завдання №3 (цикли з невідомим числом ітерацій).

№	Завдання 3
1	Підрахувати k – кількість цифр у десятковому записі цілого невід’ємного числа n .
2	Підрахувати k – суму цифр у десятковому записі цілого невід’ємного числа n .
3	Визначити старшу цифру у десятковому записі цілого невід’ємного числа n .
4	Задані натуральні числа n та k . Значення k не більше за кількість цифр у десятковому записі числа n . Отримати суму k молодших цифр десяткового запису числа n .
5	Задане натуральне число n та натуральне число k . Значення k не більше за кількість цифр у десятковому записі числа n . Визначити k -ту цифру десяткового запису числа n .
6	Задано натуральне число n та цифра m . Визначити чи є цифра m у числі n .
7	Перевірити чи у заданого натурального числа n усі цифри однакові.
8	Для довільного цілого числа $n > 1$ знайти найбільше ціле k , для якого $4^k < n$.
9	Перевірити, чи є задане натуральне число n простим.
10	Підрахувати k – добуток цифр у десятковому записі цілого невід’ємного числа n .
11	Задані натуральні числа n та k . Значення k не більше за кількість цифр у десятковому записі числа n . Отримати суму k старших цифр десяткового запису числа n .

№	Завдання 3
12	Підрахувати k – суму цифр, що стоять на парних позиціях (молодша цифра – перша позиція) у десятковому записі натурального числа n .
13	Перевірити, чи є задане натуральне число n степенем двійки.
14	Вивести на екран квадрати кожної цифри заданого натурального числа n .
15	Знайти найбільшу цифру у заданому натуральному числі n .
16	Підрахувати k – суму цифр, що стоять на непарних позиціях (молодша цифра – перша позиція) у десятковому записі натурального числа n .
17	Задані натуральні числа n та k . Значення k не більше за кількість цифр у десятковому записі числа n . Отримати добуток k молодших цифр десяткового запису числа n .
18	<p>Дано дійсні числа $x, \varepsilon > 0$. Обчислити з точністю ε нескінченну суму та вказати кількість порахованих доданків:</p> $\sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{k! (2k+1)}$
19	<p>Дано дійсні числа $x, \varepsilon > 0$. Обчислити з точністю ε нескінченну суму та вказати кількість порахованих доданків:</p> $\sum_{k=0}^{\infty} \frac{x^{2k}}{2k!}$
20	<p>Дано дійсні числа $x, \varepsilon > 0$. Обчислити з точністю ε нескінченну суму та вказати кількість порахованих доданків:</p> $\sum_{k=0}^{\infty} \frac{(-1)^k x^k}{(k+1)^2}$
19	<p>Дано дійсні числа $x, \varepsilon > 0$. Обчислити з точністю ε нескінченну суму та вказати кількість порахованих доданків:</p> $\sum_{k=0}^{\infty} \frac{x^{2k}}{2^k k!}$

№	Завдання 3
22	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $\frac{1}{(1+x)^3} = 1 - \frac{2 \cdot 3}{2} \cdot x + \frac{3 \cdot 4}{2} \cdot x^2 - \frac{4 \cdot 5}{2} \cdot x^3 + \dots$
23	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $\frac{1}{(1+x)^2} = 1 - 2x + 3x^2 - \dots$
24	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $\frac{1}{1+x} = 1 - x + x^2 - x^3 + \dots$
25	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots$
26	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $\operatorname{sh} x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots$
27	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $\operatorname{ch} x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots$
28	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots$
29	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p>

№	Завдання 3
	$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$
30	<p>За допомогою розкладу функції у ряд Тейлора обчислити з заданою точністю $\varepsilon > 0$ її значення для заданого значення x:</p> $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

Приклади виконання

Лістинг 3.1: Обчислення квадратного кореня заданого числа

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
int main(void)
{
    double res, x;
    printf("Enter x:");
    scanf("%lf", &x);
    if (x >= 0)
    {
        res = sqrt(x);
    }
    else
    {
        res = sqrt(-x);
    }
    printf("Square root is %f\n", res);
    return 0;
}
```

Результати роботи програми:

```
Enter x:56.25
Square root is 7.500000
```

Лістинг 3.2: Обчислення $y = \sum_{i=1}^{50} \frac{1}{i^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{50^2}$

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
int main()
{
    int i;
    double y;
    i = 1;
    y = 0;
    for (i = 1, y = 0; i <= 50; i++)
        y = y + 1 / ((double)i * i);
    printf("y=%f", y);
}
```

Результати роботи програми:

y=1.625133

Лістинг 3.3: Обчислення інтеграла $\sin(x)$ на відрізку від a до b .

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
int main(void)
{
    double a, b, x, h, S1, S2, S;
    int N;
    printf("Enter a, b and N: ");
    scanf("%lf%lf%d", &a, &b, &N);
    /*-- Trapezium algorithm: --*/
    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);
    S = h * (S1 + S2);
    /*-----*/
    printf("\nFor a = %f, b = %f, N = %d, S = %f", a, b, N, S);
    return 0;
}
```

Результати роботи програми:

Enter a, b and N: 0 5 10

For a = 0.000000, b = 5.000000, N = 10, S = 0.701352

ЛАБОРАТОРНА РОБОТА №4 РОЗВ'ЯЗУВАННЯ НА МОВІ C ЗАДАЧ, В ЯКИХ ВИКОРИСТОВУЄТЬСЯ ВИЗНАЧЕННЯ І ВИКЛИК ФУНКЦІЙ

Мета роботи: познайомитися з засобами опису функцій та їх виклику в мові програмування C.

Теоретичний вступ:

Засоби відлагодження інтегрованого середовища Microsoft Visual Studio 2019.

Інтегроване середовище розробки програм Visual Studio 2019 має вбудований відлагоджувач, призначений для виявлення помилок (їх локалізації) у програмі. Процес відлагодження полягає в корекції чи модифікації програми таким чином, щоб програма виконувалася правильно і відповідала поставленій задачі. Для цього відлагоджувач має цілий ряд засобів, які спрощують відслідковування та виявлення помилок в програмі.

У Visual Studio 2019 основними можна вважати такі засоби:

- точки переривання;
- покрокове виконання програми;
- вікно спостереження за змінними.

Точка переривання - це місце в програмі, в якій виконання програми буде призупинене. В програмі можна встановити довільну кількість точок переривання.

При *покроковому виконанні* за одне натискання на відповідну клавішу покрокового виконання виконується один "крок" програми. "Кроком" вважається один рядок програми, якщо у цьому рядку розміщується один чи декілька операторів програми, або декілька рядків, якщо оператор записаний у декількох рядках.

У відлагоджувачі є два режими покрокового виконання: *Step Into* і *Step Over*. Різниця між ними полягає у способі виконання функцій. Коли виконується *Step Into* і у поточному рядку зустрічається звертання до функції, починається виконання операторів цієї функції, коли виконується *Step Over* виконання операторів функції не здійснюється, і всі її оператори виконуються за один крок.

Точку переривання можна створити наступним чином: перемістити курсор в те місце програми, де необхідно встановити точку переривання і натиснути на клавішу *F9*.

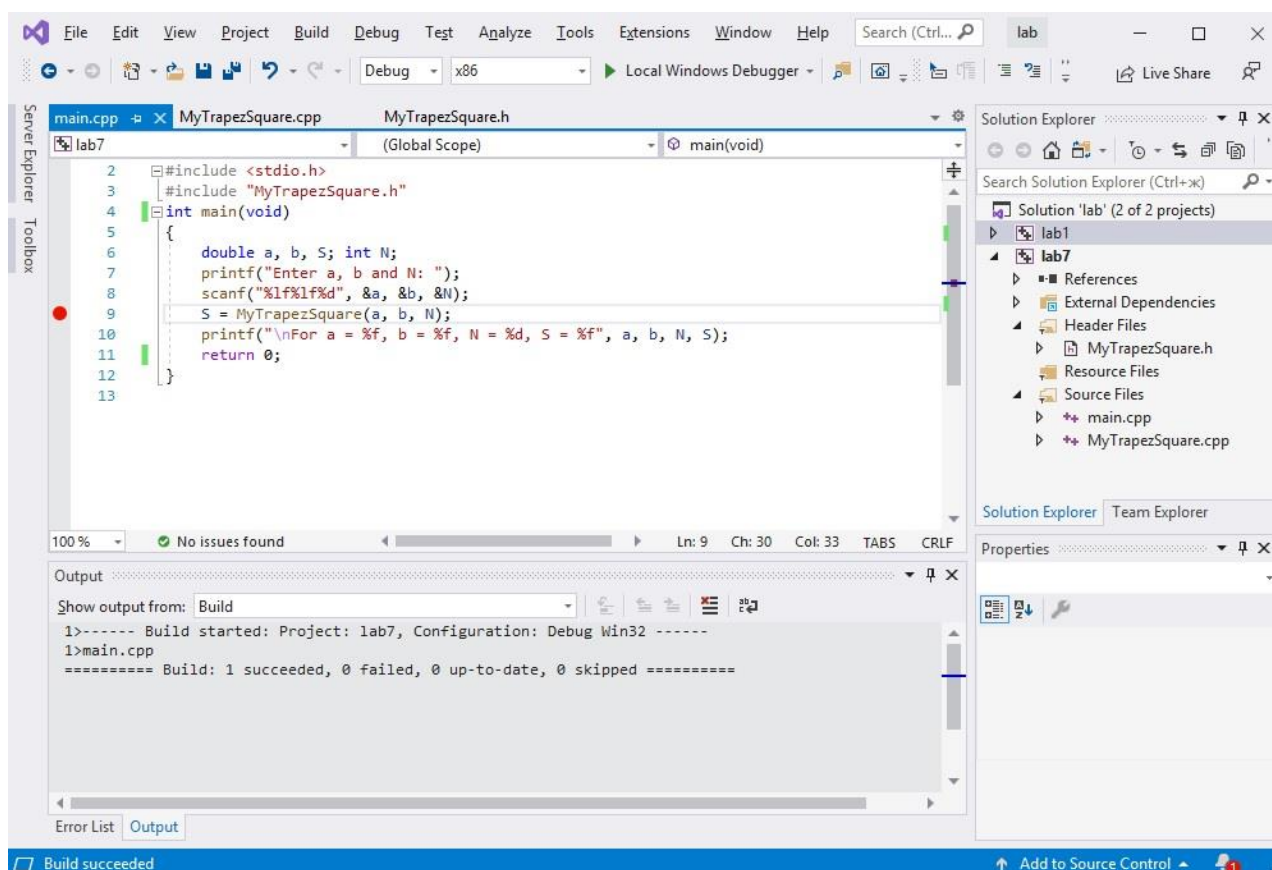


Рисунок 4.1 – Встановлення точки переривання

Місце встановлення точки переривання відзначається кружком зліва від відповідного рядка програми у вікні з текстом програми. Для того, щоб відмінити ("зняти") точку переривання, потрібно повторно натиснути на клавішу *F9*.

Встановивши точку переривання, можна запустити програму на виконання (відлагодження). Це здійснюється при натисканні на клавішу *F5*. При зупинці програми в точці переривання можна переглянути значення змінних програми та продовжити виконання програми до наступної точки поривання чи в покроковому режимі.

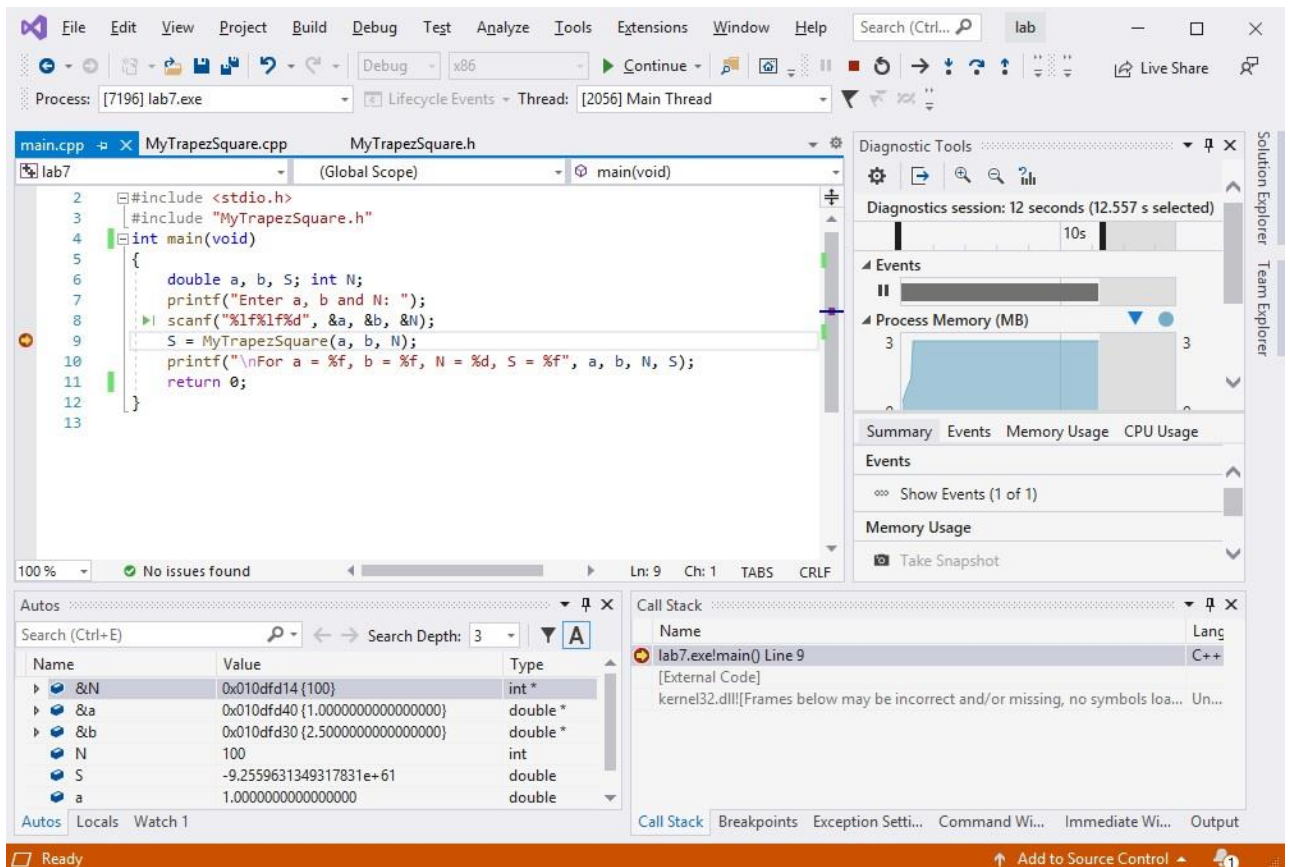


Рисунок 4.2 – Виконання програми з точкою зупинки

Для виконання програми в покроковому режимі використовуються клавішу *F10* (*Step Over*) чи *F11* (*Step Into*).

Якщо відлагодження програми потрібно взагалі припинити, то потрібно натиснути на комбінацію клавіш *Shift+F5*.

На точку програми, яка виконується в даний момент, вказує стрілка жовтого кольору зліва від відповідного рядка програми. При натисканні на клавішу покрокового виконання стрілка зміщається на наступний рядок.

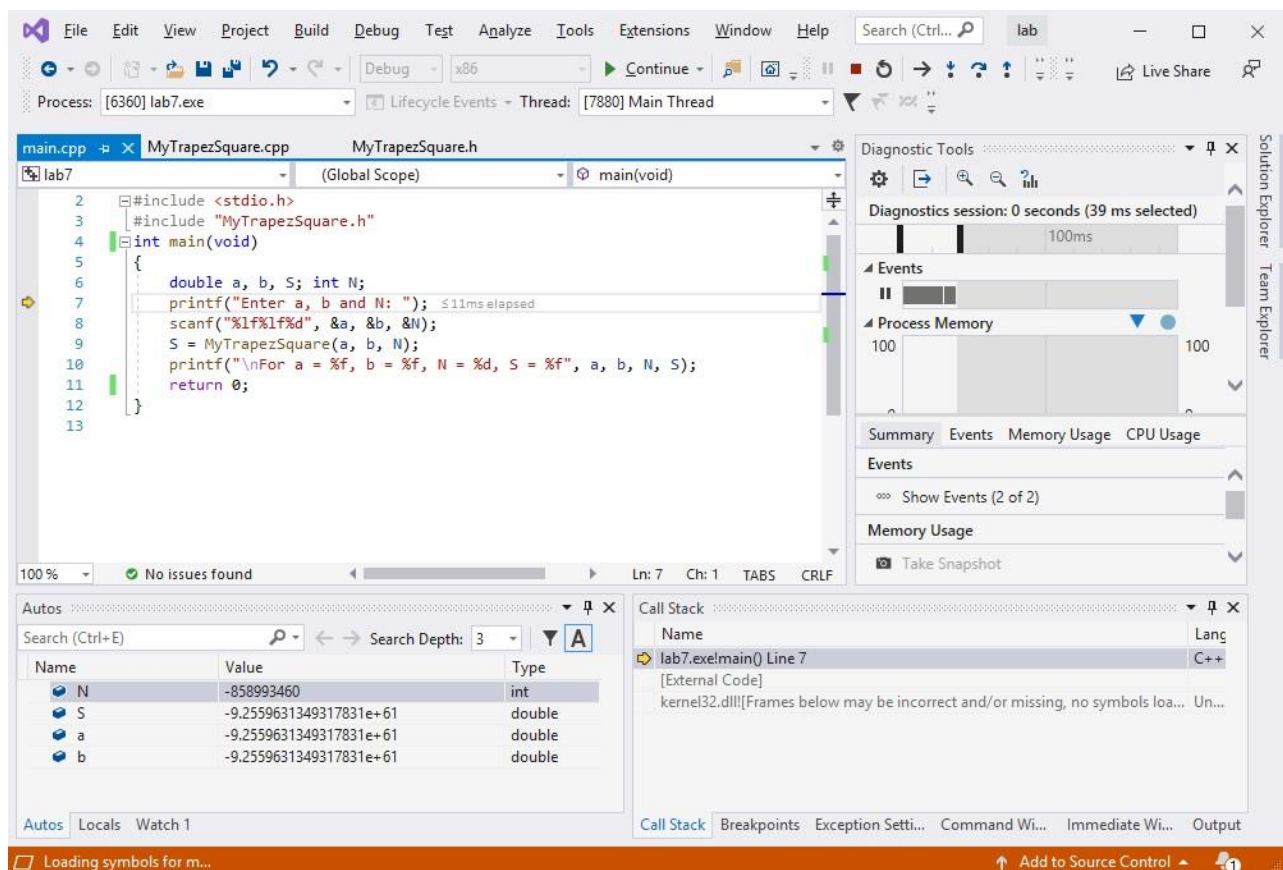


Рисунок 4.3 – Виконання програми в покроковому режимі

Основними засобами спостереження за змінними відлагоджуваної програми є вікно *Autos*, вікно *Locals* та вікно *Watch*.

У вікні *Autos* відображаються значення змінних з класом пам'яті *Auto*.

У вікні *Locals* відображаються значення локальних змінних функції, яка виконується в даний час.

У вікні *Watch* можна задати ідентифікатори змінних чи вирази, значення яких виводяться у цьому вікні під час виконання програми. Можна встановити декілька вікон *Watch* (до чотирьох).

Одне з вікон відображається у лівому нижньому куті, там можна перемкнути на те вікно, яке цікавить в певний момент.

Встановити одне з вікон спостереження активним вікном можна з меню *Debug*→*Windows*, коли інтегроване середовище знаходиться в режимі відлагодження.

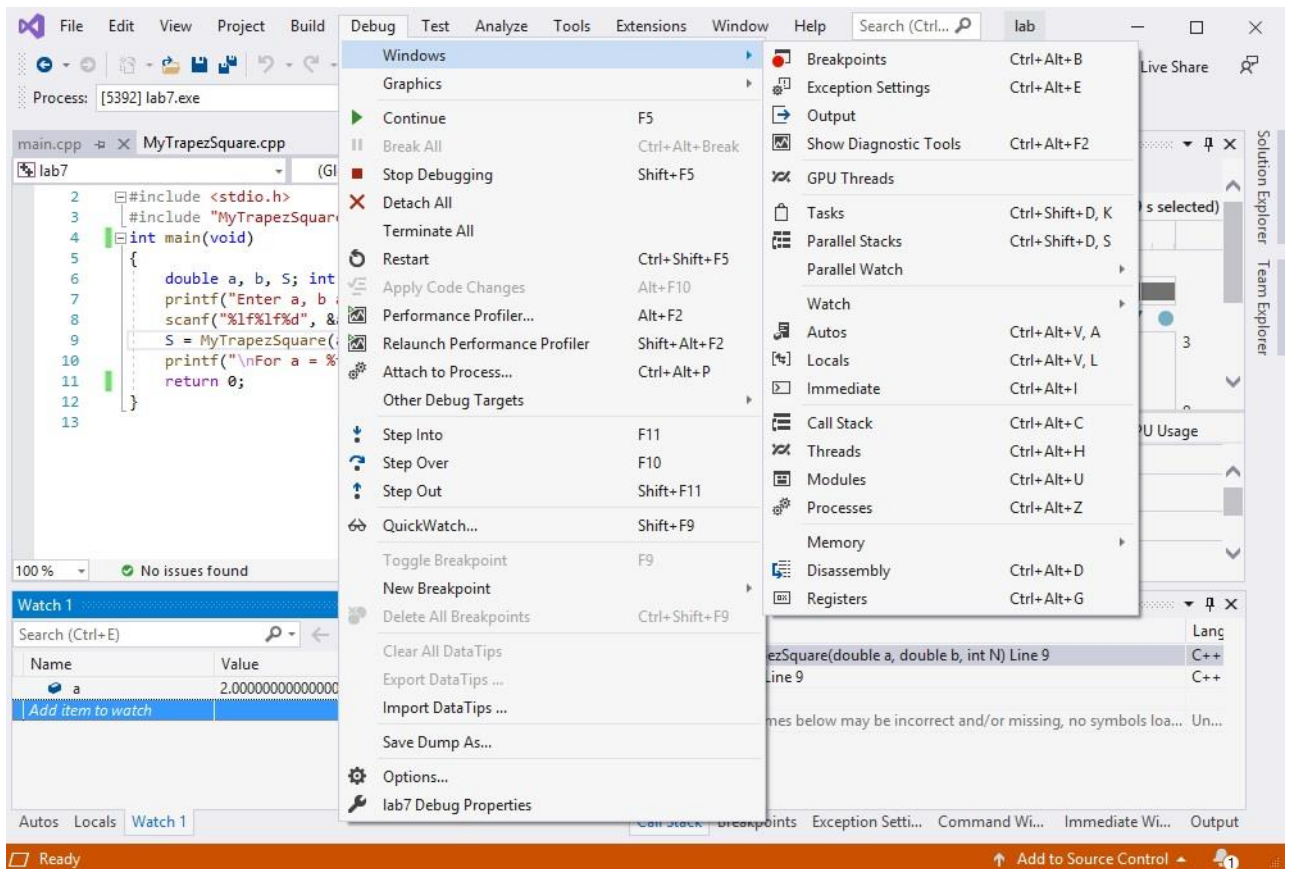


Рисунок 4.4 – Вибір активного вікна спостереження за змінними

Функції в мові програмування С.

Коли програма стає зовеликою за обсягом і складною для сприйняття, є сенс розділити її за змістом на невеликі логічні частини, називані *функціями*, кожна з яких виконуватиме певне завдання. Унаслідок цього програма стане більш легкою і для розуміння при створенні, і для процесу налагодження. Окрім того, створення функції позбавляє потреби створення двох, а то й і більшої кількості, майже однакових фрагментів програмного коду для розв'язання однакових завдань з різними вхідними даними. Розподіл програми на функції є базовим принципом структурного програмування.

Ім'я `main()` є спеціальним іменем тієї функції, з якої починається виконання програми. Інші функції є підпрограмами функції `main()`. Кожна функція описується лише один раз. В програмі може бути описано довільна кількість функцій. Кожна описана функція може викликатися стільки раз скільки необхідно.

Функція — це незалежна іменована частина програми, яка може багаторазово викликатися з інших частин програми, маніпулювати даними та повертати результати. Кожна функція має власне ім'я, за яким здійснюють її виклик.

З використанням функцій в мові С пов'язані три поняття - визначення функції (опис дій, які виконує функція), оголошення функції (задання форми звернення до функції) і виклик функції.

Визначення функції задає тип значення, що повертається, ім'я функції, типи і кількість формальних параметрів, а також оголошення змінних і оператори, звані тілом функції, що визначають дію функції. У визначенні функції також може бути заданий клас пам'яті.

У мові С немає вимоги, щоб визначення функції обов'язково передувало її виклику. Визначення функцій, які використовуються, можуть слідувати за визначенням функції `main()`, перед нею, або перебувати в іншому файлі.

Однак для того, щоб компілятор міг здійснити перевірку відповідності типів переданих фактичних параметрів типам формальних параметрів до виклику функції потрібно помістити оголошення (прототип) функції.

Оголошення функції має такий же вигляд, що і визначення функції, з тією лише різницею, що тіло функції відсутнє, і імена формальних параметрів теж можуть бути опущені.

Функція повинна бути *оголошеною і визначеною*. Оголошення функції має бути написаним у програмі раніш за її використання. Визначення може перебувати у будь-якому місці програми, за винятком тіла (середини) інших функцій. *Оголошення* функції складається з *прототипу* (чи *заголовка*) і має форму

```
тип_результату ім'я_функції(перелік формальних параметрів з  
вказанням типу);
```

Окрім оголошення, кожна функція повинна мати визначення (реалізацію). *Визначення* функції, окрім заголовку, містить тіло функції (дії, які виконує функція):

```
тип_результату ім'я_функції(перелік параметрів)  
{ тіло функції }
```

Якщо функції виконують певні обчислення й дії, які не потребують повернення конкретних результатів, то їх тип вказують як тип `void` (тобто порожній, без типу). У таких функціях оператор `return` може бути відсутнім чи записуватись без значення, яке повертається.

Інструкція для виклику функції складається з імені функції і `()`. Тут круглі дужки означають операцію виклику функції.

Приклад оформлення задачі обчислення інтегралу за формулою трапецій з використанням функції.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
double MyTrapezSquare(double a, double b, int N);
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;
}
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);
}
```

Порядок виконання лабораторної роботи:

- Познайомитися з функціями у мові C.
- Нарисувати блок-схему алгоритму вирішення задачі згідно індивідуального завдання, визначену частину реалізувати у вигляді окремої підпрограми.
- Написати програму згідно індивідуального завдання, визначену частину реалізувати у вигляді окремої функції, відлагодити її та отримати результати роботи програми.

Контрольні запитання:

- 1) Який формат опису функції в мові програмування C?
- 2) Для якої мети служить опис параметрів у заголовку функції?
- 3) Як задається тип результату, що повертає функція?
- 4) Яким оператором повертається значення, що обчислює функція?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальне завдання.
- Блок-схему алгоритму вирішення задачі.
- Текст програми та результати її роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальні завдання:

№	Завдання
1	<p>Задане дійсне число x і ціле число n. Отримати:</p> $n+x+x^n,$ <p>де x^n обчислюється згідно з формулою:</p> $x^n = \begin{cases} 1, & \text{if } n = 0 \\ 1/x^{ n }, & \text{if } n < 0. \\ x^n, & \text{if } n > 0 \end{cases}$ <p>Обчислення x^n описати з допомогою функції мови C.</p>
2	<p>Дано два натуральних числа. Знайти найбільшу цифру у цих числах, для пошуку найбільшої цифри в числі написати функцію.</p>

№	Завдання
3	<p>Задані натуральні числа a, b, c. Отримати:</p> $S(a) + S(b) + S(c),$ <p>де $S(x)$ - функція, яка обчислює суму цифр десяткового представлення натурального числа x.</p>
4	<p>Задані натуральні числа m, n. Отримати:</p> $\frac{n! \cdot m!}{(n + m)!},$ <p>обчислення факторіала описати з допомогою функції мови С.</p>
5	<p>Задані дійсні числа, які визначають відрізки a, b, c і d. Для кожної трійки тих відрізків, з яких можна побудувати трикутник, обчислити площу даного трикутника. Обчислення площі трикутника описати з допомогою функції мови С.</p>
6	<p>Задані натуральні числа a, b, c. Отримати найбільший спільний дільник (НСД) цих чисел. Для визначення НСД двох чисел використати алгоритм Евкліда і описати з допомогою функції мови С.</p>
7	<p>Задані дійсні числа a, b. Отримати:</p> $\min(u + v^2, 3.14),$ <p>де</p> $u = \min(a, b), v = \min(a \cdot b, a + b),$ <p>$\min(a, b)$ – функція, яка повертає менше з значень a і b.</p>
8	<p>Задане дійсне число y. Отримати:</p> $\frac{1.7t(0.25) + 2t(1 + y)}{6 - t(y^2 - 1)},$ <p>де</p> $t(x) = \sum_{k=0}^{10} \frac{x^k}{k!}$

№	Завдання
9	<p>Задані дійсні числа x, y. Отримати:</p> $(sign(x) + sign(y)) \cdot (\sqrt{x \cdot sign(x)} + \sqrt{y \cdot sign(y)}),$ <p>де</p> $sign(a) = \begin{cases} -1, & \text{if } a < 0 \\ 0, & \text{if } a = 0. \\ 1, & \text{if } a > 0 \end{cases}$
10	<p>Задані дійсні числа s, t. Отримати:</p> $h(s, t) + h^4(s - t, s + t) + h(1, 1),$ <p>де</p> $h(a, b) = \frac{a}{1 + b^2} + \frac{b}{1 + a^2} - (a - b)^3.$
11	<p>Задані дійсні числа a, b, c. Отримати:</p> $\frac{\max(a, a + b) + \max(a, b + c)}{1 + \max(a + b \cdot c, 1.15)},$ <p>де $\max(a, b)$ – функція, яка повертає більше з значень a і b.</p>
12	<p>Задані дійсні числа s, t. Отримати:</p> $g(1.2, s) + g(t, s) - g(2s - 1, s \cdot t),$ <p>де</p> $g(a, b) = \frac{a^2 + b^2}{a^2 + 2ab + 3b^2 + 4}.$
13	<p>Задані дійсні числа s, t. Отримати:</p> $f(t, -2s, 1.17) + f(2.2, t, s - t),$ <p>де</p> $f(a, b, c) = \frac{2a - b - \sin c}{5 + c }.$

№	Завдання
14	Перевірити чи задане шестизначне число є щасливим. Щасливим називають таке шестизначне число, у якого сума перших трьох цифр дорівнює сумі останніх трьох. Для визначення суми цифр тризначного числа написати функцію.
15	Дано два натуральних числа. Знайти в якому з них більше цифр, для визначення кількості цифр в числі написати функцію.
16	<p>Задані натуральні числа a, b, c. Отримати:</p> $S(a) + S(b) + S(c),$ <p>де</p> $S(x) = \frac{\sqrt{x} + x}{2}.$
17	Написати програму визначення периметра трикутника, заданого координатами його вершин. Для визначення довжини сторони написати функцію.
18	<p>Задані дійсні a, b. Отримати</p> $\max(a, 2b) + \max(2a - b, b)$ <p>визначивши і використавши функцію</p> $\max(x, y) - \text{максимальне з двох чисел.}$
19	Дано два натуральних числа. Знайти число, сума цифр якого буде більшою. Для визначення суми цифр в числі написати функцію.
20	Вивести на екран усі двозначні прості числа. Для перевірки чи число є простим, написати функцію.
21	<p>Задані натуральні числа a, b, c. Отримати:</p> $S(a) - S(2*b) + S(3*c),$ <p>де</p> $S(x) = \frac{\sin(x) + x}{3}.$

№	Завдання
22	Задано два тризначних цілих числа. Визначити, яке більше з чисел, у яких поміняні місцями старша і молодша цифри. Для зміни цифр у тризначному числі написати функцію.
23	Надрукувати перші десять чотиризначних щасливих номерів. Щасливим називається номер, у якого сума перших двох цифр номера дорівнює сумі останніх двох цифр. Для перевірки чи чотиризначне число є щасливим написати функцію.
24	Знайти усі медіани трикутника, заданого довжинами сторін. Для обчислення медіани написати функцію.
25	Написати власну функцію обчислення $\sin(x)$, використовуючи розклад функції в ряд Тейлора. В основній програмі порівняти роботу власної функції з бібліотечною.
26	Написати власну функцію обчислення $\cos(x)$, використовуючи розклад функції в ряд Тейлора. В основній програмі порівняти роботу власної функції з бібліотечною.
27	<p>Задані дійсні числа c і d. Обчислити</p> $\int_c^d \arctg^2(x) dx + \int_0^{\pi} \arctg(x) dx.$ <p>Інтеграл обчислювати приблизно за формулою прямокутників при $n=200$:</p> $\int_a^b f(x) dx \approx h \cdot \{f(a) + f(a+h) + f(a+2h) + \dots + f(a+(n-1) \cdot h)\},$ <p>де $h = (b-a)/n$.</p>

№	Завдання
28	<p>Задані дійсні числа c і d. Обчислити</p> $\int_c^d \arctg^2(x) dx + \int_0^{\pi} \arctg(x) dx.$ <p>Інтеграл обчислювати приблизно за формулою трапецій при $n=100$:</p> $\int_a^b f(x) dx \approx h \cdot \left\{ \frac{f(a)}{2} + f(a+h) + f(a+2h) + \dots + f(a+(n-1) \cdot h) + \frac{f(b)}{2} \right\},$ <p>де $h = (b-a)/n$.</p>
29	Дано два натуральних числа. Визначити, чи є хоча б одне з них степенем двійки. Для перевірки, чи є число степенем двійки, написати функцію.
30	Вивести на екран усі двозначні числа, які є повними квадратами. Для перевірки, чи число є повним квадратом, написати функцію.

Приклад виконання

Лістинг 4.1: Обчислення інтеграла $\sin(x)$ на відрізку від a до b з використанням функції.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>
double MyTrapezSquare(double a, double b, int N);
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;
}
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);
}
```

Результати роботи програми:

Enter a, b and N: 0 5 10

For a = 0.000000, b = 5.000000, N = 10, S = 0.701352

ЛАБОРАТОРНА РОБОТА №5 РОЗВ'ЯЗУВАННЯ НА МОВІ C ЗАДАЧ, В ЯКИХ ВИКОРИСТОВУЮТЬСЯ ЧИСЛОВІ МАСИВИ

Мета роботи: познайомитися з використанням масивів в мові програмування C.

Теоретичний вступ:

Масив у програмуванні – це впорядкована сукупність однотипних елементів. Масиви широко застосовуються для зберігання і опрацювання однорідної інформації, приміром таблиць, векторів, матриць, коефіцієнтів рівнянь тощо.

Кожен елемент масиву однозначно можна визначити за ім'ям масиву та індексами. *Ім'я масиву* (ідентифікатор) вибирають за тими самими правилами, що й для змінних. *Індекси* визначають місцезнаходження елемента в масиві. Приміром, елементи вектору мають один індекс – номер по порядку; елементи матриць чи таблиць мають по два індекси: перший означає номер рядка, другий – номер стовпчика. Кількість індексів визначає вимірність масиву. Приміром, *вектори* у програмах – це одновимірні масиви, *матриці* – двовимірні.

Індексами можуть бути лише змінні, константи чи вирази цілого типу. Значення індексів записують після імені масиву в квадратних дужках. При оголошенні масивів у квадратних дужках зазначається кількість елементів, а нумерація елементів завжди розпочинається з нуля.

Відмінності масиву від звичайних змінних:

- спільне ім'я для всіх значень;
- доступ до конкретного значення за його номером (індексом);
- можливість опрацювання у циклі.

Одновимірний масив (вектор) оголошується у програмі в такий спосіб:

`тип_даних ім'я_масиву [розмір_масиву];`

Тип_даних задає тип елементів масиву. Елементами масиву не можуть бути функції й елементи типу `void`. *Розмір_масиву* у квадратних дужках задає кількість елементів масиву. На відміну від інших мов, у C не перевіряється вихід за межі масиву, тому, щоб уникнути помилок у програмі, слід стежити за розмірністю оголошених масивів. Значення *розмір_масиву* при оголошенні масиву може бути не вказано в таких випадках:

- при оголошенні масив ініціалізується;
- масив оголошено як формальний параметр функції (докладніше див. далі);
- масив оголошено як посилання на масив, явно визначений в іншому модулі.

Використовуючи ім'я масиву та індекс, можна звертатися до елементів масиву:

`ім'я_масиву [значення_індексу]`

Значення індексів перебувають в діапазоні від нуля до величини, на одиницю меншу за розмір масиву, визначений при його оголошенні, оскільки в *C нумерація індексів розпочинається з нуля*.

Наприклад,

```
int A[10];
```

оголошує масив з ім'ям `A`, який містить 10 цілих чисел; при цьому виділяє і закріплює за цим масивом пам'ять для усіх 10-ти елементів відповідного типу (`int` – 4 байти), тобто 40 байтів. Отже, при оголошенні масиву виділяється пам'ять, потрібна для розташування усіх його елементів. Елементи масиву з першого до останнього зберігаються у послідовно зростаючих адресах пам'яті. Поміж елементами масиву в пам'яті проміжків немає. Елементи масиву зберігаються один за одним поелементно.

Зауважте на звертання у програмі до елементів масиву: `A[0]` – перший елемент, `A[1]` – другий, `A[9]` – останній.

Вводити чи виводити значення одновимірних масивів можна лише поелементно, для чого слід організовувати цикли зі змінням значень індексу зазначеного масиву.

При оголошенні масивів елементам масиву можна (необов'язково всім) присвоювати початкові значення, які у подальшому в програмі може бути змінено. Якщо реальна кількість ініціалізованих значень є меншою за розмірність масиву, то решта елементів масиву набуває значення 0:

```
тип_даних ім'я_масиву [розмір_масиву] =
{ значення_елементів_масиву };
```

Наприклад,

```
int a[5] = { 9,33,-23,8,1 };
```

```
//a[0]=9, a[1]=33, a[2]=-23, a[3]=8, a[4]=1
```

Вимірність масиву визначається кількістю індексів. Елементи одновимірного масиву (вектора) мають один індекс, двовимірного масиву (матриці, таблиці) – два індекси: перший з них – номер рядка, другий – номер стовпчика. Кількість індексів у масивах є необмежена. При розміщуванні елементів масиву в пам'яті комп'ютера першою чергою змінюється крайній правий індекс, потім решта – справа наліво.

Багатовимірний масив оголошується у програмі в такий спосіб:

```
тип ім'я [ розмір1 ] [ розмір2 ] ... [ розмірN ];
```

Кількість елементів масиву дорівнює добуткові кількості елементів за кожним індексом. У прикладі

```
int a[3][4];
```

оголошено двовимірний масив з 3-х рядків та 4-х стовпчиків (12-ти елементів) цілого типу:

```
a[0][0], a[0][1], a[0][2], a[0][3],  
a[1][0], a[1][1], a[1][2], a[1][3],  
a[2][0], a[2][1], a[2][2], a[2][3];
```

Під масив надається пам'ять, потрібна для розташування усіх його елементів. Елементи масиву один за одним, з першого до останнього, запам'ятовуються у послідовно зростаючих адресах пам'яті так само, як і елементи одновимірного масиву.

Порядок виконання лабораторної роботи:

- Познайомитися з масивами у мові C.
- Нарисувати блок-схему алгоритму вирішення задачі згідно індивідуального завдання.
- Написати програму згідно індивідуального завдання з використанням масивів, відлагодити її та отримати результати роботи програми.

Контрольні запитання:

- 1) Який формат опису масиву в мові програмування C?
- 2) Який тип мають елементи масиву?
- 3) Для якої мети служить кількість елементів в описі масиву?
- 4) Яким чином на мові програмування C позначається звернення до окремого елемента масиву?
- 5) Яким чином можна визначити розмір масиву?
- 6) Яке найменше значення має індекс елемента масиву?
- 7) Яке найбільше значення має індекс елемента масиву?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальне завдання.
- Блок-схему алгоритму вирішення задачі.
- Текст програми та результати її роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальні завдання:

№	Завдання
1	Задані дійсні числа x_1, x_2, \dots, x_{55} . Обчислити $x_1(x_2+x_3)(x_4+x_5+x_6)(x_7+x_8+x_9+x_{10}) \dots (x_{46}+x_{47}+ \dots +x_{55})$.
2	Задані дійсні числа x_1, x_2, \dots, x_{25} . Знайти три найбільших серед них.
3	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити кількість чисел в найдовшій послідовності із підряд розміщених нулів.

№	Завдання
4	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити, скільки із них більші за своїх "сусідів", тобто попереднього і наступного числа.
5	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити, скільки у цій послідовності змінюється знак. (Наприклад, в послідовності 1, -34, 8, 14, -5 знак змінюється три рази.)
6	Задані дійсні числа $a_1, b_1, a_2, b_2, \dots, a_{24}, b_{24}$. ($a_i < b_i$). Розглядаючи пари a_i і b_i як ліві і праві кінці відрізків на тій самій прямій, визначити найдовший відрізок.
7	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити порядковий номер того із них, який найближчий до заданого цілого числа.
8	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити, скільки із них приймають найбільше значення.
9	Задані дійсні числа x_1, x_2, \dots, x_{25} . Отримати суму додатних та суму від'ємних членів цієї послідовності.
10	Задані дійсні числа x_1, x_2, \dots, x_{25} . Чи кількість від'ємних членів цієї послідовності більша за кількість додатних, і на скільки?
11	Задані дійсні числа x_1, x_2, \dots, x_{25} . В заданій послідовності поміняти місцями найбільший та найменший члени.
12	Задані дійсні числа x_1, x_2, \dots, x_{25} . В заданій послідовності замінити всі члени, які менші за середнє значення нулями.
13	Задані дійсні числа x_1, x_2, \dots, x_{25} . В заданій послідовності замінити всі від'ємні члени на значення мінімального члена, а всі додатні – на значення максимального члена.
14	Задані дійсні числа x_1, x_2, \dots, x_{25} . Знайти три найменших серед них.
15	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити, скільки із них приймають найменше значення.
16	Задані дійсні числа x_1, x_2, \dots, x_{25} . Знайти суму чисел, розташованих між найбільшим та найменшим числом.

№	Завдання
17	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити суму чисел, розташованих після найменшого числа.
18	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити суму чисел, розташованих до найбільшого числа.
19	Задані дійсні числа x_1, x_2, \dots, x_{25} . Знайти два найменших і два найбільших серед них.
20	Задані дійсні числа $a_1, b_1, a_2, b_2, \dots, a_{24}, b_{24}$. ($a_i < b_i$). Розглядаючи пари a_i і b_i як ліві і праві кінці відрізків на тій самій прямій, визначити найкоротший відрізок.
21	Задані дійсні числа x_1, x_2, \dots, x_{25} . В заданій послідовності замінити всі члени, які більші за середнє значення одиницями.
22	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити, скільки із них більші за попереднє число.
23	Задані дійсні числа x_1, x_2, \dots, x_{25} . Знайти суму порядкових номерів найменшого і найбільшого чисел.
24	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити, скільки із них менші за наступне число.
25	Задані дійсні числа x_1, x_2, \dots, x_{25} . Переписати числа, розташовані між найбільшим та найменшим числом, в зворотному порядку.
26	Задані дійсні числа x_1, x_2, \dots, x_{25} . Переписати числа у такому порядку – спочатку усі додатні, потім усі від’ємні значення.
27	Задані дійсні числа x_1, x_2, \dots, x_{25} . Замінити числа, які повторюються, нулями.
28	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити кількість чисел у найдовшій послідовності з однакових чисел.
29	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити кількість чисел у найдовшій послідовності чисел, розташованих за зростанням.

№	Завдання
30	Задані дійсні числа x_1, x_2, \dots, x_{25} . Визначити кількість чисел у найдовшій послідовності чисел, розташованих за спаданням.

Приклад виконання

Лістинг 5.1: Обчислення середнього арифметичного елементів масиву

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    int size, i;
    float Sum = 0, Aver;
    int Arr[10];
    size = 10;
    for (i = 0; i < size; i++)
    {
        printf("Enter Arr[%d]:", i);
        scanf("%d", &Arr[i]);
    }
    for (i = 0; i < size; i++)
        Sum = Sum + Arr[i];
    Aver = Sum / size;
    printf("\nAverage is %f\n", Aver);
    return 0;
}
```

Результати роботи програми:

```
Enter Arr[0]:5
Enter Arr[1]:2
Enter Arr[2]:4
Enter Arr[3]:5
Enter Arr[4]:2
Enter Arr[5]:7
Enter Arr[6]:4
Enter Arr[7]:2
Enter Arr[8]:8
Enter Arr[9]:3

Average is 4.200000
```

ЛАБОРАТОРНА РОБОТА №6 РОЗВ'ЯЗУВАННЯ НА МОВІ С ЗАДАЧ, В ЯКИХ ВИКОРИСТОВУЮТЬСЯ ДИНАМІЧНІ ЧИСЛОВІ МАСИВИ

Мета роботи: познайомитися з використанням динамічних масивів в мові програмування С.

Теоретичний вступ:

Відмінності *динамічного масиву* від звичайного полягають у тому, що:

- пам'ять під динамічний масив виділяється в процесі виконання програми за допомогою спеціальних функцій;
- кількість елементів динамічного масиву може бути задано змінною (але у програмі її неодмінно має бути визначено до виділення пам'яті під масив).

Синтаксис оголошення динамічного одновимірного масиву за допомогою функції `malloc()` є такий:

```
тип* ім'я = (тип*)malloc(sizeof(тип)*кількість_елементів);
```

Приклад оголошення дійсного динамічного масиву зі змінною кількістю елементів за допомогою функції `malloc()`:

```
int N = 10;  
float* a = (float*)malloc(sizeof(float) * N);
```

Приклад оголошення дійсного динамічного масиву зі змінною кількістю елементів за допомогою функції `calloc()`:

```
int N = 10;  
float* a = (float*)calloc(N, sizeof(float));
```

Звільнення пам'яті з-під цього масиву `a`:

```
free(a);
```

Існує можливість звертатися до елементів масиву без індексації за допомогою вказівників. У циклі за допомогою індексу поточний елемент масиву записується як `a[i]`. Згадаймо, що ім'я масиву `a` можна використовувати як вказівник на початок (нульовий елемент) масиву. Тоді, згідно з арифметикою вказівників, `a+i` – це вказівник на елемент, який міститься на `i` комірок далі від початку масиву, тобто вказівник на `a[i]`. Значення елемента `a[i]` можна записати за допомогою операції розадресації: `*(a+i)`.

Двовимірний динамічний масив з m рядків і n стовпчиків займає в пам'яті сусідні $m*n$ комірок, тобто зберігається так само, як і одновимірний масив з $m*n$ елементів. При розміщенні елементи двовимірних масивів розташовуються в пам'яті підряд один за одним з першого до останнього без проміжків у послідовно зростаючих адресах пам'яті. Наприклад, масив 3×5 зберігається у пам'яті в такий спосіб:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0-й рядок					1-й рядок					2-й рядок				

У такому масиві перші п'ять елементів належать до першого рядка, наступні п'ять – до другого і останні п'ять – до третього.

Нагадаємо, що a – вказівник на початок масиву, тобто на елемент $a[0][0]$. Щоб звернутися, наприклад, до елемента $a[1][3]$, слід “перестрибнути” від початку масиву через 5 елементів нульового рядка й 3 елементи першого рядка, тобто написати: $*(a+1*5+3)$. У загальному випадку до елемента $a[i][j]$ можна звернутися в такий спосіб: $*(a+i*5+j)$. Але цей спосіб роботи з двовимірним масивом є не надто зручний, тому що в програмі при звертанні до елемента масиву доводиться розадресовувати вказівник і обчислювати індекс елемента.

Оголосити дійсний динамічний масив 3×5 можна як одновимірний з 15-ти елементів:

```
float* a = (float*)malloc(3 * 5 * sizeof(float));
```

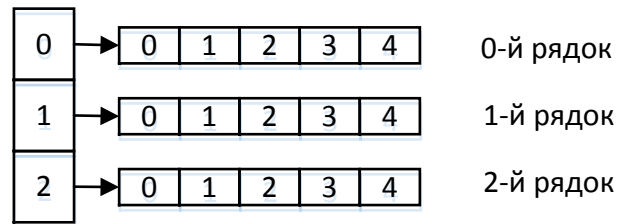
або

```
float* a = (float*)calloc(3 * 5, sizeof(float));
```

Пам'ять від створеного в такий спосіб масиву очищується за допомогою функції `free()`:

```
free(a);
```

Розглянемо інший спосіб роботи з динамічним двовимірним масивом. Для цього розмістимо в пам'яті матрицю 3×5 :



При цьому буде виділено пам'ять під кожний рядок матриці окремо, тобто буде утворено три різні одновимірні масиви. Адреси нульових елементів цих масивів зберігатимуться в допоміжному масиві **a** (пам'ять під нього слід виділити заздалегідь). Елементами цього масиву будуть адреси дійсних чисел, тому вони матимуть тип “вказівник на дійсне число”, тобто **float***. Нагадаємо, що загальний вигляд оголошення вказівника на динамічний масив є такий:

тип елемента* ім'я масиву;

Тоді при оголошенні масиву **a** слід записати дві зірочки:

```
float** a = (float**)malloc(3 * sizeof(float*));
```

// Оголошення й розміщення в пам'яті допоміжного масиву з 3-х елементів типу float*

```
for (int i = 0; i < 3; i++)
```

```
a[i] = (float*)malloc(5 * sizeof(float));
```

// У циклі виділяється пам'ять під 3 масиви по 5 елементів (рядки матриці)

// Значення адрес нульових елементів цих масивів записуються у відповідні елементи масиву **a**

Після цього можна працювати з матрицею як зі звичайним двовимірним масивом, звертаючись до кожного елемента за його індексом, наведеним у квадратних дужках: **a[i][j]**, – що є більш природним і зручним, ніж попередній спосіб.

Звільнення пам'яті необхідно виконувати в зворотному порядку – спочатку звільнити пам'ять під рядки в циклі, а потім звільнити пам'ять, що виділялась під масив вказівників.

```
for (int i = 0; i < 3; i++)
```

```
    free(a[i]);
```

```
free(a);
```

Аналогічним є оголошення за допомогою **calloc()**:

```
float** a = (float**)calloc(3, sizeof(float*));  
for (int i = 0; i < 3; i++)  
    a[i] = (float*)calloc(5, sizeof(float));
```

Звільнення пам'яті аналогічне як при роботі з функцією `malloc()`.

Порядок виконання лабораторної роботи:

- Познайомитися з динамічними масивами у мові C.
- Нарисувати блок-схему алгоритму вирішення задачі згідно індивідуального завдання.
- Написати програму згідно індивідуального завдання з використанням динамічних масивів, відлагодити її та отримати результати роботи програми. Основні дії над масивом, ввід (або заповнення випадковими значеннями), вивід елементів масиву реалізовувати через окремі власні функції.

Контрольні запитання:

- 1) Для чого використовуються динамічні масиви?
- 2) З допомогою яких функцій можна створити динамічний масив?
- 3) З допомогою якої функції звільняється пам'ять під виділений масив?
- 4) Яким чином можна звертатись до елементів динамічного масиву?
- 5) Що таке адресний вираз при звертання до елементів масиву?

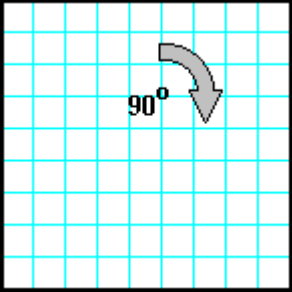
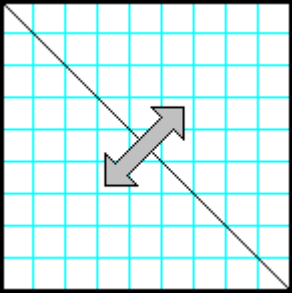
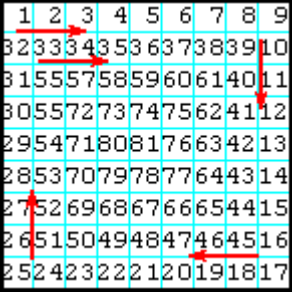
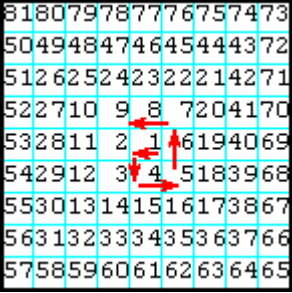
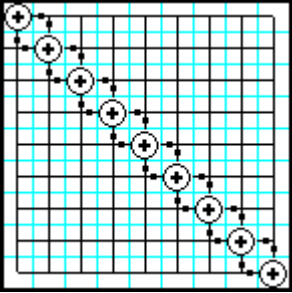
Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальне завдання.
- Блок-схему алгоритму вирішення задачі.
- Текст програми та результати її роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальні завдання:

№	Завдання	Ілюстрація
1	Заповнити матрицю випадковими числами. Повернути матрицю на 90° за годинниковою стрілкою.	
2	Заповнити матрицю випадковими числами. Відобразити матрицю симетрично відносно головної діагоналі.	
3	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від лівого верхнього кута по спіралі: вправо – вниз – вліво – вверх. (Примітка. На малюнку вказані індекси елементів вектора b)	
4	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від центра по спіралі: вліво – вниз – вправо – вверх. (Примітка. На малюнку вказані індекси елементів вектора b)	
5	Заповнити матрицю випадковими числами. На головній діагоналі розмістити суми елементів, які лежать у тому ж рядку і у тому ж стовпці.	

№	Завдання	Ілюстрація
6	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від лівого верхнього кута по діагоналі: вправо - вгору. (Примітка. На малюнку вказані індекси елементів вектора b)	
7	Заповнити сектори матриці, які лежать вліво і вправо від головної та бічної діагоналей, значеннями вектора b_1, b_2, \dots, b_{32} , від лівого верхнього кута вниз – вправо. Решта елементів матриці заповнити нулями. (Примітка. На малюнку вказані індекси елементів вектора b)	
8	Заповнити матрицю випадковими числами. Відобразити матрицю симетрично відносно вертикальної осі сектори матриці, які лежать вліво і вправо від головної і бічної діагоналей.	
9	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від лівого нижнього кута по діагоналі: вліво – вгору. (Примітка. На малюнку вказані індекси елементів вектора b)	
10	Заповнити матрицю випадковими числами. Відобразити головну і бічну діагоналі симетрично відносно вертикальної осі.	

№	Завдання	Ілюстрація
11	Заповнити матрицю випадковими числами. Розмістити на головній діагоналі суми елементів, котрі лежать на діагоналях, перпендикулярних до головної.	
12	Заповнити матрицю випадковими числами. Відобразити верхню половину матриці на нижню дзеркально симетрично відносно горизонтальної осі.	
13	Заповнити матрицю випадковими числами. Розбити матрицю на квадрати розміром 3x3. В центрі кожного квадрата помістити суму решти елементів квадрата.	
14	Заповнити матрицю випадковими числами. Відобразити праву половину матриці на ліву дзеркально симетрично відносно вертикальної осі.	
15	Заповнити сектори матриці, які лежать вліво і вправо від головної та бічної діагоналей значеннями вектора b_1, b_2, \dots, b_{32} від лівого верхнього кута вправо – вниз. Решту матриці заповнити нулями. (Примітка. На малюнку вказані індекси елементів вектора b)	

№	Завдання	Ілюстрація
16	Заповнити матрицю випадковими числами. Повернути матрицю на 90° проти годинникової стрілки.	
17	Заповнити матрицю випадковими числами. Відобразити матрицю симетрично відносно бічної діагоналі.	
18	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від лівого верхнього кута по спіралі: вниз – вправо – вверх – вліво. (Примітка. На малюнку вказані індекси елементів вектора b)	
19	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від центра по спіралі: вниз – вліво – вверх – вправо. (Примітка. На малюнку вказані індекси елементів вектора b)	
20	Заповнити матрицю випадковими числами. На бічній діагоналі розмістити суми елементів, які лежать у тому ж рядку і у тому ж стовпці.	

№	Завдання	Ілюстрація
21	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від лівого верхнього кута по діагоналі: вліво – вниз. (Примітка. На малюнку вказані індекси елементів вектора b .)	
22	Заповнити сектори матриці, які лежать вище і нижче від головної та бічної діагоналей, значеннями вектора b_1, b_2, \dots, b_{32} , від лівого верхнього кута вправо – вниз. Решта матриці заповнити нулями. (Примітка. На малюнку вказані індекси елементів вектора b .)	
23	Заповнити матрицю випадковими числами. Відобразити матрицю симетрично відносно горизонтальної осі сектори матриці, які лежать вище і нижче від головної і бічної діагоналей.	
24	Заповнити матрицю значеннями вектора b_1, b_2, \dots, b_{81} від правого верхнього кута по діагоналі: вліво – вниз. (Примітка. На малюнку вказані індекси елементів вектора b .)	
25	Заповнити матрицю випадковими числами. Відобразити головну і бічну діагоналі симетрично відносно горизонтальної осі.	

№	Завдання	Ілюстрація
26	Заповнити матрицю випадковими числами. Розмістити на бічній діагоналі суми елементів, котрі лежать на діагоналях, перпендикулярних до бічної.	
27	Заповнити матрицю випадковими числами. Відобразити ліву половину матриці на праву дзеркально симетрично відносно вертикальної осі.	
28	Заповнити матрицю випадковими числами. Повернути матрицю на 180° за годинниковою стрілкою.	
29	Заповнити матрицю випадковими числами. Відобразити нижню половину матриці на верхню дзеркально симетрично відносно горизонтальної осі.	
30	Заповнити сектори матриці, які лежать вище і нижче від головної та бічної діагоналей, значеннями вектора b_1, b_2, \dots, b_{32} , від лівого верхнього кута вниз – вправо. Решта матриці заповнити нулями. (Примітка. На малюнку вказані індекси елементів вектора b)	

Приклад виконання

Лістинг 6.1: Програма у динамічному двовимірному масиві множить кожен елемент на 2

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
int** CreateMatrix(int m, int n)
{
    int** a;
    a = (int**)malloc(m * sizeof(int*));
    for (int i = 0; i < m; i++)
        a[i] = (int*)malloc(n * sizeof(int));
    return a;
}
void DeleteMatrix(int** a, int m, int n)
{
    for (int i = 0; i < m; i++)
        free(a[i]);
    free(a);
}
void FillMatrix(int** a, int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
        {
            a[i][j] = rand() % 10;
        }
}
void PrintMatrix(int** a, int m, int n)
{
    int i, j;
    printf("\n");
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}
```

```

void ChangeMatrix(int** a, int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
        {
            a[i][j] = a[i][j]*2;
        }
}

int main(void)
{
    int M, N, ** A;
    printf("Enter the matrix A sizes (m, n):");
    scanf("%d%d", &M, &N);
    A = CreateMatrix(M, N);
    FillMatrix(A, M, N);
    printf("Matrix A:\n");
    PrintMatrix(A, M, N);
    ChangeMatrix(A, M, N);
    printf("Matrix A:\n");
    PrintMatrix(A, M, N);
    DeleteMatrix(A, M, N);
    return 0;
}

```

Результати роботи програми:

Enter the matrix A sizes (m, n):3 7						
Matrix A:						
1	7	4	0	9	4	8
8	2	4	5	5	1	7
1	1	5	2	7	6	1
Matrix A:						
2	14	8	0	18	8	16
16	4	8	10	10	2	14
2	2	10	4	14	12	2

ЛАБОРАТОРНА РОБОТА №7 РОЗРОБКА НА МОВІ С БАГАТОФАЙЛОВИХ ПРОЕКТІВ

Мета роботи:

- познайомитися із принципами розробки багатофайлових проектів на мові програмування С;
- познайомитися із прийомами розробки багатофайлових проектів в інтегрованому середовищі Microsoft Visual Studio 2019.

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 Studio 2019) 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 проєкті додаємо файл `main.cpp`, де буде визначення функції `main()` за допомогою пункту меню *Project* → *Add New Item*.

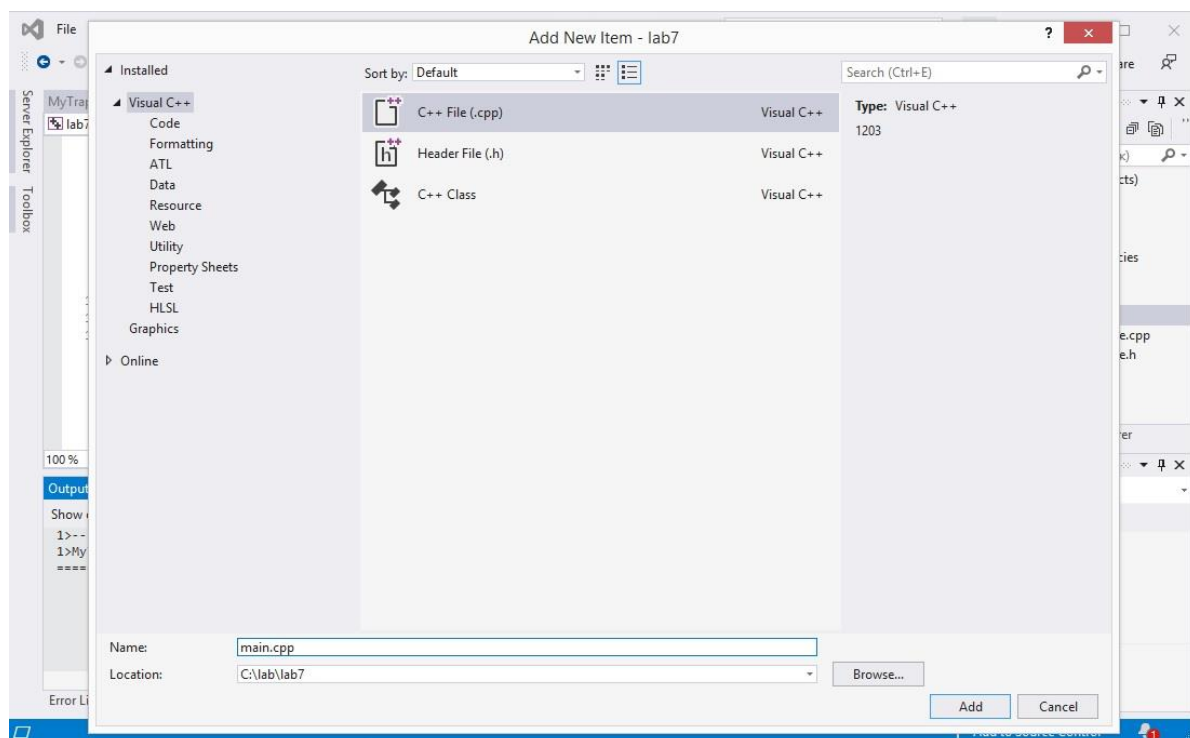


Рисунок 7.1 – Додавання до проєкту файлу `main.cpp`

Далі додаємо аналогічним чином файл `MyTrapezSquare.h`:

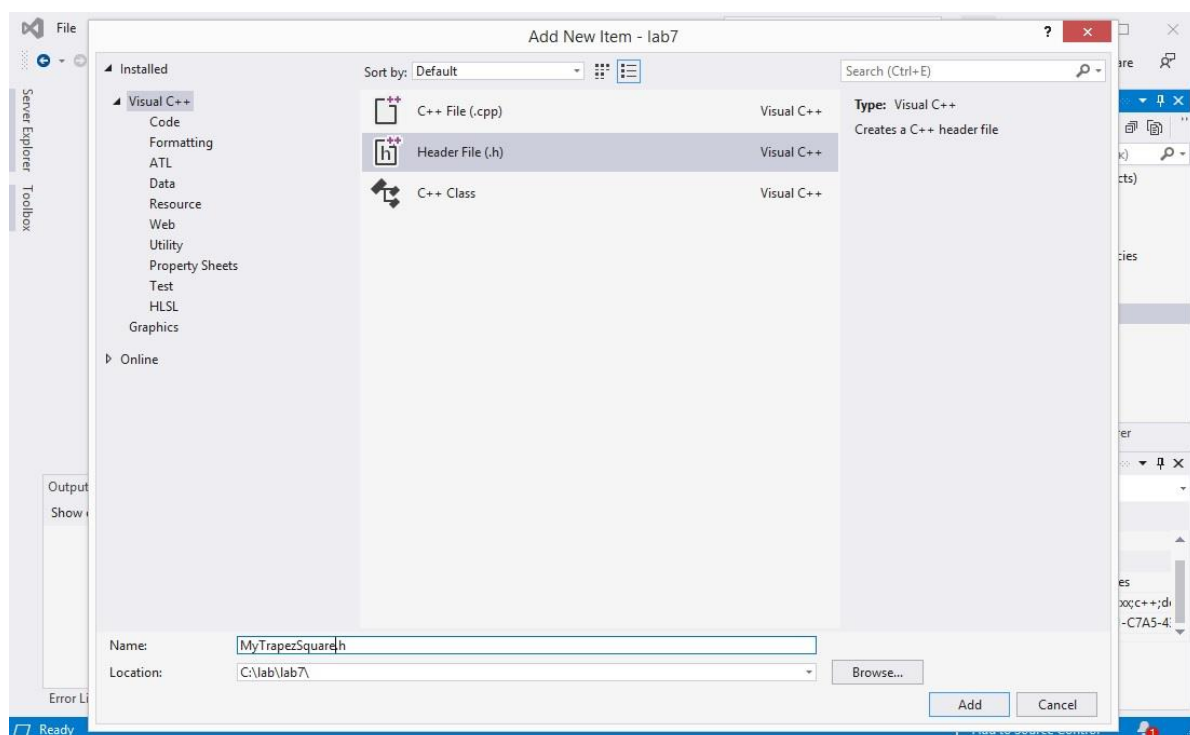


Рисунок 7.2 - Додавання до проєкту файлу `MyTrapezSquare.h`

І файл `MyTrapezSquare.cpp`:

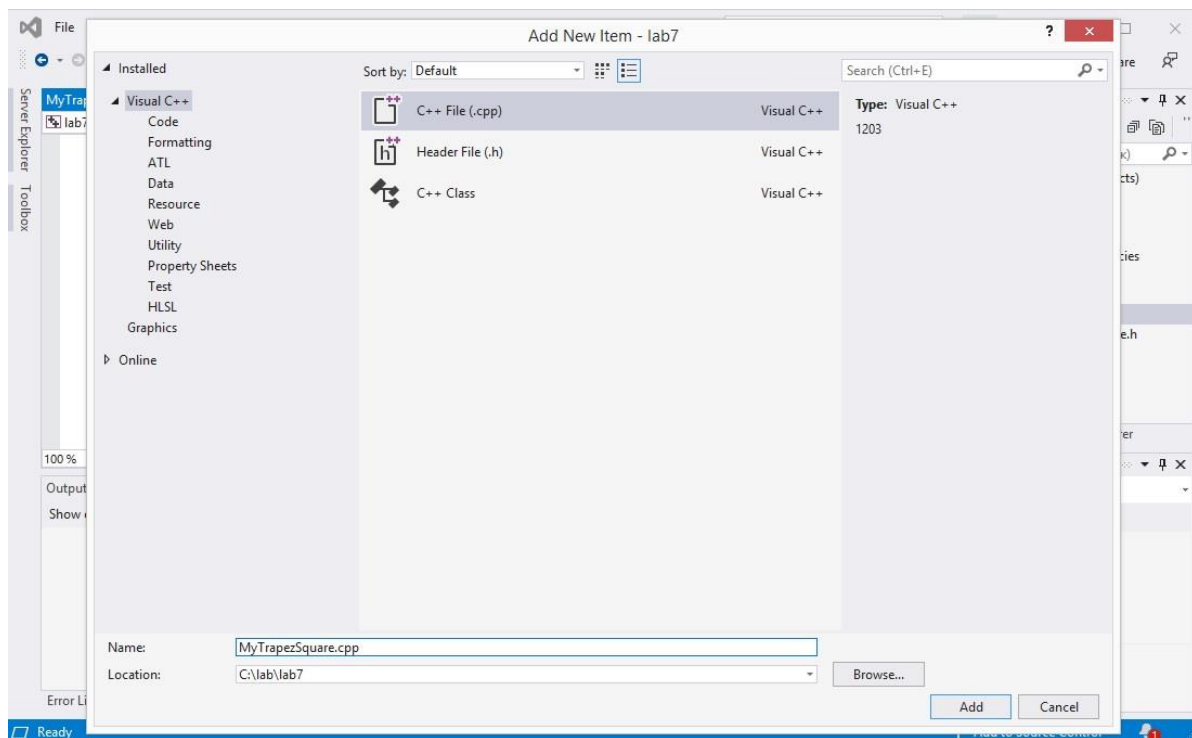


Рисунок 7.3 - Додавання до проекту файлу `MyTrapezSquare.cpp`

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

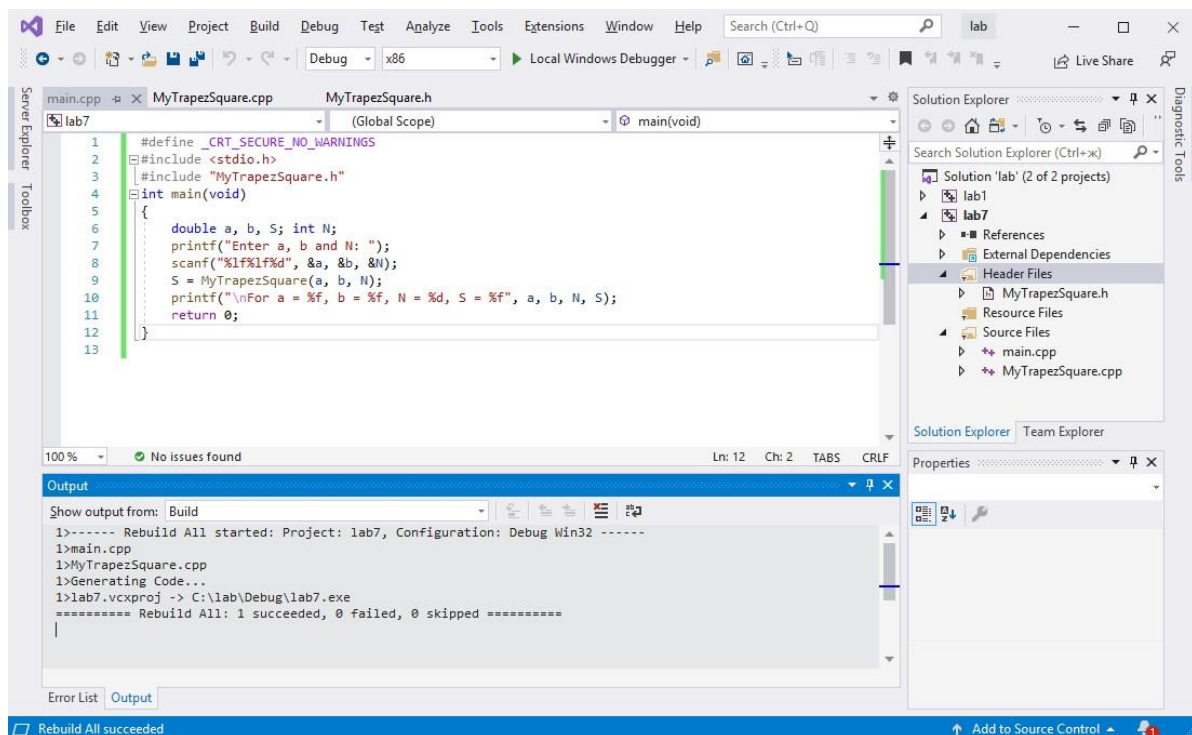


Рисунок 7.4 – Створення виконавчого файлу багатобайтового проекту

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:

- Використовуючи вище наведений опис, познайомитися із засобами створення багатофайлових проектів в інтегрованому середовищі Microsoft Visual Studio 2019.
- 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?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальне завдання.
- Тексти програми та результати її роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальні завдання:

Індивідуальні завдання такі ж, як і до **лабораторної роботи №6**. Програму необхідно оформити у вигляді багатофайлового проекту, в якому є декілька файлів з функціями програми та заголовний файл.

Основні дії над масивом, ввід (або заповнення випадковими значеннями), вивід елементів масиву реалізовувати через власні функції, які мають бути розміщені в окремому файлі. Оголошення (прототипи) цих функцій мають бути в окремому заголовному файлі, який треба підключити до основної програми.

Функція `main()` має демонструвати роботу основної задачі. Тобто, виклик функції для вводу або заповнення випадковими значеннями елементів масиву, виклик функції для основних дії та виклик функції для виводу результатів.

Приклад виконання

Лістинг 7.1: Програма у динамічному двовимірному масиві множить кожен елемент на 2

```
//файл з прототипами функцій MyFunc.h
int** CreateMatrix(int m, int n);
void DeleteMatrix(int** a, int m, int n);
void FillMatrix(int** a, int m, int n);
void PrintMatrix(int** a, int m, int n);
void ChangeMatrix(int** a, int m, int n);
```

```
//файл з реалізацією функції main main.cpp
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include "MyFunc.h"
int main(void)
{
    int M, N, ** A;
    printf("Enter the matrix A sizes (m, n):");
    scanf("%d%d", &M, &N);
    A = CreateMatrix(M, N);
    FillMatrix(A, M, N);
    printf("Matrix A:\n");
    PrintMatrix(A, M, N);
    ChangeMatrix(A, M, N);
    printf("Matrix A:\n");
    PrintMatrix(A, M, N);
    DeleteMatrix(A, M, N);
    return 0;
}
```

```

//файл з реалізаціями власних функцій MyFunc.cpp
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include "MyFunc.h"
int** CreateMatrix(int m, int n)
{
    int** a;
    a = (int**)malloc(m * sizeof(int*));
    for (int i = 0; i < m; i++)
        a[i] = (int*)malloc(n * sizeof(int));
    return a;
}

void DeleteMatrix(int** a, int m, int n)
{
    for (int i = 0; i < m; i++)
        free(a[i]);
    free(a);
}

void FillMatrix(int** a, int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
        {
            a[i][j] = rand() % 10;
        }
}

void PrintMatrix(int** a, int m, int n)
{
    int i, j;
    printf("\n");
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}

void ChangeMatrix(int** a, int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
        {
            a[i][j] = a[i][j] * 2;
        }
}

```


Результати роботи програми:

Enter the matrix A sizes (m, n):4 6

Matrix A:

1	7	4	0	9	4
8	8	2	4	5	5
1	7	1	1	5	2
7	6	1	4	2	3

Matrix A:

2	14	8	0	18	8
16	16	4	8	10	10
2	14	2	2	10	4
14	12	2	8	4	6

ЛАБОРАТОРНА РОБОТА №8 РОЗВ'ЯЗУВАННЯ НА МОВІ C ЗАДАЧ, В ЯКИХ ВИКОРИСТОВУЮТЬСЯ МАСИВИ ТИПУ CHAR І РЯДКИ

Мета роботи: познайомитися з використанням масивів типу char і рядків у мові програмування C.

Теоретичний вступ:

Символами вважаються: великі й малі літери, цифри, знаки арифметичних дій ('+', '-', '*', '/', '='), пробіл, розділові знаки ('.', ',', ';', ':', '!', '?', '-'), службові символи, що відповідають клавішам <Enter>, <Esc>, <Tab> тощо. В C значення символьних констант записуються у одинарних лапках: '3', 'f', '+', '%'.

Як було зазначено раніше для кодування усіх символів використовується восьми розрядна послідовність 0 і 1, тобто один байт. Наприклад: символ цифри '9' кодується послідовністю бітів 0011 1001, символ літери латиниці 'W' – 0101 0111. За допомогою одного байта можна закодувати $2^8 = 256$ різних комбінацій бітів, а отже, 256 різних символів.

Щоб не було розходжень у кодуванні символів, існує єдиний міжнародний стандарт – так звана таблиця ASCII-кодів (American Standard Code for Information Interchange – американський стандартний код для обміну інформацією, див. додаток А). Символи ASCII мають коди від 0 до 127, тобто значення першої половини можливих значень байта, хоча часто кодами ASCII називають всю таблицю з 256 символів. Перші 128 ASCII-кодів є єдині для всіх країн, а коди від 128 до 255 називають розширеною частиною таблиці ASCII, де залежно від країни розташовується національний алфавіт і символи псевдографіки.

У таблиці ASCII всі символи пронумеровано, тобто вони мають власний унікальний код. Так само як у кожній мові людського спілкування існує алфавіт (перелік усіх літер у чітко визначеному порядку), усі комп'ютерні символи теж є суворо упорядкованими. Символ ' ' (пробіл) має код 32, цифри мають коди від 48 для '0' до 57 для '9', великі латинські літери – від 65 для 'A' до 90 для 'Z', малі літери латиниці – від 97 для 'a' до 122 для 'z'. Кодування другої половини таблиці ASCII має різні варіанти. Найпоширенішими є DOS-кодування (866 кодова сторінка) і кодування 1251, яке є основним для Windows. Звернімо увагу на різницю поміж цифрами і їхнім символьним зображенням.

Наприклад, символ цифри '4' має ASCII-код 52 і не має безпосереднього відношення до числа 4.

Тип символних змінних у C називається `char`.

Зауважимо, що у C, на відміну від більшості мов програмування, дані типу `char` змінюються у діапазоні $-128 \dots 127$, причому додатні числа $0 \dots 127$ зайняті символами спільної частини ASCII-таблиці, а символи розширеної частини ASCII-таблиці у C відповідають від'ємним числам. Наприклад, літера кирилиці 'ч' – має код -9 , а кодом літери 'я' є -1 (для таблиці 1251).

Окрім типу `char`, існує його беззнакова модифікація `unsigned char`. Дані типу `unsigned char` мають значення у діапазоні $0 \dots 255$. В ASCII-таблиці значення кодів літер кирилиці є більшими за 127, тому, якщо треба мати справу зі змінними, значеннями яких є літери кирилиці, їх слід оголошувати типом `unsigned char`.

Символи можна порівнювати. Більшим вважається той символ, у якого код є більший, тобто символ, розташований у таблиці ASCII-кодів пізніше. Наприклад: `'a' < 'h'`, `'A' < 'a'`.

Оскільки символний тип `char` вважається у C за цілий тип, змінні цього типу можна додавати й віднімати. Результатом додавання буде символ, код якого дорівнює сумі кодів символів-доданків.

Рядки у C являють собою послідовність (масив) символів із завершальним нуль-символом. Нуль-символ (нуль-термінатор) – це символ з кодом 0, який записується у вигляді керуючої послідовності `'\0'`. За розташуванням нуль-символу визначається фактична довжина рядка.

Відмінною рисою рядка є те, що в ньому насправді може бути менше символів, аніж зазначено при оголошенні. Окрім того, з рядками можна виконувати певні специфічні дії, які не можна здійснювати з числовими масивами (наприклад перевіряти наявність у масиві літери чи послідовності літер, копіювати масив як одне ціле, порівнювати масиви за алфавітом, дописувати один масив наприкінці іншого тощо).

Пам'ять під розміщення рядків, як і для будь-яких масивів, може виділятися як компілятором, так і динамічно – при виконуванні програми. Довжина динамічного рядка може задаватися змінною з визначеним заздалегідь значенням, а довжина статичного рядка має задаватися лише константою.

Рядок може бути оголошеним в один з нижче наведених способів:

1. `char* s;` // Оголошення вказівника на перший символ рядка;

// пам'ять під сам рядок не виділяється

2. `char ss[32];` // Оголошення рядка ss з 31-го символу;

// пам'ять виділяється компілятором

3. `int n;`

`scanf("%d", &n);`

`char* str = new char[n];`

// Оголошення рядка str з n-1 символів, пам'ять виділяється динамічно

При зазначенні довжини рядка слід враховувати завершальний нуль-символ.

Зауважимо, що при оголошенні рядка першим способом пам'ять під рядок не виділяється і це може бути дуже небезпечним, оскільки до тієї самої ділянки пам'яті може бути розміщено інші змінні й рядок буде втрачено.

При оголошенні рядок можна ініціалізувати рядковою константою, при цьому нуль-символ формується автоматично після останнього символу:

`char str[10] = "world";`

Як і числові масиви, рядки опрацьовуються поелементно у циклі. Операція присвоювання одного рядка іншому є невизначена (оскільки рядок є масивом) і може виконуватися за допомогою циклу чи за допомогою функцій стандартної бібліотеки.

Порядок виконання лабораторної роботи:

- Познайомитися з символьними масивами та рядками в мові програмування C.
- Нарисувати блок-схему алгоритму вирішення задачі згідно індивідуального завдання.
- Написати програму згідно індивідуального завдання, в якій використовуються рядкові дані, відлагодити її та отримати результати роботи програми.

Контрольні запитання:

- 1) Як називається символьний тип даних в С?
- 2) Для чого служить таблиця ASCII?
- 3) Що таке рядок в мові С?
- 4) Що таке символьна константа в мові С?
- 5) Що таке рядкова константа в мові С?
- 6) Для чого служить нуль-символ в рядку?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальне завдання.
- Блок-схему алгоритму вирішення задачі.
- Текст програми та результати її роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальні завдання:

Індивідуальне завдання до даної лабораторної роботи необхідно виконати, **не використовуючи бібліотечних функцій** для роботи з рядками!

№	Завдання
1	Перевірити, чи в заданий рядок, що завершується крапкою, входить кожна з букв слова <i>key</i> .
2	Перевірити, чи в заданому рядку правильно розставлені дужки (тобто справа від кожної відкриваючої дужки є відповідна закриваюча, а зліва від кожної закриваючої – є відповідна відкриваюча).
3	Перевірити, чи заданий рядок, що завершується крапкою, є правильним записом цілого числа (можливо, зі знаком).
4	Вивести заданий рядок, вилучивши із нього всі зайві пробіли, тобто із декількох пробілів, що розміщені підряд, залишити лише один.
5	Заданий рядок роздрукувати по словах, вважаючи, що словом є або чергові 10 символів, якщо серед них немає коми, або та частина символів, яка розміщена до коми включно.
6	Заданий рядок, який складається з послідовності непорожніх слів із латинських букв; сусідні слова відділяються одне від другого комами, а за останнім словом є крапка. Визначити кількість слів, які починаються і закінчуються однією і тією ж самою буквою.
7	Заданий рядок, який складається з послідовності непорожніх слів із латинських букв; сусідні слова відділяються одне від другого комами, а за останнім словом є крапка. Визначити кількість слів, які закінчуються буквою <i>w</i> .
8	Заданий рядок, який складається з послідовності непорожніх слів із латинських букв; сусідні слова відділяються одне від другого комами, а за останнім словом є крапка. Визначити кількість слів, які містять хоча би одну букву <i>d</i> .

№	Завдання
9	Заданий рядок, який складається з послідовності непорожніх слів із латинських букв; сусідні слова відділяються одне від другого комами, а за останнім словом є крапка. Визначити кількість слів, які містять рівно три букви <i>a</i> .
10	Побудувати рядок символів, який є записом заданого цілого числа <i>k</i> в двійковій системі числення.
11	Заданий рядок, який завершується крапкою. Визначити чи впорядковані символи у цьому рядку у алфавітному порядку.
12	Заданий рядок, який є записом цілого числа, що завершується крапкою, а на початку може містити знак числа. Знайти суму цифр цього числа.
13	Заданий рядок, який має наступний вигляд: $d_1 \pm d_2 \pm \dots \pm d_n$ (d_i – цифри, $n > 1$), який закінчується крапкою. Обчислити значення цієї алгебраїчної суми.
14	Перевірити, чи заданий рядок є правильним записом дійсного числа (дійсне число обов'язково містить кому для відокремлення цілої частини від дійсної і можливо знак).
15	Побудувати рядок символів, який є записом заданого цілого числа <i>k</i> в шістнадцятковій системі числення.
16	Заданий рядок, який є записом цілого числа, що завершується крапкою, а на початку може містити знак числа. Перетворити цей рядок на число типу <code>int</code> .
17	Перевірити, чи в заданий рядок входить слово <i>key</i> . Слова в рядку відокремлені один від одного комами.
18	В заданому рядку, який завершується крапкою, вилучити усі символи, які повторюються.
19	Задано рядок, що завершується крапкою. Відсортувати його у алфавітному порядку.

№	Завдання
20	Задано рядок, що завершується крапкою. Замінити в рядку усі малі літери латиниці на великі і навпаки. Символи, які не належать до літер латиниці залишити без змін.
21	В заданому рядку, який завершується крапкою, вилучити усі входження слова <i>key</i> .
22	В заданому рядку, який завершується крапкою, знайти найбільшу кількість цифр, що йдуть підряд.
23	В заданому рядку, який завершується крапкою, підрахувати скільки разів зустрічається слово <i>key</i> .
24	В заданому рядку, що завершується крапкою, слова відокремлені один від одного комами. Знайти довжину найдовшого слова.
25	В заданому рядку, що завершується крапкою, знайти цифру, яка зустрічається найбільшу кількість раз.
26	В заданому рядку, який завершується крапкою, вилучити усі цифри.
27	В заданому рядку, що завершується крапкою, подвоїти усі цифри.
28	В заданому рядку, що завершується крапкою, вилучити усі символи, які не є цифрами.
29	В заданому рядку, що завершується крапкою, знайти цифру, яка зустрічається найменшу кількість раз.
30	В заданому рядку, що завершується крапкою, слова відокремлені один від одного комами. Знайти довжину найкоротшого слова.

Приклад виконання

Лістинг 8.1: В заданому рядку, що завершується крапкою, знайти скільки разів зустрічається заданий символ.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    char str[128];
    char s;
    int i, n;
    printf("Enter the string, please: ");
    gets_s(str, sizeof(str)); //gets(str1);
    printf("Enter the character, please: ");
    scanf("%c", &s);
    i = 0;
    n = 0;
    while (str[i] != '.')
    {
        if (str[i] == s) n++;
        i++;
    }
    printf("The number of %c characters in a string is %d", s, n);
    return 0;
}
```

Результати роботи програми:

```
Enter the string, please: enter the string, please. please
Enter the character, please: e
The number of e characters in a string is 5
```

ЛАБОРАТОРНА РОБОТА №9 РОЗВ'ЯЗУВАННЯ НА МОВІ C ЗАДАЧ, ЯКІ ВИКОРИСТОВУЮТЬ ФАЙЛИ ДЛЯ ВВОДУ ТА ВИВОДУ ДАНИХ

Мета роботи: познайомитися з засобами файлового вводу та виводу в мові програмування C.

Теоретичний вступ:

Файлами є іменовані області пам'яті на зовнішньому носії, призначені для довготривалого зберігання інформації. Файли мають *імена* й є організовані в ієрархічну деревовидну структуру з *каталогів* (тек) і простих файлів.

Для доступу до даних файлу з програми в ній слід прописати функцію відкриття цього файлу, тим самим встановити зв'язок між ім'ям файлу і певною файловою змінною у програмі.

Файли відрізняються від звичайних масивів тим, що:

- вони можуть змінювати свій розмір;
- звертання до елементів файлів здійснюється не за допомогою операції індексації `[]`, а за допомогою спеціальних системних викликів та функцій;
- доступ до елементів файлу відбувається з так званої позиції зчитування-записування, яка автоматично просувається при операціях зчитування-записування, тобто файл є видимим послідовно. Існують, щоправда, функції для довільного змінювання цієї позиції.

Мова C надає засоби опрацювання двох типів файлів: *текстових* та *бінарних*.

Текстові файли призначено для зберігання текстів, тобто сукупності символічних рядків змінної довжини. Кожен рядок завершується керуючою послідовністю `'\n'`, а розділювачами слів та чисел у рядку є пробіли й символи табуляції. Оскільки вся інформація текстового файлу є символічною, програмне опрацювання такого файлу полягає в читанні рядків, відокремленні з рядка слів і, за потреби, перетворенні цифрових символічних послідовностей на числа відповідними функціями перетворювання. Створювати, редагувати текстові файли можна не лише в програмі, а й у якому завгодно текстовому редакторі, наприклад Блокноті чи Word.

Бінарні файли зберігають дані в тому самому форматі, в якому вони були оголошені, і їхній вигляд є такий самий, як і в пам'яті комп'ютера. І тому

відпадає потреба у використанні розділювачів: пробілів, керуючих послідовностей, а отже, обсяг використовуваної пам'яті порівняно з текстовими файлами з аналогічною інформацією є значно меншим. Окрім того, немає потреби у застосуванні функцій перетворення числових даних. Але кожне опрацювання даних бінарних файлів можливе лише за наявності програми, якій має бути відомо, що саме і в якій послідовності зберігається у цьому файлі.

У мові C *файл* розглядається як потік послідовності байтів. Інформація про файл заноситься до змінної типу `FILE*`. Цей тип оголошує вказівник потоку, який використовується надалі у всіх операціях з цим файлом. Тип `FILE` означено у бібліотеці `stdio.h`.

Файлова змінна це – вказівник потоку, який в подальшому буде передаватися у функції введення-виведення у якості параметра:

```
FILE* f;
```

Функція `fopen()` для відкривання файлу має такий синтаксис:

```
FILE* fopen(const char* filename, const char* mode);
```

Перший параметр `filename` визначає ім'я файлу, який відкривається. Другий параметр `mode` задає режим відкривання файлу.

Таблиця 9.1 – Специфікатори режиму відкриття файлів

Параметр	Опис
<code>r</code>	Відкрити файл для читання даних
<code>r+</code>	Відкрити файл для читання і запису даних
<code>a</code>	Відкрити чи створити файл для запису даних в кінець файлу
<code>a+</code>	Відкрити чи створити файл для читання і запису даних в кінець файлу
<code>w</code>	Створити файл для запису даних
<code>w+</code>	Створити файл для читання і запису даних

До зазначених специфікаторів наприкінці чи перед знаком “+” може дописуватись символ чи то “`t`” – для текстових файлів, чи “`b`” – для бінарних (двійкових) файлів.

Функція `fopen()` повертає вказівник на об'єкт, який керує потоком. Якщо файл відкрити не вдалося, `fopen()` повертає нульовий вказівник `NULL`. Для уникання помилок після відкриття файлу слід перевірити, чи насправді файл відкрився.

Функція `fopen()` в середовищі Visual Studio 2019 вважається небезпечною.

Припинити роботу з файлом можна за допомогою функції

```
int fclose(FILE * stream);
```

```
int _fcloseall(void);
```

Функція **`fclose`** закриває файл, на який посилається параметр функції.

Функція **`_fcloseall`** закриває усі відкриті файли.

Перевірка кінця файлу здійснюється функцією `feof()`:

```
int feof(FILE * stream);
```

З текстового файлу можна читати цілі рядки й окремі символи.

Зчитування рядка з файлу здійснюється функцією `fgets()`:

```
char* fgets(char* str, int numChars, FILE * stream);
```

де `str` – перший параметр – рядок типу `char*`; `numChars` – максимальна кількість читаних символів (байтів); `stream` – вказівник на потік даних файлу.

Записування даних до текстового файлу можна здійснювати за допомогою функції

```
int fputs(const char* str, FILE * stream);
```

де `str` – рядок типу `char`, `stream` – файловий потік.

Зчитування форматованих даних можна також здійснювати за допомогою функції `fscanf()`:

```
int fscanf(FILE * stream, const char* format[, argument]);
```

Записування до текстового файлу можна здійснити також за допомогою функції `fprintf()`:

```
int fprintf(FILE * stream, const char* format[, argument]);
```

Текстові файли дозволяють переміщувати поточну позицію зчитування-запису. Для визначення поточної позиції файлу, яка автоматично зміщується на кількість опрацьованих байт, використовується функція `ftell()`:

```
long int ftell(FILE * stream);
```

А змінити поточну позицію файлу можна за допомогою функції `fseek()`:

```
int fseek(FILE * stream, long offset, int whence);
```

Ця функція задає зсув на `offset` байтів щодо точки відліку, яка задається параметром `whence`. Параметр `whence` може набувати таких значень: 0 – початок файлу, 1 – поточна позиція, 2 – кінець файлу.

У *бінарному* (двійковому) файлі число, на відміну від текстового, зберігається у внутрішньому його поданні. У двійковому форматі можна зберігати не лише числа, а й рядки та цілі інформаційні структури. Причому останні зберігати зручніше, завдяки тому що відсутня потреба явно зазначати кожен елемент структури, а зберігається вся структура як цілковита одиниця даних. Хоча цю інформацію не можна прочитати як текст, вона зберігається більш компактно і точно. Тому, що саме і в якій послідовності розміщено в бінарному файлі, має бути відомо програмі.

Функція `fread()` читає інформацію у вигляді потоку байтів і в незмінному вигляді розміщує її в пам'яті. Слід розрізнявати текстове подавання чисел і їхнє бінарне подавання.

```
size_t fread
(char* buffer, // Масив для зчитування даних,
size_t elemSize, // розмір одного елемента,
size_t numElems, // кількість елементів для зчитування
FILE * f // і вказівник потоку
);
```

Тут `size_t` означений як беззнаковий цілий тип у системних заголовних файлах. Функція намагається прочитати `numElems` елементів з файлу, який задається вказівником `f`, розмір кожного елемента становить `elemSize`. Функція повертає реальну кількість прочитаних елементів, яка може бути менше за `numElems`, у разі завершення файлу чи то помилки зчитування.

Функція бінарного записування до файлу `fwrite()` є аналогічна до функції зчитування `fread()`. Вона має такий прототип:

```
size_t fwrite
(char* buffer, // Масив записуваних даних,
size_t elemSize, // розмір одного елемента,
size_t numElems, // кількість записуваних елементів
FILE * f // і вказівник потоку
);
```

Функція повертає кількість реально записаних елементів, яка може бути менше за `numElems`, якщо при записуванні виникла помилка: приміром, не вистачило вільного простору на диску.

Так само, як і в текстових, у бінарних файлах можна визначати позицію зчитування-записування і переміщувати її у довільне місце файлу командами відповідно `ftell()` та `fseek()`. Можливість переміщувати позицію є корисна для файлів, які складаються з однорідних записів однакового розміру.

Порядок виконання лабораторної роботи:

- Познайомитися з засобами вводу з файлів та виводу у файл в мові програмування C.
- Написати програму згідно індивідуального завдання, в якій використовуються файли для вводу і виводу інформації та отримати результати роботи програми.

Контрольні запитання:

- 1) Для чого використовуються файли?
- 2) З якими типами файлів можуть працювати програми, написані на мові програмування C?
- 3) В якому вигляді зберігаються дані у текстових файлах?
- 4) В якому вигляді зберігаються дані у бінарних файлах?
- 5) Як оголосити файлову змінну?
- 6) Якою функцією можна відкрити файл?
- 7) Якою функцією можна закрити файл?
- 8) Які функції читають дані з текстового файлу?
- 9) Які функції дозволяють записати дані у текстовий файл?
- 10) Які функції дозволяють працювати з бінарними файлами?
- 11) Що таке режим відкриття файлу?

Вимоги до звіту:

У звіті до лабораторної роботи необхідно навести:

- Назву лабораторної роботи.
- Мету лабораторної роботи.
- Індивідуальне завдання.
- Текст програми та результати її роботи (вхідні та вихідні дані).
- Короткий висновок.

(Примітка. Уточнити у викладача вимоги до звіту.)

Індивідуальні завдання:

Індивідуальні завдання такі ж, як і до **лабораторної роботи №5**. Відмінність полягає у тому, що ввід та вивід виконується через файли.

Необхідно написати два варіанти програми:

- 1) програма у якій ввід здійснюється з текстового файлу і вивід виконується в текстовий файл;
- 2) програма у якій ввід здійснюється з бінарного файлу і вивід відбувається в бінарний файл.

У бінарному файлі дані зберігаються у внутрішньому представленні.

Для варіанту, який використовує бінарні файли, необхідно написати дві допоміжні програми:

- 1) допоміжну програму яка формує бінарний файл з вхідними даними;
- 2) допоміжну програму яка читає вихідний бінарний файл і виводить на екран монітора вихідні дані.

Приклад виконання

Лістинг 9.1: Обчислення середнього арифметичного елементів масиву. Вхідні дані беруться з файлу MyFile.txt, кожне число починається з нового рядка. Результат зберігається у тому ж файлі.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int size, i;
    float Sum = 0, Aver;
    int Arr[100];
    FILE* pf;
    char Buf[128] = ""; char* pch;
    /*----- file reading -----*/
    size = 0;
    pf = fopen("MyFile.txt", "rt");
    if (pf != 0)
    {
        pch = fgets(Buf, sizeof(Buf), pf);
        while (pch != 0 && size < 100)
        {
            Arr[size] = atoi(Buf);
            size++;
            pch = fgets(Buf, sizeof(Buf), pf);
        }
        fclose(pf);
    }

    for (i = 0; i < size; i++)
        Sum = Sum + Arr[i];
    Aver = Sum / size;
    pf = fopen("MyFile.txt", "at");
    fprintf(pf, "\nAverage is %f\n", Aver);
    fclose(pf);
    return 0;
}
```


Результати роботи програми (вміст файлу MyFile.txt після виконання програми):

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

Average is 5.500000

Лістинг 9.2: Обчислення середнього арифметичного елементів масиву. Вхідні дані беруться з файлу MyFileIn.bin. Результат зберігається у файлі MyFileOut.bin. Є дві допоміжні програми – одна формує файл MyFileIn.bin, інша читає результат з файлу MyFileOut.bin і виводить його на екран.

```
// програма для формування файлу з вхідними даними MyFileIn.bin
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    FILE* pf;
    int size, i;
    int Arr[10];
    size = 10;
    for (i = 0; i < size; i++)
    {
        printf("Enter Arr[%d]:", i);
        scanf("%d", &Arr[i]);
    }
    pf = fopen("MyFileIn.bin", "wb");
    if (pf != 0)
    {
        fwrite(Arr, sizeof(int), size, pf);
        fclose(pf);
    }
    return 0;
}
```

```

// основна програма
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    FILE* pf;
    int size, i;
    float Sum = 0, Aver;
    int Arr[10];
    size = 10;
    pf = fopen("MyFileIn.bin", "rb");
    if (pf != 0)
    {
        fread(Arr, sizeof(int), size, pf);
        fclose(pf);
    }
    for (i = 0; i < size; i++)
        Sum = Sum + Arr[i];
    Aver = Sum / size;
    pf = fopen("MyFileOut.bin", "wb");
    if (pf != 0)
    {
        fwrite(&Aver, sizeof(float), 1, pf);
        fclose(pf);
    }
    return 0;
}

```

```

// програма для читання результату з файлу MyFileOut.bin і виводу його
на екран
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    FILE* pf;
    float Aver;
    pf = fopen("MyFileOut.bin", "rb");
    if (pf != 0)
    {
        fread(&Aver, sizeof(float), 1, pf);
        fclose(pf);
    }
    printf("\nAverage is %f\n", Aver);
    return 0;
}

```

РЕКОМЕНДОВАНІ ДЖЕРЕЛА

1. Програмування, частина 1 (Основи алгоритмізації та програмування) [Електронний ресурс] – Режим доступу до ресурсу: <http://vns.lpnu.ua/course/view.php?id=1545>.
2. C++ language documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-gb/cpp/cpp/?view=vs-2019>.
3. Шпак З. Я. Програмування мовою С / Зореслава Ярославівна Шпак. – Львів: Оріяна-Нова, 2006. – 432 с.
4. Ковалюк Т. В. Алгоритмізація та програмування. Підручник / Тетяна Володимирівна Ковалюк. – Львів: Магнолія 2006, 2013. – 400 с.
5. КРЕНЕВИЧ А. П. С у задачах і прикладах : Навчальний посібник із дисципліни "Інформатика та програмування / А. П. КРЕНЕВИЧ, О. В. ОБВІНЦЕВ. – Київ: Видавничо-поліграфічний центр "Київський університет", 2011. – 200 с.
6. Керниган Б. Язык программирования С / Б. Керниган, Д. Ритчи. – Москва: Вильямс, 2015. – 304 с.

НАВЧАЛЬНЕ ВИДАННЯ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

з дисципліни

ПРОГРАМУВАННЯ, ЧАСТИНА 1 (ОСНОВИ АЛГОРИТМІЗАЦІЇ І ПРОГРАМУВАННЯ)

для студентів галузі знань «12 Інформаційні технології»
спеціальності 123 «Комп'ютерна інженерія»

Автор:

Ногаль Марія Василівна

Редактор

Комп'ютерне верстання