УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ
І ТЕХНОЛОГІЙ

Кафедра обчислювальної техніки та систем управління

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних і самостійної робіт із дисципліни
«*КОМП'ЮТЕРНА ТЕХНІКА ТА ПРОГРАМУВАННЯ*»

Частина 1

Харків 2024

Методичні вказівки розглянуто і рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 29 травня 2024 р., протокол № 9.

Методичні вказівки розроблено відповідно до освітніх програм «Організація перевезень і управління на транспорті», «Митний контроль на транспорті (залізничний транспорт)», «Організація міжнародних перевезень», «Транспортний сервіс та логістика», «Організація правової та експедиторської діяльності», «Безпека та охорона праці на залізничному транспорті».

Метою виконання лабораторних і самостійної робіт є ознайомлення з інтегрованим середовищем MSVC, відпрацювання основних прийомів програмування базових обчислювальних процесів; розв’язання задач, що мають лінійну, розгалужену або циклічну структуру і структурування даних.

Рекомендовано для здобувачів вищої освіти першого (бакалаврського) рівня спеціальностей 275.02 «Транспортні технології (на залізничному транспорті)» і 263 «Цивільна безпека» факультету управління процесами перевезень заочної форми навчання.

Рецензенти:

доц. В. О. Сотник (УкрДУЗТ),
О. Г. Панченко (Onapsis Inc., Берлін, Німеччина)

ЗМІСТ

ВСТУП

Сучасний розвиток обчислювальної техніки не має кордонів. ЕОМ застосовують у всіх сферах життя, а це означає, що навички роботи з обчислювальною технікою необхідні кожному інженерові.

Сьогодні ЕОМ є приємним співрозмовником, підказує, нагадує. Діалог між ЕОМ і людиною ведеться мовою, яка допускає однозначне тлумачення з обох сторін. Мов та інструментальних систем програмування для реалізації задач багато. Найбільш популярною та широко використовуваною довгі роки є С++, що поєднує елементи мов високого рівня з функціональністю Асемблера. При використанні мови С++ користувачі мають у своєму розпорядженні швидкий і ефективний компілятор та інтегроване середовище, що дає змогу максимально автоматизувати процес розроблення програм.

У методичних вказівках розглянуто елементарні конструкції мови С++, основні оператори та функції для програмування лінійних, розгалужених, циклічних обчислювальних процесів і правила опрацювання масивів. Завдання до самостійної роботи знаходяться після розібраних типових тестових прикладів.

У методичних вказівках у записах форматів застосовують такі позначення:

<*значення*> – обов’язковий елемент, на місці якого має бути послідовність символів, допущена в такому форматі;

[рядок_символів] – рядок сиволів у квадратних дужках вважають необов’язковим і може бути відсутній.

Порядок виконання лабораторної роботи:

1) вивчити теоретичний матеріал перед виконанням лабораторної роботи. Перед кожним тестовим прикладом розміщено теоретичний матеріал до його виконання;

2) виконати тестові завдання. У кожному тестовому прикладі є умова, схема алгоритму, программа і результат її виконання. Тексти программ забезпечені коментарями, які пояснюють, як повинен відпрацювати кожен оператор. Завдання здобувача вищої освіти: розібрати тестовий приклад, ввести, налагодити і протестувати роботу програми на основі результатів тестового прикладу. Отримані результати розмістити у звіті;

3) протестувати тестове завдання, увівши власні дані. Отримані результати розмістити у звіті;

4) розв'язати задачу для самостійної роботи. Відповідно до завдання самостійної роботи розроботи схему алгоритму і програму. Ввести, налагодити і протестувати роботу програми, увівши власні дані. Отримані результати розмістити у звіті;

5) файли зі звітами подати викладачеві для перевірки.

Приклад завантаження MSVC подано в додатку А, приклад оформлення завдання до звіту – додатку Б, відомості з мови С++ – додатку В.

ЛАБОРАТОРНА РОБОТА 1. Проєктування в інтегрованому середовищі MSVC додатків, що містять структури слідування, галуження та циклів

Мета: *набування навичок створення і налагодження програм в інтегрованому середовищі MSVC із алгоритмами лінійної, галуженої та циклічної структури.*

1.1 Форматне і потокове введення-виведення даних. Розрахунки з використанням функцій

1.1.1 Теоретичний матеріал «Структури слідування»

Структура програми [3, 5, 7, 10, 14, 20]. Мова С++ побудована на концепції складених блоків, що називають функціями, які являють собою програму, що складається і реалізує певну задачу. Основною функцією будь-якої програми є функція main().

Формат функції:

<тип_значення,_що_повертається> <ім'я_функції> (<параметри>)
{ <оголошення змінних>;
<тіло програми>; }.

Розглянемо кожну складову формату.

1 <Тип_значення,_що_повертається> – функція завжди повертає значення, тип якого вказаний перед ім'ям функції (таблиця 1.1).

Таблиця 1.1 – Діапазони значень і типи даних

Тип змінної	**Діапазон значень**	**Кількість байтів**
*char (*символи*)*	від 0 до 255	1
*int (*цілі числа*)*	від -32768 до 32767	2
*float (*числа з рухомою крапкою*)*	від 3.4e-38 до 3.4e+38 ($3.4*10^{+38}$)	4
*double (*числа з рухомою крапкою подвійної точності*)*	від 1.7e-308 до 1.7e+308	8
*void (*змінні без значення*)*	значень не має	

Наприклад:
int i, j, l; // змінні цілого типу
double less, roz; // числа з рухомою крапкою подвійної точності.

2 <Ім'я_функції> – ідентифікатор, який використовують для виклику функції для відпрацювання.

Ідентифікатори – імена, що використовують для звернення до змінних, міток, функцій та інших визначених користувачем об'єктів і можуть містити від одного до декількох символів.

Вимоги до ідентифікаторів:

– перший символ має бути літерою англійської абетки чи символом підкреслення;

– не повинен містити спеціальних символів;

– значущими символами є тільки перші 32. Це означає, що якщо ідентіфикатори двох змінних мають однакові перші 32 символи, то розрізняють тільки з 33-го символу, С++ ці змінні не розрізняє;

– у мові С++ прописні та рядкові літери трактовані як різні;

– ідентифікатор змінної не має співпадати з ключовим словом (додаток В).

Наприклад:
i01, _01, i_01, // правильно записаний ідентифікатор
01i, 01_, 01_i, i 01, sin // неправильно записаний ідентифікатор
count, Count і COUNT // три різні ідентифікатори.

3 <Параметри> – це змінні величини, які передають функції для опрацювання. Слід зазначити, що при описі функції в дужках задано *формальні параметри*, а при виклику функції – *фактичні параметри.* Назви формальних і фактичних параметрів можуть не співпадати.

4 <Оголошення змінних> є обов'язковим перед використанням.

Формат оператора оголошення змінних:

<тип> <список_змінних>;

де <тип> – будь-який допустимий тип змінних (таблиця 1.1);
<список_змінних> – перелік ідентифікаторів, розділених комами.

Змінні, що оголошені перед функцією main() і можуть бути використані всіма функціями, називають *глобальними*. Змінні, оголошені та використовувані всередині функції, називають *локальними*.

Константами вважають величини, що не змінюються у процесі виконання програми (таблиця 1.2).

Таблиця 1.2 – Основні види сталих величин

Тип даних	Вид константи
char	*'a', '\n', '9'*
int	*1, 123, 21000*
float	*12.23, 4.34E-3*
double	*12345212, 1.0E100*

Рядкові константи – набір символів у подвійних апострофах, наприклад "це текст". Не плутайте рядкові константи з символьними: перші укладені в подвійні апострофи, інші – в одинарні.

Символьні константи зі зворотним слешем. Є символи, що не можна ввести з клавіатури, наприклад Enter. Для таких символів можна використовувати код таблиці символів ASCII або спеціальні символьні константи зі зворотним слешем:

\f – встановлення на нову сторінку;

\n – встановлення на новий рядок;

\r – повернення каретки;

\t – горизонтальна табуляція;

\v – вертикальна табуляція.

Ініціалізація змінних. Змінним можна надавати значення під час їх оголошення. Загальний формат ініціалізації:

<тип> <ім'я_змінної> = <константа>.

Наприклад: char ch = 'a';
int first = 0.

5 <Тіло програми> – перелік виконуваних операторів. Кожен *оператор закінчується крапкою з комою* (наприклад x = y; y = y+1;), *група* логічно пов'язаних *операторів*, включена в інший оператор, *об'єднана фігурними дужками* (наприклад if(x == y) {x = y; y = y+1;}).

Коментарі – пояснення до тексту програми, які знаходяться всередині коду. Також при налагодженні програми за допомогою коментаря можна відключити частину коду. Коментарі бувають *рядкові* (наприклад // x = y; y = y+1; компілятор ігнорує обидва оператори) і *блокові* (наприклад /* x = y; */ y = y+1; компілятор ігнорує тільки перший оператор).

Зазвичай функція описана до того, як вона буде визначена. Опис (прототип) інформує компілятор про існування функції, тип значення, що повертається, а також тип параметрів, що їй передають.

Загальний формат:

<тип_значення,_що_повертається> <ім'я_функції> (<параметри>).

Препроцесор – це програма, що обробляє вхідний модуль до того, як він пройде через компілятор, заміняючи визначені імена еквівалентними їм

рядками. *Компілятор* – це програма, яка перетворює вхідний модуль (програма мовою програмування) в об’єктний (машинний) код. *Інтерпретатор* – це програма, яка перетворює вхідний модуль в об’єктний порядково.

Рядки, що містять директиви препроцесора, розпочинаються з символу номера *(#).* Директива *#include* підставляє замість себе текст, зазначений у директиві файлу. Вхідний файл, що буде читатися, має бути укладений у подвійні лапки або гострі дужки.

Наприклад *#include "stdio.h" або #include <stdio.h>*.

Якщо ім'я файлу укладено в лапки, то компілятор спочатку буде шукати ім'я файлу в поточному робочому каталозі і, якщо компілятор не знаходить файл, шукати ім'я файлу в будь-якому каталозі, який специфіковано в командному рядку.

Укладення ім'я файлу в гострі дужки каже про те, що пошук цього файлу буде здійснюватися у специфікованому каталозі. Якщо компілятор не знаходить файл, то пошук буде виконуватися у стандартному каталозі.

У мові С++ файли з поширенням *.h* називають файлами-заголовками (Header File). Вони містять опис змінних, функцій, типів, використовуваних багатьма програмами. У такому випадку у файлі *stdio.h* міститься опис, необхідний для використання стандартної бібліотеки введення/виведення. Ім'я файлу утворено від скорочування *Standard Input/Output*. Цей файл буде включатися перед усіма програмами, де є введення або виведення.

Директиву препроцесора *#define* застосовують для того, щоб визначити ідентифікатор і ланцюжок, який компілятор буде підставляти замість ідентифікатора кожен раз, коли він зустрічається у вхідному файлі. Ідентифікатор називають макроіменем, а процес підстановки називають макропідстановкою. Загальний формат цієї директиви:

#define <ідентіфикатор> <ланцюжок>.

Наприклад, для використання *True* у вигляді значення *1* і *False* як значення *0* необхідно оголосити два *#define*:

#define True 1

#define False 0.

Ці рядки змушують компілятор підставляти *1* або *0* кожен раз, коли він зустрічає у вхідному файлі *True* або *False*.

Наприклад, функція *printf()* надрукує на екрані *0 1 2*:

printf(" %d %d %d", False, True, True+1).

Після того як макроім'я визначено, його можна використовувати у вигляді частини визначення інших макроімен. Наприклад,

#define One 1
#define Two One+One
#define Three One+Two.

Ведення/виведення

Форматне введення та виведення *(тестове завдання 1)*. Розглянемо функції введення/виведення, прототипи яких знаходяться у стандартній бібліотеці <*stdio.h*>.

Функція scanf() – універсальна *функція введення*, яка може читати всі типи даних і автоматично перетворює значення в необхідний внутрішній формат. Формат функції *scanf()*:

scanf ("<управляючий_рядок>", &<список_аргументів>),

де ***<управляючий_рядок>*** – визначає тип значення, що вводять, за допомогою специфікаторів формату (таблиця 1.3);

<список_аргументів> – список змінних, що отримують значення шляхом введення з клавіатури. Кількість і тип аргументів мають співпадати з кількістю і типом специфікаторів форматів в управляючому рядку;

& – усі змінні, які використовують для приймання значень, що мають бути передані їхніми адресами. Це означає, що всі аргументи мають бути покажчиками на змінні, які використовують як аргументи.

Таблиця 1.3 – Специфікатори формату функції *scanf()/printf():*

Код	Значення
%d	Читати/Виводити десяткове ціле
%f	Читати/Виводити число з рухомою крапкою
%c	Читати/Виводити окремий символ
%s	Читати/Виводити рядок символів
%o	Читати/Виводити вісімкове число
%x	Читати/Виводити шістнадцяткове число
%p	Читати/Виводити покажчик

Наприклад, зчитування трьох значень у три змінні відповідно

int i, j;
float k;
scanf("%d,%d,%f", &i, &j, &k).

Функція printf() – універсальна *функція форматного виведення*.

Формат функції printf():

printf("<управляючий_рядок>", <список_аргументів>),

де ***<управляючий_рядок>*** – окрім специфікаторів формату (таблиця 1.3) можуть містити символи, які виводять на екран;

<список_аргументів> – список змінних, значення яких отримують шляхом введення з клавіатури. Кількість і тип аргументів мають співпадати з кількістю і типом специфікаторів форматів в управляючому рядку.

Наприклад,

printf (" %s %d", "це рядок", 100); // виводить на екран: *це рядок 100*

printf ("це рядок %d", 100); // виводить на екран: *це рядок 100*

printf ("число %d - десяткове, %f - дробове.", 10, 110.789);

//виводить на екран: *число 10 - десяткове, 110.789 - дробове.*

Команди формату можуть мати модифікатори, які специфікують ширину поля, кількість десяткових розрядів і прапорець вирівнювання за першим лівим знаком чи розрядом. Ціле, розташоване між знаком % і командою формату, діє як специфікатор мінімальної ширини поля. Цей специфікатор доповнює виведення пропусками чи нулями, щоб забезпечити визначену мінімальну його довжину. Якщо ланцюжок символів чи число перевищує цей мінімум, то функція *printf()* буде друкувати їх повністю, навіть якщо вони виходять за межу цього мінімуму. За замовченням, для доповнення виведення використовують пропуски. Щоб доповнити виведення нулями, необхідно розташувати *0* перед специфікатором ширини поля. Наприклад, *%05d* буде доповнювати число, яке складається менш ніж із п’яти цифр, нулями перед значенням числа для того, щоб загальна довжина дорівнювала п'яти.

Щоб специфікувати кількість десяткових розрядів, які необхідно надрукувати в разі числа з десятковою крапкою, десяткову крапку розміщують після специфікатора ширини поля, за нею – кількість десяткових розрядів. Наприклад, *%10.4f* буде виводити число, загальне поле виведення якого *10* позицій, з них чотири позиції - дробова частина, одна позиція – десяткова крапка. Коли формат, подібний цьому, застосовують до ланцюжків символів чи цілих, число, що стоїть після крапки, специфікує максимальну довжину поля. Наприклад, *%5.7s* буде виводити ланцюжок символів довжиною не менше п'яти і не більше семи символів. Якщо ланцюжок більший, ніж максимальна ширина поля, то символи, що залишилися, будуть відсічені.

За замовченням, усе виведення вирівнюють праворуч: якщо ширина поля більше, ніж друковані дані, то такі дані розташовані на правому краю цього поля. Можна викликати вирівнювання інформації ліворуч, розташовуючи знак мінус безпосередньо після %. Наприклад, *%-10.2f* буде вирівнювати ліворуч число з рухомою крапкою з двома десятковими розрядами в десятипозиційному полі.

Приклади управляючих рядків для виведення даних наведено в таблиці 1.4.

Таблиця 1.4 – Приклади управляючих рядків для виведення даних

Функція printf()	Виведення
("%9.2f", 123.321) ("%-9.2f", 123.321) ("%5.7s", "123456789") ("%9d", 555) ("%09d", 555)	123.32 123.32 1234567 555 000000555

Потокове введення та виведення *(тестове завдання 2)*. *Потік* – механізм перетворення значень різного типу на послідовність символів (виведення) або, навпаки (введення), значення змінної. Напрямок даних у потік виведення заданий за допомогою оператора << (extractor).

Формат оператора cout – стандартний потік виведення (екран):

cout<<"x="<<x.

Введення даних із потоку здійснюється аналогічно з використанням зворотного оператора >> (inserter).

Формат оператора cin – стандартний потік введення (клавіатура)

cin>>x.

Розрахунки. Для розрахунків використовують арифметичні оператори (таблиця 1.5).

Таблиця 1.5 – Арифметичні оператори

Оператор	**Дія**	**Оператор**	**Дія**
+	Додавання	*	Множення
–	Віднімання та унарний мінус	/	Ділення
++	Додатний приріст	––	Від'ємний приріст
%	Цілочисельне ділення (дає остачу від ділення цілих чисел)		

Пріоритети арифметичних операцій:

1) ++, −− додатній та від'ємний приріст (збільшення зменшення на одиницю). Наприклад,

```
x=10; y=++x;      // y=11, тому що спочатку виконується приріст x,
                  // а після цього його значення присвоюється y
x=10; y=x++;      // y=11, тому що спочатку y присвоється значення 10,
                  // а після цього виконається збільшення значення x;
```

2) – (наприклад $y = -a;$);
3) *, /, %;
4) +, – .

Оператори одного пріоритету виконуються за правилами читання зліва направо. Для зміни пріоритету використовують дужки.

Якщо в арифметичному виразі зустрічаються операнди різних типів, то один з операндів має змінити свій тип так, щоб він відповідав типу іншого операнда за правилом

символьні < цілі < рухомі < подвійної точності.

В арифметичних операціях можуть використовувати стандартні математичні функції. У таблиці 1.6 наведені стандартні математичні функції та приклади їх використання. Для виклику функцій необхідно підключити бібліотеку <math.h> за допомогою директиви #include <math.h>.

Таблиця 1.6 – Стандартні математичні функції

Функція	***Опис***	***Приклад***
1	2	3
abs(a)	модуль або абсолютне значення **a**	abs(-3.0)=3.0abs(5.0)= 5.0
sqrt(a)	корінь квадратний з **a,** де **a** >=0	sqrt(9.0)=3.0
pow(a, b)	піднесення **a** до степеня **b**	pow(2,3)=8 pow(27,(1/3))=3
ceil(a)	округлення **a** до найменшого цілого, але не менш ніж **a**	ceil(2.3)=3.0 ceil(-2.3)=-2.0

Продовження таблиці 1.6

1	2	3
floor(a)	округлення **a** до найбільшого цілого, але не більш ніж a	floor(12.4)=12 floor(-2.9)=-3
fmod(a, b)	обчислення остачі від **a/b**	fmod(4.4, 7.5) = 4.4 fmod(7.5, 4.4) = 3.1
exp(a)	обчислення експоненти $\mathbf{e^a}$	exp(0)=1
sin(a)	**a** задано в радіанах	
cos(a)	**a** задано в радіанах	
log(a)	натуральний логарифм **a** (основою є експонента)	log(1.0)=0.0
log10(a)	десятковий логарифм **a**	Log10(10)=1
asin(a)	арксинус **a**, де **-1.0 < a < 1.0**	asin(1)=1.5708

Оператор присвоювання дає змогу надати значення певному ідентифікатору. *Формат оператора присвоювання*:

<ім'я_змінної> = <значення>;

Наприклад, *int aud; aud = 222.*

1.1.2 Тестові завдання 1-2 «Структури слідування»

Тестове завдання 1. Знаючи довжину кожної зі сторін **a, b, c** трикутника, знайти площу **s** і периметр **p**, використовуючи формулу Герона. Формула Герона [12] дає змогу обчислити площу трикутника **S** за його сторонами **a, b, c**:

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \tag{1.1}$$

де $p = \frac{a+b+c}{2}$ – півпериметр трикутника.

Порядок виконання завдання

1 етап *– постановка та формалізація завдання.* Формалізований запис умови завдання дає змогу використати безпосередньо в алгоритмі та програмі отримані залежності:

$$s = \sqrt{r(r-a)(r-b)(r-c)} \quad ;$$

$$r = \frac{a+b+c}{2} \quad ; \qquad p=2r. \qquad (1.2)$$

2 етап *– складання схеми алгоритму (*рисунок 1.1*).*

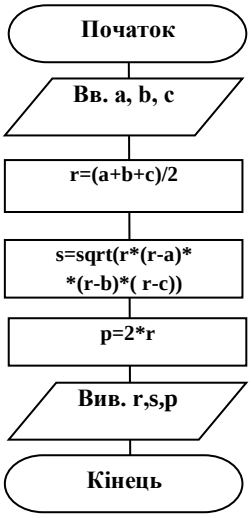


Рисунок 1.1

3 етап *– програмування (ввести і протестувати)*

```
#include <stdio.h>            // заголовний файл стандартної бібліотеки stdio.h
// – вставляє  прототипи функцій введення/виведення  scanf() і printf()
// (ім'я, тип  і параметри), які потрібні для компіляції, після якої буде отриманий
// фактичний їхній код, необхідний  для зв'язування та  виконання
#include <math.h>             // заголовний файл стандартної бібліотеки math.h
// для виконання математичних операцій
#include <stdlib.h>           //заголовний файл бібліотеки стандартного призначення;
// включає функції, які займаються розподілом пам'яті,
// управлінням процесами, перетвореннями  та ін.;
#include "windows.h"          // головний заголовний файл Windows API
// (API, Application Programming Interface) – це набір функцій,
// що працюють під управлінням ОС Windows, за допомогою яких
// можна створювати різні віконні процедури, діалогові вікна програми
char * Rus(const char * text)          //група операторів для виведення кирилиці
{char *buf = new char [strlen(text)]; //
CharToOem(text, buf);       //перетворює рядок з ANSI в OEM  параметри - покажчик на вихідний буфер
return buf;}                //

int main()                  //перший оператор – головна функція, яку викликає  виконуюча система
{float  a,b,c,r;            //опис дійсних змінних довжиною 4 байти
double s;                   //опис дійсної змінно, довжиною 8 байтів
printf("a=");               //виведення на екран символів 'a='
scanf("%f",&a);             //запис у змінну a введеного з  клавіатури значення
printf("b=");       scanf("%f",&b);
printf("c=");       scanf("%f",&c);
r=(a+b+c)/2;                          //обчислення півпериметра
s=sqrt(r*(r-a)*(r-b)*(r-c));          //обчислення площі
printf("s=%5.2f \t",s);               // символи,  виштовхувані в потік виведення  для  форматування та
// відображення на екрані виведення  символів 's=', значення s і символу управління \t– горизонтальна  // табуляція –
для вирівнювання тексту по горизонталі  (4 або 8 пробілів )
printf(Rus("периметр "));
printf(" p=%5.2f \n",2*r);            // виведення  символів ' p=', значення виразу 2*r і
// символу  управління \n – переведення  каретки на новий рядок
return 0;}                  // повернення  ОС як сигналу про успішне закінчення програми числа 0.
```

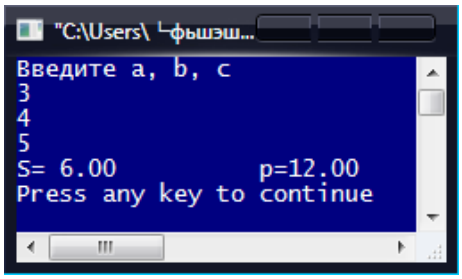


Рисунок 1.2

4 етап *– тестування додатка.*

5 етап *– аналіз результатів.*

У завданні не передбачено перевіряти коректність вихідних даних (від'ємні чи нульові значення сторін трикутника, виконання умови існування трикутника –

сума двох сторін більше третьої і т. ін.). Тому потрібно вводити коректні дані. Для такого випадку одержано правильний результат (рисунок 1.2).

Тестове завдання 2. Задано щільність **ρ**, висоту **h** та радіус основи **R** циліндричного зливка, отриманого в металургійний лабораторії. Знайти об'єм **V**, масу **m** і площу **S** основи зливка [16].

Порядок виконання завдання

1 етап – *постановка та формалізація завдання*. Вихідні дані: **ρ, h, R.** Розрахункові формули для результатів:

$$V=\pi R^2 h,\ m=\rho V,\ S=2\pi R. \qquad (1.3)$$

2 етап – *складання схеми алгоритму (рисунок 1.3).*

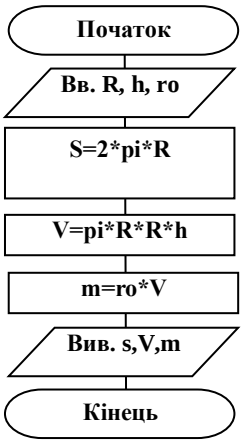


Рисунок 1.3

3 етап – *програмування (ввести і протестувати)*

```
#include <stdio.h>
#include <iostream>          //опис об'єктів для управління введенням/виведенням утримується в заголовному
//файлі iostream при його підключенні в програмі автоматично створюються об'єкти-потоки cin
//(для введення з клавіатури) і cout (для виведення на екран),
//а також операції " << " - поміщення в потік і " >> " - зчитування з потоку.
using namespace std;         // використовують для виключення конфліктів змінних, що повторюються
#define pi 3.14159           //визначення констант
int main()  {
double R,h,ro,S,V,m;
cout<<"R=";                  //виведення на екран символу 'R='  cout  та операція ' << ' дають змогу
//вивести на екран значення будь-якої змінної, текст у подвійних лапках або спеціальні символи, //наприклад, cout<<i;
- виводить на екран значення змінної i
//cout<<x<<"\t"<<y;  - виводить на екран значення змінних x і y, розподілені пробілами
cin>>R;                      //введення значення змінної R – cin та операція ' >> ' дають змогу присвоїти значення
          //будь-якій змінній, наприклад, cin>> i;  -  змінна i буде набувати цілого значення, введеного з клавіатури,
//якщо сама i ціла cin>>x>>y>>z; (вводить декілька змінних)
cout<<"h=";
cin>>h;
cout<<"ro=";                 //наприклад, щільність заліза 7,874 г/см³
cin>>ro;
S=2*pi*R;                    //обчислення площі
V=pi*R*R*h;                  //обчислення об'єму
m=ro*V;                      //обчислення маси
cout<<"S="<<S;               //виведення на екран символів 'S=' і значення змінної S
cout<<"\n V="<<V;            //виведення на екран з нового рядка символів 'V=' та значення змінної V
cout<<"\n m="<<m;            //виведення на екран з нового рядка символів 'm=' та значення змінної m
return 0;    }
```

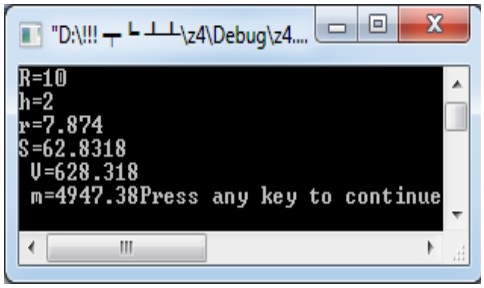


Рисунок 1.4

4 етап *– тестування додатка.*

5 етап *– аналіз результатів (рисунок 1.4).* Помилок у процесі тестування не виникло.

1.1.3 Варіанти індивідуальних завдань[1] до самостійної роботи «Структури слідування»

Варіанти індивідуальних завдань до самостійної роботи «Структури слідування» [13] наведені в таблицях 1.7, 1.8.

Таблиця 1.7 – Завдання 1 «Структури слідування»

Варіант	Завдання	**Варіант**	Завдання
1	$r=\frac{mc+\sqrt{2x}+f^2}{\sin^2 x+3c}$	9	$f=\frac{a}{x^2+b}+\frac{m}{2c}$
2	$q=\frac{2x+\ln^2 x+\sqrt{x}}{a+mc+t};$	10	$z=\frac{a+bx}{m}+\frac{c}{a+x}-q;$
3	$t=\frac{mb}{a+x}$	11	$f=\frac{3y}{m+x}+\sqrt{\frac{b}{m+2x}}+\sin\left(m^2+x\right);$
4	$t=\frac{2s+m^2}{a-x}+\sqrt{s+c^3}$	12	$q=\frac{b+cx}{2}+\frac{1}{a-cx};$
5	$c=\frac{a+\sin^2 x}{m-r}+\frac{cx}{a+2x}$	13	$c=\frac{a+\sin 2x}{m-r}+\frac{\cos^2 x}{ax}$
6	$s=\frac{a+2c}{m}+q;$	14	$m=\frac{b+ax}{2}+\frac{3f}{b-x}$
7	$y=\frac{cx^2+mb^3+\ln x+7}{b+m+\sqrt{c}}$	15	$m=\frac{c+2{,}5d}{a+x};$
8	$m=\frac{2x}{s+\frac{2}{d}}+\sqrt{x}$		

[1] Значення незалежних змінних, що не вказані в умовах задач, вважати довільними.

Таблиця 1.8 – Завдання 2 «Структури слідування»

Варіант	Формули для розрахунків	Вивести на екран
1	$q=\frac{2x+\ln^2 x+\sqrt{x}}{a+mc+t};\ t=\frac{mb}{a+x};\quad m=a+2b^2,\ якщо\ b=4.3$	***q, t, m***
2	$t=\frac{2s+m^2}{a-x}+\sqrt{s+c^3}-x^2;\ s=\frac{a+2c}{m}+q;\quad m=2\sqrt{a^2+c\quad},\ якщо\ a=5{,}6$	***t, s, m***
3	$y=\frac{cx^2+mb^3+\ln x+7}{b+m+\sqrt{c}};\ m=2x+\sqrt{x};\ b=ac^2-m$	***y, m, b***
4	$r=\frac{mc+\sqrt{2x}+f^2}{\sin^2 x+3c};\ f=\frac{a}{x^2+b}+\frac{m}{2c}-x;\ m=b+\cos(a-x)\quad ,якщо\ \ b=0{,}27$	***r, f, m***
5	$z=\frac{a+bx}{m}+\frac{c}{a+x}-q;\ q=\frac{b+cx}{2}+\frac{1}{a-cx};a=bx-mt^2,якщоt=0{,}5$	***z, q, a***
6	$c=\frac{a+\sin 2x}{m-r}+\frac{\cos^2 x}{ax};\ r=\frac{2m}{x}+arctg2x;\ m=\frac{b+ax}{2}+\frac{3f}{b-x}$	***c, r, m***
7	$f=\frac{3y}{m+x}+\sqrt{\frac{b}{m+2x}}+\sin(m^2+x);\ m=2a-cd;\ c=a+x^2$	***f, m, c***
8	$r=\sqrt{t+2q};\quad q=\frac{m+x}{\ln x};\quad m=\frac{c+2{,}5d}{a+x};\quad c=\sqrt{x^2+a}$	***r, m, q***
9	$s=\frac{7}{m+2t}+\frac{a}{b+cx}+m^2-4z;\quad z=3m+at^2;m=\sqrt{a+2cx^2},якщо\quad b=3{,}25$	***s, z, m***
10	$z=\frac{2x}{b+y}+\frac{c}{b-y}+\frac{2bx}{b-xy};\quad y=2\ln(\lvert x-a\rvert);\quad b=c-2x,\quad якщо\quad c=12$	***z, y, b***
11	$q=mc^2+2f^5+\cos^2 x;\quad m=\frac{a}{c}+\frac{b}{x}+\sqrt{f};\quad f=\ln(a+2{,}5x),\quad якщо\quad a=3{,}7;$	***q, m, f***
12	$z=\frac{m^2cx}{\sqrt{a}}+bc^3-\ln a;\quad b=a+\sin 2x^2;\ m=a+b^2-\cos x,\ якщо\ a=2$	***z, b, m***
13	$r=\frac{c+2x^2}{m}+at^2+\sqrt{x};\quad c=b+2x-\sin x;\quad m=\frac{2+x}{c}+arctgx-a$	***r, c, m***
14	$q=\frac{3m}{ax}+\sqrt{b+cx}+\lg(\lvert b-mx\rvert);\quad m=a+r^3+\cos x;\ \ r=c+2x;c=2a-x$	***q, m, r***
15	$q=\frac{at^2+rc-ax}{b+\sqrt{c+x}-5};r=(mx+2t+c)^3;m=2cx;\quad t=ax^2+bx-c,якщо\quad c=7{,}5$	***q, r, m, t***

1.2 Використання оператора if, умовної тернарної операції та оператора switch

1.2.1 Теоретичний матеріал «Структури галуження»

Оператори порівняння та *логічні оператори*. Оператори порівняння дають змогу визначити співвідношення двох величин. Логичні оператори встановлюють взаємозв'язок цих співвідношень [3, 5, 7, 14, 15, 20].

В основі роботи логічних операторів і операторів порівняння лежать поняття «істинно» та «хибно» (таблиця 1.9). Істинним вважають будь-який вираз, відмінний від нуля, нульове значення – «хибно».

Таблиця 1.9 – Логічні оператори та оператори порівняння

Оператори порівняння	
Оператор	**Дія**
>	більше ніж
>=	більше або дорівнює
<	менше ніж
<=	менше або дорівнює
==	Дорівнює
!=	не дорівнює

Логічні оператори	
Оператор	**Дія**
&&	AND
||	OR
!	NOT

Усі оператори порівняння і логічні оператори мають пріоритет нижче, ніж арифметичні оператори.

Пріоритети логічних операторів і операторів порівняння:

1) !; 2) >, >=, <, <=; 3) ==, !=; 4) &&; 5) ||.

Як і в арифметичних виразах, порядок виконання цих операторів можна змінювати за допомогою дужок.

Оператор умовного передавання управління if призначений для будь-якого типу галуження (тестове завдання 3). Формат оператора if:

if (<умова>) <оператор1>;

[else <оператор2>],

де ***<умова>*** – аналізують умовний вираз, при отриманні ненульового значення «істина» виконується оператор1, при отриманні нульового значення «хибність» виконується оператор2. Замість оператора може стояти група операторів.

Якщо замість оператора1 або оператора2 стоїть інший оператор if, то такий оператор називають ускладненим. Складність у тому, що важко зразу сказати, якому *if* відповідає який *else*.

Наприклад,

if (x) if (y) printf (" 1 "); else printf (" 2 "); // else пов'язано з if (y)

if (x) {if (y) printf (" 1 ");} else printf (" 2 "); // else пов'язано з if (x).

Щоб запам'ятати, є просте правило: *else* пов'язане з ближчим *if*, що не має парного *else*. І *if,* і *else* мають знаходитися всередині одного блока.

Умовна тернарна операція (тестове завдання 4) є аналогом оператора if. Формат умовної тернарної операції:

(<умова>? <оператор1> : <оператор2>).

Наприклад,

(*x<1****? printf (" 1 ");> :)? printf (" 2 ");).***

Оператор вибору switch призначений для вибору на основі значення змінної відповідного оператора або групи операторів (тестове завдання 5). Формат оператора switch:

switch (змінна)

{case значення1:

{<оператор1> break;}

case значення2:

{<оператор2> break;}

…

default:

{<оператор3> },

де ***< змінна> –*** аналізована змінна;

case значення1: – якщо змінна набуває значення 1, тоді виконується група ***{***оператор1 і break***}***;

break – оператор переривання закінчення оператора switch;

default: – якщо змінна не набуває жодного значення, виконується оператор3 (аналог else).

1.2.2 Тестові завдання 3-5 «Структури галуження»

Тестове завдання 3. Розробити проєкт, у якому обчислюють значення ***y*** залежно від виконання умови за формулою ***y=x***, якщо ***x<0***, або формулою ***y=0***, якщо ***0<=x<30***, або формулою ***y=x²*** , якщо ***x>=30.*** Використовувати оператори **if** [13].

Порядок виконання завдання

1 етап – *постановка та формалізація завдання.* Складемо математичну модель. Формальний запис доцільно надати в такому вигляді:

$$y = \begin{cases} x & якщо \quad x < 0 \\ 0 & якщо \quad 0 <= x < 30 \\ x^2 & ,в\ інших\ випадках \end{cases} . \qquad (1.4)$$

2 етап – *складання схеми алгоритму (рисунок 1.5).*

3 етап *– програмування.*

```
#include <stdio.h>
#include <iostream>
using namespace std;

int main()
{int x, y;
cout << "Vvedite x: ";
cin >> x;

if (x < 0)      y = x; //виконується, якщо x < 0
else if ( (x >= 0) && (x < 30) )  y = 0;
            // виконується, якщо 0 <= x < 30
else    y = x * x;      //виконується в інших випадках
            //  ( якщо x більше або дорівнює 30)
cout << "y=" << y << endl;
// маніпулятор формату  endl
// виводить символ переходу на новий рядок
// та очищує буфер потоку
return 0;}
```

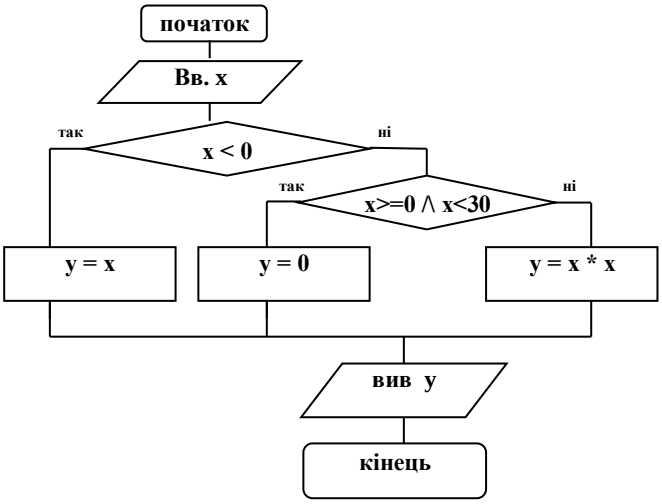


Рисунок 1.5

4 етап *– тестування додатка (рисунок 1.6).*

5 етап *– аналіз результатів.* Синтаксичних і семантичних помилок у процесі тестування не виникло. Відсутність логічних помилок доведена на трьох тестових наборах, що дало змогу перевірити кожну з трьох можливих гілок обчислювального процесу.

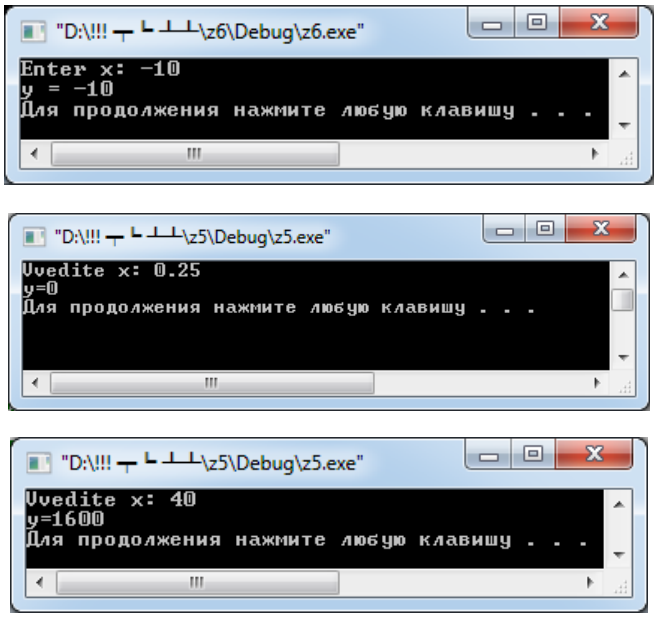


Рисунок 1.6

Тестове завдання 4. Вирішити ***завдання 3*** із застосуванням ***умовної тернарної операції.***

Порядок виконання завдання

1 *та* ***2 етапи*** *- як у завданні 3.*

3 етап *– програмування.*

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main()
{int x;
cout << "Enter x: ";
cin >> x;                                                    // вводимо x
cout << "y = " << (x < 0 ? x : (x >= 0) && (x < 30) ? 0 : x * x  ) << endl;       // дві тернарні операції;
//тернарна умовна операція має три аргументи і повертає свій другий або третій операнд
// залежно від значення логічного виразу, заданого першим  операндом
return 0; }
```

4 *та* ***5 етапи*** *- як у завданні 3.*

Тестове завдання 5. Розробити проєкт, у якому реалізовано додавання, віднімання, множення та ділення двох чисел, що вводять із клавіатури з застосуванням оператора **switch.**

Порядок виконання завдання

1 етап *– постановка та формалізація завдання.* Складемо модель. Формальний запис можна надати в такому вигляді:

$$операція = \begin{cases} a+b & якщо \quad flag = 1 \\ a-b & якщо \quad flag = 2 \\ a*b & якщо \quad flag = 3 \\ a/b & якщо \quad flag = 4 \end{cases}. \qquad (1.5)$$

2 етап *– складання схеми алгоритму (рисунок 1.7).*

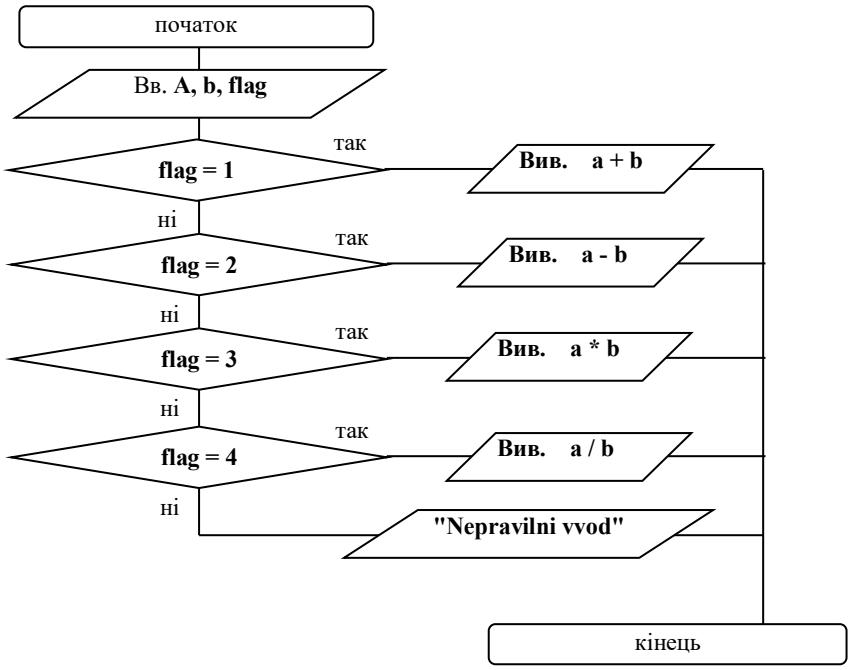


Рисунок 1.7

3 етап *– програмування.*

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main()
{int flag;                                          // змінна для вибору в switch
double a,b;                                         // змінні для збереження операндів
  cout << "Vvedite pervoe chislo: ";
  cin >> a;
  cout << "Vvedite vtoroe chislo: ";
  cin >> b;
  cout << "Vibirite deistvie: 1-clojenie; 2-vichitanie; 3-ymnojenie; 4-delenie: ";
  cin >> flag;
switch (flag)                                       // початок  оператора switch
{case 1:                                            // якщо flag = 1
 {cout << a << " + " << b << " = " << a + b << endl;   break;} // виконати додавання та закінчити
вибір
case 2:                                             // якщо flag = 2
 {cout << a << " - " << b << " = " << a - b << endl;   break; } // виконати віднімання
case 3:                                             // якщо flag = 3
 {cout << a << " * " << b << " = " << a * b << endl;   break;     }       // виконати множення
case 4:                                             // якщо flag = 4
 {cout << a << " / " << b << " = " << a / b << endl;   break;     }       // виконати ділення
default:                                            // якщо flag дорівнює будь-якому іншому значенню
 cout << "Nepravilni vvod" << endl; }
return 0;}
```

4 етап *– тестування додатка (рисунок 1.8).*

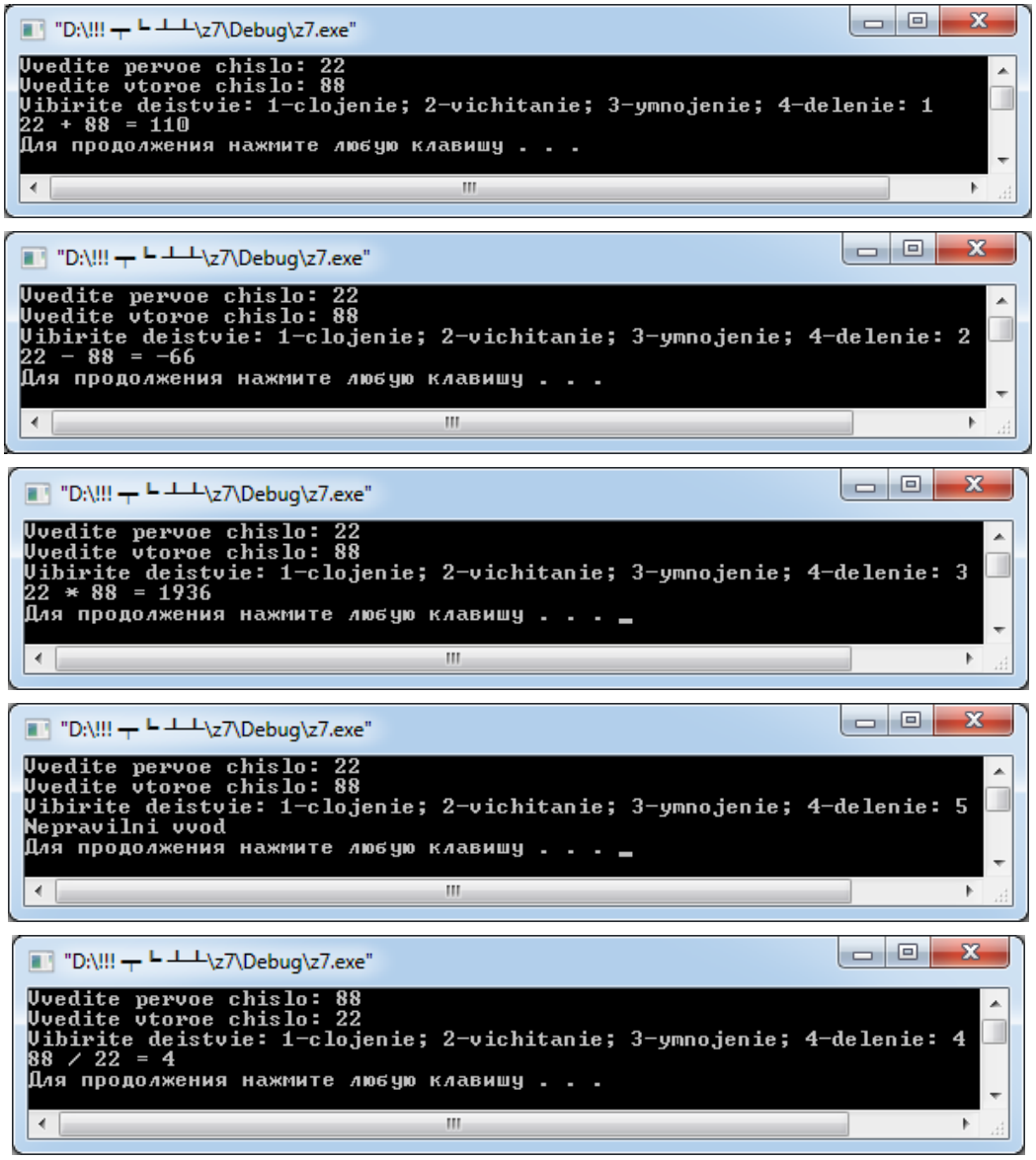


Рисунок 1.8

5 етап *– аналіз результатів.* Синтаксичних і семантичних помилок у процесі тестування не виникло. Відсутність програмних помилок доведена на п’яти тестових наборах, що дало змогу перевірити кожну з гілок обчислювального процесу. Відсутність break; призвела до виконання всіх наступних case; і default.

1.2.3 Завдання до самостійної роботи «Структури галуження»

Варіанти індивідуальних завдань до самостійної роботи «Структури галуження» [13] наведені в таблиці 1.10.

Таблиця 1.10 – Завдання 1 «Структури галуження»

Варіант	Формула для розрахунків
1	2
1	$Q=\begin{cases}2a-r^{x};\text{якщо } 0<x<5;\text{та } r\neq 0\\ \sqrt{x+r};\text{якщо } 5\le x<9;\text{та } a=0\\ a+x+r;\text{в інших випадках.}\end{cases}$ $S=\begin{cases}2Q,\text{якщо } Q>12{,}6\\ 3Q,\text{якщо } Q\le 12{,}6\end{cases}$
2	$Y=\begin{cases}x^{2}+a,\text{якщо } 0<x<1{,}5\\ 2\ln x-a,\text{якщо } 1{,}5\le x<3\\ \sqrt{a+x},\text{якщо } 3\le x<5\\ 12x^{2},\text{в інших випадках.}\end{cases}$ $Z=\begin{cases}3Y,\text{якщо } Y\le 3{,}8\\ 7Y,\text{якщо } Y>3{,}8\end{cases}$
3	$S=\begin{cases}2x^{m}-\ln c^{2},\text{якщо } x^{m}=396,\text{та } c>5\\ x^{5}-\lg m,\text{якщо } x^{m}=458 \text{ або } c=3\\ c^{5}+\sqrt{x},\text{якщо } x^{m}>458\\ b+3mx+x^{m},\text{в інших випадках}\end{cases}$ $Q=\ln\frac{S}{m}+x^{m+1}$ $Z=5.2S+\lg Q-2x$
4	$T=\begin{cases}x^{2}-am^{2},\text{якщо } c>x>a \text{ та } b>5\\ cx^{5}+2b,\text{якщо } x\le a \quad \text{та} \quad x\neq c\\ \sqrt{b+cx^{2}},\text{якщо } x=c\\ 0,\text{в інших випадках}\end{cases}$ $Z=3{,}5T$ $Q=\begin{cases}Z+2T,\text{якщо} \quad Z\le 29\\ \ln(Z+T),\text{якщо} \quad Z>29\end{cases}$

Продовження таблиці 1.10

1	2
5	$R=\begin{cases} a+2x, \text{якщо } x<0 \quad та \quad a>2 \\ b-cx, \text{якщо } x>0 \quad або \quad a=0 \quad або \quad x\neq 4 \\ a^2-2\sqrt{x}, \text{якщо } x=4 \quad та \quad c=7 \\ \sqrt{b+x}, \text{в} \quad інших\ випадках \end{cases}$ $S=\begin{cases} 2R+a^x, \text{якщо } R\geq 7.2 \\ 3R-x, \text{якщо } R<7.2 \end{cases}$
6	$Y=\begin{cases} x+2a^m, \text{якщо } a+m=4.6 \text{ та } x\geq 7 \\ \sqrt{x}+3a, \text{якщо } a+m\geq 5{,}2 \\ \ln\left(a+x^m\right), \text{в} \quad інших\ випадках \end{cases}$ $Z=\begin{cases} 2Y, \quad \text{якщо } Y<16{,}5 \\ 5Y-c, \text{якщо } Y\geq 16{,}5 \end{cases}$ $Q=a+2Y-3Z$
7	$Y=\begin{cases} cx^3+a, \text{якщо } x=5 \text{ та } b+c<12 \\ \ln(x-a), \text{якщо } c\leq x<b \quad та \quad x\neq 5 \\ b+3c, \text{в} \quad інших\ випадках \end{cases}$ $R=\begin{cases} 3Y, \text{якщо } Y>15 \\ a-Y, \text{якщо } Y=15 \\ b+2x, \text{якщо } Y<15 \end{cases}$
8	$Z=\begin{cases} ax^3+b\ln 2x^2, \text{якщо } x>a \quad та \quad b=7 \\ c\sqrt{\lvert b+cx^2-a\rvert}, \text{якщо } x<a \\ \lg\left(2x+3b^3\right), \text{в інших} \quad випадках \end{cases}$ $Q=\begin{cases} 2Z^2+\ln a, \text{якщо } Z>153 \\ 2Z^2+e^x, \text{якщо } Z\leq 153 \end{cases};$
9	$F=\begin{cases} a+mx, \text{якщо } x<0 \\ b+cx^2-a, \text{якщо } 0\leq x<2 \text{ та } a>b \\ 3x+2b, \text{якщо } 2\leq x<5 \\ x^2, \quad \text{в інших} \quad випадках \end{cases}$ $S=\begin{cases} 2F, \text{якщо } F<7 \\ 5F, \text{якщо } F>7 \\ 3F, \text{якщо } F=7. \end{cases}$
10	$R=\begin{cases} bx^2+\ln 2x+\sqrt{x}, \text{якщо } x\geq 5 \text{ або } b=3 \text{ та } c<8 \\ \lg^2\lvert x-a\rvert+b\dfrac{c-x}{a-m}, \text{в інших випадках} \end{cases}$ $Q=\begin{cases} e^{2x}+\sqrt{\lvert R\rvert-a}, \text{якщо } R>56.4 \\ \sin^2 2x+3R-b\sqrt{cx}, \text{якщо } R<56.4 \\ R+arctgx, \quad \text{якщо } R=56.4 \end{cases}$

Продовження таблиці 1.10

1	2
11	$T=\begin{cases} a+2x^{c}, \text{якщо } x+c=2.9 \text{ та } c<5 \\ m-3b^{2}, \text{якщо } x+c>2.9 \text{ або } m=4; \\ c+\ln x, \quad \text{в інших випадках} \end{cases}$ $Q=\frac{x}{T}+\frac{T}{2m}+\sqrt{a+T};$ $S=\begin{cases} a-T, \text{якщо} \quad T>Q \\ b+T, \text{якщо} \quad T\le Q \end{cases}$
12	$R=\begin{cases} cx^{2}+\sqrt{b}-c, \text{якщо } 2<x<5 \\ \sqrt{cx}+a, \text{якщо } 0<x\le 2 \text{ та } c=3; \\ b-2ac, \quad \text{в} \quad \text{інших випадках} \end{cases}$ $Q=3R+2bx;$ $T=\begin{cases} 2Q, \text{ якщо} \quad Q<12.8 \\ Q-3a, \text{ якщо} \quad Q\ge 12{,}8 \end{cases}$
13	$R=\begin{cases} cx^{2}+\ln x, \text{ якщо } 2<x<4 \\ b-cx, \text{ якщо } 4\le x<6 \\ a^{3}+\lg x, \text{якщо } 6\le x<8 \\ cx+b^{x}, \text{ якщо } x\le 2 \quad \text{або } x\ge 8 \end{cases}$ $S=\begin{cases} 2R, \text{ якщо} \quad R=14.2 \\ 3R, \text{ якщо} \quad R\ne 14.2 \end{cases}$
14	$Y=\begin{cases} 3x+b\sin x, \text{ якщо } 2<x<4 \quad \text{та } b<8 \\ 5x-a\cos 2x, \text{ якщо } 4\le x<9 \\ 7x+2ab, \text{в} \quad \text{інших випадках} \end{cases}$ $Z=3Y+2ax-\sqrt[3]{Y};$ $R=\begin{cases} Z+Y, \text{якщо} \quad Z\ge 25 \\ Y-Z, \text{якщо} \quad Z<25 \end{cases}$
15	$Z=\begin{cases} \sqrt{b}+\ln x-2, \text{якщо } bx=6, \text{ та } \ a\ne 2 \\ \sqrt[3]{cx+5a-c}, \text{якщо } 6<bx<9 \\ \ln\left(b+ax^{2}\right)-b, \text{ в інших випадках} \end{cases}$ $R=\begin{cases} 2Z; \text{якщо} \quad Z\ge 12 \\ 3Z; \text{якщо} \quad Z<12 \end{cases}$

1.3 Використання операторів for, while, do while

1.3.1 Теоретичний матеріал «Циклічні структури»

Оператор безумовного передавання управління goto. Формат оператора *goto*:

goto <мітка>,

де мітка – це повноправний ідентифікатор, після якого стоїть двокрапка. Мітка має знаходитися в тій самій функції, що і оператор *goto* [3, 5, 7, 14, 15, 20].

Наприклад, можна організувати цикл, що працює 100 разів, за допомогою операторів *if* та *goto*:

```
x=1;
loop:
x++;
if(x<100)  goto loop.
```

Цикли дають можливість багаторазово повторювати деякий набір операцій до тих пір, поки не буде виконана деяка умова.

Оператор for *реалізує цикли з лічильником (управляючої змінної)*. Такі цикли ще називають арифметичними (тестове завдання 6). Умову перевіряють на початку циклу, тому цикл може не бути виконаним жодного разу. Формат циклу *for*:

for(<ініціалізація>; <умова>; <крок>) <оператор>,

де <*ініціалізація*> – оператор визначення початкового значення управляючої змінної циклу;

<*умова*> – умова, за виконання якої цикл закінчується;

<*крок*> – приріст, на який змінюється управляюча змінна на кожній ітерації.

Наприклад,

for(x=1; x<=100; x++) printf ("%d", x);

// виводить на екран значення *x (цифри)* від 1 до 100

for (x=100; x>0; x--) printf ("%d", x);

// виводить на екран значення *x (цифри)* від 100 до 1

for (x=0; x<=100; x=x+5) printf ("%d", x);

// виводити на екран числа від *0* до *100* з кроком *5*

for (i=0; i<100; i++) {група операторів}

// виконати групу операторів

for (x=0; x!=123;) scanf ("%d", &x);

// цикл буде працювати, поки не буде введено число 123

```
for(i=0; i<10; i++)              // укладений цикл for
    {
       оператори
       for(j=0; j<8; j++)    { оператори }
    }.
```

Оператор while *реалізує ітераційні цикли* (тестове завдання 6). Цикл виконується до тих пір, поки умова «істина», в іншому випадку цикл закінчується, і управління передається на наступний за while оператор. Умову перевіряють на початку циклу, тому цикл може не бути виконаний жодного разу. Формат циклу *while*:

while(<умова>) <оператор>,

де ***<умова>*** **- будь-який допустимий вираз;**

<оператор> **- один оператор або група операторів.**

Наприклад,

while(i < 5) printf("%d\n", i++);

// виведення на екран числа від 0 до 4.

Оператор do-while *реалізує ітераційні цикли* (тестове завдання 7). На відміну від циклів *for* та *while*, у циклі *do-while* перевіряють умову наприкінці циклу, тому цикл *do-while* завжди буде виконаний хоча б один раз. Формат циклу *do-while:*

do {<оператор>;} while(<умова>).

Оператор break використовують для вимушеного переривання циклів for, while та do-while під час нормального виконання. Якщо оператор break стоїть усередині циклу, комп'ютер припиняє виконання циклу, і управління передається на оператор після цього циклу. Break здійснює вихід тільки з внутрішнього циклу.

За допомогою оператора goto можна вийти із цикла в будь-яку точку програми.

Оператор continue. По дії оператор continue подібен оператору break, однако він не завершує цикл, а дає команду на наступну ітерацію. При цьому оператори тіла циклу, що залишилися, не виконуються. Наприклад, наступна програма виводить на екран тільки парні числа:

```
#include <stdio.h>
main ()
{
int  x;
for (x=0; x<100; x++)
      {
        if (x%2)  continue;
        printf ("%d", x);
       }
}
```

1.3.2 Тестові завдання 6-8 «Циклічні структури»

Тестове завдання 6. Табулювання функції. Розрахувати значення функції $y = x^3\left(x + 7\right) + \frac{10}{x}$, якщо ***x*** змінюється від **2** до **3** з кроком **0.05,** і визначити суму обчислених значень [13].

Порядок виконання завдання

1 етап *– постановка та формалізація завдання.*

$$sum = \sum_{x=2}^{3} y = x^3\left(x + 7\right) + \frac{10}{x}, \quad де \quad h_x=0.05. \qquad (1.6)$$

2 етап *– складання схеми алгоритму (рисунок 1.9).*

3 етап *– програмування.*

```
#include  <stdio.h>
int main ()
{double x, y, sum=0;
for (x=2; x<=3; x+=0.05)                    // організація циклу по x від 2 до 3 з кроком 0.05
{ y=x*x*x*(x+7)+10/x;
printf(" x= %4.2f \t y= %8.3f  \n ",x,y); //виведення значень  x та y в відформатованому вигляді
sum+=y;}                                    //складене присвоювання (те саме, що і  sum=sum+y)
                                            // накопичування суми обчислених значень функції y
printf(" ----------------------------\n ");
printf(" sum=  %9.3f  \n ",sum);
return 0; }
```

4 етап *– тестування додатка (рисунок 1.10).*

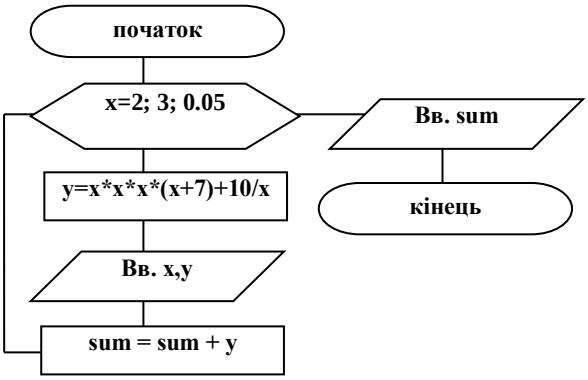


Рисунок 1.9

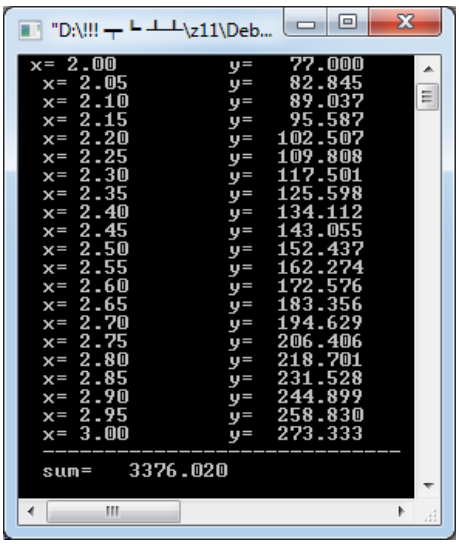


Рисунок 1.10

5 етап *– аналіз результатів. Виведення результатів у вигляді таблиці дало можливість переконатися в наявності всіх обчислених значень і правильності підрахування суми.*

Тестове завдання 7. Дослідним шляхом визначити, яка клавіша відповідає коду **27**. Застосувати оператор **i**з передумовою **while**.

Порядок виконання завдання

1 етап *– постановка та формалізація завдання.* Послідовно натискаємо клавіші у вибраному порядку та кожного разу перевіряємо відповідний їм код із числом 27 до збігу.

2 етап *– складання схеми алгоритму (рисунок 1.11).*

3 етап *– програмування.*

```
#include <stdio.h>
#include <conio.h>

main()
{ int i;
  while(i!=27)   // kod
 {    i=getch();
     printf("%d \n", i);  }
return 0;}
```

4 етап *- тестування додатка (рисунок 1.12).*

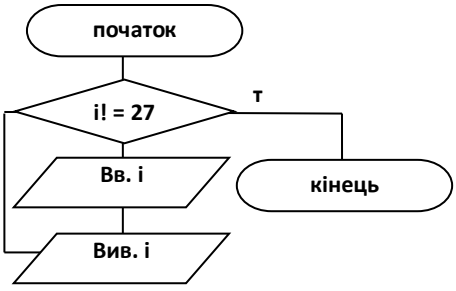


Рисунок 1.11 – Схема алгоритму

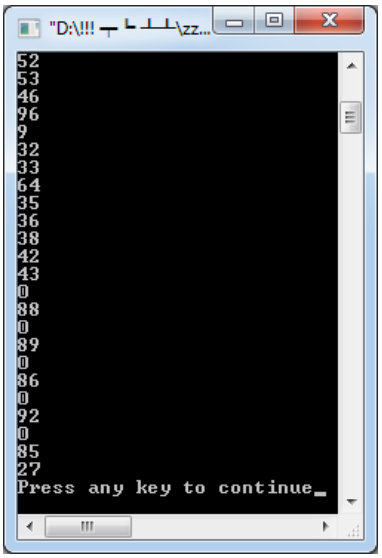


Рисунок 1.12 – Результати

5 етап *– аналіз результатів.* Помилок нема. На 24 кроці стався збіг натиснутої клавіші з заданим кодом, і цикл закінчився.

Тестове завдання 8.** З використанням генератора випадкових чисел створюють цілі числа з інтервалу [a,b***], якщо ***a <=b***. Розробити додаток, де підраховують суму всіх згенерованих на інтервалі [***a,b***] чисел, які потрапили до меншого інтервалу [***a-2, b-2***], до появи першого числа, що не потрапляє в [***a-2, b-2***]. Застосувати оператор з після умовою **do… while**.

Порядок виконання завдання

1 етап *– постановка та формалізація завдання.*

$$sum = \sum_{a-2}^{b-2} x; \qquad x \in \left[a, b \right], \qquad a \le b \quad . \tag{1.7}$$

2 етап *– складання схеми алгоритму (рисунок 1.13).*

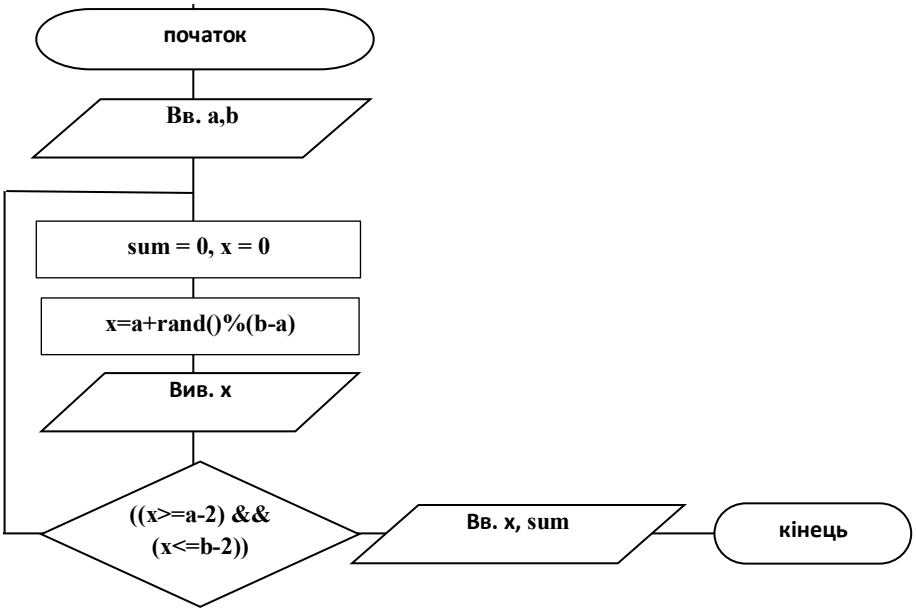


Рисунок 1.13 – Схема алгоритму

3 етап *– програмування.*

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <ctime>
#include <iostream>
using namespace std;
int main()
{srand(time(0));
int a,b;
cout << "Enter a: ";
cin >> a;                          // початкове  значення   інтервалу формування
cout << "Enter b: ";
cin >> b;                          // кінцеве    значення   інтервалу формування
cout << "початок інтервалу формування a = " << a << endl;
cout << "кінець інтервалу формування b = " << b << endl;
cout << " початок інтервалу вибірки a-2 = " << a-2 << endl;
cout << " кінець інтервалу вибірки  b-2 = " << b-2 << endl;
int sum = 0, x = 0;

do                          // початок циклу do while
{sum = sum+x;               // накопичування суми чисел, що потрапили до інтервалу вибірки
x=a+rand()%(b-a);           // генерація випадкового числа з інтервалу формування
cout << "x = " << x << endl; } while ((x>=a-2) && (x<=b-2));          // виведення поточного випадкового числа
cout << "pershe poza diapazony x = " << x << endl;          //виведення першого випадкового числа поза інтервалом вибірки
  cout << "sum = " << sum << endl;              // виведення  суми
  return 0; }
```

```
random_number = firs_value + rand() % last_value;
```

де **firs_value** - мінімальне число з бажаного діапазону

last_value - ширина вибірки

4 етап *– тестування додатка (рисунок 1.14).*

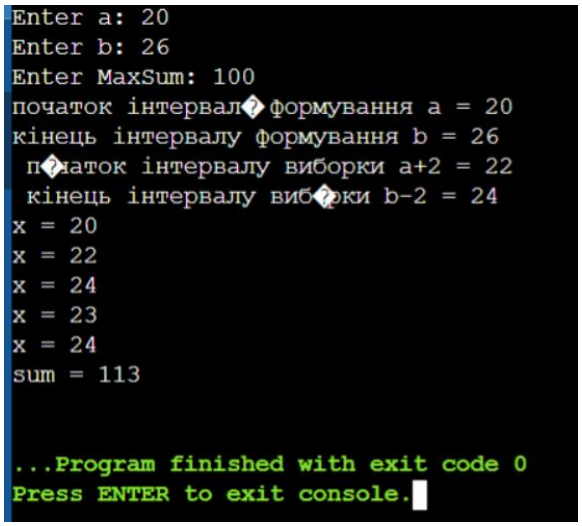


Рисунок 1.14 – Результати

5 етап *– аналіз результатів.* У програмному коді нема виведення першого випадкового числа x, яке поза інтервалом вибірки.

1.3.3 Завдання до самостійної роботи «Циклічні структури»

Варіанти індивідуальних завдань [13] до самостійної роботи «Циклічні структури» задачі табулювання функції одного аргументу і накопичення наведені в таблиці 1.11.

Таблиця 1.11 – Завдання 1 «Циклічні структури»

Варі-ант	**Завдання 1**	**Завдання 2**
1	2	3
1	$R=\dfrac{2{,}5m!+a\sum\limits_{k=1}^{5}k^3}{m\prod\limits_{i=1}^{m}(i+2)+a^{m!-1}}$	$Q=\dfrac{mb^2+2x}{\prod\limits_{i=1}^{7}(i+2)}-\dfrac{\sqrt{k!+m!}}{m+5x};$ $Z=\dfrac{k+m+2Q}{m+c+k!}+2^x;$ *якщо* $x\in[a;c];\ h_x=0{,}15a$
2	$Q=\dfrac{2(m+k)!+a\sum\limits_{i=1}^{12}(i-a)^3}{\sqrt{k!+7a}+3{,}2k!}$	$F=\dfrac{3\sqrt{(m+a)!}+2{,}2x^2+\sin x}{\lg^2 x+\ln^2 x};$ $Z=\sqrt{F}+\sum\limits_{i=3}^{9}(i+a^2)+\sqrt{x};$ $R=a+Z-13{,}5x,$ *якщо* $x\in[2;9;];\ h_x=0{,}1a$
3	$S=\dfrac{2{,}3\lg\left(\prod\limits_{i=1}^{11}(i+b)^2+3x\right)}{\sum\limits_{j=1}^{3}(j+mc)^3+2{,}5x^2};$ *якщо* $x\in[3;12];\ hx=0{,}5$	$R=\sqrt{\prod\limits_{i=2}^{9}\dfrac{i+a}{2}+3f^3};$ $Z=\ln(a+R)+\lg\left(\sum\limits_{i=3}^{5}i^3\right);$ $Q=R+fZ-5a;$ *якщо* $f\in[2;10];\ h_f=0{,}2a$
4	$Q=\dfrac{x\sqrt{m+k!}+a\ln(m!+k)}{2x+(m+k)!};$ *якщо* $x\in[3;6{,}5];$ $h_x=0{,}1$	$L=\dfrac{1}{2x}+\dfrac{2}{m!}+\dfrac{(m+2)!}{a+x};$ $S=x\prod\limits_{i=10}^{20}(i-m)^2+3{,}45c;$ *якщо* $x\in[a;2c];\ h_x=0{,}25$

Продовження таблиці 1.11

1	2	3
5	$Z=\dfrac{\sqrt{m!+k}+2k!}{\prod\limits_{i=1}^{7}(i-a)^2+\ln\sum\limits_{j=2}^{10}(b-j)^3};$	$Z=\prod\limits_{i=1}^{7}\left(i+2a^2\right)+3x\sum\limits_{k=2}^{12}k^5;$ $R=3\sqrt{Z}+m!x-17,2;$ *якщо* $x\in[a;m];\ hx=0,125$
6	$Q=\dfrac{\ln(k!+4,2)-\lg(m^3-3)}{m!+\prod\limits_{j=1}^{5}(j+k)^2+3,7\sum\limits_{i=2}^{9}(i-a)^2}$	$Z=\dfrac{2m-1}{k!+x}+\dfrac{\prod\limits_{i=1}^{7}(i+2a)^2}{(m+k)!}+2^{x+m};$ $Q=aZ+2mx;$ *якщо* $x\in[m;k];\ hx=1$
7	$Z=\dfrac{2m!+3k!+\sqrt{m!-k!}}{\prod\limits_{i=1}^{m}(i+m)^2+5\sum\limits_{i=3}^{7}(m+k+i)^3}$	$Z=\dfrac{a^x}{m!}+\dfrac{R-3}{(k+2)!}+\dfrac{(b-a)^3}{3S};$ $R=\prod\limits_{i=3}^{5}(i+k);$ $S=\sum\limits_{i=1}^{7}(i+a)^3;$ *якщо* $x\in[a;5];\ hx=0,15$
8	$R=\dfrac{3m!+k+x^2+b^x}{\lg(a+2k+x)};$ *якщо* $x\in[0,2;3,5]$ $hx=0,05$	$Q=\dfrac{b^2-a}{m!+x}+\sqrt{\dfrac{c+2m!}{\sum\limits_{k=3}^{7}(a+k)^2}};$ $F=2\sqrt{Q-c};$ $R=mx+2F^3;$ *якщо* $x\in[1;2m];\ hx=0,1$
9	$Z=\dfrac{e^x}{n!}+\dfrac{(m+n)!}{x}+\dfrac{a^x+b}{(n+3)!};$ *якщо* $x\in[2;5];\ hx=0,25$	$Z=\dfrac{a\sum\limits_{i=1}^{7}ik^2+(k+a)^x}{\sqrt{k!+x}+(k+a)!};$ $R=3Z+ax^5;$ *якщо* $x\in[2;5];\ hx=0,25$
10	$Z=\dfrac{(m!+2k)!-x\sqrt{2k+1}}{\prod\limits_{i=1}^{3}(i+2)^2+\ln(m!+4)+e^x};$ *якщо* $x\in[m;k];\ hx=0,2m$	$Q=\dfrac{a+mt}{\prod\limits_{i=1}^{7}(i+2m)}+2\sum\limits_{j=4}^{7}\dfrac{m!}{j+2};$ $Z=\dfrac{Q}{a+t}+\sqrt[3]{a+mt};$ *якщо* $t\in[5;8];\ ht=0,25$

Продовження таблиці 1.11

1	2	3
11	$Y = \dfrac{\sqrt{m!+2k} + \prod\limits_{i=2}^{5}(i+2k)^2}{3\sum\limits_{i=1}^{7}\left(a^i + 2x^2 + k!\right)};$ $якщо\ x \in [3;k];\ hx = 0{,}5k$	$F = \sqrt[3]{\dfrac{b + 2m! - 4x^2}{3\sum\limits_{i=2}^{7}(i+m)^2}};$ $R = \dfrac{a}{F+x} + 3{,}4x^2 + \prod\limits_{i=2}^{9}\dfrac{a}{i+1};$ $якщо\ x \in [5;10];\ hx = 0{,}5$
12	$R = \dfrac{k!+1}{t+2k} + \prod\limits_{i=1}^{5} i^3 + \dfrac{t^k}{2\sum\limits_{i=1}^{3}(i+a)^2};$ $якщо\ t \in [12;15];\quad ht = 0{,}5$	$Q = \dfrac{2t}{r} + \dfrac{a}{m!} + \dfrac{b+x}{(k+2)!};$ $Z = ax\sum\limits_{i=1}^{5}(i+m)^2 + 3{,}7ct;$ $якщо\ t \in [2;6];\ h_t = 0{,}25$
13	$Y = \dfrac{k^{x+1}}{\left(\prod\limits_{i=1}^{3} i^2\right) + x} + \dfrac{m!+x^5 + (m+2)!}{k!+2x+3\sum\limits_{i=2}^{7} i^5};$ $якщо\ x \in [4;7];\ hx = 0{,}25$	$M = \sqrt{a + x\sqrt{c + \ln^2 x}};$ $T = \dfrac{2n!}{k\prod\limits_{i=2}^{7} i^2} + \lg\left(\dfrac{x}{n!}\right)^3;$ $якщо\ x \in [12;31];\ hx = 0{,}5n$
14	$R = \dfrac{x^2 + \lg(m!+3k!)}{\sqrt[3]{x + m!+k!} - a^x};$ $якщо\ x \in [3{,}2;6{,}1];$ $hx = 0{,}1$	$D = \dfrac{a}{2m+x} + \prod\limits_{i=1}^{9}\dfrac{c}{i+2} + \sqrt{m!};$ $Q = 3{,}2mx^2 + \sqrt{\sum\limits_{i=4}^{9}(i+a)^3};$ $якщо\ x \in [2;6];\ hx = 0{,}2$
15	$B = (2P)!;$ $P = x^3 + \sqrt{ax^2 + mt + 1};$ $x = \sum\limits_{i=1}^{15}(i + \sin i)$	$Y = \dfrac{P+2a}{2m-x} + \dfrac{\sqrt{m!+1}}{a-x} - \dfrac{(m+5)!}{S+x};$ $P = \prod\limits_{i=1}^{3}(i+2);$ $S = \sum\limits_{j=5}^{2m}(j+a)^2;$ $x \in [3;5];\ hx = 0{,}1$

Варіанти індивідуальних завдань до самостійної роботи «Циклічні структури» задачі обчислення суми ряду з заданою точністю: обчислити

суми елементів рядів, що сходяться, із заданою точністю $E = 10^{-5}$ і визначити кількість елементів ряду (таблиця 1.12).

Таблиця 1.12 – Завдання 3 «Циклічні структури»

Варіант	**Завдання**
1	$Y = 1 + \frac{X}{2} + \frac{X^2}{4} + \frac{X^3}{8} + \frac{X^4}{16} + \frac{X^5}{32} + \ldots + \frac{x^i}{2^i}$
2	$Y = 0.5 + 0.25X^2 - 0.125X^3 + 0.0625X^4 - 0.03125X^5 + \cdots$
3	$Y = A + XA^{1/2} + 0.5XA^{1/3} + 0.25XA^{1/4} + 0.125XA^{1/5} + \ldots$
4	$Y = A + B - \frac{A}{X^2} + \frac{B}{X^3} - \frac{A}{X^4} + \frac{B}{X^5} - \frac{A}{X^6} + \frac{B}{X^7} - \ldots$
5	$Y = \frac{A}{3X} + \frac{A^2}{9X} + \frac{A^3}{27X} + \frac{A^4}{81X} + \frac{A^5}{243X} + \ldots$
6	$Y = R + \frac{3R - 2K}{6A^2} - \frac{4R - 4K}{12A^4} + \frac{5R - 6K}{24A^6} - \frac{6R - 8K}{48A^8} + \ldots$
7	$Y = \frac{X}{M} - \frac{M^2}{2X} + \frac{2M^3}{X^3} - \frac{6M^4}{X^4} + \frac{9M^5}{X^5} - \frac{12M^6}{X^6} + \ldots$
8	$Y = 2.63K + \frac{1}{K^4} + \frac{A}{2K^6 + A} + \frac{2A}{4K^8 + A^2} + \frac{3A}{8K^{10} + A^3} + \ldots$
9	$Y = 0.35A + \frac{A}{8S} + \frac{A + B}{16S} + \frac{A + 2B}{32S} + \frac{A + 3B}{64S} + \frac{A + 4B}{128S} + \ldots$
10	$Y = \frac{1}{\sin^2(X)} + \frac{1}{A - \sin^3(X)} + \frac{1}{A^2 - \sin^4(X)} + \frac{1}{A^3 - \sin^5(X)} + \ldots$
11	$Y = A^2 + \frac{8B}{2A + K} - \frac{16B^2}{6A + K} + \frac{32B^3}{24A + K} - \frac{64B^4}{120A + K} + \ldots$
12	$Y = 0.1 + \frac{A}{6M^3} - \frac{2A}{24M^4} + \frac{3A}{120M^5} - \frac{4A}{720M^6} + \ldots$
13	$Y = R^3 + \frac{BX}{X + \sin^2(X)} + \frac{ABX}{2X - \sin^3(X)} + \frac{A^2BX}{6X + \sin^4(X)} + \ldots$
14	$Y = \frac{A + X}{BX^2} + \frac{2X}{2X + B} - \frac{4X^3}{4X^2 + 2B} + \frac{8X^5}{12X^3 + 3B} - \frac{16X^7}{48X^4 + 4B} + \ldots$
15	$Y = X^4 - \frac{8A}{3X^2 - A} + \frac{9A}{7X^2 - 2A} - \frac{10A}{25X^2 - 3A} + \frac{11A}{121X^2 - 4A} - \ldots$

ЛАБОРАТОРНА РОБОТА 2. Проєктування в інтегрованому середовищі MSVC додатків, що містять масиви

Мета: *набуття навичок створення і налагодження програм в інтегрованому середовищі MSVC із алгоритмами масивів.*

2.1 Заповнення масиву, пошук, розрахунки та формування нового масиву

2.1.1 Теоретичний матеріал «Масиви»

Масивом називають поіменовану сукупність даних, що має розмір і розмірність. Формат оголошення одновимірного масиву:

<тип> <ім'я_масиву> [<розмір>],

де *тип* - визначає базовий тип масиву. Базовий тип визначає тип кожного елемента масиву;

***розмір* – кількість елементів, що може містити масив.**

Наприклад, *int sample [10]; int a[10];* /* оголошення двох масивів цілого типу з ім'ям *sample та a*, що містять по 10 елементів. Нумерація починається з нуля *sample[0], sample[1], sample[2], …,* sample[9]*/.

У мові С++ не перевіряють межі масиву – ніщо не контролює індекс при виході за межі значення в масиві. Наприклад, якщо перевищення значення індексу масиву трапиться під час виконання оператора присвоювання, то зайві значення можуть бути присвоєні іншим змінним, розташованим у пам'яті ЕОМ після поля елементів масиву [3, 5, 7, 14, 15, 20].

Ініціалізація одновимірних масивів. Загальний формат ініціалізації масиву:

<тип> <ім'я_масиву> [<розмір>] = { <список_значень> },

де список_значень - це список розділених комами констант, сумісних за типом із базовим типом масиву.

Оператор ініціалізації розмістить першу константу в перший елемент масиву, другу константу – у другий і т. д.

*int i[10] = {1,2,3,4,5,6,7,8,9,10}; /** ініціалізується 10-елементний масив цілих чисел із значеннями від 1 до 10*/.

Оскільки всі рядки у С++ мають закінчуватися нулем, масив повинен містити місце і для нього.

Ініціалізація безрозмірних символьних масивів. Підраховувати вручну кількість символів у кожному повідомленні для задавання розміру масиву дуже трудомістко. Однак можна змусити С++ автоматично оброзмірити масиви за допомогою ініціалізації безрозмірних масивів. Для цього в операторі ініціалізації не слід призначати розмір масиву, і С++ автоматично створить масив, що зможе містити присутній ініціалізатор. Наприклад, *char e[] = "cannot open file\n"*.

Мова С++ дає змогу оголошувати багатовимірні масиви. Найпростішим із багатовимірних масивів є двовимірний масив. Наприклад, для оголошення двовимірного масиву цілих чисел із ім'ям *dvmas* і розмірами *10*20* необхідно записати *int dvmas[10][20]*.

У наступній програмі організований двовимірний масив із значеннями елементів від *1* до *12*.

```
main()
{       int i, j, num[3][4];
for (i=0; i<3; i++)
        for (j=0; j<4; j++)
                num[i][j] = (i*4)+j+1;  }.
```

Елемент *num[0][0]* буде мати значення *1*, елемент *num[0][1]* – значення *2*, елемент *num[0][2]* – значення *3* і т. д.

Ініціалізація двовимірних масивів. Двовимірні масиви ініціалізовані так само, як і одновимірні. Наприклад, так можна проініціалізувати двовимірний масив *sqrs* числами від *1* до *10* і їхніми квадратами:

```
int  sqrs[10][2] = {1, 1,
                    2, 4,
                    3, 9,
                    4, 16,
                    5, 25,
                    6, 36,
                    7, 49,
                    8, 64,
                    9, 81,
                    10, 100 }.
```

Ініціалізація безрозмірних числових масивів. Ініціалізація безрозмірних масивів не обмежена тільки одновимірними масивами. Однак для багатовимірних масивів необхідно зазначати всі індекси вимірів, крім останнього лівого. Як приклад розглянемо оголошення двовимірного масиву *sqrs* як безрозмірного:

```
int  sqrs[][2] = {  1, 1,
                    2, 4,
                    3, 9,
                    4, 16,
                    5, 25,
                    6, 36,
                    7, 49,
                    8, 64,
                    9, 81,
                    10, 100}.
```

Перевага в тому, що можна подовжити або скоротити таблицю, не змінюючи індекс за першим виміром.

2.1.2 Тестові завдання 9-11 «Масиви»

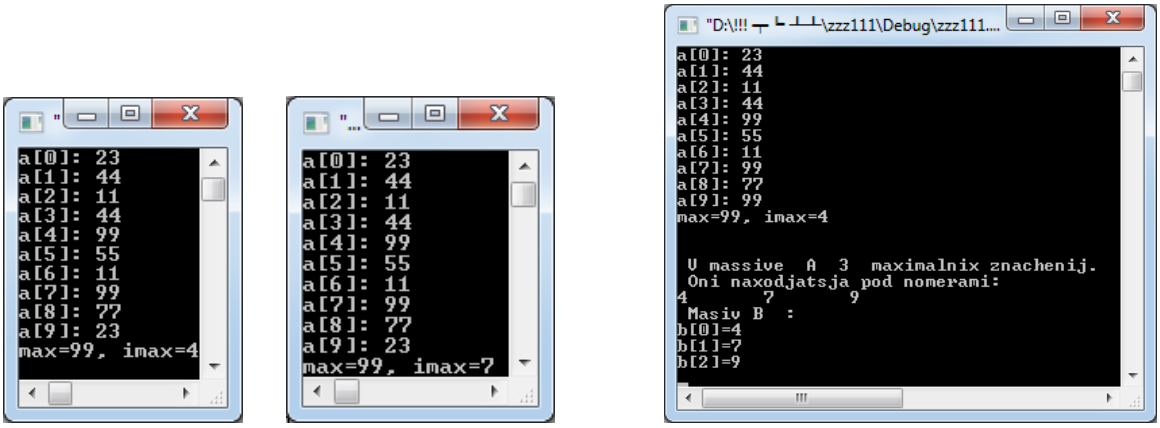
Тестове завдання 9. *Порядок виконання завдання* [13]

1 етап *– постановка та формалізація завдання.* Задано одновимірний масив А, що містить 10 довільних цілих чисел. Визначити найбільший

елемент масиву, вказати його номер за порядком за умови, що таких елементів:

- один або два (рисунок 2.1, а, б);

- більше двох (рисунок 2.1, в) – у цьому випадку сформувати одновимірний масив B, що міститиме номери за порядком найбільших елементів масиву A.



а) б) в)

а) один або перший із двох; б) один або останній із двох;
в) усі найбільші значення (формування масиву B)

Рисунок 2.1 – Результат пошуку найбільшого значення масиву

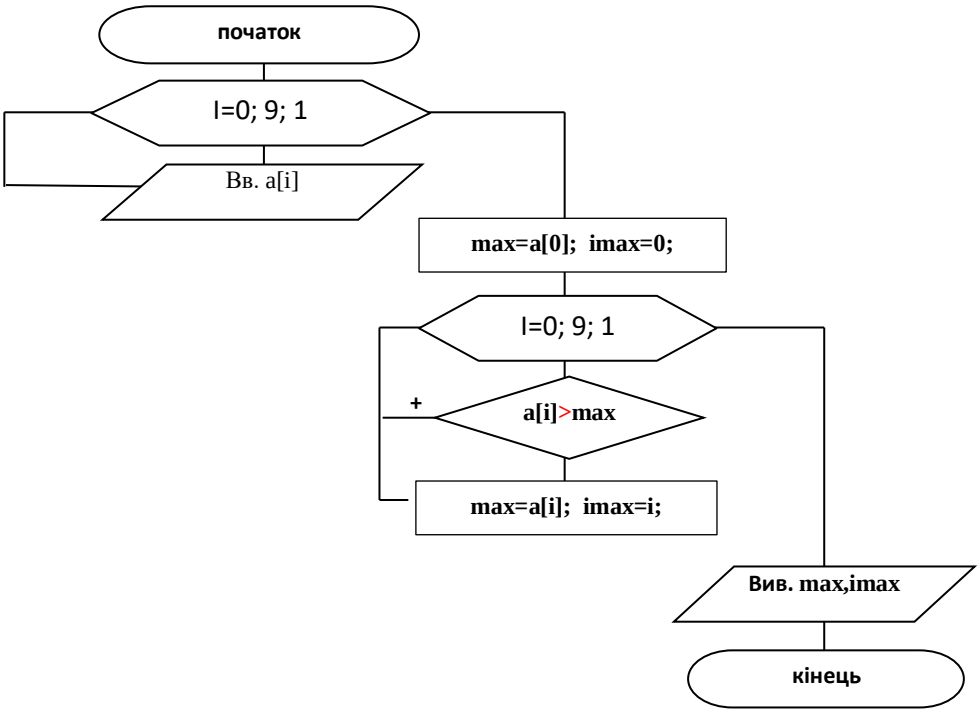
2 етап *– складання схеми алгоритму (рисунок 2.2).*

3 етап *– програмування.*

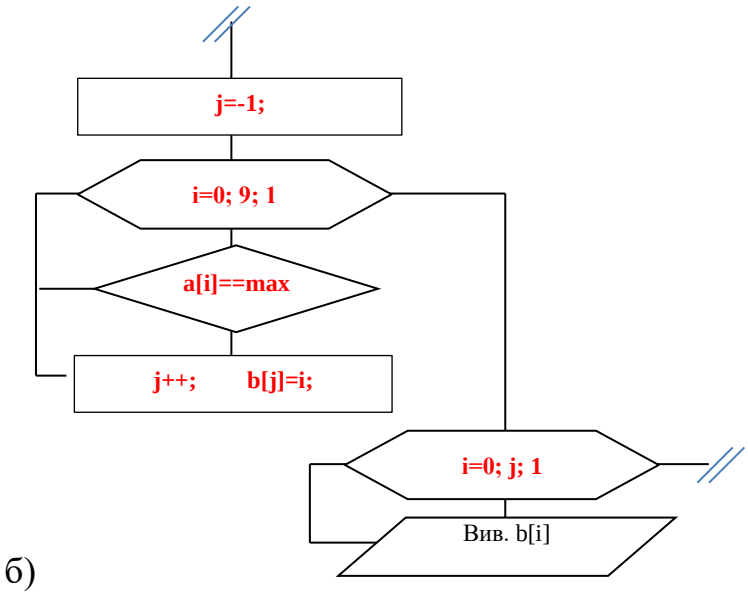
4 етап *- тестування додатка (рисунок 2.1).*

```
1)#include<stdio.h>                    // текст програми для а) і б)
#include<conio.h>
main()
{
int i, a[10], max, imax;
//clrscr();

for(i=0; i<=9; i++)
        {
        printf("a[%d]: ",i);
        scanf("%d",&a[i]);
        }
max=a[0];
```

```
imax=0;
for(i=0; i<=9; i++)
        {
        if(a[i]>max)                    // див. рис. а)
// замінити умову на (a[i]>=max)  див. рис. б)
//замінити умову на (a[i]<max)
                                //замінити умову на (a[i]<=max)
                {
                max=a[i];
                imax=i;
                }
        }
printf("max=%d, imax=%d", max,imax);
getch();                return 0; }

2) #include<stdio.h>                    //червоні  рядки треба додати для вирішення
цього завдання
#include<conio.h>
main()
{
        int i, a[10], max, imax, j, b[10];

for(i=0; i<=9; i++)
        {
        printf("a[%d]: ",i);
        scanf("%d",&a[i]);
        }
max=a[0];
imax=0;
for(i=0; i<=9; i++)
        {
        if(a[i]>max)
                {
                max=a[i];
                imax=i;
                }
        }
printf("max=%d, imax=%d  \n \n", max, imax);

3) j=-1;
for(i=0; i<=9; i++)             if(a[i]==max)           {j++;   b[j]=i; }

printf("  \n V massive  A  %d  maximalnix znachenij.  ",j+1);
printf("  \n Oni naxodjatsja pod nomerami: \n");

for(i=0; i<=j; i++)             printf("%d \t",b[i]);

printf("  \n Masiv B  : \n");
for(i=0; i<=j; i++)             printf("b[%d]=%d \n",i,b[i]);

getch();                return 0;}
```



а)



б)

а) перший або останній із двох найбільших елементів масиву;

б) формування масиву b із номерів найбільших елементів масиву a

Рисунок 2.2 – Схема алгоритму до задачі
пошуку найбільшого значення масиву

Тестове завдання 10. ***Порядок виконання завдання***

1 етап *– постановка та формалізація завдання.* Сформувати двовимірний цілочисельний масив A[4][4]. Визначити суму елементів кожного рядка масиву A окремо. Суми занести до одновимірного масиву Sum.

2 етап *– складання схеми алгоритму (рисунок 2.3).*

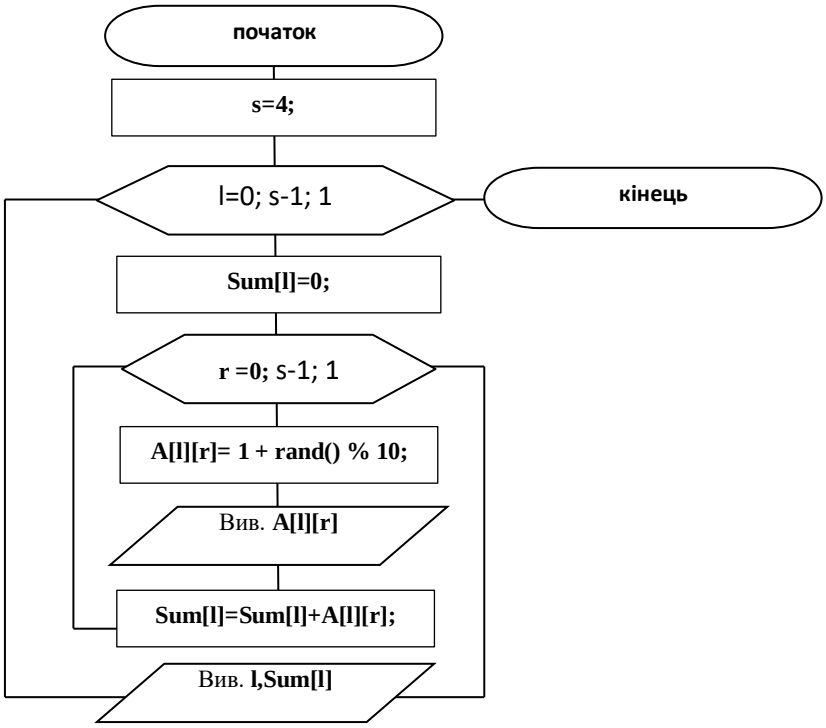


Рисунок 2.3 – Схема алгоритму

3 етап *– програмування.*

4 етап *– тестування додатка (рисунок 2.4).*

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctime>
#define s 4
int A[s][s],Sum[s];

int main (void)
{ short l, r ;

printf("Matrica A:\t\t Rezul'tat:Summa d ctroke: \n");
```

```
srand( time( 0 ) );

for (l=0; l<s; l++)
 {Sum[l]=0;
  for (r=0; r<s; r++)
  {A[l][r]= 1 + rand() % 10;
   printf(" %3d",A[l][r]);
   Sum[l]=Sum[l]+A[l][r];}
printf("\t\t Sum[%2d]= %3d",l,Sum[l]);
printf("\n"); }
  return 0; }
```

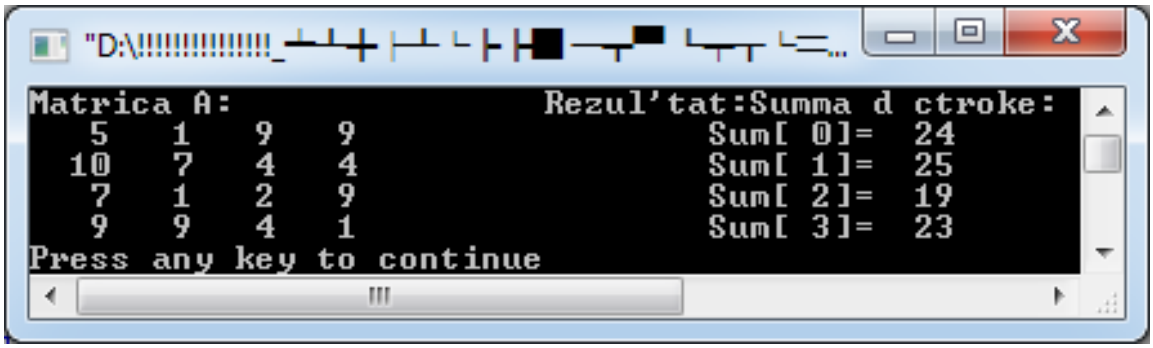


Рисунок 2.4 – Сума в рядках масиву

Тестове завдання 11. ***Порядок виконання завдання***

1 етап *– постановка та формалізація завдання.* Сформувати двовимірний цілочисельний масив A[4][4]. Впорядкувати за зростанням вказаний користувачем стовпець матриці A.

2 етап *– складання схеми алгоритму (рисунок 2.5).*

3 етап *– програмування.*

4 етап *– тестування додатка (рисунок 2.6).*

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctime>
#define s 4
int A[s][s];

int main (void)
```

```
{ int l, r, vr;
  bool fl;

printf("Matrica A: \n");
srand( time( 0 ) );

for (l=0; l<s; l++)
 {for (r=0; r<s; r++)
 {A[l][r]= 1 + rand() % 10;
  printf(" %3d",A[l][r]); }
  printf("\n"); }

do
{printf("\n Vvedite N stolbika ot 0 do%2d  \t ",s-1);
 scanf("%d",&r);
 if ((r>=0)&&(r<=s-1)) printf("\n Rezul'tat  dlja %d - stolbika \n",r);else printf("
Takogo stolbika net\n ");
}while ((r<0)||(r>=s));

do
{fl=0;

for (l=0; l<s-1; l++)
        if (A[l][r]> A[l+1][r])
                {vr=A[l][r];
                A[l][r]=A[l+1][r];
                A[l+1][r]=vr;
                fl=1;}
}while (fl==1);


printf("Matrica A: \n");

for (l=0; l<s; l++)
 { for (r=0; r<s; r++)    printf(" %3d",A[l][r]);
   printf("\n"); }

return 0; }
```

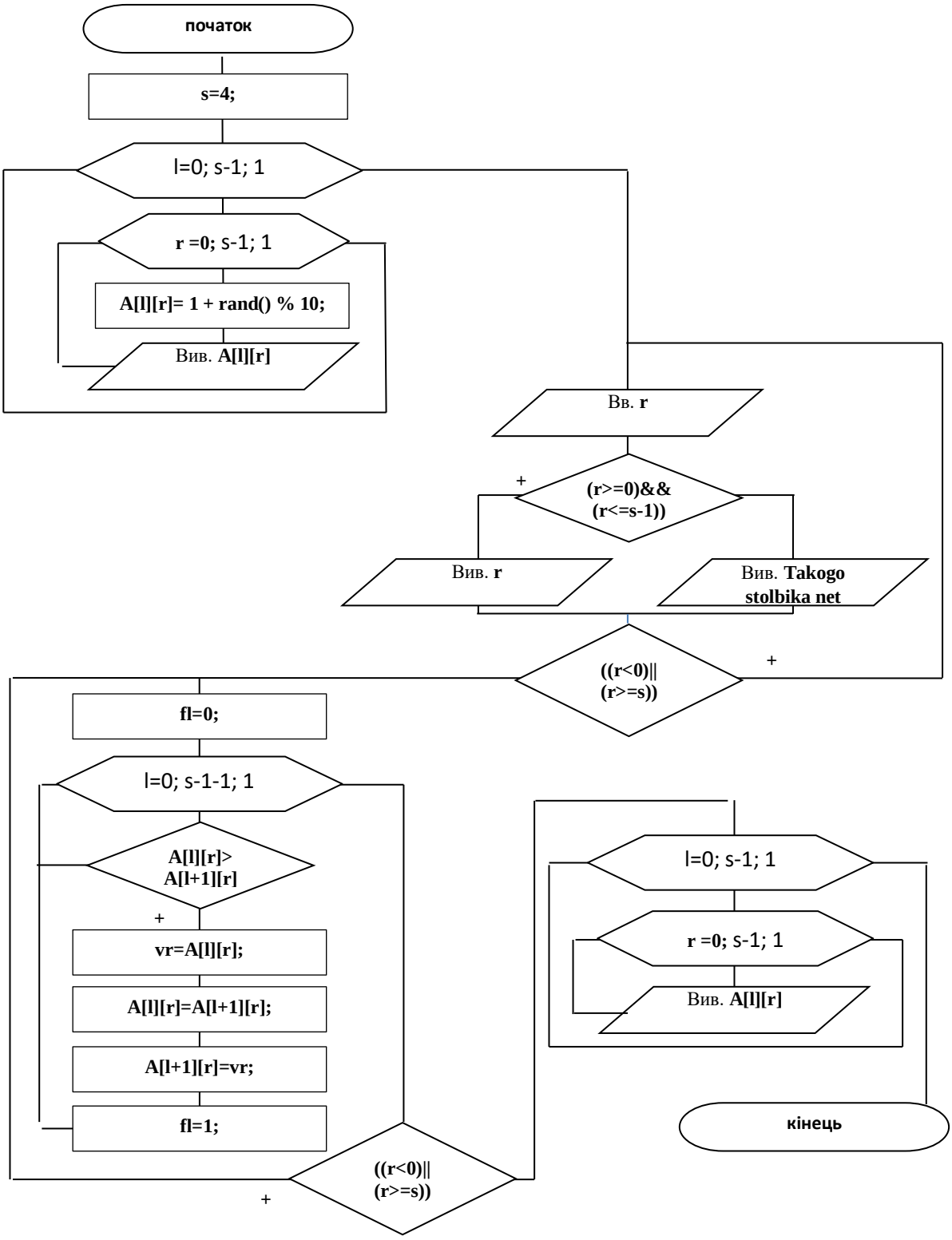


Рисунок 2.5 – Схема алгоритму

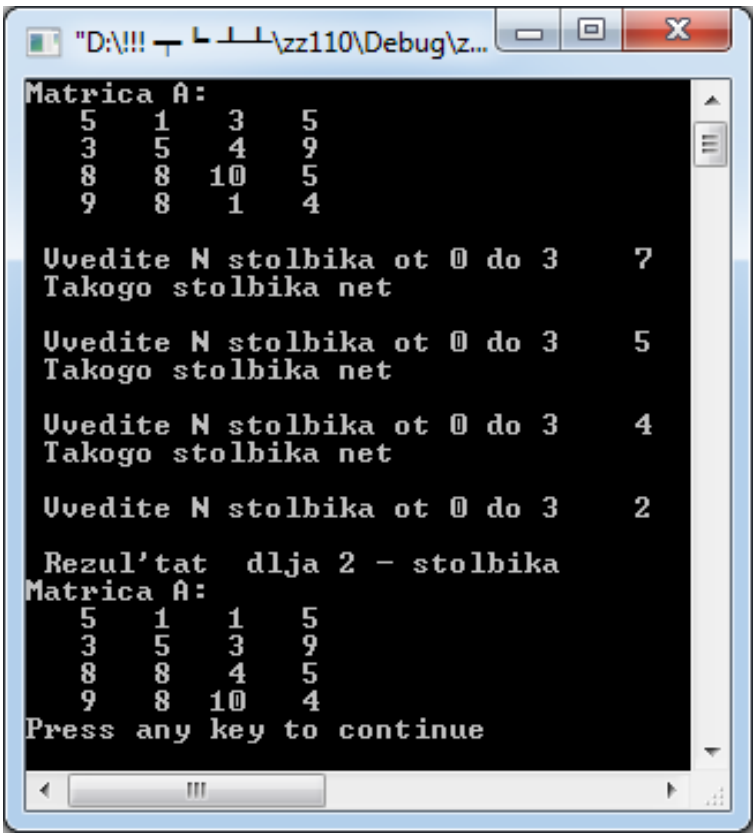


Рисунок 2.6 – Впорядкування вказаного стовпця

2.1.3 Завдання до самостійної роботи «Масиви»

Варіант 1. Для перевезення пасажирів поїздами дальнього сполучення виділені локомотиви з номерами, які зберігаються в ЕОМ як масив X, що містить M однорідних елементів. Ввести значення елементів масиву та визначити номер першого елемента X(i) з ключовою ознакою X(i) < 100000 і X(i) > 50000.

Варіант 2. Для перевезення пасажирів поїздами місцевого сполучення виділені локомотиви з номерами, які зберігаються в ЕОМ як масив Y. Масив Y містить N елементів Y(i). Скласти схему алгоритму та програму введення значень елементів масиву, визначення добутку значень елементів із номерами i = 1,3,5..., ключова ознака яких Y(i) ≥ 300000 або Y(i) > 120000.

Варіант 3. Для перевезення пасажирів поїздами приміського сполучення виділені локомотиви з номерами, які зберігаються в ЕОМ як

масив Z, що містить N однорідних елементів. Ввести значення елементів масиву Z і визначити номер першого елемента Z(i) з ключовою ознакою Z(i) = 50000 або Z(i) = 10000.

Варіант 4. Для формування вантажних вивізних поїздів обирають локомотиви з номерами, які зберігаються в ЕОМ як масив S, що містить N елементів S(i). Скласти схему алгоритму та програму введення елементів масиву, визначення добутку значень елементів із номерами i = 2,4,6..., ключова ознака яких S(i) ≤ 100000 або S(i) = 130000.

Варіант 5. Для формування вантажних передаточних поїздів обирають локомотиви з номерами, які зберігаються в ЕОМ як масив T, що містить N однорідних елементів. Ввести значення елементів масиву T і визначити кількість елементів T(i) з ключовою ознакою T(i) > 10000 і T(i) < 28000.

Варіант 6. При проведенні роботи на сортувальній станції для формування поїздів обирають локомотиви з номерами, які зберігаються в ЕОМ як масив Z, що містить N однорідних елементів. Ввести значення елементів масиву Z і визначити номер першого елемента Z(i) з ключовою ознакою Z(i) ≥ 25000 або Z(i) ≤ 10000.

Варіант 7. При проведенні роботи на сортувальній станції для розформування поїздів обирають локомотиви з номерами, які зберігаються в ЕОМ як масив X, що містить N однорідних елементів. Ввести значення елементів масиву X і визначити номер першого елемента X(i) з ключовою ознакою X(i) > 12000 або X(i) < 13000.

Варіант 8. При подаванні поїздів на завантаження обирають локомотиви з номерами, які зберігаються в ЕОМ як масив Z, що містить N однорідних елементів. Ввести значення елементів масиву Z і визначити кількість елементів Z(i) з ключовою ознакою Z(i) > 15000 і Z(i) ≤ 80000.

Варіант 9. При подаванні поїздів на розвантаження обирають локомотиви з номерами, які зберігаються в ЕОМ як масив S, що містить N однорідних елементів. Ввести значення елементів масиву S і визначити номер першого елемента S(i) з ключовою ознакою S(i) > 20000 і S(i) < 28000.

Варіант 10. При проведенні поточного ремонту обирають локомотиви з номерами, які зберігаються в ЕОМ як масив X, що містить M однорідних елементів. Ввести значення елементів масиву X і створити масив Z з елементів X(i), ключова ознака яких 20000 > X(i) ≥ 13000.

Варіант 11. Для виключення з неексплуатованого парку обирають локомотиви з номерами, які зберігаються в ЕОМ як масив K, що містить N елементів K(i). Ввести значення елементів масиву K і визначити кількість елементів з ключовою ознакою K(i) = 4000, K(i) = 11000 або K(i) > 30000.

Варіант 12. Для включення до інвентарного парку обирають локомотиви з номерами, які зберігаються в ЕОМ як масив H, що містить K однорідних елементів. Ввести значення елементів масиву H і створити масив S з елементів H(i), ключова ознака яких H(i) ≥ 15000 або H(i) = 10000.

Варіант 13. Для включення до інвентарного парку обирають локомотиви з номерами, які зберігаються в ЕОМ як масив X, що містить M однорідних елементів. Ввести значення елементів масиву X і створити масив S з елементів X(i), ключова ознака яких 24000 ≤ X(i) < 120000.

Варіант 14. Для виключення з експлуатованого парку обирають локомотиви з номерами, які зберігаються в ЕОМ як масив K, що містить S однорідних елементів. Ввести значення елементів масиву K і створити масив Z з елементів K(i), ключова ознака яких K(i) ≥ 60000 або K(i) < 50000.

Варіант 15. Для включення до експлуатованого парку обирають локомотиви з номерами, які зберігаються в ЕОМ як масив A, що містить N елементів A(i). Ввести значення елементів масиву A, визначити кількість елементів з ключовою ознакою A(i) > 50000 або A(i) < 30000.

Література [3, 5, 7, 14, 15, 20].

Список літератури

1 Cyganek B. Introduction to Programming with C++ for Engineers. Hoboken: Wiley, 2021. 649 p.

2 George S. Tselikis, Nikolaos D. Tselikas. C. From Theory to Practice. CRC Press, 2020. 708 p.

3 Gonzalez Avelino J. Computer Programming in C for Beginners. Springer, 2020. 203 p.

4 Kelley Al., Pohl I. A Book on C: Programming in C. Addison-Wesley Professiona, 1998. 724 p.

5 Leonard Robbie. Programming in C Part Two: Advanced Data Types (Programming in C by Eucoding Book 2). Independently Published, 2023. 338 p.

6 Leonard Robbie. Programming in C. Part 1: Introduction to C. (Programming in C by Eucoding Book 1). Independently Published, 2023. 280 p.

7 Вступ до програмування мовою С++. Організація обчислень: навч. посіб. /Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. Київ: ВПЦ «Київський Університет», 2012. 175 с.

8 Васильєв О. Програмування на С++ в прикладах і задачах: навч. посіб. Київ: Видавництво Ліра, 2017. 382 с.

9 Веклич Р. А., Карнаух Т. О., Ставровський А. Б. Вступ до програмування мовою С++. Структури даних: навч. посіб. Київ: ВПЦ «Київський університет», 2018. 99 с.

10 Грицюк Ю., Рак Т. Програмування Мовою С++: навч. посіб. Львів: Вид-во ЛДУ БЖД, 2011. 290 с.

11 Іванов Є. О., Ліндер Я. М., Жереб К. А. Основи мови програмування С++: навч. посіб. Київ: Логос, 2020. 90 с.

12 Мерзляк А. Г., Полонський В. Б., Якір М. С. Геометрія: підруч. для 8 кл. з поглибленим вивченням математики. Харків: Гімназія, 2016. 234 с.

13 Основи алгоритмізації базових обчислювальних процесів: навч. посіб. / В. С. Меркулов, В. О. Гончаров, І. Г. Бізюк та ін. Харків: УкрДАЗТ, 2008. 164 с.

14 Основи програмування мовою С++: метод. вказ. до лаб. робіт / С. Є. Бантюков, В. М. Бутенко. О. В. Головко та ін. Харків: УкрДУЗТ, 2017. Ч. 1. 58 с.

15 Лабораторний практикум з програмування мовою С/С++: навч. посіб. для студ. техн. спец. закл. вищ. освіти І–IV рівнів акредит. /П. А. Пех, С. В. Лавренчук, М. В. Делявський, С. В. Гринюк. Луцьк: Вежа-Друк, 2020. 228 с.

16 Середа Б. П., Прищип М. Г., Кругляк І. В. Експериментальні дослідження процесів ОМТ: навч.-метод. посіб. Запоріжжя: ЗДІА, 2011. 170 с.

17 Сухова О. В. Програмування мовою С: навч. посіб. для здоб. вищ. освіти за спец. 104 Фізика та астрономія. Дніпро: ДНУ, 2018. 52 с.

18 Вступ до алгоритмів / Томас Г. Кормен, Чарлз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн. К.І.С., 2023. 1288 с.

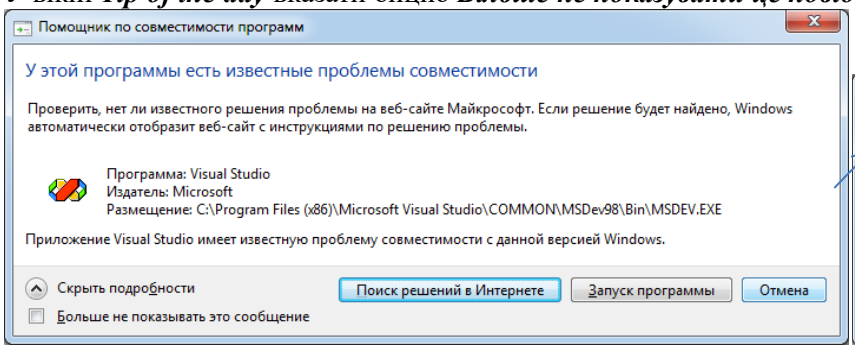
19 С++. Алгоритмізація та програмування: підручник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, О. В. Задерейко. Вид. 2-ге, перероб. і доп. Одеса: Фенікс, 2019. 477 с.

20 Шпак З. Програмування мовою С. Львів: Львівська політехніка, 2011. 436 с.

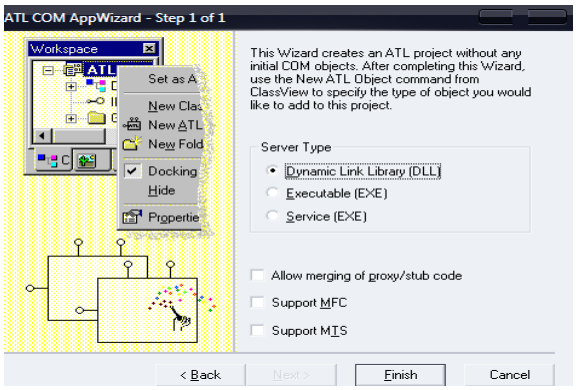
ДОДАТОК А

ЗАВАНТАЖЕННЯ MSVC

1 *Завантажити С++* (***Пуск → Усі програми → Microsoft Visual Studio6.0 → Microsoft C++ 6.0).***
У разі появи вікна Помічника за сумісністю програм вибрати ***Запуск програми.***
У вікні ***Tip of the day*** вказати опцію ***Більше не показувати це повідомлення,*** вибрати ***Close.***

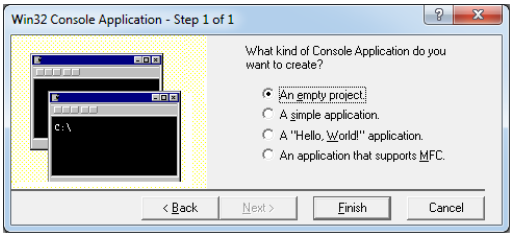


2 *Створити новий проєкт* ***(File→ New→*** вкладка ***Projects*** *(здійснити налаштування))*:

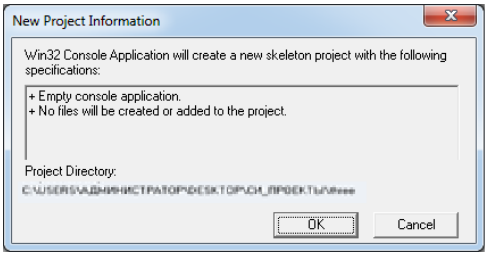


- у списку вибрати ***Win32 Console Aplication***;
- **Project name**: задати ім'я проєкту (наприклад ***LR1_Petrenko***);
- **Location**: вибрати папку ***Home;***
- **Plaforms**: вибрати ***win32;***
- **ATL COM**….. вибрати ***Finish.***

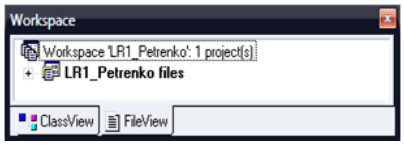
Створити стандартний проєкт. У вікні **Win32 Console Aplication** вибрати ***An empty project.***



У вікні New Project Information вибрати ***OK.***

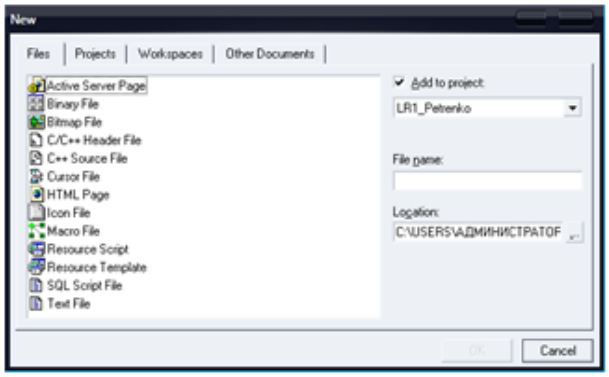


У вікні Workspace (провідник проєкту) з'явиться LR1_Petrenko classes.



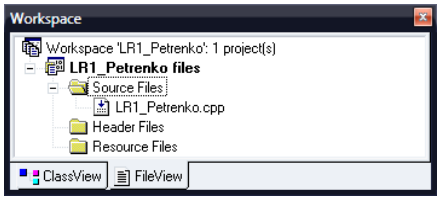
Побачити структуру проєкту можна, якщо активізувати вкладку FileView.

3 *Створити новий файл* ***(File→New→***вкладка ***Files*** *(здійснити налаштування))*:



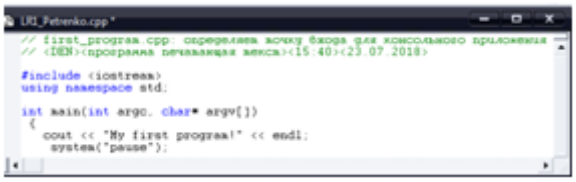
- у списку вибрати ***C++Source File***;
- вікні **File name**: задати ім'я файлу (наприклад ***LR1_Petrenko***)
- додати в проєкт (***Add to project***).

Побачити структуру проєкту можна, якщо активізувати вкладку Workspace.

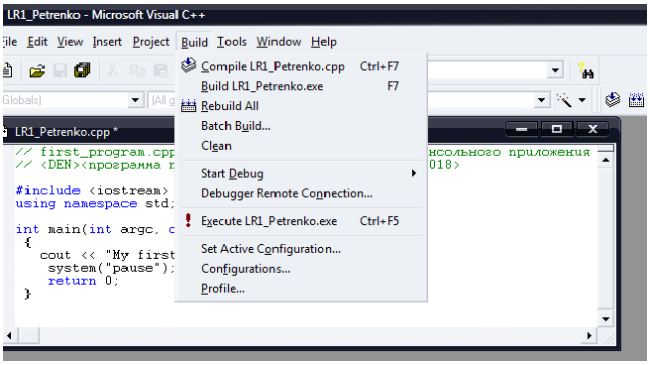


4 Ввести текст програми:

- у вікні Workspace вибрати файл LR1_Petrenko.cpp;
- вікні, що відкрилось (**Code**) із заголовком LR1_Petrenko.cpp, ввести ***текст коду.***



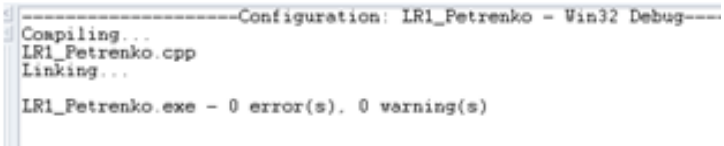
5 Компіляція програми. У рядку заголовка вибрати вкладку ***Build,*** у вікні, що відкрилося, вибираємо ***Compile LR1_Petrenko.cpp.***

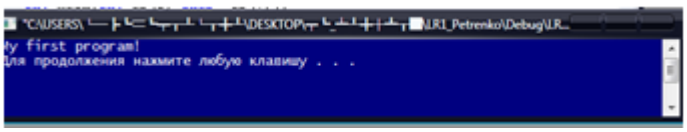


У вікні з заголовком Configuration: LR1_Petrenko - Win32 Debug аналізуємо результати, якщо є помилки, виправляємо їх і компілюємо програму повторно.

6 Збирання програми (створення об'єктного коду). Знову активізуємо вкладку **Build** і вибираємо ***Build LR1_Petrenko .exe.***

7 Виконання програми. Знову активізуємо вкладку **Build** і вибираємо ***Execute LR1_Petrenko .exe,*** аналізуємо результати розрахунків.





ДОДАТОК Б
ПРИКЛАД ОФОРМЛЕННЯ ЗАВДАННЯ ДО ЗВІТУ

ЛАБОРАТОРНА РОБОТА. Проєктування в інтегрованому середовищі MSVC додатків, що містять базові обчислювальні структури (слідування, галуження, повторення)

Мета: *набуття навичок створення і налагодження програм в інтегрованому середовищі MSVC із алгоритмами лінійної структури. Форматне та потокове введення-виведення даних. Розрахунки з використанням функцій.*

Тестове завдання 1. Знаючи довжину кожної зі сторін **a, b, c** трикутника, знайти площу **s** і периметр **p**, використовуючи формулу Герона. Формула Герона дає змогу обчислити площу трикутника **S** за його сторонами **a, b, c**:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

де $p = \frac{a+b+c}{2}$ – півпериметр трикутника.

1 етап *– постановка та формалізація завдання*:

$s = \sqrt{\mathrm{r(r-a)(r-b)(r-c)}}$; $r = \frac{a+b+c}{2}$; p=2r.

2 етап *– складання схеми алгоритму (*рисунок Б.1*).*

3 етап *– програмування (ввести і протестувати).*

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "windows.h"

int main()
{float  a,b,c,r;                  //опис дійсних змінних, довжиною 4 байти
double s;                         //опис дійсної змінної, довжиною 8 байтів
printf("a=");                     //виведення на екран символів ‘a=’
scanf("%f",&a);                   //запис у змінну a введеного з  клавіатури значення
printf("b=");       scanf("%f",&b);
printf("c=");       scanf("%f",&c);
r=(a+b+c)/2;                              //обчислення півпериметра
s=sqrt(r*(r-a)*(r-b)*(r-c));              //обчислення площини
printf("s=%5.2f \t",s);                   // форматування та
                                          // виведення  на екрані виведення  s
printf(" p=%5.2f \n",2*r);                // форматування та виведення  на екрані p
printf("r=%5.2f \n", r);                  // форматування та виведення  на екрані r
return 0;}
```

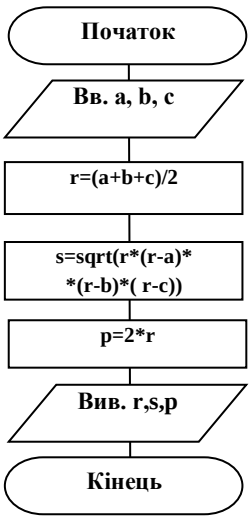


Рисунок Б.1

4 етап *– тестування додатка.*

5 етап *– аналіз результатів.*

У завданні не передбачено перевіряти коректність вихідних даних (від'ємні чи нульові значення сторін трикутника, виконання умови існування трикутника – сума двох сторін більше третьої і т. ін.). Тому потрібно вводити коректні дані. Для цього випадку одержано правильний результат (рисунок Б.2).

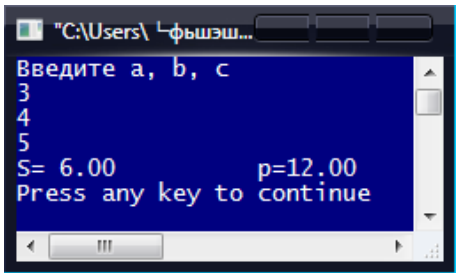


Рисунок Б.2 – Результати виконання тестового завдання 1.
Структури слідування

Усі інші завдання з лабораторних і самостійної робіт оформлюють аналогічно.

ДОДАТОК В
ОСНОВИ С++

Таблиця В.1 – Базові типи даних С

Тип	**Діапазон значень**	**Роз-мір**	**Тип**	**Діапазон значень**	**Роз-мір**
bool	true та false	1	*unsigned int*	0 … 4 294 967 295	4
signed char	-128 … 127	1	*signed long int*	-2 147 483 648 … 2 147 483 647	4
unsigned char	0 … 255	1	*unsigned long int*	0 … 4 294 967 295	4
signed short int	-32 768 … 32 767	2	*float*	3.4e-38 … 3.4e+38	4
unsigned short int	0 … 65 535	2	*double*	1.7e-308 … 1.7e+308	8
signed int	-2 147 483 648 … 2 147 483 647	4	*long double*	3.4e-4932 … 3.4e+4932	10
void	Безліч значень порожньо. Використовують для визначення функцій, які не повертають значення, для зазначення порожнього списку аргументів функції, як базовий тип для покажчиків і в операції приведення типів				

Ключові (зарезервовані) слова С: and; and_eq; asm; auto; bitand; bitor; bool; break; case; catch; char; class; compl; const; const_cast; continue; default; delete; do; double; dynamic_cast; else; enum; explicit; export; extern; false; float; for; friend; goto; if; inline; int; long; mutable; namespace; new; not; not_eq; operator; or; or_eq; private; protected; public; register; reinterpret_cast; return; short; signed; sizeof; static; static_cast; struct; switch; template; this; throw; true; try; typedef; typeid; typename; union; unsigned; using; virtual; void; volatile; wchar_t; while; xor; xor_eq.

Таблиця В.2 – Унарні операції С++

Бінарні операції	*Дія*	*Бінарні операції*	*Дія*	*Унарні операції*	*Дія*
?:	**умовна операція (тернарна)**	*****	**множення**	**++**	**збільшення на 1**
=	**присвоювання**	**/**	**ділення**	**- -**	**зменшення на 1**
***=**	**множення з присвоюванням**	**%**	**остача від ділення**	***sizeof***	**розмір**
/=	**ділення з присвоюванням**	**+**	**додавання**	**-**	**порозрядне заперечення**
%=	**остача від ділення з присвоюванням**	**-**	**віднімання**	**!**	**логічне заперечення**
+=	**додавання з присвоюванням**	**<<**	**зсув ліворуч**	**-**	**арифметичне заперечення (унарний мінус)**
- =	**віднімання з присвоюванням**	**>>**	**зсув праворуч**	**+**	**унарний плюс**
<<=	**зсув ліворуч із присвоюванням**	**<**	**менше**	**&**	**отримання адреси**
>>=	**зсув праворуч із присвоюванням**	**<=**	**менше або дорівнює**	*****	**звернення за адресою (реадресація)**
&=	**Порозрядне і з присвоюванням**	**>**	**більше**	**->**	**звернення до члена структури або класу через покажчик**
| =	**порозрядне або з присвоюванням**	**>=**	**більше або дорівнює**	**.**	**звернення до члена структури або класу**
^=	**порозрядне виключне або з присвоюванням**	**= =**	**дорівнює**	**::**	**дозвіл області дії**
.	**послідовне обчислення**	***!=***	**не дорівнює**	***new***	**виділення пам'яті**
		&	**порозрядна кон'юнкція (І)**	***delete***	**звільнення пам'яті**
		^	**порозрядне виключне АБО**	***(type)***	**перетворення типу**
		|	**порозрядна диз'юнкція (АБО)**		
		&&	**логічне І**		
		||	**логічне АБО**		

Таблиця В.3 – Пріоритети і порядок операцій

Пріоритет	*Позначення операції*	*Тип операції*	*Порядок виконання*
1	**( ) [ ] . -> ::**	вираз	зліва направо
2	**++ - -**	унарні	зліва направо (постфіксний інкремент і декремент)
3	**++ - -** ***! ~ + - & * (тип) sizeof new delete тип()***	унарні	справа наліво (префіксний інкремент і декремент)
4	**. -> ***		зліва направо
5	*** / %**	мультиплікативні	зліва направо
6	**+ -**	адитивні	
7	**<< >>**	зсув	
8	**< > <= >=**	відношення	
9	**== !=**	відношення (рівність)	
10	**&**	порозрядне І	
11	**^**	порозрядне виключне АБО	
12	**|**	порозрядне АБО	
13	**&&**	логічне І	
14	**||**	логічне АБО	
15	**? :**	умовна	
16	**= *= /= %= += -= &= |= >>= <<= ^=**	звичайне та складене присвоювання	справа наліво
17	**.**	послідовне обчислення	зліва направо

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних і самостійної робіт із дисципліни

«*КОМП'ЮТЕРНА ТЕХНІКА ТА ПРОГРАМУВАННЯ*»

Частина 1

Відповідальний за випуск Бізюк І. Г.

Редактор Ібрагімова Н. В.

Підписано до друку 05.07.2024 р.
Умовн. друк. арк. 4,0. Тираж . Замовлення № .

Видавець та виготовлювач Український державний університет
залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха,7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.