Міністерство освіти і науки України
Сумський державний університет

Тищенко К. В., Логвинов А. М., Пилипенко О. В.

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ В ЕЛЕКТРОННИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЯХ (ПРАКТИКУМ)

Навчальний посібник

Рекомендовано вченою радою Сумського державного університету

Суми
Сумський державний університет
2024

УДК 004.4:681.5(075.8)
Т 47

Рецензенти:
С. О. Лебединський – кандидат фізико-математичних наук, науковий співробітник відділу квантової електродинаміки сильних полів Інституту прикладної фізики НАН України;
Ю. О. Шкурдода – доктор фізико-математичних наук, професор, доцент кафедри електроніки, загальної та прикладної фізики Сумського державного університету

Рекомендовано до видання
вченою радою Сумського державного університету
як навчальний посібник
(протокол № 15 від 24 червня 2024 року)

Т 47 **Тищенко К. В.**
Алгоритмічні мови програмування в електронних інформаційних системах та комп'ютерних технологіях (практикум) / К. В. Тищенко, А. М. Логвинов, О. В. Пилипенко. – Суми : Сумськкий державний університет, 2024. – 95 с.
ISBN 978-966-657-976-1

У навчальному посібнику подано навчальні матеріали з дисципліни «Алгоритмічні мови програмування в комп'ютерних технологіях» і «Програмування електронних систем оброблення даних».

Призначений для здобувачів закладів вищої освіти спеціальності «Електроніка».

УДК 004.4:681.5(075.8)
ISBN 978-966-657-976-1
© Сумський державний університет, 2024

ЗМІСТ

С.

Вступ

Цей навчальний посібник із практикумів допоможе здобувачам освіти розібратися в основах алгоритмічних мов програмування, алгоритмах, а також навчитися розв’язувати типові задачі засобами та методами мови програмування С++. Видання містить інформацію щодо налаштування інтегрованого середовища розробника Microsoft Visual Studio та 8 лабораторних робіт, розрахованих на виконання мовою С++. Коло питань, охоплених посібником, стосується використання класичних методів, засобів та алгоритмів розв’язання типових задач і покликане надати здобувачеві базові навички для вирішення більш складних завдань, пов’язаних із розробленням програмного забезпечення. Особливістю цього навчального посібника є надання в кожній лабораторній роботі алгоритму й коду розв'язання подібної задачі, що надає правильний вектор пошуку алгоритму реалізації власного завлання згідно з варіантом.

Обрання мови програмування С++ для виконання лабораторних робіт аргументовано тим, що вона є універсальною, й тому підходить для розв’язання широкого кола задач, починаючи низькорівневим програмуванням, закінчуючи Stand Alone додатками та Back End частиною клієнт серверних чи Web-застосунків. Також синтаксис С++ став основою для більшості сучасних мов програмування, що в подальшому дозволить швидко опановувати нові мови, підвищуючи свій рівень як програміста.

Автори впевнені, що виконавши запропоновані в навчальному посібнику лабораторні роботи здобувачі одержать практичні навички для розв’язання широкого кола задач, та закладуть основу для становлення їх як розробників програмного забезпечення для найрізноманітніших проєктів.

Розділ 1
Початок роботи з Microsoft Visual Studio

Microsoft Visual Studio – це набір інструментів для створення програмного забезпечення: від планування до розроблення інтерфейсу користувача, написання коду, тестування, налагодження, аналізування якості коду та продуктивності. Нижче описано процес створення проєкту та написання простої програми в Microsoft Visual Studio 2022.

1.1 Створення порожнього проєкту

Після запуску середовища програмування перед користувачем відкривається початкова сторінка (рис. 1.1), де презентовані останні розроблені проєкти, передбачена можливість відкриття вже наявних проєктів та створення нового.

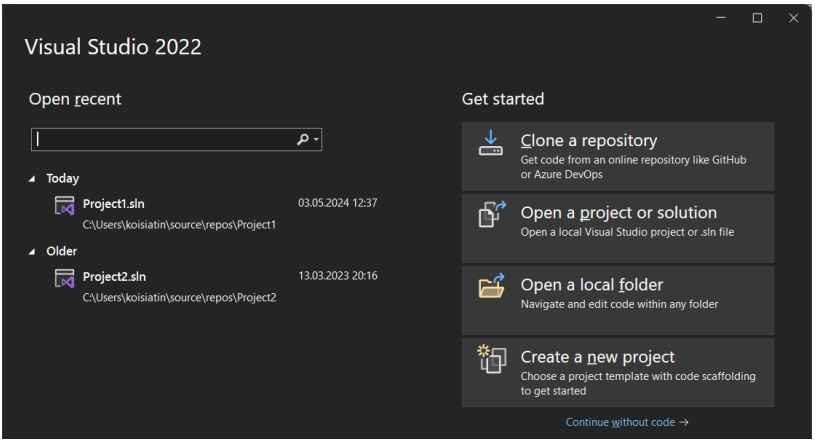


Рисунок 1.1 – Початкова сторінка середовища програмування Microsoft Visual Studio

Для створення проєкту необхідно вибрати розділ «Створити проєкт», також для цього може бути використана комбінація клавіш Ctrl + Shift + N.

Далі відкривається вікно властивостей проєкту, що складається з кількох частин (рис. 1.2). У правій частині вікна необхідно вибрати Visual C++, водночас може бути обраний тип створюваного проєкту. В межах курсу «Алгоритмічні мови програмування в комп'ютерних технологіях» рекомендується використовувати «Порожній проєкт». На наступному етапі вказують ім'я проєкту та вибирають папку, у якій буде його збережено. Завершується процес створення натисканням кнопки «ОК».

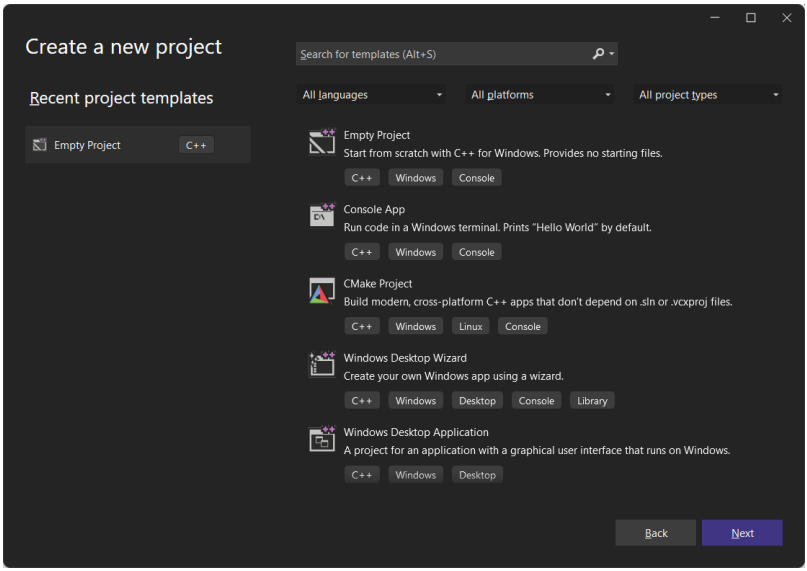


Рисунок 1.2 – Вікно властивостей проєкту

Проєкт створено. Для початку роботи необхідно додати файл із кодом програми.

1.2 Додавання файлу з кодом програми

Існує три способи додавання файлу до проєкту:

– використання оглядача рішень (рис. 1.3);

– вибір у меню Проєкт → Додати новий елемент → Створити елемент;

– використання «гарячих клавіш» «Ctrl + Shift + A».

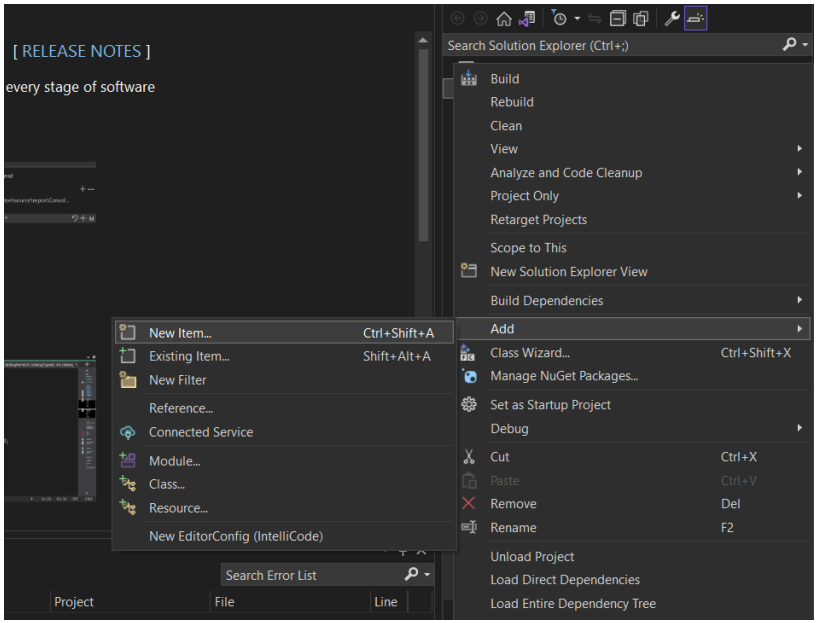


Рисунок 1.3 – Додавання файлу до проєкту через оглядач рішень

Незалежно від вибраного способу відкривається вікно додавання нового елемента. У вікні необхідно вибрати пункт «Файл C++(.cpp)» і написати ім’я файлу в нижній частині вікна (рис. 1.4).

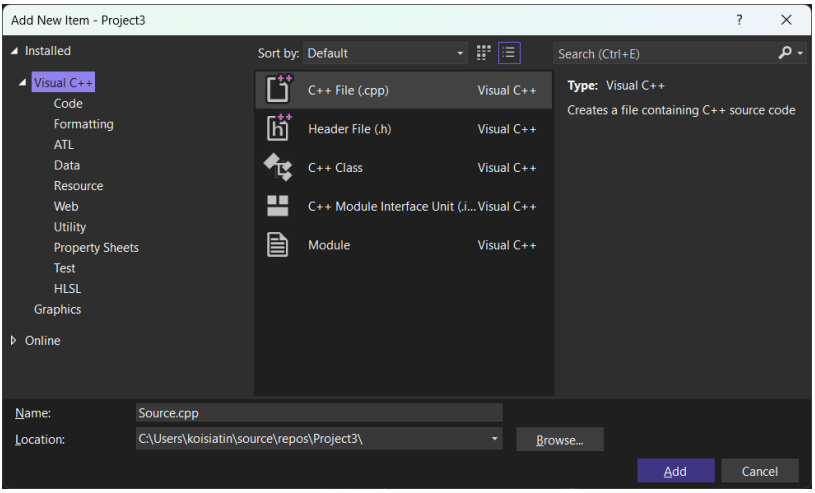


Рисунок 1.4. – Додавання нового елемента

Після натискання кнопки «Додати» відкривається сторінка редактора, призначена для написання коду програми.

1.3 Запуск програми

Після закінчення написання програми, необхідно провести її компіляцію з перевіркою на можливі синтаксичні помилки, складання та в разі успішної компіляції, запустити програму для виконання. Для цього можна вибрати на панелі інструментів кнопку «Запуск» або скористатися клавішею «F5» у разі успішної компіляції програми відкривається вікно консолі, у якому вводяться / виводяться на екран дані, якщо це передбачено програмою. На рисунку 1.5 у результаті успішного запуску програми в консолі виведено повідомлення Hello, World!

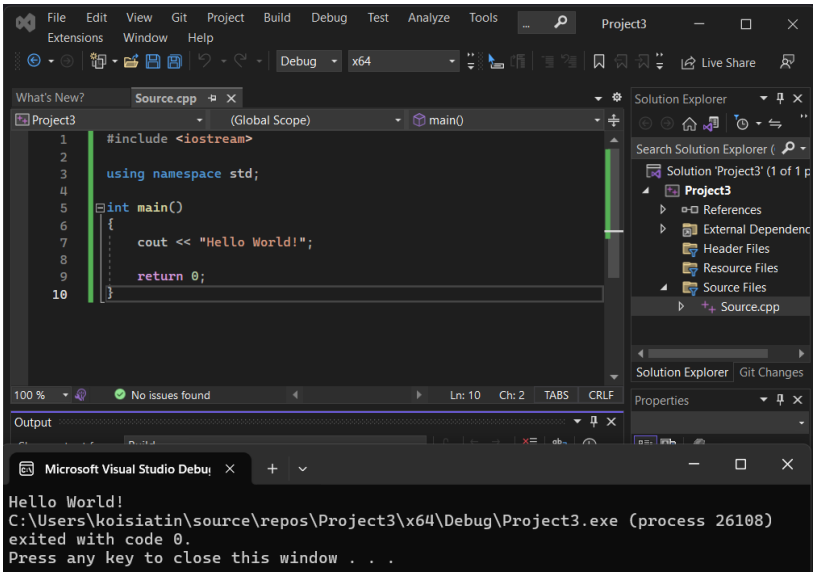


Рисунок 1.5 – Результат успішного виконання програми

Якщо програмний код має помилки, програма не буде запущена. Ви можете переглянути список помилок, вибравши меню «Перегляд → Список помилок». З рисунка 1.6 бачимо, що в коді є синтаксична помилка, а саме, немає «;» перед return.

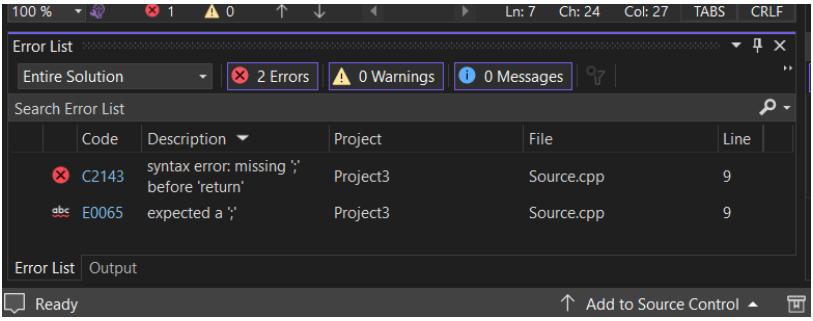


Рисунок 1.6 – Список помилок

1.4 Використання української мови в консолі

Як бачимо з рисунка 1.5 у консолі виводиться текст англійською мовою. У Visual Studio передбачено можливість підключити українську мову для використання в консолі. Для цього необхідно додати до коду програми такі рядки:

```
#include <Windows.h>
   SetConsoleOutputCP(1251);
   SetConsoleCP(1251);
```

Тепер можна виводити в консоль текст українською мовою. Це проілюстровано на рисунку 1.7.

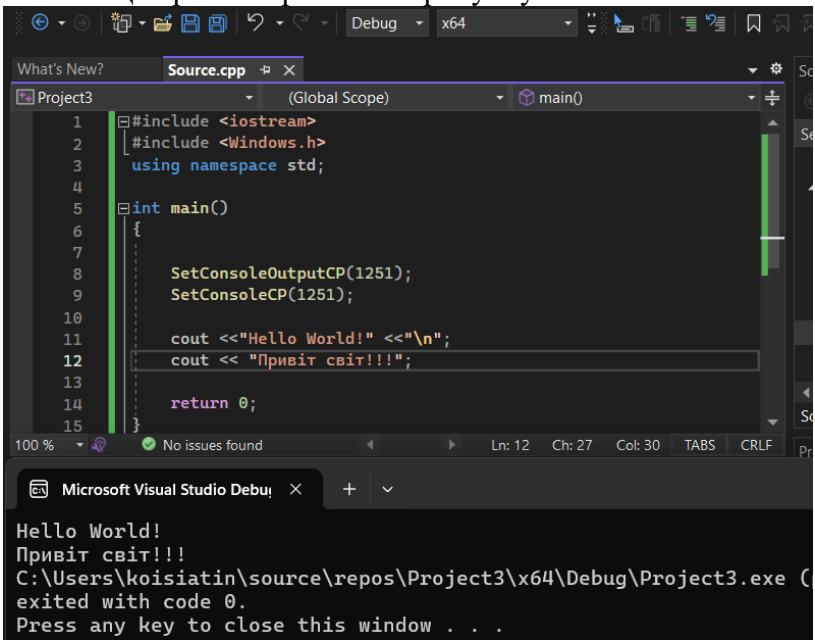


Рисунок 1.7 – Підключення української мови до консолі

Розділ 2
Лабораторний практикум

2.1 Лінійне програмування

Завдання для роботи

Скласти програму для обчислення еквівалентних пар виразів $z1$, $z2$, $y1$, $y2$ відповідно до завдання в таблиці 2.1.6. Для всіх виразів підібрати вхідні дані згідно з областю допустимих значень.

Вивести на екран результати обчислень і вхідні дані. Уведення даних організувати з клавіатури.

Теоретичні відомості

Потокове введення / виведення

Підтримка для потокового введення / виведення даних вбудованих типів мови С ++ реалізована в бібліотеці iostream. Для використання бібліотеки iostream у програмі необхідно додати заголовки:

```
#include <iostream>
```

Застосовуючи iostream, необхідно використовувати наступну директиву простору імен, щоб визначення в iostream були доступні в програмі: using namespace std;

Операції введення / виведення виконуються за допомогою класів istream (потоковий ввід) і ostream (потокове виведення). Третій клас, iostream, є похідним від них і підтримує двонаправлене введення / виведення. Для зручності в бібліотеці визначені три стандартних об'єкти-потоку:

- cin – об'єкт класу iostream, відповідний стандартному введенню. Загалом він дозволяє читати дані з терміналу користувача (консолі);
- cout – об'єкт класу iostream, відповідний стандартному виведенню. Загалом він дозволяє виводити дані на термінал користувача (консоль);

• cerr – об’єкт класу iostream, відповідний стандартному виведенню для помилок. У цей потік спрямовані повідомлення про помилки програми.

Виведення зазвичай здійснюють за допомогою оператора зсуву вліво (<<), а введення – за допомогою оператора зсуву вправо (>>). Виведення повідомлення на екран:

```
cout << "Hello, World!"; / * На екран буде виведене повідомлення Hello, World! * /
```

Введення з клавіатури:

```
int new;
cin >> new; / * Введення з клавіатури значення цілочислової змінної new * /
```

Форматне виведення

Налаштування ширини полів

Для розміщення чисел різної довжини в полях постійної ширини можна, можливо скористатися функцією-членом width. Може бути застосоване двома способами:

```
cout.width (); cout.width(i);
```

Перша форма повертає поточну установку ширини поля. Друга встановлює ширину поля рівної *i* пробілів та повертає попереднє значення ширини, метод width () стосується лише наступного відображуваного елемента, після чого ширина поля повертається до значення за замовчуванням.

Приклад:

```
cout << '#'; cout.width (12);
cout << 12 << "#" << 24 << "# \ n";
```

Оператор виведення створює такий рядок виведення:

```
# 12 # 24 #
```

Символи-наповнювачі

За замовчуванням cout заповнює невикористовувані частини поля пробілами. Для зміни цього можна скористатися функцією-членом fill (). Наприклад, наступний виклик змінює символ-заповнювач на зірочку: **cout.fill ('*'),** приклад:

```
     int main ()
     {Using std :: cout; cout.fill ( '*');
     const char * staff [2] = { "Waldo
Whipsnade", "Wilmarie
     Wooper "}; int bonus [2] = {900
1350}; for (int i = 0; i <2; i ++)
     {cout << staff [i] << ": $";
cout.width (7);
     cout << bonus [i] << "\ n";
     } Return 0;
     }
```

Нижче показано виведення програми з прикладу:

```
Waldo Whipsnade: $ **** 900
Wilmarie Wooper: $ *** 1350
```

Необхідно звернути увагу, що на відміну від використання функції по установці ширини поля, новий символ-заповнювач залишається чинним до тих пір, поки він не буде замінений.

Установка точності відображення чисел із плаваючою точкою

Функція-член precission () дозволяє задати кількість знаків після десяткового дробу. cout.precision (n); // n – число знаків після десяткового дробу Приклад:

```
     int main () {
     float pricel = 20.40;
     float price2 = 1.9 + 8.0 / 9.0;
cout.precision (2);
```

```
    cout << "Furry Friends is $" <<
pricel << "! \ n"; cout << "Fiery Fiends
is $" << price2 << "! \ n"; return 0;
    }
```

Виведення програми із застосуванням поточного форматування С ++ має такий вигляд:

```
    Furry Friends is $ 20.! Fiery Fiends
is $ 2.8!
```

Основні алгебраїчні функції

Бібліотека cmath мови С ++ визначає набір функцій для виконання спільних математичних операцій і перетворень:

```
#include <Cmath>
```

У таблицях 2.1–2.5 надані основні функції математичних операцій і перетворень бібліотеки cmath.

Таблиця 2.1.1 – Тригонометричні функції

cos(a)	Обчислення косинуса кута *a*, кут *a* в радіанах
sin(a)	Обчислення синуса кута *a*, кут *a* в радіанах
tan(a)	Обчислення тангенса кута *a*, кут *a* в радіанах
acos(a)	Обчислення арккосинуса кута *a*, кут *a* в радіанах
asin(a)	Обчислення арксинуса *a*. Результат в радіанах
atan(a)	Обчислення арктангенса *a*. Результат в радіанах

Таблиця 2.1.2 – Гіперболічні функції

cosh(a)	Обчислення гіперболічного косинуса
sinh(a)	Обчислення гіперболічного синуса
tanh(a)	Обчислення гіперболічного тангенса

Таблиця 2.1.3 – Експонентні й логарифмічні функції

exp(a)	Обчислення експоненти
log(a)	Натуральний логарифм
log10(a)	Десятковий логарифм *a*

Таблиця 2.1.4 – Функції ступеня

pow(a,b)	Зведення числа *a* в степінь *b*
sqr(a)	Корінь квадратний числа *a*, ($a \geq 0$)

Таблиця 2.1.5 – Округлення, модуль та інші функції

ceil(a)	Округлення до найменшого цілого значення, але не менше *a*
abs(a)	Обчислення модуля *a*
floor(a)	Округлення до найбільшого цілого значення, але не більше *a*
fmod(a,b)	Залишок від ділення *a* на *b*

Функцію sec □ обчислюють зі співвідношення sec □ · cos □ = 1.

Приклад програми

Вихідні дані: □ = 3,3

Вирази для обчислення

$$y_1 = \frac{x^2 + 2x - 3 + (x+1)\sqrt{x^2 - 9}}{x^2 - 2x - 3 + (x-1)\sqrt{x^2 - 9}};$$

$$y_2 = \frac{\sqrt{x+3}}{\sqrt{x-3}}.$$

```
// лабораторна робота № варіант № група №
студент ПІБ
#include <iostream> // стандартний потік
консольного введення / виведення
#include <cmath> / * заголовки, що містить
основні
математичні функції * /

using namespace std; // використовується
стандартне простір імен

int main ()
{
     // оголошення та ініціалізація
змінних
     double y1 = 0.0, y2 = 0.0, s1 = 0.0,
yd = 0.0;
     double x = 3.3;
     s1 = sqrt (x * x - 9); // повторює
частину формули
     yd = x * x - 2 * x - 3 + (x-1) * s1;
// знаменник
     y1 = (x * x + 2 * x -3 + (x + 1) *
s1) / yd;
     y2 = sqrt (x + 3) / sqrt (x-3);
    cout << "Y1 =" << y1 << "Y2 =" << y2 <<
endl;
```

```
    cin.get (); // затримка екрана
 return 0;
}
```

Зміст звіту

Звіт до лабораторної роботи повинен містити:

1) текст завдання, варіант;
2) лістинг програми;
3) скриншоти результатів виконання програми.

Варіанти завдань для роботи

Таблиця 2.1.6 – Варіанти завдань

№ пор.	Вихідні дані	Формули для обчислень
1	x, α, β	$z_1 = \left(\frac{1+\sqrt{x}}{\sqrt{1+x}} - \frac{\sqrt{1+x}}{1+\sqrt{x}}\right)^2 - \left(\frac{1-\sqrt{x}}{\sqrt{1+x}} - \frac{\sqrt{1+x}}{1-\sqrt{x}}\right)^2;$ $z_2 = \frac{16x\sqrt{x}}{(1-x^2)(x-1)};$ $y_1 = (\cos\alpha - \cos\beta)^2 + (\sin\alpha - \sin\beta)^2;$ $y_2 = 4\sin^2\frac{\alpha-\beta}{2}.$
2	m, n, α	$z_1 = \frac{\left(\sqrt[4]{m}+\sqrt[4]{n}\right)^2 + \left(\sqrt[4]{m}-\sqrt[4]{n}\right)^2}{2(m-n)} : \frac{1}{\sqrt{m^3}-\sqrt{n^3}} - 3\sqrt{mn};$ $z_2 = \left(\sqrt{m}-\sqrt{n}\right)^2;$ $y_1 = \frac{1-2\,sin^2\alpha}{1+sin\,2\alpha};$ $y_2 = \frac{1-tg\,\alpha}{1+tg\,\alpha}.$

3	a, α	$z_1 = \sqrt{\frac{2a}{(1+a)\sqrt[3]{1+a}}} \cdot \sqrt[3]{\frac{4+\frac{8}{a}+\frac{4}{a^2}}{\sqrt{2}}}$; $z_2 = \frac{2\sqrt[6]{a^5}}{a}$; $y_1 = \frac{\cos 4\alpha + 1}{\operatorname{ctg}\alpha - \operatorname{tg}\alpha}$; $y_2 = \frac{1}{2} sin\, 4\alpha.$
4	x, α	$z_1 = \frac{4x\left(x+\sqrt{x^2-1}\right)^2}{\left(x+\sqrt{x^2-1}\right)^4 - 1}$; $z_2 = \frac{1}{\sqrt{x^2-1}}$; $y_1 = \frac{\cos^{-1} 2\alpha + \sin 2\alpha \operatorname{tg} 2\alpha}{1+\cos 4\alpha} + \frac{1}{4\sin^2\left(\frac{\pi}{4}-\alpha\right)\operatorname{ctg}\left(\frac{\pi}{4}-\alpha\right)}$; $y_2 = \frac{1}{cos^3\, 2\alpha}.$
5	a, α	$z_1 = \left(\frac{\sqrt{a}}{2} - \frac{1}{2\sqrt{a}}\right)^2 \left(\frac{\sqrt{a}-1}{\sqrt{a}+1} - \frac{\sqrt{a}+1}{\sqrt{a}-1}\right)$; $z_2 = \frac{1-a}{\sqrt{a}}$; $y_1 = \cos\alpha(1+\cos^{-1}\alpha + \operatorname{tg}\alpha)(1-\cos^{-1}\alpha + \operatorname{tg}\alpha)$; $y_2 = 2\, sin\, \alpha.$
6	a, α	$z_1 = \frac{1}{2\left(1+\sqrt{a}\right)} + \frac{1}{2\left(1-\sqrt{a}\right)} - \frac{a^2+2}{1-a^3}$; $z_2 = -\frac{1}{a^2+a+1}$; $y_1 = \frac{1+\operatorname{ctg} 2\alpha \operatorname{ctg}\alpha}{\operatorname{tg}\alpha + \operatorname{ctg}\alpha}$; $y_2 = \frac{1}{2} ctg\, \alpha.$

7	m, n, α	$z_1 = \frac{\left(m^2 - \frac{1}{n^2}\right)^m \left(n + \frac{1}{m}\right)^{n-m}}{\left(n^2 - \frac{1}{m^2}\right)^n \left(m - \frac{1}{n}\right)^{m-n}}$; $z_2 = \left(\frac{m}{n}\right)^{m+n}$; $y_1 = \frac{\cos^2\alpha - \operatorname{ctg}^2\alpha + 1}{\sin^2\alpha + \operatorname{tg}^2\alpha - 1}$; $y_2 = ctg^2\,\alpha$.
8	x, y, α	$z_1 = \left(\frac{\sqrt[3]{x+y}}{\sqrt[3]{x-y}} + \frac{\sqrt[3]{x-y}}{\sqrt[3]{x+y}} - 2\right) : \left(\frac{1}{\sqrt[3]{x-y}} - \frac{1}{\sqrt[3]{x+y}}\right)$; $z_2 = \sqrt[3]{x+y} - \sqrt[3]{x-y}$; $y_1 = \sin^2\left(\frac{9\pi}{8} + \alpha\right) - \sin^2\left(\frac{17\pi}{8} - \alpha\right)$; $y_2 = \frac{1}{\sqrt{2}} sin\,2\alpha$.
9	x, y, α	$z_1 = \left(\left(\frac{x^2}{y^3} + \frac{1}{x}\right) : \left(\frac{x}{y^2} - \frac{1}{y} + \frac{1}{x}\right)\right) : \frac{(x-y)^2 + 4xy}{1 + yx^{-1}}$; $z_2 = \frac{1}{xy}$; $y_1 = \left(1 + \frac{1}{\cos 2\alpha} + \operatorname{tg} 2\alpha\right)\left(1 - \frac{1}{\cos 2\alpha} + \operatorname{tg} 2\alpha\right)$; $y_2 = 2\,tg\,2\alpha$.
10	x, y, α	$z_1 = \left(\frac{3}{2x-y} - \frac{2}{2x+y} - \frac{1}{2x-5y}\right) : \frac{y^2}{4x^2 - y^2}$; $z_2 = \frac{24}{5y - 2x}$; $y_1 = \frac{\cos(3\pi - 2\alpha)}{2\sin^2\left(\frac{5\pi}{4} + \alpha\right)}$; $y_2 = tg\left(\alpha - \frac{5\pi}{4}\right)$.

11	a, b, α, β	$z_1 = \dfrac{2b + a - \frac{4a^2 - b^2}{a}}{b^3 + 2ab^2 - 3a^2b} \cdot \dfrac{a^3b - 2a^2b^2 + ab^3}{a^2 - b^2}$; $z_2 = \dfrac{a - b}{a + b}$; $y_1 = \dfrac{\operatorname{tg} 2\alpha + \operatorname{ctg} 3\beta}{\operatorname{ctg} 2\alpha + \operatorname{tg} 3\beta}$; $y_2 = \dfrac{tg\, 2\alpha}{tg\, 3\beta}$.
12	x, α	$z_1 = \left(x \cdot \sqrt[3]{\dfrac{x-1}{(x+1)^2}} + \dfrac{x-1}{\sqrt[3]{(x^2-1)^2}} \right)^{-3/5} : (x^2 - 1)^{4/5}$; $z_2 = \dfrac{1}{x^2 - 1}$; $y_1 = \operatorname{tg}\alpha + \operatorname{ctg}\alpha + \operatorname{tg} 3\alpha + \operatorname{ctg} 3\alpha$; $y_2 = \dfrac{8\cos^2 2\alpha}{\sin 6\alpha}$.
13	a, b, α	$z_1 = \dfrac{(ab^{-1} + a^{-1}b + 1)(a^{-1} - b^{-1})^2}{a^2b^{-2} + a^{-2}b^2 - (ab^{-1} + a^{-1}b)}$; $z_2 = \dfrac{1}{ab}$; $y_1 = \dfrac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$; $y_2 = ctg\left(\dfrac{5\pi}{4} + \dfrac{3\alpha}{2}\right)$.
14	a, α	$z_1 = \left(\dfrac{1}{\sqrt{a} + \sqrt{a+1}} + \dfrac{1}{\sqrt{a} - \sqrt{a-1}} \right) : \left(1 + \sqrt{\dfrac{a+1}{a-1}} \right)$; $z_2 = \sqrt{a - 1}$; $y_1 = \dfrac{\sin 2\alpha - \sin 3\alpha + \sin 4\alpha}{\cos 2\alpha + \cos 3\alpha + \cos 4\alpha}$; $y_2 = tg\, 3\alpha$.

15	x, α	$z_1 = \dfrac{x-1}{x^{3/4}+x^{1/2}} \cdot \dfrac{x^{1/2}+x^{1/4}}{x^{1/2}+1} \cdot x^{1/4} + 1;$ $z_2 = \sqrt{x};$ $y_1 = \dfrac{\operatorname{tg} 2\alpha}{\operatorname{tg} 4\alpha - \operatorname{tg} 2\alpha};$ $y_2 = cos\ 4\alpha.$
16	x, α	$z_1 = \sqrt[6]{4x\left(11+4\sqrt{6}\right)} \cdot \sqrt[3]{4\sqrt{2x} - 2\sqrt{3x}};$ $z_2 = \sqrt[3]{20x};$ $y_1 = \dfrac{1+\cos\alpha+\cos 2\alpha+\cos 3\alpha}{\cos\alpha + 2\cos^2\alpha - 1};$ $y_2 = 2cos\ \alpha.$
17	p, α	$z_1 = \dfrac{\sqrt{(2p+1)^3} + \sqrt{(2p-1)^3}}{\sqrt{4p + 2\sqrt{4p^2-1}}};$ $z_2 = 4p - \sqrt{4p^2-1};$ $y_1 = \cos^{-4}\alpha - \sin^{-4}\alpha;$ $y_2 = -\dfrac{16\ cos\ 2\alpha}{sin^4\ 2\alpha}.$
18	x, α	$z_1 = \dfrac{1-x^{-2}}{x^{1/2}-x^{-1/2}} - \dfrac{2}{x^{3/2}} + \dfrac{x^{-2}-x}{x^{1/2}-x^{-1/2}};$ $z_2 = -\sqrt{x}\left(1+\dfrac{2}{x^2}\right);$ $y_1 = \dfrac{\operatorname{tg}^4\alpha - \operatorname{tg}^6\alpha}{\operatorname{ctg}^4\alpha - \operatorname{ctg}^2\alpha};$ $y_2 = \operatorname{tg}^8\alpha.$

19	x, α, β	$z_1 = \left(\frac{\sqrt[4]{x^3} - \sqrt[4]{x}}{1-\sqrt{x}} + \frac{1+\sqrt{x}}{\sqrt[4]{x}}\right)^2 + \left(1 + \frac{2}{\sqrt{x}} + \frac{1}{x}\right)^{-1/2};$ $z_2 = \frac{1-\sqrt{x}}{1-x};$ $y_1 = \sin^2\left(\beta - \frac{\pi}{2}\right) - \cos^2\left(\alpha - \frac{3\pi}{2}\right);$ $y_2 = cos(\alpha+\beta)\,cos(\alpha-\beta).$
20	x, α	$z_1 = \left(\sqrt{1-x^2}+1\right) : \left(\frac{1}{\sqrt{1+x}} + \sqrt{1-x}\right);$ $z_2 = \sqrt{1+x};$ $y_1 = 3 - 4\sin^2\left(\frac{3\pi}{2} - \alpha\right);$ $y_2 = 4\,cos\left(\frac{\pi}{6}+\alpha\right) cos\left(\frac{\pi}{6}-\alpha\right).$

2.2 Умовні та циклічні алгоритмічні конструкції. Визначення потрапляння променя у вхідну зіницю складної форми

Завдання для роботи

Для кожної лінії, що обмежує вхідну зіницю, скласти рівняння $y = f_i(x)$ відповідно до свого варіанта в таблиці 2.5.3. Перевірити правильність рівнянь, побудувавши графіки функцій. Створити програму, що генерує координати точок прямокутника, рівномірно заповненого точками (20–30 точок по кожній осі) й накриває область зіниці. У разі потрапляння променя в зіницю в точці з координатами (x, y), вивести інформацію про це на екран. Уведення даних організувати з клавіатури.

Теоретичні відомості

Оператори порівняння та логічні операції мови С ++

Оператори порівняння призначені для порівняння між собою двох значень. У таблиці 2.2.1 наведені оператори порівняння мови С ++.

Таблиця 2.2.1 – Оператори порівняння

Символ операції	Значення	Використання
<	менше	a <b
<=	менше або дорівнює	a <= b
>	більше	a> b
> =	більше або дорівнює	a> = b

==	дорівнює	a == b
! =	не дорівнює	a! = b

У мові С ++ існує чотири основних логічних операції:

- логічна операція І;
- логічна операція АБО;
- логічна операція НІ або логічне заперечення;
- логічна операція ВИКЛЮЧНЕ АБО.

Логічні операції утворюють складне (складене) умова з декількох простих (два або більше) умов. Приклад умов показаний у таблиці 2.2.2.

Таблиця 2.2.2 – Логічні операції мови С ++

Операції	Позначення	Умова	Короткий опис
І	&&	a == 3 && b> 4	Складений умова істинно, якщо істинні обидва простих умови
АБО	||	a == 3 || b> 4	Складений умова істинно, якщо істинно, хоча б одна з простих умов

НІ	!	`!(A == 3)`	Умова істинно, якщо а НЕ дорівнює 3
ВИКЛЮЧНЕ АБО		`!(a == 3 || b> 4)`	Складена умова помилкова, якщо обидві простих умови істинні або помилкові

Операції порівняння та логічні операції в результаті дають значення типу bool, тобто true або false. Якщо ж такий вислів трапляється в контексті, що вимагає цілого значення, true перетвориться на 1, а false – на 0.

Умовний оператор if

Умовний оператор у мові С ++ має таку структуру:

```
if (логічний вираз) оператор_1;
[else оператор_2;]
```

Спочатку обчислюють значення логічного виразу, якщо воно не дорівнює нулю (true), виконується перший оператор, інакше – другий. Другу гілку оператора разом з else можна опустити, тоді, якщо вираз помилковий, відбудеться вихід із розгалуження.

Умовний оператор із двома гілками:

```
if (num <10)
{// якщо введене число менше 10
cout << "Це число менше 10." << endl;
}
```

```
    else {// інакше     cout << "Це число
більше або дорівнює 10" << endl;
    }
```

Умовний оператор з однією гілкою:

```
    if (num!= 10) // якщо введене число
не дорівнює 10
      cout << "Це число не дорівнює 10."
<< endl;
```

Поширені помилки

1. Використання в виразі під час перевірки рівності замість оператора (==) просте присвоєння (=).

2. Неправильний запис перевірки на приналежність діапазону. Умову $0 < x < 1$ необхідно записати так:

```
    if (0 <x && x <1)
```

Оператори циклу

Оператори циклу використовуються для організації багаторазово повторюваних обчислень. У мові С ++ існує три типи циклічних конструкцій:

- цикл із передумовою (while);
- цикл із післяумовою (do while);
- цикл із параметром (із заданою кількістю повторень (for)).

Виконання циклу з передумовою починається з умови, якщо воно істинне, виконується оператор циклу. Якщо в разі першої перевірки умова помилкова, то цикл не виконається жодного разу. Загалом цикл із передумовою наведено нижче:

```
    while (умова)
    {
    блок інструкцій
    }
```

Приклад:

```
int i = 1;     // ініціалізація
лічильника циклу while (i <= 10) //
виконуємо цикл поки i ≤ 10
{
cout << i * i << endl;
++ i;     // збільшення i на 1
}
```

Тіло циклу з післяумовою завжди виконують хоча б один раз. Спочатку виконують простий або складений оператор у тілі циклу, а потім перевіряють умову. Якщо умова істинна, тіло циклу виконують ще раз. Цикл завершується, коли умова стане хибним або в тілі циклу буде виконано оператор передачі управління.

Цикл із післяумовою в загальному вигляді

```
do {
блок інструкцій
}
while (умова);
```

Приклад:

```
int i = 0;   // ініціалізація лічильника
             циклу
int sum = 0; //  ініціалізація  лічильника
             суми
do {         // виконуємо цикл
     i ++;     sum + = i;
}
while (i <1000); // поки виконується умова
```

Цикл із параметром має такий вигляд:

```
for (ініціалізація; умова;
модифікації)
{
блок інструкцій
}
```

У частині ініціалізації можна записати кілька змінних, використовуваних у циклі, розділених комою. Цикл із параметром виконується як цикл із передумовою: спочатку перевіряють істинність умови, а потім виконують або не виконують тіло циклу.
приклад:

```
for (int i = 0; i <= 10; i ++) / * ініціалізація лічильника, умова, збільшення лічильника на 1 * / cout << i * i << endl; // тіло циклу
```

Поширені помилки

1. Використання в тілі циклу неініціалізованих змінних.

2. Неправильний запис умови виходу із циклу. Рекомендації щодо виконання завдання.

Для виконання завдання потрібно скласти нерівності, яким задовольняють координати точок усередині області зіниці. Рекомендовано розбити область на частини та скористатися симетрією заданої області зіниці.

Приклад виконання завдання

Для кожної лінії, що обмежує вхідну зіницю, зображену на рисунку 2.2.1, складемо рівняння $y = f_i(x)$. Визначимо умови потрапляння променя в область зіниці:

1. Умова потрапляння в I або III квадрант: $x \cdot y > 0$.

2. Умови потрапляння між двома похилими прямими:

$-(x+1) < y < -(x-1)$.

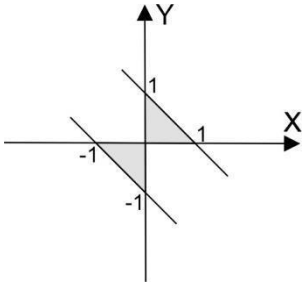


Рисунок 2.2.1 – Вхідна зіниця складної форми

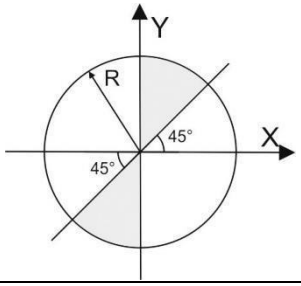
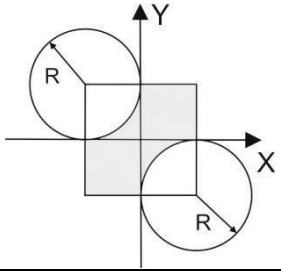
Програма, яка генерує координати точок прямокутника, що накриває область зіниці, рівномірно заповненого точками та визначає потрапляння променя до зіниці:

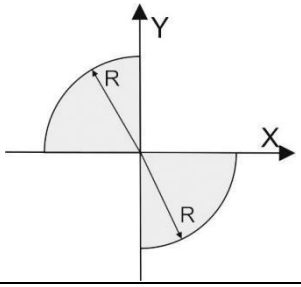
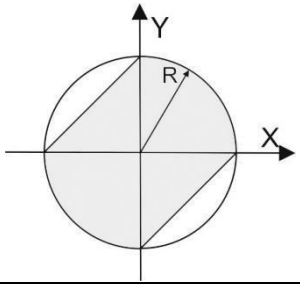
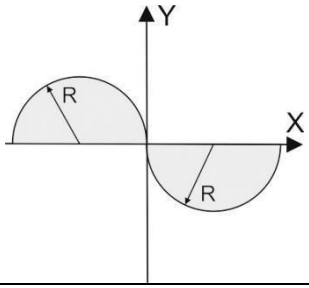
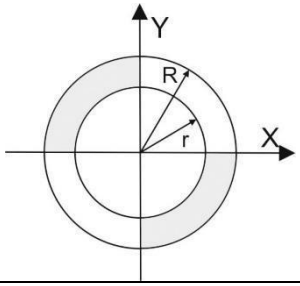
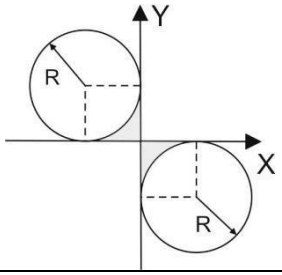
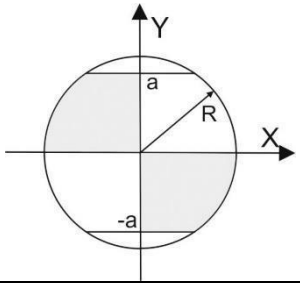
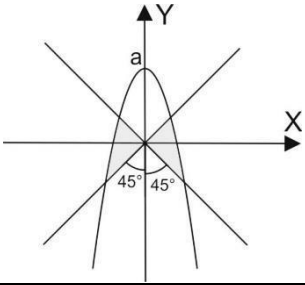
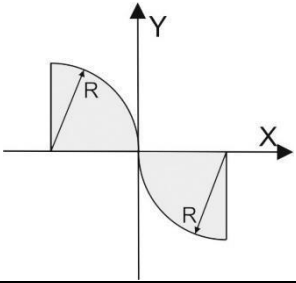
```
// лабораторна робота № варіант № група №
студент ПІБ
#include <iostream>
 using namespace std;
 int main () {
double x = 0.0, y = 0,0; // координати по
oci X і У double step = 0.1; // крок по
oci
    for (x = -1.5; x <= 1.5; x = x + step)
      {
            for (y = -1.5; y <= 1.5; y = y +
step)
           {
               if (((x * y)> 0) && (y> (-x -
1)) && (y <(-x + 1)))
// перевірка умови потрапляння в область
                {
                                cout << "Точка з
координатами x =" << x << ",
y = "<< y <<" потрапляє в область "<<
endl;
```

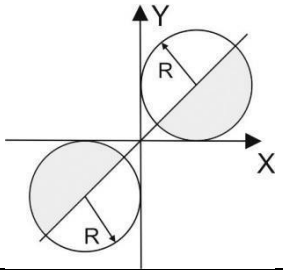
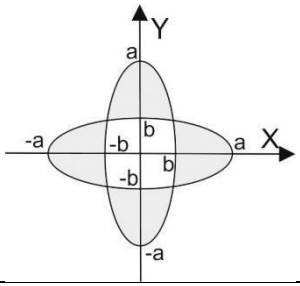
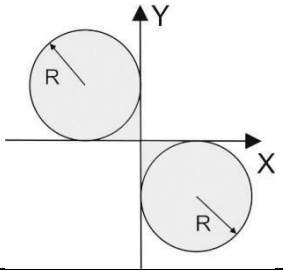
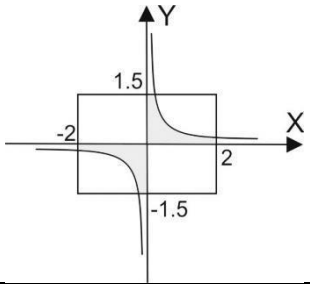
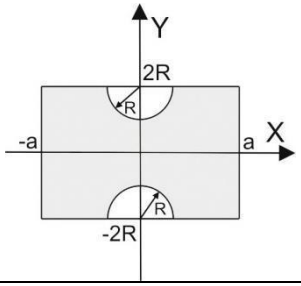
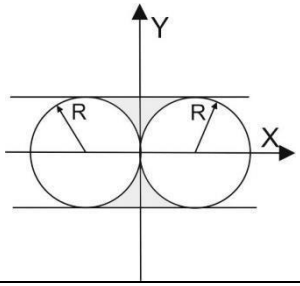
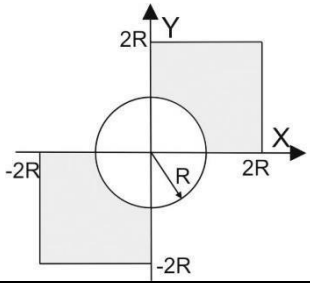
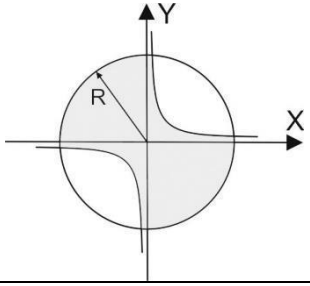
```
                }
                else
                {
                                cout << "Точка з
координатами x =" << x << ",
y = "<< y <<" не попадає в область "<<
endl;
                }
            }
        }
    cin.get ();
```

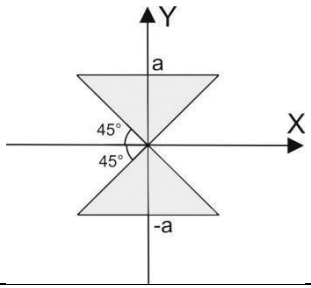
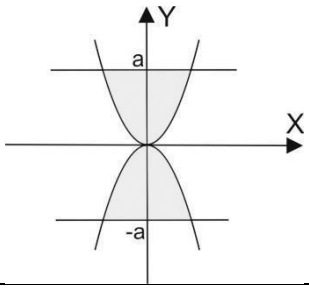
Варіанти завдань для роботи

Таблиця 2.2.3 – Варіанти завдань

№ пор.	Форма вхідної зіниці	№ пор.	Форма вхідної зіниці
1		2	

3	 Y X R R	4	 Y X R
5	 Y X R R	6	 Y X R r
7	 Y X R R	8	 Y X a R -a
9	 Y X a 45° 45°	10	 Y X R R

11		12	
13		14	
15		16	
17		18	

19		20	

2.3 Умовні та циклічні алгоритмічні конструкції. Робота з файлами. Побудова меж вхідної зіниці складної форми

Завдання для роботи

Для кожної ділянки вхідної зіниці складної форми відповідно до варіанта в таблиці 3.1 задати кордону та скласти рівняння $y = f_i(x)$. Написати програму табулювання функції $y = f_i(x)$. Одержані значення зберегти в текстовий файл. Побудувати графік функції за даними з файлу.

Теоретичні відомості

Робота з файлами в С ++

Для роботи з файлами використовують спеціальні типи даних, що називають потоками. У програмах на мові С ++ під час робот із текстовими файлами необхідно підключати бібліотеку fstream:

```
#include <fstream>
```

Потік ifstream служить для роботи з файлами в режимі читання, а ofstream у режимі запису. Для роботи з файлами в режимі як запису, так і читання служить потік fstream.

Для того, щоб записувати дані в текстовий файл, необхідно:

1) описати зміну типу ofstream;
2) відкрити файл за допомогою функції open;
3) вивести інформацію у файл;
4) обов'язково закрити файл.

Як було сказано раніше, для того, щоб почати працювати з текстовим файлом, необхідно описати змінну типу ofstream. Наприклад, так: ofstream outFile;

Буде створена змінна outFile для запису інформації у файлі. На наступному етапі файл необхідно відкрити для

запису. Загалом оператор відкриття потоку матиме вигляд outFile.open («file.txt», mode).

Тут outFile змінна, описана як ofstream, file.txt – повне ім'я файлу на диску, mode – режим роботи з файлом, що відкривається. Необхідно звернути увагу на те, що під час вказівки повного імені файлу потрібно ставити подвійний бекслеш. Для звернення, наприклад до файлу «accounts.txt», що знаходиться в папці sites на диску D, у програмі необхідно вказати «D:\\sites\\accounts.txt».

Функція open () вимагає як аргумент рядки в стилі C. Це може бути літеральний рядок або ж рядок, збережений у символьному масиві.

Файл може бути відкритий в одному з таких режимів:

• ios :: in – відкрити файл у режимі читання даних; режим є режимом за замовчуванням для потоків ifstream;

• ios :: out – відкрити файл у режимі запису даних (при цьому інформація про наявний файл знищується); режим є режимом за замовчуванням для потоків ofstream;

• ios :: app – відкрити файл у режимі запису даних в кінець файлу;

• ios :: ate – пересунутися в кінець уже відкритого файлу;

• ios :: trunc – очистити файл, це саме відбувається в режимі ios :: out;

• ios :: nocreate – не виконувати операцію відкриття файлу, якщо його не існує;

• ios :: noreplace – не відкривати наявний файл.

Параметр mode може бути відсутнім, у цьому разі файл відкривається в режимі за замовчуванням для даного потоку.

```
ifstream file;
file.open("Test.txt", ios::in); // відкрити файл в режимі для читання
ifstream file;
file.open("Test.txt");  // відкрити файл в режимі для читання (за замовчуванням)
```

Після вдалого відкриття файлу (в будь-якому режимі) в змінної outFile буде зберігатися true, в іншому випадку false. Це дозволить перевірити коректність операції відкриття файлу.

Для зчитування даних із текстового файлу, необхідно:

1) описати зміну типу ifstream;
2) відкрити файл за допомогою функції open;
3) зчитати інформацію з файлу, під час зчитування кожної порції даних необхідно перевіряти, чи досягнуто його кінеця;
4) закрити файл.

Для того щоб прочитати інформацію з текстового файлу, необхідно описати змінну типу ifstream. Після цього потрібно відкрити файл для читання за допомогою оператора open. Якщо змінну назвати fromFile, то перші два оператора будуть такими:

```
ifstream fromFile;
fromFile.open("D: \\ sites \\ accounts.txt");
```

Після відкриття файлу в режимі читання з нього можна зчитувати інформацію точно так, як і з клавіатури, вказавши ім’я потоку, із якого відбуватиметься читання даних.

Наприклад, для читання даних із потоку fromFile в змінну a, оператор уведення буде виглядати так: fromFile >> a;

Два числа в текстовому редакторі вважають розділеними, якщо між ними є хоча б один із символів: пробіл, табуляція, символ кінця рядка. Добре, коли програмісту заздалегідь відомо, скільки та які значення зберігаються в текстовому файлі, однак часто відомий лише тип таких значень, водночас їх кількість може бути різною. Для вирішення цієї проблеми необхідно зчитувати значення з файлу по черзі, а перед кожним зчитуванням перевіряти, чи досягнуто його кінця за допомогою функції fromFile.eof (). Тут fromFile – ім'я потоку, eof () – функція, що повертає логічне значення (true або false), залежно від того, чи досягнуто кінця файлу.

Отже, цикл для читання вмісту всього файлу можна записати так:

```
// організуємо для читання значень з файлу
цикл, виконання
// циклу перерветься, коли досягнемо
кінець файлу, // в цьому випадку
fromFile.eof () поверне істину while (!
FromFile.eof ())
{
// читання чергового значення з потоку
fromFile в змінну a      fromFile >> a;
// далі йде обробка значення змінної a
}
```

Для читання окремих символів можна використовувати функцію get () і функцію getline () – для читання цілих рядків.

Приклад виконання завдання

Задати та побудувати кордону зіниці складної форми згідно з графіком на рисунку 2.3.1.

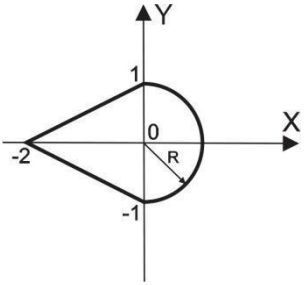


Рисунок 2.3.1 – Межі зіниці

```
// лабораторна робота № варіант № група № студент П І П/б
#include <fstream> // бібліотека для роботи з файлами
#include <cmath>
using namespace std;
int main() {
    const double PI = 2 * acos(0.); // розрахунок числа ПІ
        ofstream outFile; /*Створення потоку для виведення інформації у файл */
        outFile.open("points.txt");
// зв'язок потоку з файлом
    double x = 0.0, y = 0.0, alpha = 0.0;
// цикл по прямій верхній частині графіка

    for (x = -2; x <= 0; x ++)
    {
        y = 0.5 * x + 1;
        outFile << x << "" << y << endl; // запис в файл
    }
// цикл по дузі від PI / 2 до -PI / 2, alpha заданий у радіанах
    for (alpha = PI / 2; alpha >= - (PI / 2); alpha = alpha - 0.05)
```

```
    {
        x = cos(alpha);
        y = sin(alpha);
        outFile << x << "" << y << endl; //
запис в файл
    }
    //  цикл  по  прямій  нижній  частині
графіка
    for (x = 0; x >= -2; x--)
    {
        y = -(0.5 * x) - 1;
        outFile << x << "" << y << endl; //
запис в файл
    }
    // завершити роботу з файлом
    outFile.close ();
}
```

На рисунку 2.3.2 зображені графіки, побудовані в середовищі MathCad за даними, записаними у файл «points.txt» у результаті виконання програми.

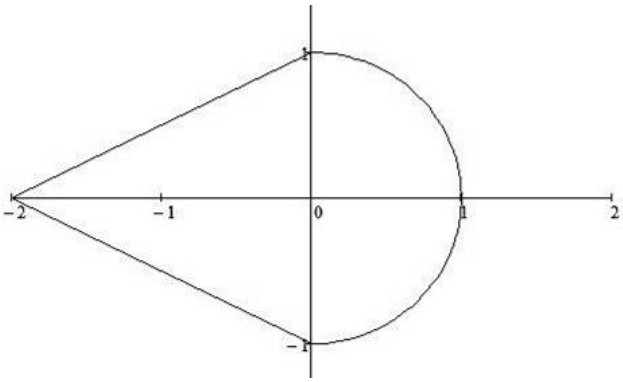
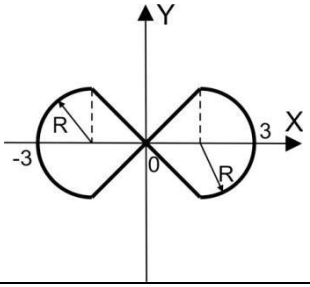
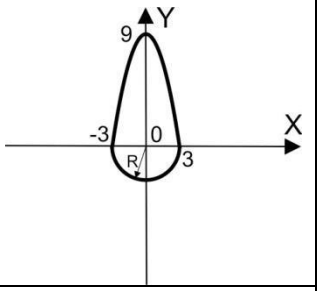
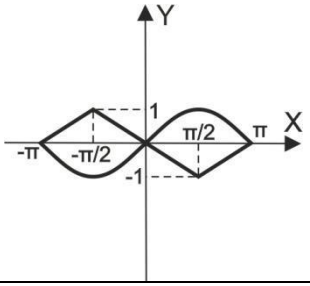
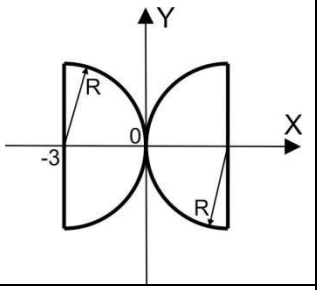
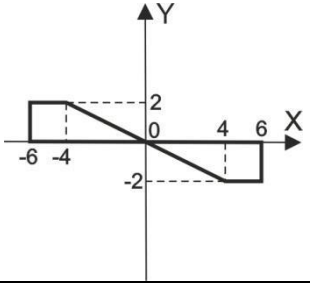
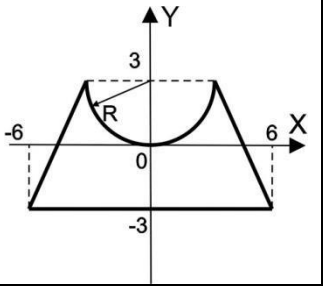
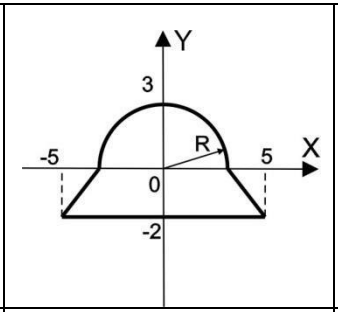
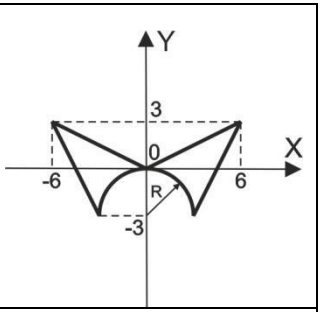
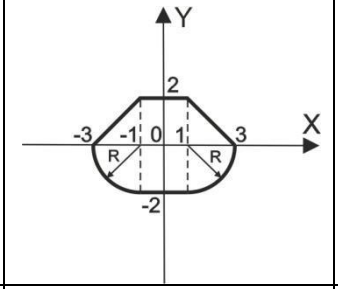
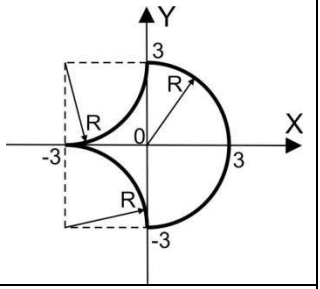
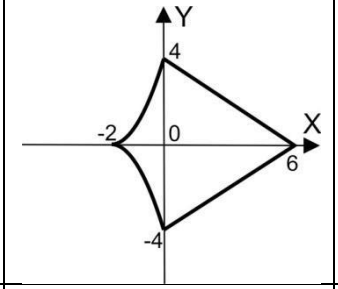
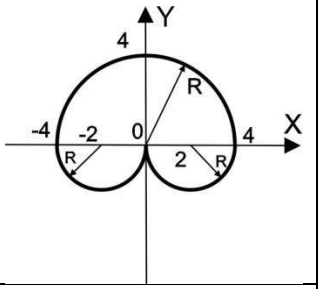
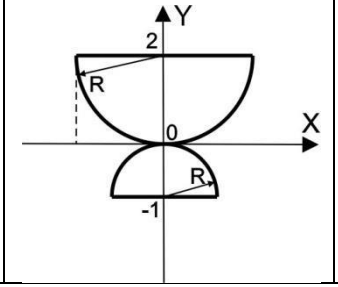
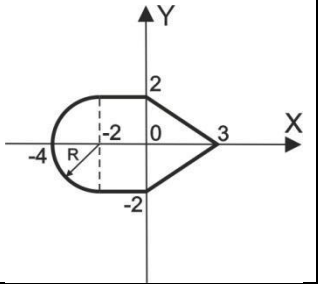
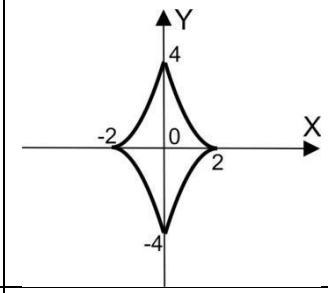
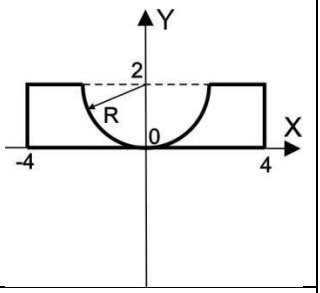
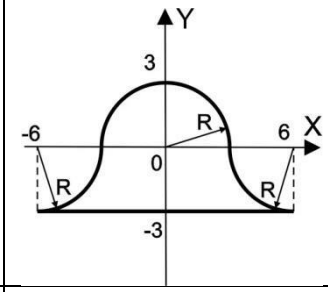
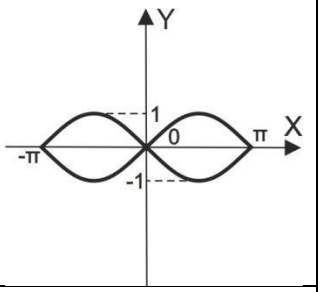
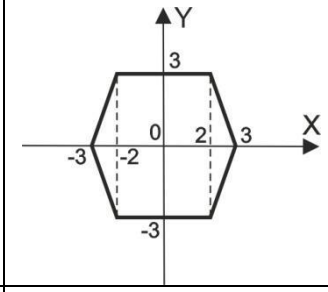
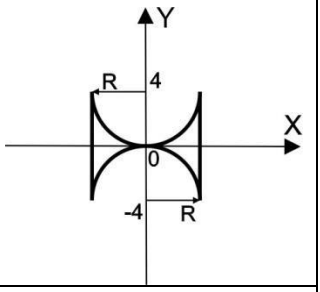
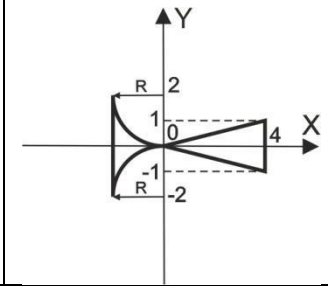
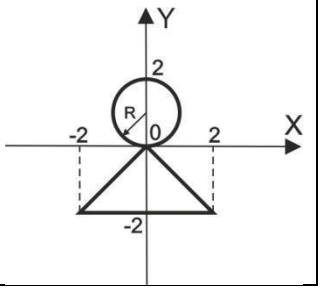


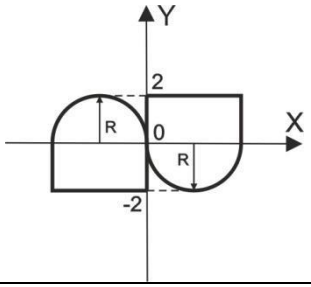
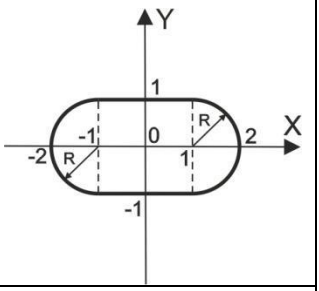
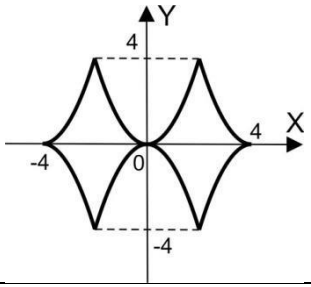
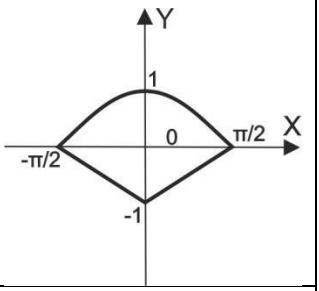
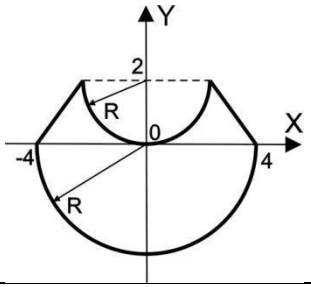
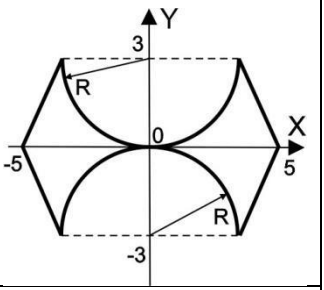
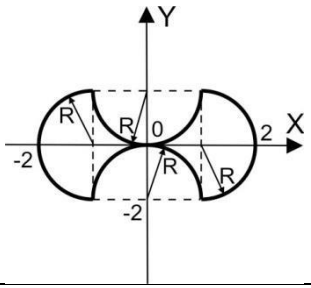
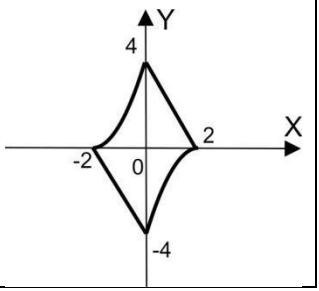
Рисунок 2.3.2 – Графіки функцій

Таблиця 2.3.1 – Варіанти завдань

№ пор.	Форма вхідного зіниці	№ пор.	Форма вхідного зіниці
1		2	
3		4	
5		6	

7		8	
9		10	
11		12	
13		14	

15		16	
17		18	
19		20	
21		22	

23		24	
25		26	
27		28	
29		30	

2.4 Робота з одновимірними масивами. Оброблення результатів вимірювань координат центру мас плями розсіювання

Завдання для роботи

Написати програму, яка:

1) зчитує з файлу в масив набір із 20 дійсних координат центрів мас плям розсіювання й виводить цей масив у результуючий файл. Серед координат повинні бути негативні, позитивні та рівні нулю. Розширені можливості пошуку, якщо це необхідно, ввести з клавіатури;

2) обчислює значення всіх змінних, що входять до складу виразу, і значення загального вираження відповідно до завдання в таблиці 4.5.2;

3) виводить значення всіх одержаних змінних і вирази з коментарями.

Теоретичні відомості

Робота з масивами в мові С ++

Масив – це структура даних, подана у вигляді послідовної групи значень одного типу, об'єднаних під одним ім'ям. Масиви використовують для оброблення великої кількості однотипних даних. Окреме значення даних масиву називають елементом масиву. Ними можуть бути дані будь-якого типу. Масиви, що мають один вимір, називають одновимірними.

Для створення масиву використовують оператор оголошення. Під час оголошення масиву необхідно вказати тип значень елементів масиву, його ім'я та розмірність (кількість елементів у масиві). Масиви бувають статичними й динамічними. Розмірність статичного масиву вказують під час його опису та разом із його типом визначають обсяг пам'яті, необхідний для розміщення масиву. Розмірність статичного масиву повинна бути цілочисленною константою або константним виразом, у якому відомі всі

значення на момент компіляції. Оголошення одновимірного статичного масиву на мові С ++ виглядає так:

```
double a [10]; // опис масиву з 10 елементів дійсного типу
```

Розмірність масивів краще ставити за допомогою іменованих констант.

На відміну від статичних масивів, у динамічних розмірність може бути змінної, тобто обсяг пам'яті, що виділяється під масив, визначається на етапі виконання програми.

Елементи масиву в мові С ++ нумерують із нуля, відповідно перший елемент масиву буде мати індекс 0, другий – 1, третій – 2 тощо.

У разі ініціалізації статичного масиву всі його елементи вказують по порядку у фігурних дужках під час оголошення масиву в програмі:

int b [3] = {1, 5, 4}; // b [0] = 1, b [1] = 5, b [2] = 4, b [3] = 0

Якщо під час ініціалізації у фігурних дужках буде вказано менше елементів, ніж розмірність масиву, то все решта елементи будуть рівні 0.

Для того, щоб привласнити значення окремого елемента масиву, використовують індекси (оператор []):

b [3] = 5; // b [0] = 1, b [1] = 5, b [2] = 4, b [3] = 0

Якщо необхідно заповнити масив своїми значеннями з клавіатури, із файлу або випадковими значеннями, використовують цикли. Приклад заповнення одновимірного масиву випадковими числами:

```
#include <iostream> // стандартний потік введення / виводу
#include <cstdlib> / * заголовки визначає кілька функцій загального призначення, в тому числі функції генерації випадкових чисел * / #include <ctime> // містить
```

```
функції для роботи з часом і датою using
namespace std;
int main () {
int randomDigits [3] = {}; // порожній масив
з трьох елементів
srand (time (NULL)); / * Установлює як базу
для генерації випадкових чисел поточний
час, отже, під час кожного запуску
генерується новий набір випадкових значень
* /
for (int i = 0; i <3; i ++) // цикл для
заповнення масиву
     {
          randomDigits [i] = rand (); / *
Функція rand () генерує випадкові числа * /
          cout << randomDigits [i] << endl;
     }
     return 0;
}
```

У наведеному вище прикладі масив із 3 елементів заповнений випадковими цілими числами.

Середні значення дійсних чисел

Середні значення n дійсних чисел $a1$, $a2$, $a3$, ..., an обчислюють за формулами наведеними в таблиці 2.4.1.

Таблиця 2.4.1 – Середні значення дійсних чисел

Середнє арифметичне	$A_n = \frac{a_1 + a_2 + a_3 + \cdots + a_n}{n}$
Середнє геометричне	$G_n = \sqrt[n]{a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_n}$

Середнє квадратичне	$Q_n = \sqrt{\frac{1}{n}(a_1^2 + a_2^2 + \cdots + a_n^2)}$
Середнє гармонійне	$M_n = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3} + \cdots + \frac{1}{a_n}}$, где $a_i \neq 0$

Обчислення середнього гармонійного потрібно проводити лише для ненульових елементів масиву.

Приклад програми

Обчислити характеристику координат центрів мас плям розсіювання за формулою $C = A + B$, де A – середнє арифметичне негативних координат (елементів масиву), великих D (D <0), B – номер мінімальної координати (елемента масиву) з тих, що мають непарний номер.

```
#include <fstream>
#include <iostream>

using namespace std;
 int main() {
     const int size = 20; // розмір масиву

     double M [size], A = 0.0, D = 0.0,
Mid = 0.0, C = 0,0;
     int B, i, n;
     ifstream inFile ( "In.txt"); // файл
із вихідним масивом
     ofstream outFile ( "Out.txt"); //
результуючий файл
     // введення масиву
     for (i = 0; i <size; i ++)
     {
         inFile >> M [i];
     }
```

```
    inFile.close ();
     // вивід масиву для контролю
    for (i = 0; i <size; i ++)
     {
         cout << M [i] << endl;
     }
     // пошук мінімального серед елементів
із номерами 1,3, ... 11  B = 1;
// нехай мінімальний - це перший елемент
масиву
for (i = 3; i <size, i + = 2)
     {
         if (M [i] <M [B])
            B = i; // оновити B, якщо M [B]
НЕ мінімум
     }
              outFile << "Номер мінімального
елемента з непарним номером =" << B <<
endl;
     // пошук середнього арифметичного
     cout << "Введіть D";
     cin >> D;
     n = 0; // лічильник для середнього
арифметичного
Mid = 0; // сума елементів для середнього
арифметичного
    for (i = 0; i <size; i ++)
     {
         if (M [i] <0 && M [i]> D)
          {               n ++;
             Mid + = M [i];
          }
     }
     if (n! = 0)
     {
```

```
          A = Mid / n;
           outFile << A << "Середнє
арифметичне>" << D << endl;
          C = A + B;
          outFile << "C =" << C;
     }
     else
     {
       outFile << "Негативних великих" << D
<< "ні";
     }
    outFile.close ();
}
```

Зміст звіту

Звіт до лабораторної роботи повинен містити:

1) текст завдання, варіант;
2) блок-схему програми;
3) листинг програми;
4) скриншоти результатів виконання програми;
5) вхідні й вихідні дані з файлів.

Варіанти завдань для роботи

Таблиця 2.4.2 – варіанти завдань

№ пор.	Вираз	Визначення змінних
1	$\frac{A+C}{B+C}$	A – середнє квадратичне позитивних координат (Елементів масиву); B – кількість ненульових координат (елементів масиву); C – твір ненульових координат (елементів масиву)

2	$\frac{A}{A+1}+C+B$	A – середнє гармонійне негативних координат (елементів масиву) великих D; B – сума координат (елементів масиву) з парним номером; C – номер останньої позитивної координати (елемента масиву).
3	$\left(1+\frac{1}{A}+\frac{1}{B}\right)C$	A– кількість координат (елементів масиву) менших D з непарним номером; B – середнє арифметичне перших M координат (елементів масиву); C – модуль найменшою координати (елемента масиву)
4	$\frac{1}{A+B+C}+3$	A – найбільша парна координата (елемент масиву); B – середнє квадратичне всіх координат (елементів масиву); C – кількість координат (елементів масиву) з інтервалу [A, B]
5	$\frac{A+B+C}{A\cdot B\cdot C}+1$	A – середнє гармонійне перших M координат (елементів масиву) не дорівнює 0; B – номер найменшою за модулем ненульовий координати (Елемента масиву); C– кількість координат (елементів масиву) менших D

6	$\frac{A+B+C}{100}+A$	A – найбільша координата (елемент масиву) з непарним номером; B – середнє гармонійне координат (елементів масиву) з парним номером; C – сума позитивних координат (елементів масиву) з парними номерами
7	$\frac{A \cdot B}{C+2}$	A – сума негативних координат (елементів масиву) великих D; B – кількість позитивних координат (елементів масиву) з непарним номером; C – сума позитивних координат (елементів масиву)
8	$\frac{A \cdot B \cdot C}{A+B+C}$	A – сума координат (елементів масиву) з інтервалу [A, B]; B–- середнє квадратичне координат (елементів масиву) з парним номером; C– номер останньої негативною координати (елемента масиву)
9	$\frac{A}{(B+1)(C+1)}$	A– номер найменшою позитивною координати (елемента масиву); B – кількість нульових координат (елементів масиву); C – середнє гармонійне позитивних координат (елементів масиву)

10	$\frac{A}{10}+\frac{C+B}{10}$	A – кількість негативних координат (елементів масиву) з непарним номером; B – сума координат (елементів масиву) з непарним номером; C – номер першої позитивної координати (елемента масиву)
11	$\frac{B}{A+1}+C$	A – середнє арифметичне негативних координат (елементів масиву); B – твір негативних координат (елементів масиву); C – сума модулів негативних координат (елементів масиву)
12	$\frac{B\cdot C+A}{A\cdot B+2}$	A – найбільша за модулем координата (елемент масиву); B – сума негативних координат (елементів масиву); C – твір модулів негативних координат (елементів масиву)
13	$C+\frac{A}{100+B}$	A – сума перших M координат (елементів масиву); B – номер останньої нульовий координати (елемента масиву); C – кількість негативних координат (елементів масиву)
14	$(A+1)\,(B+2)\,(C+3)$	A – сума координат (елементів масиву) великих 0 з непарним номером; B – середнє арифметичне координат (елементів масиву); C – твір перших M координат (елементів масиву) не дорівнює 0

15	$\frac{C}{(A+1)\,(B+1)}+C$	A – середнє арифметичне позитивних координат (елементів масиву); B – сума всіх координат (елементів масиву); C – твір координат (елементів масиву) з непарним номером
16	$\frac{A \cdot C + B}{B \cdot C + A}$	A – сума координат (елементів масиву) великих D з непарним номером; B – номер останньої негативною координати (елемента масиву); C– кількість позитивних координат (елементів масиву).

2.5 Робота з двовимірними масивами. Попіксельне оброблення цифрових зображень

Завдання для роботи

Написати програму, що зчитує з текстового файлу значення інтенсивностей пікселів цифрового зображення (значення інтенсивності пікселя в діапазоні від 0 до 255), заповнює ними двовимірний масив $A\,[m, n]$, виконує його оброблення відповідно до варіанта завдання в таблиці 2.5.1 і виводить із формату в результуючий файл такі дані: розмір зображення (масиву), вихідні та одержані значення інтенсивностей усіх пікселів обробленого зображення (вихідний і результуючі масиви).

Розмір оброблюваного зображення не повинен перевищувати 20 × 20 пікселів.

Теоретичні відомості

Двовимірний масив – це одновимірний масив одновимірних масивів. Основні характеристики двовимірного масиву: кількість рядків і кількість стовпців. У пам'яті двовимірні масиви зберігаються послідовно по рядках.

Елементи двовимірного масиву, як і одновимірного нумерують із нуля. Оголошення масиву повинно мати таку послідовність:

- тип значень;
- ім'я масиву;
- розмірність (кількість елементів у масиві, а саме – кількість рядків і стовпців).

Під час вказівки розмірності двовимірного масиву обидві розмірності вказують у квадратних дужках: matr [6] [8]; // опис масиву з 6 рядків і 8 стовпців.

Якщо під час ініціалізації двовимірний масив задається як масив із масивів, тоді першу розмірність, а саме кількість рядків, можна опустити: matr [] [2] = {{1,1}, {0,2}, {1,0}}.

Також під час ініціалізації двовимірного масиву можна вказати елементи списком через кому в тому порядку, у якому вони знаходяться в пам'яті. Водночас обидві розмірності масиву необхідно вказати.

```
matr [3] [2] = {1, 1, 0, 2, 1, 0};
```

Для доступу до будь-якого елемента двовимірного масиву вказують усі його індекси:

```
matr [i] [j]; // i - номер рядка,
j - номер стовпця matr [2] [2] = 5; // елемент у 3 рядку і 3 стовпці дорівнюватиме 5.
```

Приклад програми

Усі пікселі чорного кольору (значення інтенсивності 0) в оригінальному документі поміняти на пікселі білого кольору (значення інтенсивності 255).

```
#include <fstream>
#include <iostream>
 using namespace std;
 int main () {
// розміри двовимірного масиву
const int rows = 4; // кількість рядків
const int cols = 4; // кількість стовпців
   int A [rows] [cols];
     ifstream inFile ( "In.txt"); // файл із вихідним масивом     ofstream outFile ( "Out.txt"); // результуючий файл
  // Уведення в двовимірний масив
  for (int i = 0; i <Rows; i ++) // цикл по рядках
     {
          for (int k = 0; k <Cols; k ++) // цикл по стовпцях
           {
```

```
                inFile>> A [i] [k]; //
запис значення в елемент масиву
            }
        }
        inFile.close (); //
закриваємо файл начитання
outFile << "Вихідна матриця:"
<< endl;
       // вивід масиву у файл
  for (int i = 0; i <Rows; i ++) // цикл по
рядках
        {
             for (int k = 0; k <Cols; k ++) //
цикл по стовпцях
             {
outFile << A [i] [k] << ""; // записуємо
значення елемента масиву у файл
             }
            outFile << endl;
        }
  // замінюємо елементи масиву, рівні 0, на
255
  for (int i = 0; i <Rows; i ++) // цикл по
рядках
        {
             for (int k = 0; k <Cols; k ++) //
цикл по стовпцях
             {
 if (A [i] [k] == 0) // якщо елемент
масиву дорівнює 0, то замінюємо значення
на 255
                  {
                      A [i] [k] = 255;
                  }
             }
```

```
      }
        outFile << "Результуюча матриця:" <<
endl;
  for (int i = 0; i <Rows; i ++) // цикл по
рядках
      {
           for (int k = 0; k <Cols; k ++) //
цикл по стовпцях
           {
               outFile << A [i] [k] << "";
           }
          outFile << endl;
      }
     outFile.close ();
      return 0;
}
```

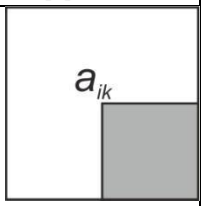
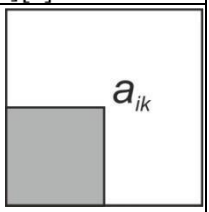
Зміст звіту

Звіт до лабораторної роботи повинен містити:

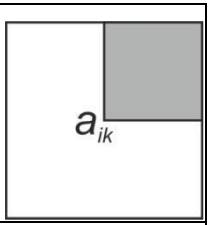
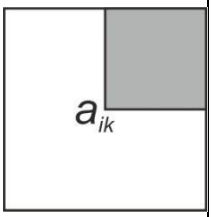
1) текст завдання, варіант;
2) вихідний і результуючий масиви;
3) лістинг програми;
4) скриншоти результатів виконання програми.

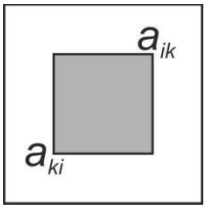
Варіанти завдань для роботи

Таблиця 5.1 – Варіанти завдань

№ пор.	Завдання
1	Відобразити вихідне зображення за горизонталлю: у кожному рядку вихідного цифрового зображення значення інтенсивності кожного пікселя переставити у зворотному порядку [0][0] [0][1] [0][2] [0][2] [0][1] [0][0] [1][0] [1][1] [1][2] → [1][2] [1][1] [1][0] [2][0] [2][1] [2][2] [2][2] [2][1] [2][0]
2	Сформувати цифрове зображення B, елемент *bi, j* якого дорівнює сумі елементів вихідного цифрового зображення A з області, що визначає індексами i, j 
3	Повернути зображення на 90 ° вправо [0][0] [0][1] [0][2] [2][0] [1][0] [0][0] [1][0] [1][1] [1][2] → [2][1] [1][1] [0][1] [2][0] [2][1] [2][2] [2][2] [1][2] [0][2]
4	Сформувати цифрове зображення B, елемент $b_{i,j}$ якого дорівнює середньому арифметичному елементів вихідного цифрового зображення A з області, що визначає індексами i, j 
5	Порахувати кількість пікселів на зображенні із кожним можливим рівнем інтенсивності. Сформувати одновимірний масив, що містить

	інформацію про кількість елементів із кожним можливим рівнем інтенсивності (0-255)
6	Виконати бінаризацію зображення: усі значення інтенсивності менше D замініть на 0, решта на 255. Значення D ввести з клавіатури за запитом
7	Усі значення інтенсивності елементів вихідного цифрового зображення більше D_{max} змінити на максимально можливе значення інтенсивності (255), а всі значення менше D_{min} на мінімальне значення інтенсивності (0). Значення D_{max} і D_{min} ввести з клавіатури за запитом
8	Збільшити значення інтенсивності кожного пікселя зображення на D. Якщо нове значення буде перевищувати максимальне значення інтенсивності 255, прийняти його рівним 255
9	Відобразити вихідне зображення за вертикаллю: в кожному стовпці вихідного цифрового зображення всі значення інтенсивності переставити у зворотньому порядку $\begin{matrix}[0][0] & [0][1] & [0][2] \\ [1][0] & [1][1] & [1][2] \\ [2][0] & [2][1] & [2][2]\end{matrix} \rightarrow \begin{matrix}[2][0] & [2][1] & [2][2] \\ [1][0] & [1][1] & [1][2] \\ [0][0] & [0][1] & [0][2]\end{matrix}$
10	Сформувати цифрове зображення, кожен елемент якого $b_{i,j}$ буде середнім арифметичним елемента вихідного цифрового зображення $a_{i,j}$ і всіх елементів, що його оточують
11	Повернути зображення на 90 ° вліво $\begin{matrix}[0][0] & [0][1] & [0][2] \\ [1][0] & [1][1] & [1][2] \\ [2][0] & [2][1] & [2][2]\end{matrix} \rightarrow \begin{matrix}[0][2] & [1][2] & [2][2] \\ [0][1] & [1][1] & [2][1] \\ [0][0] & [1][0] & [2][0]\end{matrix}$

12	Сформувати цифрове зображення B, елемент $b_{i,j}$ якого дорівнює сумі елементів вихідного цифрового зображення A з області, що визначає індексами i, j 
13	Знайти всі значення інтенсивності вихідного цифрового зображення рівні D, порахувати їх кількість, замінити їх на 255, інші значення замінити на 0 Значення D ввести з клавіатури за запитом
14	Перевернути зображення щодо верхнього лівого кута (переставити елементи вихідного цифрового зображення у зворотному порядку) $\begin{matrix}[0][0] & [0][1] & [0][2] \\ [1][0] & [1][1] & [1][2] \\ [2][0] & [2][1] & [2][2]\end{matrix} \rightarrow \begin{matrix}[2][2] & [2][1] & [2][0] \\ [1][2] & [1][1] & [1][0] \\ [0][2] & [0][1] & [0][0]\end{matrix}$
15	Виконати фільтрацію, у результаті якої сформувати нове цифрове зображення, у якому кожен елемент $b_{i,j}$ буде серединою одновимірного масиву відсортованого за зростанням і складається з елемента $a_{i,j}$ і всіх елементів, що його оточують
16	Сформувати масив інтенсивностей пікселів зображення розмірністю m × n який є центральною частиною вихідного цифрового зображення
17	Сформувати цифрове зображення, кожен елемент якого $b_{i,j}$ буде середнім арифметичним елемента вихідного цифрового зображення $a_{i,j}$ і елементів що стоять у зображенні праворуч і ліворуч від нього
18	Сформувати цифрове зображення B, елемент $b_{i,j}$ якого дорівнює середньому арифметичному елементів вихідного цифрового зображення A з області, що визначають індексами i, j 

19	Сформувати цифрове зображення B, елемент $b_{i,j}$ якого дорівнює сумі елементів вихідного цифрового квадратного зображення A з області, що визначає індексами i, j 
20	Виконати фільтрацію, у результаті якої сформувати цифрове зображення, у якому кожен елемент $b_{i,j}$ буде серединою одновимірного масиву відсортованого за зростанням і складається з елемента вихідного цифрового зображення $a_{i,j}$ і всіх елементів, що його оточують
21	Виконати оброблення цифрового зображення: усі значення елементів, що лежать в діапазоні від D_{min} до D_{max}, замінити на 255, усі інші елементи вихідного зображення залишити без змін. Значення D_{max} і D_{min} ввести з клавіатури за запитом
22	Перетворити вихідне зображення (масив значень інтенсивностей пікселів) на негатив. Для цього кожен елемент цифрового зображення змінити за формулою: $b_{i,j} = 255 - a_{i,j}$
23	Вважати з вихідного файлу два цифрових зображення однакової розмірності та порівняти їх поелементно: якщо значення елемента першого зображення більше або дорівнює відповідному елементу другого зображення, то записати в результуюче зображення 255, якщо менше – 0

2.6 Робота з рядками в мові С++

Завдання для роботи

Написати програму, яка зчитує з файлу вихідний текст, згідно із завданням у таблиці та формує новий текст. Одержаний текст записати у файл.

Вихідний текст складається з рядка, що містить трохи більше 200 символів. Наприкінці пропозиції є крапка. Кожному слову, крім першого, передує одна прогалина, усередині пробілів немає. Розділові знаки, якщо вони є, пишуть відразу після слова.

Теоретичні відомості

Рядки мовою С++

Рядок – масив символів, що закінчується нуль-символом.

```
char a [5] = { 'a' , 's' , ' ' , '1' , '!' , \0 };
```

Оголошення рядків аналогічне одновимірним масивам типу char. Ініціалізувати рядок можна рядковим літералом (константою), під час ініціалізації рядка при його визначенні розмірність можна опустити. У цьому разі буде виділено необхідну кількість байт для цього рядка. char str [10] = «Vasia»; // ініціалізація рядка літералом.

Для рядка буде виділено 10 елементів від 0 до 9. Перші 6 елементів рядка: 'V', 'a', 's', 'i', 'a', '\0'.

```
char str [] = "Vasia"; //при ініціалізації опущено розмірність рядка.
```

Для рядка буде виділено 6 елементів від 0 до 5: 'V', 'a', 's', 'i', 'a', '\0'. Символьні рядки складаються з набору символьних констант узятих у подвійні лапки. Під час оголошення рядкового масиву необхідно враховувати наявність кінці кінці нуль-символу та відводити додатковий байт під нього.

У мові С++ існує дві можливості роботи з рядками.

1. Функції, успадковані з бібліотеки мови С (<cstring>) Бібліотека <cstring> містить функції копіювання рядків, порівняння, об'єднання рядків, пошуку підрядка, пошуку входження символу, визначення довжини рядка та інші.

2. Бібліотечний клас мови С++ string.

Функції бібліотеки <cstring>

Для визначення довжини рядка в бібліотеці <cstring> існує функція strlen, яка визначає довжину вказаного рядка без урахування нуль-символу. strlen (ім'я_рядки);

```
char str [10] = "Vasia";
a = strlen (str); //змінною a буде
присвоєно значення 5.
```

Для копіювання рядків реалізовано дві функції: strcpy та strncpy. Перша виконує побайтне копіювання символів з рядка_2 до рядка_1. Друга функція виконує побайтне копіювання n символів із рядка_2 до рядка_1.

```
strcpy (рядок_1, рядок_2); strncpy
(рядок_1, рядок_2, n);

char str_1[] = "Бути чи не бути"; char
str_2[10];
char str_3[10];

strncpy(str_2, str_1); //Копіює рядок str_1
в рядок str_2 strncpy(str_3, str_1, 8); /*
копіює у рядок str_3 8 символів із str_1 */
```

Щоб уникнути переповнення, рядок, на який указують str_2 і str_3 повинні бути достатньо довгими, щоб у них помістився рядок, що копіюється (включаючи зокрема завершальний нульовий символ). Рядок і рядок

призначення, що копіюється, не повинні перекриватися в пам'яті.

Конкатенацію рядків здійснюєть за допомогою функцій strcat та strncat:

```
strcat (рядок_1, рядок_2); /* поєднує
рядок_2 з рядком_1. Результат зберігається
в рядку_1 */
strncat(рядок_1,рядок_2,n); /* поєднує n
символів рядка_2
сострокой_1. Результат зберігається в
рядку_1 */ char str_1[] = "Hello "; char
str_2[] = "World!";
strcat(str_1, str_2); //В str_1 буде
"Hello World!".
```

Функція додає копію рядка_2 до кінця рядка_1. Нульовий символ кінця рядка_1 замінюється першим символом рядка_2, і новий нуль-символ додається до кінця вже нового рядка, сформованого об'єднанням символів двох рядків у рядку_1.

Бібліотека <cstring> дозволяє порівнювати рядки між собою з урахуванням або без урахування регістру. Функція strcmp порівнює рядок_1 із рядком_2 з урахуванням регістру й повертає результат типу int: 0 – рядки еквівалентні, більше 0 – рядок_1 менше рядка_2, менше 0 – рядок_1 більше рядка_2: strcmp (рядок_1, рядок_2);

Функція strncmp порівнює n символів рядка_1 із рядком_2 з урахуванням регістру та повертає результат типу int: 0 – рядки еквівалентні, більше 0 – рядок_1 менше рядка_2, менше 0 – рядок_1 більше рядка_2: strncmp (рядок_1, рядок_2, n);

З першого символу попарно порівнює всі символи двох рядків, доки знайдено різні символи чи буде досягнуто кінця рядка.

Функції для порівняння без урахування регістру працюють аналогічно функціям порівняння рядків з його врахуванням:

stricmp(рядок_1, рядок_2);

strnicmp(рядок_1, рядок_2, n);

У бібліотеці <cstring> існують такі функції пошуку:

strchr (рядок, символ).

Функція здійснює пошук першого входження символу в рядку. У разі успішного пошуку повертає покажчик на місце першого входження символу. Якщо символ не знайдено, повертається нуль.

Завершальний нульовий символ вважають частиною рядка. Отже, він може бути знайдений для одержання покажчика на кінець рядка. strcspn (рядок_1, рядок_2).

Визначає довжину початкового сегмента рядка_1, що містить символи, які не входять у рядок_2. Пошук враховує також і завершальний нуль-символів, тому якщо функція повертає довжину рядка_1, це означає, що жоден із символів рядка_2 не входить до складу рядка_1. strspn (рядок_1, рядок_2);

Функція strspn повертає довжину початкового сегмента рядка_1, що містить лише ті символи, які входять до рядка_2. Пошук ураховує й завершальний нуль-символ, тому, якщо функція повертає довжину рядка_1, це означає, що всі символи рядка_2 входять до складу рядка_1.**strprbk(s1, s2).**

Повертає покажчик першого входження будь-якого символу рядка_2 у рядку_1.

За потреби можна перетворити значення рядка на інший тип даних. Для перетворення рядка на тип double необхідно використовувати наступну таку функцію atof (рядок_1);

Для перетворення на тип int:

atoi (рядок_1).

Для перетворення на тип longint:

```
atol(рядок_1);
```

Функції стандартної бібліотеки введення / виведення

p align = "justify" > Для роботи з рядками в мові C++ також використовують функції стандартної бібліотеки введення / виведення: getchar(c);

Функція getchar зчитує символ зі стандартного потоку введення, повертає символ у форматі int. gets(s);

Функція gets зчитує потік символів зі стандартного пристрою введення в рядок s, доки не буде натиснута клавіша ENTER.

Функції бібліотеки <cctupe> для оброблення символів

У таблиці 2.6.1 наведено функції, які дають можливість перевірити належність символу до того чи іншого типу.

Таблиця 2.6.1 – Функції оброблення символів

Функція	Опис функції
`isalnum(c)`	Повертає значення true, якщо є буквою або цифрою, і false в інших випадках
`isalpha(c)`	Повертає значення true, якщо є буквою, і false в інших випадках
`isdigit(c)`	Повертає значення true, якщо є цифрою, і false в інших випадках
`islower(c)`	Повертає значення true, якщо є буквою нижнього регістру, і false в інших випадках

isupper(c)	Повертає значення true, якщо є буквою верхнього регістра, і false в інших випадках
isspace(c)	Повертає значення true, якщо є пробілом, і false в інших випадках
toupper(c)	Якщо символ c є символом нижнього регістру, то функція повертає перетворений символ у верхньому регістрі, інакше символ повертається без змін

Приклад програми

У вихідному рядку довжиною 70 символів – пропозиція, що складається не менше ніж із двох слів. Сформувати новий рядок, у якому з останнього слова речення видалено першу літеру.

```
#include <iostream>
#include <cstdio>
 int main() {
using namespace std;
     char  StIn[70],  StOut[70];  // Вхідний і вихідний рядки
     int Nend, N1; // Позиції в тексті
     cout << "Уведіть текст: ";
gets_s(StIn); // функція gets() зчитує всі введені символи з пробілами доти, доки натиснута клавіша Enter.
Nend  =  strlen(StIn);  //  Поверне довжину рядка
char * pch = strrchr (StIn, ''); // Повертає покажчик на останнє входження символу у рядку
```

```
    N1 = (pch - StIn + 1); // кількість
символів до останньої пробілу
          // скопіювати в результуючий
рядок N1 символів  strncpy_s(StOut, StIn,
N1);
      // додати останнє слово без першої
літери
      strncat_s (StOut, & StIn [N1 + 1],
(Nend - N1 - 1));
      cout << StOut;
      cin.get();
     }
```

Зміст звіту

Звіт до лабораторної роботи повинен містити:

1) текст завдання, варіант;
2) вихідний та результуючий тексти;
3) лістинг програми;
4) скриншоти результатів виконання програми.

Варіанти завдань для роботи

Таблиця 2.6.2 – Варіанти завдань

№ пор.	Початковий текст	Завдання
1	Для зручності читання оптичних схем та комп'ютерних розрахунків в оптиці прийнято єдині правила знаків. Позитивним напрямом світла вважають поширення зліва направо	Поміняти речення місцями
2	Хвильова аберація пропорційна відхиленням оптичних довжин променів пучка	Слова розділити комами

3	Якщо розкласти поле на монохроматичні складові (кожна з певною довжиною хвилі), то вся енергія певним чином розподілиться між ними	Текст у дужках перенести до кінця речення (перед крапкою)
4	В ідеальній оптичній системі всі промені, що виходять із точки, перетинаються у сполученій із нею точці	Видалити другу кому
5	Кома з'являється при зміщення точки предмета з осі. Кома додається	Слова другого речення поміняти місцями
6	Тому вплив хвильової аберації на якість зображення залежить від типу зображення а визначається тим, скільки довжин хвиль вона становить	Вставити кому, що бракує, перед сполучником "а"
7	Апланатизм може виконуватися лише для частини предмета. На околиці осі	Друге та третє слова другого речення поміняти місцями
8	Спектральна щільність потоку випромінювання показує розподілення енергії за спектром випромінювання. Поверхнева густина потоку енергії – це величина потоку, що припадає на одиницю площі	Перші слова речень поміняти місцями

9	Світловий потік, поширюючись в оптичному середовищі, частково поглинається	Вивести на друк частину тексту між комами
10	Монохроматичні аберації: сферична, кома, астигматизм та кривизна зображення, дисторсія	Третє та останнє слова речення поміняти місцями
11	Поверхні: пласкі, сферичні, асферичні	Кожне слово з нового рядка
12	Абсолютно чорне тіло – це тіло, яке повністю поглинає енергію, що падає на нього	Слова до та після дефісу поміняти місцями
13	Енергетичний коефіцієнт пропускання	Новий порядок слів: 2,3,1
14	Дисперсія оптичних матеріалів	Середнє слово – у круглі дужки
15	Оптична поверхня – це регулярна гладка поверхня точно відомої форми. Оптичне середовище – прозоре однорідне середовище з точним значенням показника заломлення	Другі слова речень поміняти місцями
16	Ідеальна оптична система	Новий порядок слів: 3,2,1
17	Кут між променем та оптичною віссю вважається позитивним, якщо для поєднання осі з променем	Замінити слово «вісь» словом «її»

	вісь потрібно обертати за годинниковою стрілкою	
18	Геометрична теорія оптичних зображень	Новий порядок слів: 4,3,2,1
19	У назві «неізопланатизм» є коріння грецьких слів: з – однаковий, рівний, планета – блукаюче тіло	Видалити слово разом із лапками
20	Якщо квадратний предмет зображується як подушки – це дисторсія позитивна. Якщо зображення квадрата має опуклі сторони (як бочки), це дисторсія негативна	Останні слова речень поміняти місцями

2.7 Використання функцій мови С++. Розв'язання оптичних задач

Завдання для роботи

Створити програму, що містить функцію для розв'язання оптичної задачі згідно з варіантом у таблиці 7.1.

Програма повинна містити функції для:

- уведення вихідних даних;
- розрахунку необхідних оптичних величин (у головній програмі необхідно передбачити можливість звернення до функції із різними вихідними даними);
- виведення результатів розв'язання задачі з коментарями.

Виведення всієї інформації в результуючий файл здійснювати за форматом. Організувати перевірку правильності введення даних користувачем.

Теоретичні відомості

Функції у мові С++

Функція – це іменована послідовність описів та операторів, зазвичай призначена на вирішення певної задачі. Функція може приймати параметри та повертати значення.

Будь-яка програма на З складається з функцій, одна з яких повинна мати ім'я main. Із функції main починається виконання програми. Виконання функції починається під час її виклику. Як і змінна, функція повинна бути оголошена та визначена, водночас оголошення функції повинне знаходитись у тексті програми раніше, ніж її виклик.

Під час оголошення функції вказують її ім'я, тип значення, що повертається, і список параметрів, що передаються в неї: int sum (int a, int b); // Оголошення функції

Визначення функції містить, крім оголошення, тіло функції, що є послідовністю описів та операторів у фігурних дужках:

```
int sum (int a, int b)
{ // Визначення функції
return (a+b);
}
```

В оголошенні, визначенні та виклику однієї й тієї самої функції типи та порядок слідування параметрів повинні збігатися. На імена параметрів обмеження не накладаються.

Для того, щоб викликати функцію, необхідно вказати її ім'я, після якого в круглих дужках через кому перераховують імена аргументів, що передаються у функцію:

```
#include <iostream>
int sum (int a, int b); //Оголошення функції
int main(){
 int a = 2, b = 3, c, d;
c = sum(a, b); /*виклик функції та передача як вхідні параметри значень змінних a та b*/
 cin >> d;
 cout << sum (c, d); /*виклик функції та передача як
вхідних параметрів значень змінних c та d*/
 return 0;
}
```

Усі величини, описані у функції, та її параметри є локальними, областю їх дії є функція. Під час виходу з функції значення локальних змінних не зберігаються, оскільки ділянка стека звільняється. Щоб уникнути цього використовують параметр static.

Під час спільної роботи функції повинні обмінюватись інформацією. Це можна здійснити за допомогою:

- глобальних змінних;
- через параметри;
- через значення, що повертається функцією.

Повернення з функції у функцію, що викликала її, реалізується оператором return: return [вираз].

Функція може містити кілька операторів `return`, проте її робота припиняється під час першого можливого оператора return. Якщо функцію описано як `void`, вираз не вказують. Оператор return можна опускати для функцій типу void, якщо повернення з неї відбувається перед фігурною дужкою, і функції main.

```
int function_1 {
 return 1;
}
double function_2 {
  return 1; //1 перетворюється на тип
double
}
```

ВАЖЛИВО: Не можна повертати вказівник на локальні змінні з функції, оскільки пам’ять, виділена локальним змінним під час входження у функцію, звільняється після повернення з неї.

Параметри функції

У функціях є два типи параметрів: формальні та фактичні. Формальні параметри (параметри) – це параметри, перелічені в заголовку опису функції, їх називають просто параметрами. Фактичні параметри (аргументи) – це параметри, записані в операторі виклику функції.

Існує чотири способи передавання параметрів функції:

- за значенням;
- за адресою;
- за вказівником;
- за посиланням.

Під час надсилання параметрів за значенням у пам'ять заносять копії значень аргументів, і оператори функції працюють із цими копіями. Доступу до вихідних значень параметрів у функції немає, а отже, немає можливості змінити їх.

Під час надсилання за адресою в пам'ять заносять копії адрес аргументів, функція здійснює доступ до осередків пам'яті за цими адресами й може змінити вихідні значення аргументів.

```
#include <iostream>
void f(int i, int * j, int & k);
int main(){ int i = 1, j = 2, k = 3;
cout << "ijk\n";
cout << i << ' '<< j << ' ' << k <<
'\n'; f(i, &j, k);
cout << i << ' << j << ' << k;
return 0; }
void f (int i, int * j, int & k) {
     i++; (*j)++;
 k++;
}
```

У прикладі вище параметр i передається за значенням. Його зміна функцією не вплине на вихідне значення. Параметр `j` передається за адресою з використанням покажчика, водночас для передавання до функції адреси фактичного параметра використовують операцію взяття адреси, а для одержання значення функції потрібна операція розіменування.

Параметр `k` надсилають за адресою за допомогою посилання. Під час надсилання за посиланням до функції

передається адреса вказаного під час виклику параметра, усередині функції всі звернення до параметра неявно розіменовуються. Під час використання посилання, а не показників читання програми краще, також це позбавляє використання операцій одержання адреси та розіменування.

Приклад програми

Написати функцію для розрахунку лінійного збільшення β. Вхідні параметри для функції: розмір зображення y', розмір предмета y.

```
#include <iostream>
using namespace std;
double CalcLinMagnification(double
objectSize, double imageSize)
{
     return imageSize/objectSize; }
int main() {
double y, y_;
     cout << "Введіть розмір
предмета";     cin >> y;
 cout << "Введіть розмір зображення";
 cin >> y_;
   cout << "Лінійне збільшення = " <<
CalcLinMagnification(y, y_)
<< endl;
     cin.get();
          cin.get();
}
```

Зміст звіту

Звіт до лабораторної роботи повинен містити:

1) текст завдання, варіант;
2) блок-схему функцій;
3) лістинг програми;
4) скриншоти результатів роботи програми.

Варіанти завдань для роботи

Таблиця 2.7.1– Варіанти завдань

№ пор.	Завдання	Формули для розрахунку
1	Написати функцію визначення передньої апертури A об'єктива для предмета далекого типу. Вхідні параметри для функції: відносний отвір об'єктива 1: k, фокусна відстань об'єктива f'	$\frac{D}{f'} = 1:k$ $A = \frac{D}{2}$
2	Написати функцію визначення діаметра вихідного зіниці об'єктива. Вхідні параметри для функції: відносний отвір об'єктива 1: k, фокусна відстань об'єктива f в мм, лінійне збільшення об'єктива β	$1:k = \frac{D}{f'}$ $\beta = \frac{D}{D'}$

3	Світло падає нормально на межу поділу двох середовищ. Написати функцію для обчислення показника заломлення другого середовища. Вхідні параметри для функції: показник заломлення першого середовища $n1$ та коефіцієнт відображення ρ	$\rho = \left(\frac{n_2 - n_1}{n_2 + n_1}\right)^2$
4	Написати функцію для обчислення яскравості квадратного розсіювача. Вхідні параметри для функції: потік розсіювача Ф в лм, сторона квадрата a в м, ступінь білизни поверхні k	$L = \frac{kE}{\pi}$ $\Phi = ES$

5	Написати функцію для обчислення оптичної сили D тонкої лінзи в повітрі з показником заломлення $n0$. Вхідні параметри для функції: показник заломлення лінзи n , радіуси передньої $R1$ та задньої $R2$ поверхонь лінзи	$D = \frac{1}{F} = \left(\frac{n_{\text{л}}}{n_{\text{ср}}} - 1\right)\left(\frac{1}{R_1} + \frac{1}{R_2}\right)$
6	У досвіді Ллойда світлова хвиля, що виходить безпосередньо з джерела, інтерферує з хвилею, відбитою від дзеркала. У результаті екрані утворюється система інтерференційних смуг. Написати функцію для обчислення ширини інтерференційної лінії. Вхідні параметри для функції: відстань	$\Delta x = \frac{\lambda}{d} l$

	від джерела до екрану l, висота від джерела до площини дзеркала $d/2$, довжина хвилі світла λ	
7	Світлова хвиля падає на дзеркала Френеля. Написати функцію визначення довжини хвилі світла λ . Вхідними параметрами для функції: кут між дзеркалами φ, ширина інтерференційної смуги Δl, відстань від джерела до ребра дзеркал r, відстань від ребра до екрану $L0$	$\lambda = 2\varphi r \cdot \frac{\Delta l}{L_0 + r}$
8	Написати функцію для обчислення кута заломленого променя α_2. Вхідні параметри для функції: кут падіння променя α_1, показники заломлення першої	$n_1 \sin \alpha_1 = n_2 \sin \alpha_2$

	n_1 та другої n_2 середовища	
9	Написати функцію для обчислення діаметра польової діафрагми $D_{\text{пд}}$ у площині предмета. Вхідні параметри для функції: величина зображення в мм, кутове збільшення системи β	$D_{\text{пд}} = 2y$ $\beta = \frac{y'}{y}$
10	Написати функцію для обчислення оптичної сили двокомпонентної оптичної системи D. Вхідні параметри для функції: оптичні сили першого $D1$ та другого $D2$ компонентів оптичної системи, відстань між компонентами d	$D = D1 + D2 - D1D2d$
11	Написати функцію для обчислення коефіцієнта пропускання за нормального падіння світла.	$\tau = \frac{4n_2n_1}{n_2 + n_1}$

	Вхідні параметри для функції: показник заломлення першого середовища n_1 та другого середовища n_2	
12	Написати функцію для обчислення радіусу лінзи кулі. Вхідні параметри для функції: фокусна відстань лінзи кулі f', показник заломлення n	$D = \frac{1}{f'}$ $D = \frac{2(n-1)}{nr}$
13	Написати функцію для обчислення положення вхідної зіниці a. Вхідні параметри для функції: відстань від тонкої лінзи до апертурної діафрагми a', фокусна відстань лінзи f'	$\frac{1}{a'} - \frac{1}{a} = \frac{1}{f}$
14	Написати функцію для розрахунку радіусу круглого майданчика. Вхідні параметри	$M = \frac{\Phi}{S}$ $S = \pi R^2$

	для функції: світність круглого майданчика M, потік, що випромінюється майданчиком Ф	
15	Написати функцію розрахунку повного потоку від плоского ламбертовского випромінювача. Вхідні дані для функції: сила світла плоского випромінювання ламберта I, тілесний кут утворений обертанням плоского кута σ	$\Phi = \frac{\pi I \sin^2 \sigma}{2}$
16	Написати функцію розрахунку показника заломлення n плоскогнутої лінзи. Вхідні параметри для функції: оптична сила лінзи D, радіус поверхні лінзи r_2	$D = (n-1)\left(-\frac{1}{r_2}\right)$

17	Світло падає нормально на межу поділу двох середовищ. Написати функцію для обчислення коефіцієнта пропускання τ. Вхідні параметри для функції: показник заломлення першого середовища n_1 і другого середовища n_2	$\rho = \left(\frac{n_2 - n_1}{n_2 + n_1}\right)^2$ $\rho + \tau = 1$
18	Написати функцію для обчислення коефіцієнта віньєтування зверху Kв. Вхідні параметри для функції: діаметр апертурної діафрагми DАД, висота верхнього променя позаосьового пучка на апертурній діафрагмі hв	$a_{\text{в}} = \frac{D_{\text{АД}}}{2 - h_{\text{в}}}$ $K_{\text{в}} = \frac{2a_{\text{в}}}{D_{\text{АД}}}$

19	Промінь світла падає на плоскопаралельну скляну пластину. Написати функцію для обчислення величини усунення променя Δx, що пройшов через цю пластину. Вхідні параметри для функції: товщина скляної пластинки d, кут падіння θ, показник заломлення n	$\Delta x = d \sin\theta \left(1 - \frac{1 - \sin^2\theta}{n^2 - \sin^2\theta}\right)$
20	Написати функцію розрахунку сили світла сферичного ламбертовского випромінювача. Вхідні дані для функції: повний потік Ф, тілесний кут утворений обертанням плоского кута σ	$I = \frac{\Phi}{\Omega}$ $\Omega = 4\pi \sin^2\left(\frac{\sigma}{2}\right)$

2.8 Типи даних користувача в мові С++. структури

Завдання для роботи

На основі даних із файлу розробити та створити структуру, що містить дані про оптичні стекла. Створити масив структур та заповнити його даними з файлу. Виконати індивідуальне завдання згідно із завданням у таблиці 2.8.1.

Теоретичні відомості

Структури у мові С++

У мові С++ передбачена можливість створення користувальницьких типів даних, тобто типів даних, склад яких визначає програміст. Одним з інструментів для створення типів даних користувача є використання структур.

На відміну від масиву, усі елементи якого є однотипними, структура поєднує елементи різних типів.

У мові С++ структури є видом класу та мають всі його властивості, але в багатьох випадках досить використовувати структури так, як вони визначені мовою С:

```
struct [ім'я_типу] {  тип_1 елемент_1;  тип_2 елемент_2; … тип_n елемент_n;} [список_описувачів].
```

Елементи структури називають полями структури, вони можуть мати будь-який тип, крім типу цієї самої структури, але можуть бути показниками на нього.

Якщо відсутнє ім’я типу, повинен бути вказаний список описників змінних, показників чи масивів. І тут опис структури служить визначенням елементів цього списку.

```
//Визначення масиву структур і покажчика на структуру struct {
```

```
        char fio [30]; int date, code;
     double salary;
    } staff [100], * ps;
```

Якщо список відсутній, опис структури визначає новий тип, ім'я якого можна використовувати поряд зі стандартними типами, наприклад:

```
   struct Worker{//опис нового типу Worker

   char fio [30];
      int date, code;
      double salary;
   }; //кінець визначення структури
   Worker staff[100], *ps; /* визначення
масиву типу Worker та вказівника на нього
*/
```

Ім'я структури можна використовувати одразу після його оголошення в тих прикладах, коли компілятору не потрібно знати розмір структури:

```
  struct List; //Оголошення структури List
struct Link{
   List * p; //Покажчик на структуру List
          Link *prev, *succ; //Покажчик на
структуру Link
  } struct List {
    визначення_структури_List
  };
```

Для ініціалізації структури значення її елементів перераховують у фігурних дужках у порядку їх опису:

```
struct {
     char fio [30]; int date, code;
    double salary;
} worker = {"Шевченко", 31, 111, 3400.55};
```

Під час ініціалізації масивів структур потрібно вкладати у фігурні дужки кожен елемент масиву:

```
struct complex {     float real, im;
} compl [2] [3] = {
 {{1,1}, {1,1}, {1,1}},
 {{2,2}, {2,2}, {2,2}}
};
```

Для змінних одного й того ж самого структурного типу визначено операцію присвоювання, у своїй відбувається поелементне копіювання. Структури можна передавати функції, повертати значення функції. Інші операції зі структурами може бути визначено користувачам.

Доступ до полів структури виконують за допомогою операції вибору. Під час звернення до поля через ім'я структури та -> під час звернення через показник:

```
     Worker worker, staff[100], *ps;
worker.fio = "Шевченко";
     staff [8]. code = 111; ps ->
     salary = 0.12;
```

Бітові поля

Бітові поля – це особливий вид полів структури. Бітові поля зручно використовуватиме зберігання прапорців типу «так/ні». Таким прапорцям достатньо одного біта, у той час, як мінімальний осередок адресованої пам'яті – 1 байт.

Під час опису бітового поля в структурі після імені через двокрапку вказують довжину поля в бітах:

```
struct Options{     bool centerX:1;
bool centerY:1;     unsigned int shadow:2;
unsigned int palette:4;
};
```

Бітові поля можуть бути будь-якого цілого типу. Доступ до поля здійснюють звичайним способом – на ім'я.

Приклад програми

Розробити та створити структуру, що містить дані про об’єктиви. Створити масив структур та заповнити його даними з файлу. Знайти всі об’єктиви з фокусом у діапазоні від 40 до 60 мм або кутовим полем більше 60 градусів.

```
#include <iostream>
#include <cmath>

using namespace std;

struct Objective // Структура об'єктива
{     char name [30]; // Назва об'єктиву
      double focus; // фокусна відстань
      double w2; // Кутове поле
      double k; // відносний отвір
};
int main() {

 Objectivecatalog[4] = { {"Індустар-7",104,
                          24, 3.5},
                         {"Індустар-50",52,
                         46, 2.9},
                         {"Геліос-10",14,
                         53, 1.9},
                         {"Юпітер-6",180,
                         14, 3.2}};

      cout << "Введіть фокусну відстань";
      double Focus;  cin >> Focus;
cout << "Список об'єктивів, у яких фокусна
відстань більша" << Focus <<endl; // Назва
колонок
cout << "Назва";
```

```
cout << "F't";
cout << "2w\t";
cout << "1: k" << endl; // цикл масиву
catalog
    for (int i = 0; i < 4; i++)
     {
         if (catalog[i].focus >= Focus)
          {
               cout << catalog[i].name <<
"\t";                                    cout <<
catalog[i].focus << "\t";
cout << catalog[i].w2 << "\t";
               cout << "1:" << catalog
[i].k;
               cout << endl;
          }
     }
     cin.get();
cin.get();
}
```

Зміст звіту

Звіт до лабораторної роботи повинен містити:

1) текст завдання, варіант;
2) текст програми;
3) зміст текстового файлу;
4) скриншоти результатів роботи програми.

Варіанти завдань для роботи

Таблиця 8.1 – Варіанти завдань

№ пор.	Завдання
1	Організувати пошук за маркою скла. Вивести на екран дані щодо марки скла, що запитується
2	Вивести у файл дані про скло, у яких число Аббе дорівнює введеному значенню
3	Вивести на екран марку скла з мінімальним показником заломлення
4	Вивести на екран марки шибок із показником заломлення в діапазоні від $n1$ до $n2$
5	Відсортувати масив структур за зростанням значення показника заломлення та вивести його у файл
6	Організувати введення з клавіатури для додавання даних про марки скла. Додати дані до масиву, використовуючи введення з клавіатури та вивести всі дані у файл
7	Вивести у файл марки скла, у яких число Аббе менше значення, введеного з клавіатури
8	Створити другий файл із даними за марками скла й порівняти його з даними, записаними в масив структур. У разі виявлення збігу вивести на екран повідомлення: «Знайдено збіг!», інакше вивести повідомлення: «Збігів не знайдено!»
9	Організувати пошук марки скла за введеним значенням показника заломлення та вивести на екран дані знайденої марки скла

10	Для кожного запису структури розрахувати загальну дисперсію та вивести на екран дані: марка скла, загальна дисперсія
11	Вивести на екран марки скла з показником заломлення більше введеного з клавіатури
12	Вивести на екран марки скла, у яких значення числа Аббе знаходиться в діапазоні від $V1$ до $V2$
13	Розрахувати середнє арифметичне показників заломлення всіх марок скла, дані про які записані в масив структур, та вивести одержане значення на екран
14	Введіть із клавіатури показник заломлення та виведіть на екран дані про марки скла, які близькі до введеного показника заломлення (відхилення не більше 10 %)
15	Відсортуйте масив структур за зростанням значення числа Аббе і виведіть відсортовані дані про марки скла у файл
16	Ввести з клавіатури значення показника заломлення та виведіть на екран дані про марки скла, у яких показник заломлення більший або дорівнює введеному значенню
17	Створити другий файл із даними за марками скла й порівняти його з даними, записаними в масив структур. Якщо даних про таку марку скла в масиві структур немає, то додати такий запис
18	Ввести з клавіатури значення показника заломлення та вивести на екран марки скла, у яких показник заломлення менший або дорівнює введеному значенню

19	Створити нову структуру, що містить такі поля: марку скла, загальну дисперсію. Розрахувати для кожної марки скла загальну дисперсію та записати назву марки скла та обчислену загальну дисперсію до масиву з новою структурою. Вивести одержаний масив структур у файл
20	Відсортувати масив структур за зростанням показника заломлення та вивести на екран три перші записи, що мають мінімальне значення показника заломлення

Каталог оптичних стекол:

URL: https://drive.google.com/file/d/1PdREgqCfpt4qiXUraNolxnjcLbR_c8b9/view?usp=sharing.

Список літератури

1. Основи програмування на C++ : навчальний посібник / О. О. Водка та ін., Харків НТУ «ХПІ», 2021. 112 с.

2. Програмування мовою C++. Технологія візуального програмування : навч. посіб. Полтава : ПНПУ імені В. Г. Короленка, 2020. 144 с.

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

4. Петрик М. Р., Петрик О. Ю. Основи програмування : курс лекцій / Тернопіль : ТНТУ 2018. 64 с.

5. Петрик О., Петрик М., Бойко І. Основи програмування : лабораторний практикум / Тернопіль : ТНТУ 2021. 64 с.

6. Герасимюк В. І. Електронний підручник. ВСП «Павлоградський фаховий коледж НТУ «Дніпровська політехніка» URL: http://cpp.dp.ua/.

7. Уроки програмування на C++. URL: https://acode.com.ua/uroki-po-cpp/.

8. Balagurusamy E. Object Oriented Programming with C++, 8th Edition. Mc Graw Hill India, 2019. 426 p.

9. Rutland Water C. N.: Its conception, impact and future development. Brighton : University of Sussex. 978 p.

10. Alexandrescu A. Modern C++ Design : Generic Programming and Design Patterns Applied / Addison-Wesley: 2001. 323 p.

Електронне навчальне видання

Тищенко Костянтин Володимирович,
Логвинов Андрій Миколайович,
Пилипенко Олександр Валерійович

АЛГОРИТМІЧНІ МОВИ ПРОГРАМУВАННЯ В ЕЛЕКТРОННИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЯХ (ПРАКТИКУМ)

Навчальний посібник

Художнє оформлення обкладинки К. В. Тищенка
Редакторка О. Ф. Дубровіна
Комп'ютерне верстання А. М. Логвинова

Формат 60x84/16. Ум. друк. арк. 5,58. Обл. -вид. арк. 5,94.

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, м. Суми, 40007
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.