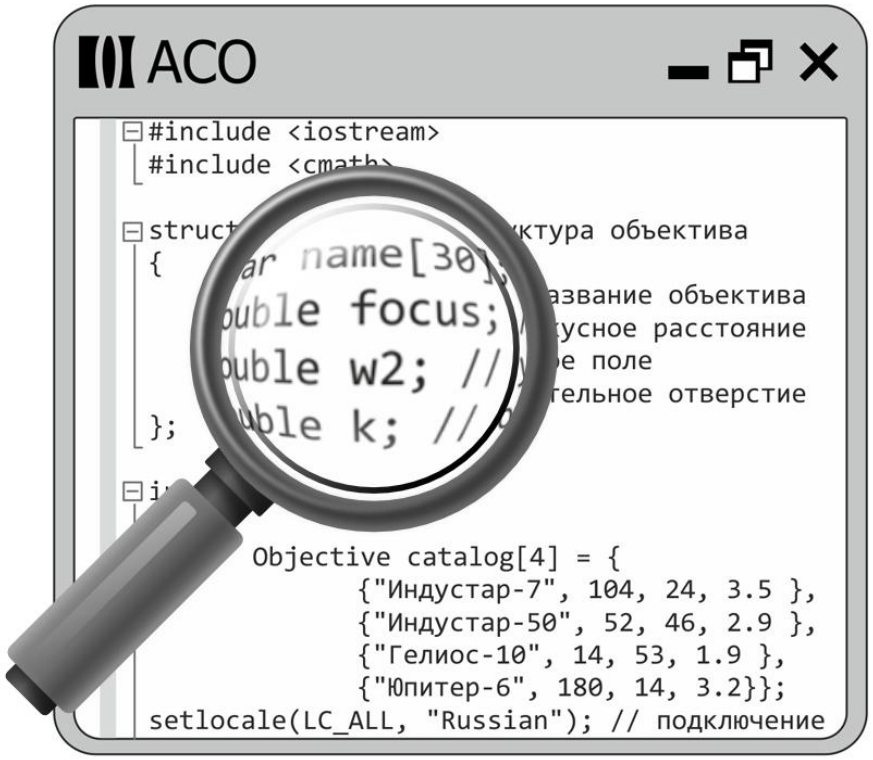
УНИВЕРСИТЕТ ИТМО

К.В.Ежова, А.А.Бурцева, Р.О.Данцаранов

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА С++



Санкт-Петербург
2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

К.В.Ежова, А.А.Бурцева, Р.О.Данцаранов

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА С++

Методические указания к лабораторным работам

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2016

К.В.Ежова, А.А.Бурцева, Р.О.Данцаранов, Основы программирования на С++.– СПб: Университет ИТМО, 2016. – 73 с.

В учебном пособии представлены краткие теоретические сведения и варианты индивидуальных заданий для выполнения лабораторных работ, проводимых в рамках дисциплины «Основы программирования».

Учебное пособие предназначено для студентов высших учебных заведений, обучающихся по направлениям подготовки бакалавров 12.03.02 «Оптотехника» и 16.03.01 «Техническая физика».

Рекомендовано к печати Ученым советом факультета Лазерной и световой инженерии Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики, протокол №10 от 11 октября 2016 г.

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

Введение

Учебное пособие предназначено для студентов, изучающих дисциплину «Основы программирования» на С++.

В пособии приводятся краткие теоретические сведения, методические указания, варианты индивидуальных заданий и требования к оформлению отчетов необходимые для выполнения восьми лабораторных работ.

Отличительной особенностью пособия является использование в качестве вариантов заданий задач, содержание которых непосредственно связано с содержанием дисциплин по направлениям подготовки бакалавров 12.03.02 «Оптотехника» и 16.03.01 «Техническая физика».

В приложении излагаются основы работы в среде программирования Microsoft Visual Studio 2010.

1. Лабораторная работа №1. Линейное программирование

1.1. Задание для работы

Составить программу для вычисления эквивалентных пар выражений z_1, z_2, y_1, y_2 в соответствии с заданием в таблице 1.5.6. Для всех выражений подобрать входные данные согласно ОДЗ.

Вывести на экран результаты вычислений и входные данные. Для вывода данных использовать форматный вывод. Ввод данных организовать с клавиатуры.

1.2. Теоретические сведения

1.2.1. Потоковый ввод/вывод

Поддержка для потокового ввода/вывода данных встроенных типов языка C++ реализована в библиотеке iostream. Для использования библиотеки iostream в программе необходимо включить заголовочный файл:

```
#include <iostream>
```

Применяя iostream, необходимо использовать следующую директиву пространства имен, чтобы определения в iostream были доступны в программе:

```
using namespace std;
```

Операции ввода/вывода выполняются с помощью классов istream (потоковый ввод) и ostream (потоковый вывод). Третий класс, iostream, является производным от них и поддерживает двунаправленный ввод/вывод. Для удобства в библиотеке определены три стандартных объекта-потока:

- cin – объект класса istream, соответствующий стандартному вводу. В общем случае он позволяет читать данные с терминала пользователя (консоли);
- cout – объект класса ostream, соответствующий стандартному выводу. В общем случае он позволяет выводить данные на терминал пользователя (консоль);
- cerr – объект класса ostream, соответствующий стандартному выводу для ошибок. В этот поток направляются сообщения об ошибках программы.

Вывод осуществляется, как правило, с помощью перегруженного оператора сдвига влево (<<), а ввод – с помощью оператора сдвига вправо (>>).

Вывод сообщения на экран:

```
cout << "Hello, World!"; /*на экран будет выведено сообщение Hello,
World!*/
```

Ввод с клавиатуры:

```
int new;
cin >> new; /*ввод с клавиатуры значения целочисленной переменной
new*/
```

1.2.2. Форматный вывод

Настройка ширины полей

Для размещения чисел различной длины в полях постоянной ширины можно воспользоваться функцией-членом width. Может применяться двумя способами:

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

Первая форма возвращает текущую установку ширины поля. Вторая устанавливает ширину поля равной i пробелам и возвращает предыдущее значение ширины. Метод width() касается только следующего отображаемого элемента, после чего ширина поля возвращается к значению по умолчанию.

Пример:

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

Оператор вывода создает следующую строку вывода:

```
#          12#24#
```

Символы-заполнители

По умолчанию cout заполняет неиспользуемые части поля пробелами. Для изменения этого можно воспользоваться функцией-членом fill().Например, следующий вызов изменяет символ-заполнитель на звездочку:

```
cout.fill('*');
```

Пример:

```
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;
}
```

Ниже показан вывод программы из примера:

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

Необходимо обратить внимание, что в отличие от использования функции по установке ширины поля, новый символ-заполнитель остается в действии до тех пор, пока он не будет заменен.

Установка точности отображения чисел с плавающей точкой

Функция-член precission() позволяет задать количество знаков после десятичной точки.

```
cout.precision (n); //n – число знаков после десятичной точки
```

Пример:

```
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;
}
```

Вывод программы с применением текущего форматирования C++ имеет следующий вид:

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

1.2.3. Основные алгебраические функции

Библиотека cmath языка C++ определяет набор функций для выполнения общих математических операций и преобразований:

```
#include <cmath>
```

В таблицах 1.2.1-1.2.5 представлены основные функции математических операций и преобразований библиотеки cmath.

Таблица 1.2.1. Тригонометрические функции

`cos(a)`	Вычисление косинуса угла a. Угол a в радианах
`sin(a)`	Вычисление синуса угла a. Угол a в радианах
`tan(a)`	Вычисление тангенса угла a. Угол a в радианах
`acos(a)`	Вычисление арккосинуса a. Результат в радианах
`asin(a)`	Вычисление арксинуса a. Результат в радианах
`atan(a)`	Вычисление арктангенса a. Результат в радианах

Таблица 1.2.2. Гиперболические функции

`cosh(a)`	Вычисление гиперболического косинуса.
`sinh(a)`	Вычисление гиперболического синуса.
`tanh(a)`	Вычисление гиперболического тангенса.

Таблица 1.2.3. Экспоненциальные и логарифмические функции

exp(a)	Вычисление экспоненты
log(a)	Натуральный логарифма
log10(a)	Десятичный логарифм a

Таблица 1.2.4. Функции степени

pow(a,b)	Возведение числа a в степень b
sqrt(a)	Корень квадратный числа a (a ≥ 0)

Таблица 1.2.5. Округление, модуль и другие функции

ceil(a)	Округлениеа до наименьшего целого значения, но не меньше a
abs(a)	Вычисление модуля a
floor(a)	Округлениеа до наибольшего целого значения, но не больше a
fmod(a,b)	Остаток от деления a на b

Функция $\sec x$ вычисляется из соотношения $\sec x \cdot \cos x = 1$.

1.3. Пример программы

Исходные данные: $x = 3{,}3$.

Выражения для вычисления:

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

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

```
// лабораторная работа № вариант № группа № студент ФИО
#include <iostream>// стандартный поток консольного ввода/вывода
#include <cmath> /* заголовочный файл, содержащий основные
математические функции */

using namespace std; //используемое стандартное пространство имен

int main()
{
      // объявление и инициализация переменных
      double y1 = 0.0, y2 = 0.0, s1 = 0.0, yd = 0.0;
      double x = 3.3;
      s1 = sqrt(x*x - 9); // повторяющая часть формулы
      yd = x*x - 2*x - 3 + (x-1)*s1; // знаменатель y1
      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(); // задержка экрана
    return 0;
}
```

1.4. Содержание отчета

Отчет к лабораторной работе должен содержать:

1. Текст задания, вариант.
2. ОДЗ (область допустимых значений).
3. Листинг программы.
4. Скриншоты результатов выполнения программы.

1.5. Варианты заданий для работы

Таблица 1.5.6. Варианты заданий

№ п/п	Исходные данные	Формулы для вычислений
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	$z_1 = \sqrt{\frac{2a}{(1+a)\sqrt[3]{1+a}}} \cdot \sqrt[3]{\frac{4+\frac{8}{a}+\frac{4}{a^2}}{\sqrt{2}}};$ $z_2 = \frac{2\sqrt[6]{a^5}}{a};$ $y_1 = \frac{\cos 4\alpha + 1}{\mathrm{ctg}\,\alpha - \mathrm{tg}\,\alpha};$ $y_2 = \frac{1}{2}sin\,4\alpha.$

№ п/п	Исходные данные	Формулы для вычислений
4	x, α	$z_1 = \dfrac{4x\left(x+\sqrt{x^2-1}\right)^2}{\left(x+\sqrt{x^2-1}\right)^4-1};$ $z_2 = \dfrac{1}{\sqrt{x^2-1}};$ $y_1 = \dfrac{\cos^{-1} 2\alpha + \sin 2\alpha \operatorname{tg} 2\alpha}{1+\cos 4\alpha} + \dfrac{1}{4\sin^2\left(\frac{\pi}{4}-\alpha\right)\operatorname{ctg}\left(\frac{\pi}{4}-\alpha\right)};$ $y_2 = \dfrac{1}{cos^3 2\alpha}.$
5	a, α	$z_1 = \left(\dfrac{\sqrt{a}}{2} - \dfrac{1}{2\sqrt{a}}\right)^2 \left(\dfrac{\sqrt{a}-1}{\sqrt{a}+1} - \dfrac{\sqrt{a}+1}{\sqrt{a}-1}\right);$ $z_2 = \dfrac{1-a}{\sqrt{a}};$ $y_1 = \cos\alpha(1+\cos^{-1}\alpha+\operatorname{tg}\alpha)(1-\cos^{-1}\alpha+\operatorname{tg}\alpha);$ $y_2 = 2\,sin\,\alpha.$
6	a, α	$z_1 = \dfrac{1}{2\left(1+\sqrt{a}\right)} + \dfrac{1}{2\left(1-\sqrt{a}\right)} - \dfrac{a^2+2}{1-a^3};$ $z_2 = -\dfrac{1}{a^2+a+1};$ $y_1 = \dfrac{1+\operatorname{ctg} 2\alpha \operatorname{ctg}\alpha}{\operatorname{tg}\alpha+\operatorname{ctg}\alpha};$ $y_2 = \dfrac{1}{2}ctg\,\alpha.$
7	m, n, α	$z_1 = \dfrac{\left(m^2-\frac{1}{n^2}\right)^m \left(n+\frac{1}{m}\right)^{n-m}}{\left(n^2-\frac{1}{m^2}\right)^n \left(m-\frac{1}{n}\right)^{m-n}};$ $z_2 = \left(\dfrac{m}{n}\right)^{m+n};$ $y_1 = \dfrac{\cos^2\alpha - \operatorname{ctg}^2\alpha + 1}{\sin^2\alpha + \operatorname{tg}^2\alpha - 1};$ $y_2 = ctg^2\,\alpha.$

№ п/п	Исходные данные	Формулы для вычислений
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	$z_1 = \frac{2b + a - \frac{4a^2 - b^2}{a}}{b^3 + 2ab^2 - 3a^2b} \cdot \frac{a^3b - 2a^2b^2 + ab^3}{a^2 - b^2};$ $z_2 = \frac{a-b}{a+b};$ $y_1 = \frac{\operatorname{tg} 2\alpha + \operatorname{ctg} 3\beta}{\operatorname{ctg} 2\alpha + \operatorname{tg} 3\beta};$ $y_2 = \frac{tg\, 2\alpha}{tg\, 3\beta}.$

№ п/п	Исходные данные	Формулы для вычислений
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	$z_1 = \sqrt[6]{4x\left(11 + 4\sqrt{6}\right)} \cdot \sqrt[3]{4\sqrt{2x} - 2\sqrt{3x}};$ $z_2 = \sqrt[3]{20x};$ $y_1 = \dfrac{1 + \cos\alpha + \cos 2\alpha + \cos 3\alpha}{\cos\alpha + 2\cos^2\alpha - 1};$ $y_2 = 2cos\ \alpha.$

№ п/п	Исходные данные	Формулы для вычислений
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	$z_1 = \dfrac{1 - x^{-2}}{x^{1/2} - x^{-1/2}} - \dfrac{2}{x^{3/2}} + \dfrac{x^{-2} - x}{x^{1/2} - x^{-1/2}}$; $z_2 = -\sqrt{x}\left(1 + \dfrac{2}{x^2}\right)$; $y_1 = \dfrac{\mathrm{tg}^4\,\alpha - \mathrm{tg}^6\,\alpha}{\mathrm{ctg}^4\,\alpha - \mathrm{ctg}^2\,\alpha}$; $y_2 = \mathrm{tg}^8\,\alpha$.
19	x, α, β	$z_1 = \left(\dfrac{\sqrt[4]{x^3} - \sqrt[4]{x}}{1 - \sqrt{x}} + \dfrac{1 + \sqrt{x}}{\sqrt[4]{x}}\right)^2 + \left(1 + \dfrac{2}{\sqrt{x}} + \dfrac{1}{x}\right)^{-1/2}$; $z_2 = \dfrac{1 - \sqrt{x}}{1 - x}$; $y_1 = \sin^2\left(\beta - \dfrac{\pi}{2}\right) - \cos^2\left(\alpha - \dfrac{3\pi}{2}\right)$; $y_2 = cos(\alpha + \beta)\,cos(\alpha - \beta)$.
20	x, α	$z_1 = \left(\sqrt{1 - x^2} + 1\right) : \left(\dfrac{1}{\sqrt{1+x}} + \sqrt{1-x}\right)$; $z_2 = \sqrt{1+x}$; $y_1 = 3 - 4\sin^2\left(\dfrac{3\pi}{2} - \alpha\right)$; $y_2 = 4\,cos\left(\dfrac{\pi}{6} + \alpha\right) cos\left(\dfrac{\pi}{6} - \alpha\right)$.
21	a, x, α	$z_1 = \sqrt{\dfrac{x}{x - a^2}} : \left(\dfrac{\sqrt{x} - \sqrt{x - a^2}}{\sqrt{x} + \sqrt{x - a^2}} - \dfrac{\sqrt{x} + \sqrt{x - a^2}}{\sqrt{x} - \sqrt{x - a^2}}\right)$; $z_2 = \dfrac{a^2}{4(a^2 - x)}$; $y_1 = \sin 5\alpha + \sin 6\alpha + \sin 7\alpha + \sin 8\alpha$; $y_2 = 4\,cos\dfrac{\alpha}{2} cos\,\alpha\, sin\dfrac{13\alpha}{2}$.

№ п/п	Исходные данные	Формулы для вычислений
22	z, α	$z_1 = \dfrac{\left(z - z\sqrt{z} + 2 - 2\sqrt{z}\right)^2\left(1 + \sqrt{z}\right)^2}{z - 2 + \frac{1}{z}} - z\sqrt{z}\sqrt{\dfrac{4}{z} + 4 + z};$ $z_2 = z(z+1)(z+2);$ $y_1 = 3 + 4\cos 4\alpha + \cos 8\alpha\,;$ $y_2 = 8\,cos^4 2\alpha.$
23	x, α	$z_1 = \dfrac{2(x^4 + 4x^2 - 12) + x^4 + 11x^2 + 30}{x^2 + 6};$ $z_2 = 1 + 3x^2;$ $y_1 = 4\cos\left(\dfrac{\pi}{6} - \alpha\right)\sin\left(\dfrac{\pi}{3} - \alpha\right);$ $y_2 = \dfrac{sin\,3\alpha}{sin\,\alpha}.$
24	p, α	$z_1 = \left((1-p^2)^{-1/2} - (1+p^2)^{-1/2}\right)^2 + 2(1-p^4)^{-1/2};$ $z_2 = \dfrac{2}{1-p^4};$ $y_1 = 3 - 4\cos(4\alpha - 3\pi) - \cos(5\pi + 8\alpha)\,;$ $y_2 = 8\,cos^4 2\alpha.$
25	x, y, α	$z_1 = \left(\left(\dfrac{x}{y-x}\right)^{-2} - \dfrac{(x+y)^2 - 4xy}{x^2 - xy}\right)^2 \dfrac{x^2}{x^2y^2 - y^4};$ $z_2 = \dfrac{x-y}{x+y};$ $y_1 = \dfrac{3 + 4\cos 4\alpha + \cos 8\alpha}{3 - 4\cos 4\alpha + \cos 8\alpha};$ $y_2 = ctg^4 2\alpha.$
26	a, α	$z_1 = \left(6a^2 + 5a - 1 + \dfrac{a+4}{a+1}\right) : \left(3a - 2 + \dfrac{3}{a+1}\right);$ $z_2 = 2a + 3;$ $y_1 = \dfrac{1 - 2\sin^2 2\alpha}{1 - \sin 4\alpha};$ $y_2 = \dfrac{1 + tg\,2\alpha}{1 - tg\,2\alpha}.$

№ п/п	Исходные данные	Формулы для вычислений
27	m, n, α	$z_1 = \dfrac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + mn + m^2 - m}$; $z_2 = \dfrac{\sqrt{m} - \sqrt{n}}{m}$; $y_1 = \sin(\pi + \alpha)\sin\left(\dfrac{4\pi}{3} + \alpha\right)\sin\left(\dfrac{2\pi}{3} + \alpha\right)$; $y_2 = \dfrac{1}{4} sin\, 3\alpha$.
28	t, α	$z_1 = \left(\dfrac{t\sqrt{t+2}}{\sqrt{t-2}} - \dfrac{2\sqrt{t-2}}{\sqrt{t+2}} - \dfrac{4t}{\sqrt{t^2-4}}\right)^{1/2} : \sqrt[4]{t^2-4}$; $z_2 = \dfrac{\sqrt{t^2-4}}{t+2}$; $y_1 = \operatorname{tg} 4\alpha - \cos^{-1} 4\alpha$; $y_2 = \dfrac{sin\, 2\alpha - cos\, 2\alpha}{sin\, 2\alpha + cos\, 2\alpha}$.
29	b, α	$z_1 = \dfrac{\frac{\lvert b-1\rvert}{b} + b\lvert b-1\rvert + 2 - \frac{2}{b}}{\sqrt{b - 2 + \frac{1}{b}}}$; $z_2 = \dfrac{b^2 - 1}{\sqrt{b}}$; $y_1 = \cos^8 \alpha - \sin^8 \alpha$; $y_2 = \dfrac{cos\, 2\alpha(3 + cos\, 4\alpha)}{4}$.
30	a, α, β	$z_1 = \left(2 - \dfrac{1}{4a^{-1}} - \dfrac{4}{a}\right)\left((a-4)\sqrt[3]{(a+4)^{-3}} - \dfrac{(a+4)^{3/2}}{\sqrt{(a^2-16)(a-4)}}\right)$; $z_2 = \dfrac{(4-a)(a^2+16)}{2a(a+4)}$; $y_1 = \operatorname{ctg}^2 \alpha - \operatorname{ctg}^2 \beta$; $y_2 = \dfrac{cos^2 \alpha - cos^2 \beta}{sin^2 \alpha\, sin^2 \beta}$.

2. Лабораторная работа №2. Условные и циклические алгоритмические конструкции. Определение попадания луча во входной зрачок сложной формы

2.1. Задание для работы

Для каждой линии, ограничивающей входной зрачок, составить уравнение $y = f_i(x)$ согласно своему варианту в таблице 2.5.3. Проверить правильность уравнений, построив графики функций. Написать программу, генерирующую координаты точек прямоугольника, равномерно заполненного точками (20-30 точек по каждой оси) и накрывающего область зрачка. В случае попадания луча в зрачок в точке с координатами (x, y), вывести информацию об этом на экран. Ввод данных организовать с клавиатуры.

2.2. Теоретические сведения

2.2.1. Операторы сравнения и логические операции языка C++

Операторы сравнения предназначены для сравнения между собой двух значений. В таблице 2.2.1 представлены операторы сравнения языка C++.

Таблица 2.2.1. Операторы сравнения

Символ операции	Значение	Использование
`<`	меньше	`a < b`
`<=`	меньше либо равно	`a <= b`
`>`	больше	`a > b`
`>=`	больше либо равно	`a >= b`
`==`	равно	`a == b`
`!=`	не равно	`a != b`

В языке C++ существует четыре основных логических операции:

- логическая операция И;
- логическая операция ИЛИ;
- логическая операция НЕ или логическое отрицание;
- логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ.

Логические операции образуют сложное (составное) условие из нескольких простых (два или более) условий. Пример условий показан в таблице 2.2.2.

Таблица 2.2.2. Логические операции языка C++

Операции	Обозначение	Условие	Краткое описание
И	`&&`	`a == 3 && b > 4`	Составное условие истинно, если истинны оба простых условия
ИЛИ	`||`	`a == 3 || b > 4`	Составное условие истинно, если истинно, хотя бы одно из простых условий
НЕ	`!`	`!( a == 3)`	Условие истинно, если a не равно 3
ИСКЛЮЧАЮЩЕЕ ИЛИ		`!(a == 3 || b> 4)`	Составное условие ложно, если оба простых условия истинны или ложны.

Операции сравнения и логические операции в результате дают значение типа bool, то есть true или false. Если же такое выражение встречается в контексте, требующем целого значения, true преобразуется в 1, a false – в 0.

2.2.2. Условный оператор if

Условный оператор в языке C++ имеет следующую структуру:

```
if (логическое выражение) оператор_1;
[else оператор_2;]
```

Сначала вычисляется значение логического выражения, если оно не равно нулю (true), выполняется первый оператор, иначе – второй. Вторую ветвь оператора вместе с else можно опустить, тогда, в случае, если выражение ложно, произойдет выход из ветвления.

Условный оператор с двумя ветвями:

```
if (num < 10)
{ // если введенное число меньше 10
      cout << "Это число меньше 10." << endl;
}
else { // иначе
      cout << "Это число больше либо равно 10" << endl;
}
```

Условный оператор с одной ветвью:

```
if (num!= 10) // если введенное число не равно 10
      cout << "Это число не равно 10." << endl;
```

Распространенные ошибки:

1. Использование в выражении при проверке равенства вместо оператора (==) простое присваивание (=).
2. Неверная запись проверки на принадлежность диапазону. Условие $0 < x < 1$ необходимо записать следующим образом: `if (0 <x&&x< 1)`

2.2.3. Операторы цикла

Операторы цикла используются для организации многократно повторяющихся вычислений. В языке C++ существует три типа циклических конструкций:

- цикл с предусловием (while);
- цикл с постусловием (do while);
- цикл с параметром (с заданным количеством повторений (for)).

Выполнение цикла с предусловием начинается с условия, если оно истинно, выполняется оператор цикла. Если при первой проверке условие ложно, то цикл не выполнится ни разу. В общем виде цикл с предусловием приведен ниже:

```
while (условие)
{
блок инструкций
}
```

Пример:

```
int i = 1;       // инициализация счетчика цикла
while (i<=10)    // выполняем цикл пока i ≤ 10
{
cout << i*i << endl;
++i;             // увеличение i на 1
}
```

Тело цикла с постусловием всегда выполняется хотя бы один раз. Сначала выполняется простой или составной оператор в теле цикла, а затем проверяется условие. Если условие истинно, тело цикла выполняется еще раз. Цикл завершается, когда условие станет ложным или в теле цикла будет выполнен оператор передачи управления.

Цикл с постусловием в общем виде:

```
do
{
блок инструкций
}
while (условие);
```

Пример:

```
int i = 0;       // инициализация счетчика цикла
int sum = 0;     // инициализация счетчика суммы
do {             // выполняем цикл
     i++;
     sum += i;
}
while (i < 1000);     // пока выполняется условие
```

Цикл с параметром имеет следующий формат:

```
for (инициализация; условие; модификации)
{
блок инструкций
}
```

В части инициализации можно записать несколько переменных, используемых в цикле, разделенных запятой. Цикл с параметром выполняется как цикл с предусловием: сначала проверяется истинность условия, а затем выполняется или не выполняется тело цикла.

Пример:

```
for ( int i =0; i<= 10; i++)     /* инициализация счетчика, условие,
увеличение счетчика на 1*/
      cout << i*i << endl;  // тело цикла
```

Распространенные ошибки:

1. Использование в теле цикла не инициализированных переменных.
2. Неверная запись условия выхода из цикла.

2.2.4. Рекомендации по выполнению задания

Для выполнения задания следует составить неравенства, которым удовлетворяют координаты точек внутри области зрачка. Рекомендуется разбить область на части и воспользоваться симметрией заданной области зрачка.

2.3. Пример выполнения задания

Для каждой линии, ограничивающей входной зрачок, изображенный на рисунке 2.3.1, составим уравнение $y = f_i(x)$. Определим условия попадания луча в область зрачка:

1. Условие попадания в I или III квадрант: $x \cdot y > 0$.
2. Условия попадания между двумя наклонными прямыми: $-(x+1) < y < -(x-1)$.

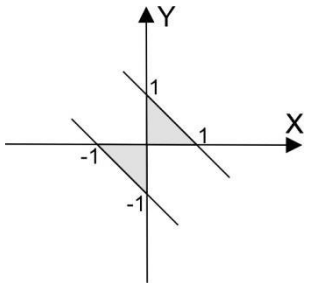


Рисунок 2.3.1. Входной зрачок сложной формы

Программа, генерирующая координаты точек прямоугольника, накрывающего область зрачка, равномерно заполненного точками и определяющая попадание луча в зрачок:

```
// лабораторная работа №  вариант №    группа №   студент  ФИО
#include <iostream>

using namespace std;

int main()
{
     setlocale(LC_ALL, "Russian");    // подключение русского языка в
                                      //консоли
     double x = 0.0, y = 0.0; // координаты по оси X и У
     double step = 0.1; // шаг по оси
     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)))
// проверка условия попадания в область
               {
                    cout << "Точка с координатами x=" << x << ",
y=" << y << " попадает в область" << endl;
               }
               else
               {
                    cout << "Точка с координатами x=" << x << ",
y=" << y << " не попадает в область" << endl;
               }
          }
     }
     cin.get();
```

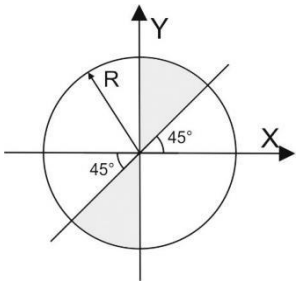
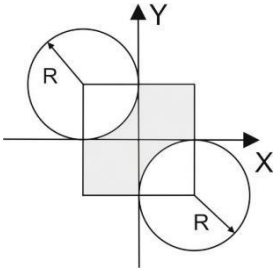
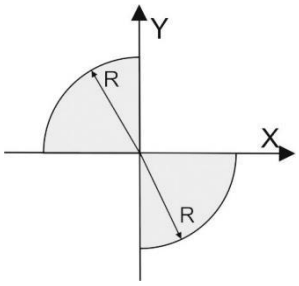
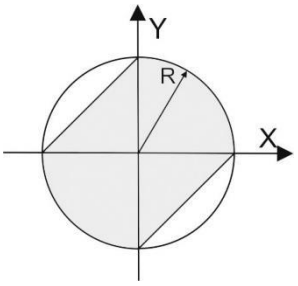
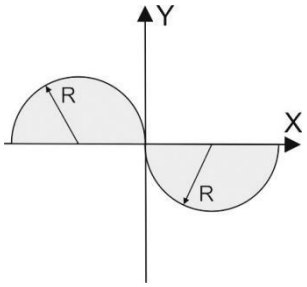
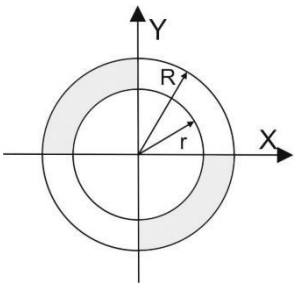
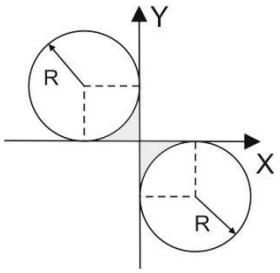
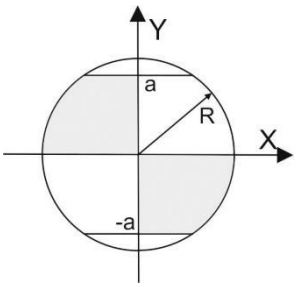
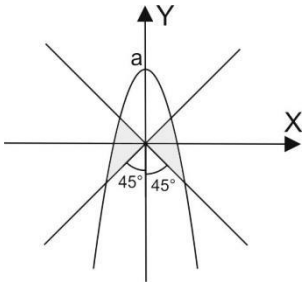
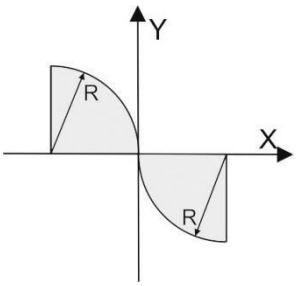
2.4. Содержание отчета

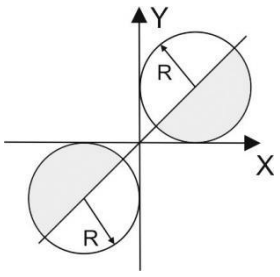
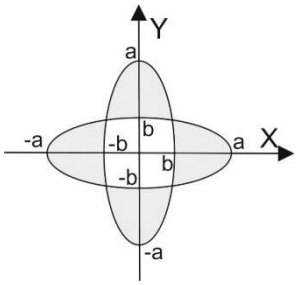
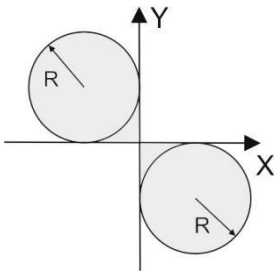
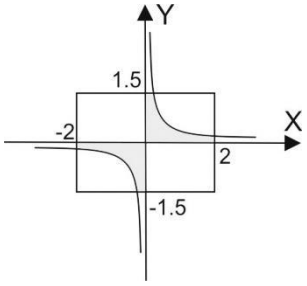
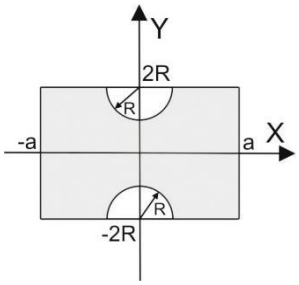
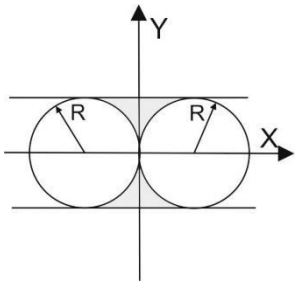
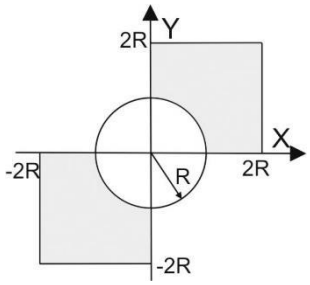
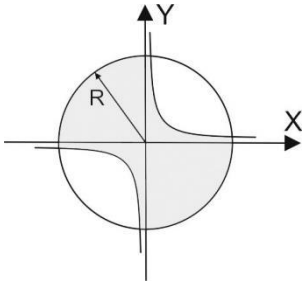
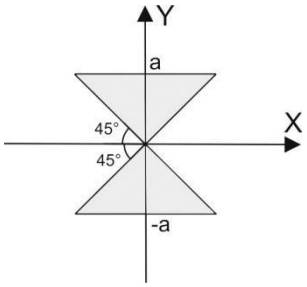
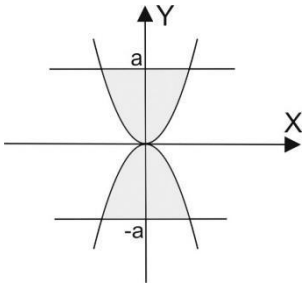
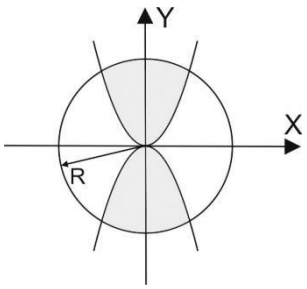
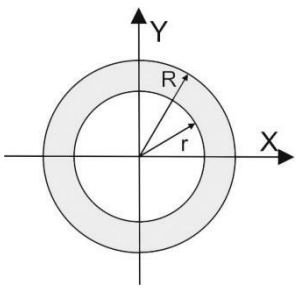
Отчет к лабораторной работе должен содержать:

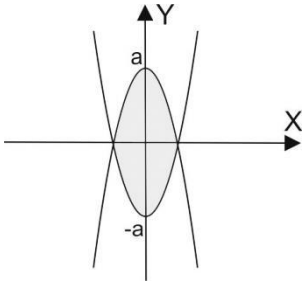
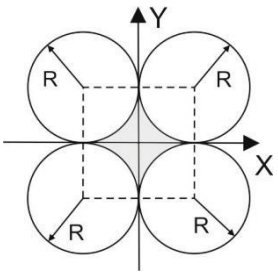
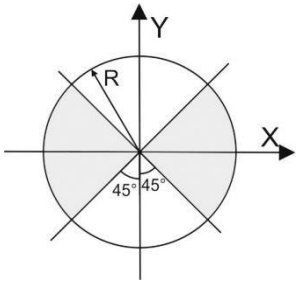
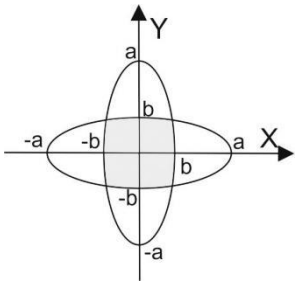
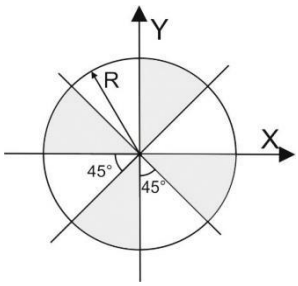
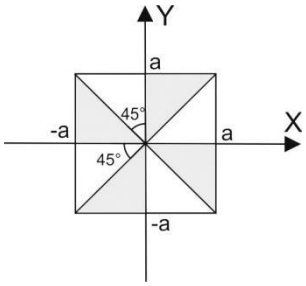
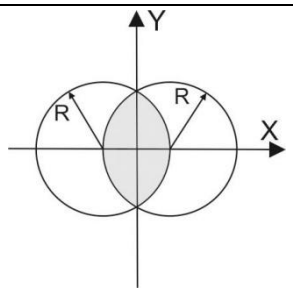
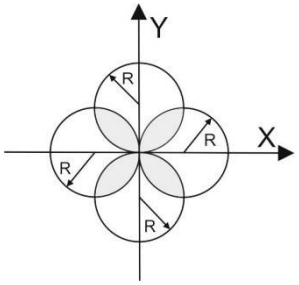
1. Текст задания, вариант.
2. Функции, соответствующие границам области входного зрачка, и их графики.
3. Блок-схему программы.
4. Листинг программы.
5. Скриншоты результатов выполнения программы.

2.5. Варианты заданий для работы

Таблица 2.5.3. Варианты заданий

№ п/п	Форма входного зрачка	№ п/п	Форма входного зрачка
1		2	
3		4	
5		6	
7		8	
9		10	

№ п/п	Форма входного зрачка	№ п/п	Форма входного зрачка
11		12	
13		14	
15		16	
17		18	
19		20	
21		22	

№ п/п	Форма входного зрачка	№ п/п	Форма входного зрачка
23		24	
25		26	
27		28	
29		30	

3. Лабораторная работа №3. Условные и циклические алгоритмические конструкции. Работа с файлами. Построение границ входного зрачка сложной формы

3.1. Задание для работы

Для каждого участка входного зрачка сложной формы согласно варианту в таблице 3.5.1 задать границы и составить уравнения $y = f_i(x)$. Написать программу табулирования функции $y = f_i(x)$. Полученные значения сохранить в текстовый файл. Построить график функции по данным из файла.

3.2. Теоретические сведения

3.2.1. Работа с файлами в C++

Для работы с файлами используются специальные типы данных, называемые потоками. В программах на языке C++ при работе с текстовыми файлами необходимо подключать библиотеку fstream:

```
#include <fstream>
```

Поток ifstream служит для работы с файлами в режиме чтения, а ofstream в режиме записи. Для работы с файлами в режиме, как записи, так и чтения, служит поток fstream.

Для того чтобы записывать данные в текстовый файл, необходимо:

1. Описать переменную типа ofstream.
2. Открыть файл с помощью функции open.
3. Вывести информацию в файл.
4. Обязательно закрыть файл.

Как было сказано ранее, для того чтобы начать работать с текстовым файлом, необходимо описать переменную типа ofstream. Например, так:

```
ofstream outFile;
```

Будет создана переменная outFile для записи информации в файл. На следующем этапе файл необходимо открыть для записи. В общем случае оператор открытия потока будет иметь вид:

```
outFile.open(«file.txt», mode);
```

Здесь outFile — переменная, описанная как ofstream, file.txt — полное имя файла на диске, mode — режим работы с открываемым файлом. Стоит обратить внимание на то, что при указании полного имени файла нужно ставить двойной слеш. Для обращения, например к файлу “accounts.txt”, находящемуся в папке sites на диске D, в программе необходимо указать: “D:\\sites\\accounts.txt”.

Функция open() требует в качестве аргумента строки в стиле C. Это может быть литеральная строка или же строка, сохраненная в символьном массиве.

Файл может быть открыт в одном из следующих режимов:

- ios::in — открыть файл в режиме чтения данных; режим является режимом по умолчанию для потоков ifstream;
- ios::out — открыть файл в режиме записи данных (при этом информация о существующем файле уничтожается); режим является режимом по умолчанию для потоков ofstream;
- ios::app — открыть файл в режиме записи данных в конец файла;
- ios::ate — передвинуться в конец уже открытого файла;
- ios::trunc — очистить файл, это же происходит в режиме ios::out;
- ios::nocreate — не выполнять операцию открытия файла, если он не существует;
- ios::noreplace — не открывать существующий файл.

Параметр mode может отсутствовать, в этом случае файл открывается в режиме по умолчанию для данного потока.

```
ifstream file;
file.open ("Test.txt", ios::in); // открыть файл в режиме для чтения

ifstream file;
file.open ("Test.txt"); // открыть файл в режиме для чтения (по умолчанию)
```

После удачного открытия файла (в любом режиме) в переменной outFile будет храниться true, в противном случае false. Это позволит проверить корректность операции открытия файла.

Для считывания данных из текстового файла, необходимо:

1. Описать переменную типа ifstream.
2. Открыть файл с помощью функции open.
3. Считать информацию из файла, при считывании каждой порции данных необходимо проверять, достигнут ли конец файла.
4. Закрыть файл.

Для того чтобы прочитать информацию из текстового файла, необходимо описать переменную типа ifstream. После этого нужно открыть файл для чтения с помощью оператора open. Если переменную назвать fromFile, то первые два оператора будут такими:

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

После открытия файла в режиме чтения из него можно считывать информацию точно так же, как и с клавиатуры, указав имя потока, из которого будет происходить чтение данных.

Например, для чтения данных из потока fromFile в переменную a, оператор ввода будет выглядеть так:

```
fromFile >> a;
```

Два числа в текстовом редакторе считаются разделенными, если между ними есть хотя бы один из символов: пробел, табуляция, символ

конца строки. Хорошо, когда программисту заранее известно, сколько и какие значения хранятся в текстовом файле. Однако часто известен лишь тип значений, хранящихся в файле, при этом их количество может быть различным. Для решения данной проблемы необходимо считывать значения из файла поочередно, а перед каждым считыванием проверять, достигнут ли конец файла с помощью функции fromFile.eof(). Здесь fromFile — имя потока, eof() – функция, возвращающая логическое значение (true или false), в зависимости от того достигнут ли конец файла.

Следовательно, цикл для чтения содержимого всего файла можно записать так:

```
//организуем для чтения значений из файла цикл, выполнение
//цикла прервется, когда достигнем конец файла,
//в этом случае fromFile.eof() вернет истину
while(!fromFile.eof())
{
//чтение очередного значения из потока fromFile в переменную a
     fromFile >> a;
//далее идет обработка значения переменной a
}
```

Для чтения отдельных символов можно использовать функцию get() и функцию getline() – для чтения целых строк.

3.3. Пример выполнения задания

Задать и построить границы зрачка сложной формы согласно графику на рисунке 3.3.1.

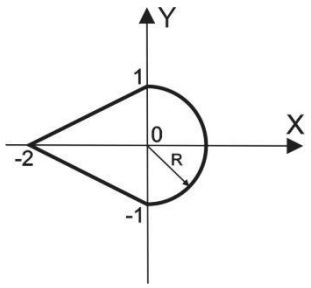


Рисунок 3.3.1. Границы зрачка

```
// лабораторная работа №  вариант №    группа №   студент  ФИО
#include <fstream> // библиотека для работы с файлами
#include <cmath>

using namespace std;

int main()
{
     const double PI =2*acos(0.); // вычислениечислап
     setlocale(LC_ALL, "Russian"); /* подключение русского языка в
консоли */
     ofstream outFile; /* создание потока для вывода информации в
файл */
     outFile.open("points.txt"); // связь потока с файлом
```

```
	double x = 0.0, y = 0.0, alpha = 0.0;
	// цикл по прямой верхней части графика
	for (x = -2; x <= 0; x++)
	{
		y = 0.5*x + 1;
		outFile << x << " " << y << endl; // запись в файл
	}
	// цикл по дуге от PI/2 до -PI/2, alpha задан в радианах
	for (alpha = PI/2; alpha >= -(PI/2); alpha = alpha - 0.05)
	{
		x = cos(alpha);
		y = sin(alpha);
		outFile << x << " " << y << endl; // запись в файл
	}
	// цикл по прямой нижней части графика
	for (x = 0; x >= -2; x--)
	{
		y = -(0.5*x) - 1;
		outFile << x << " " << y << endl; // запись в файл
	}
	// завершить работу с файлом
	outFile.close();
}
```

На рисунке 3.3.2 изображены графики, построенные в среде MathCad по данным, записанными в файл “points.txt” в результате выполнения программы.

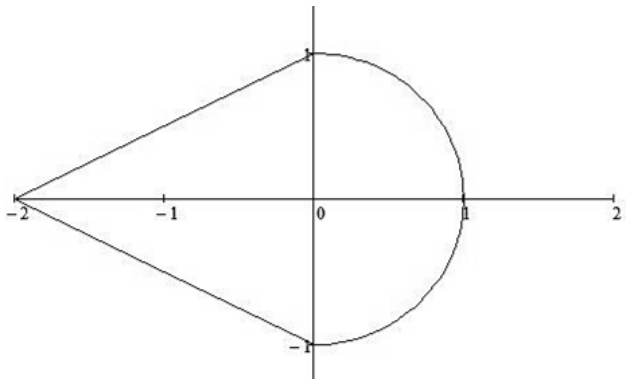


Рисунок 3.3.2. Графики функций

3.4. Содержание отчета

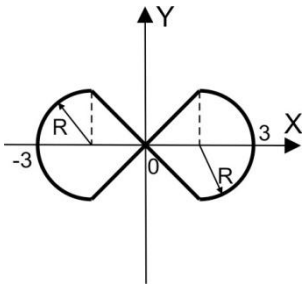
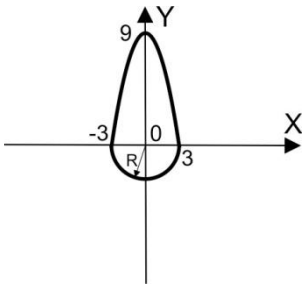
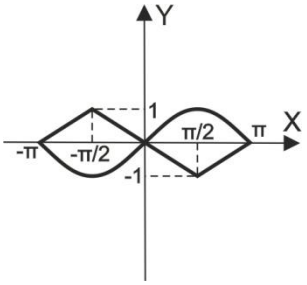
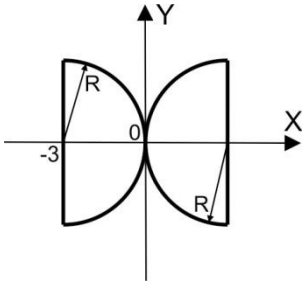
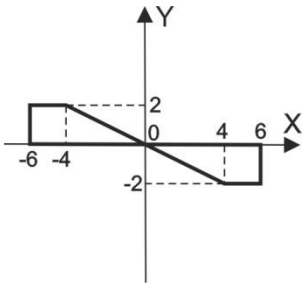
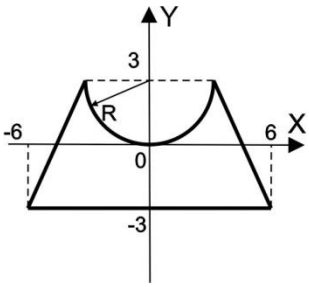
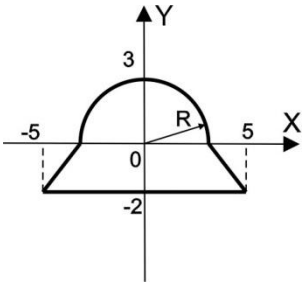
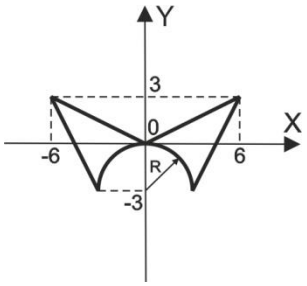
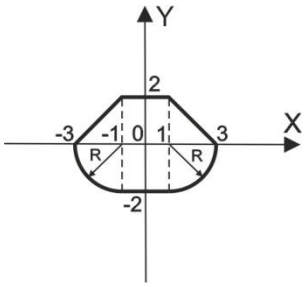
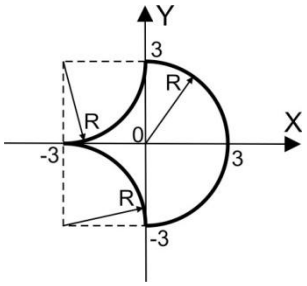
Отчет к лабораторной работе должен содержать:

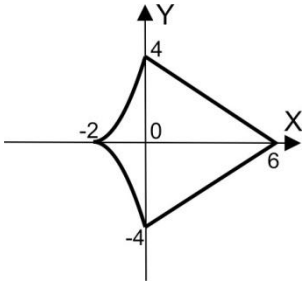
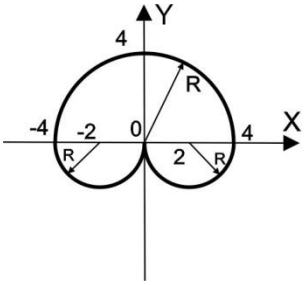
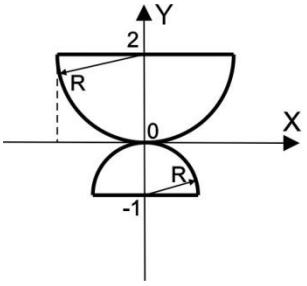
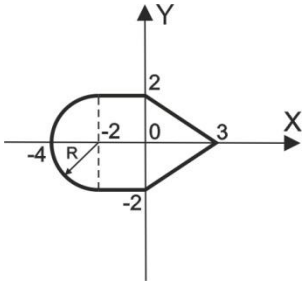
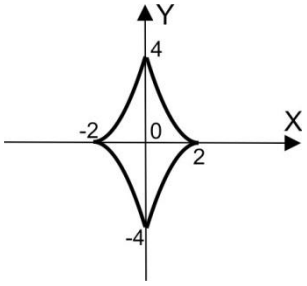
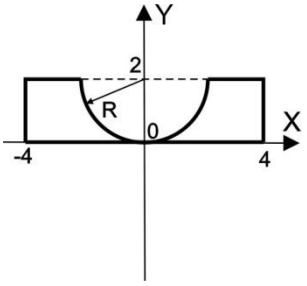
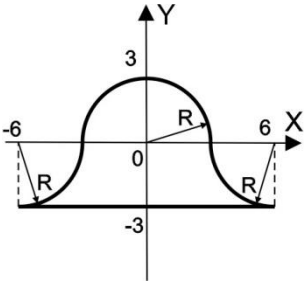
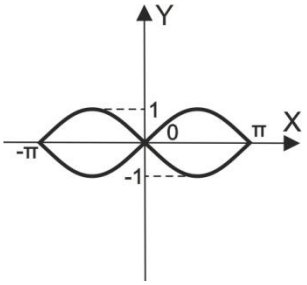
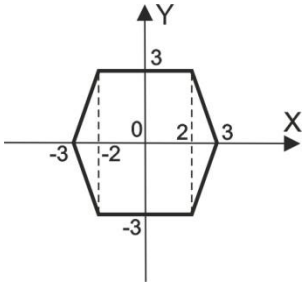
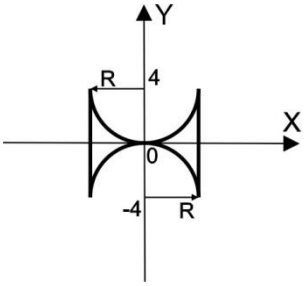
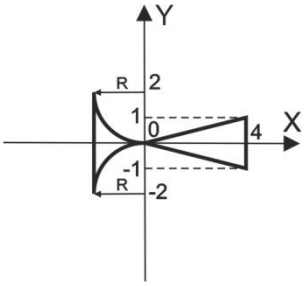
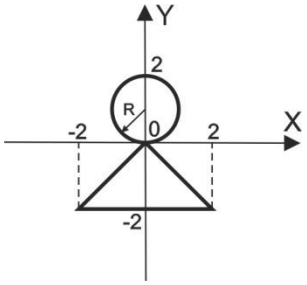
1. Текст задания, вариант.
2. Уравнения всех участков границ зрачка согласно варианту.
3. Блок-схему программы.
4. Листинг программы.
5. Скриншоты результатов выполнения программы.
6. Данные, записанные в файл.

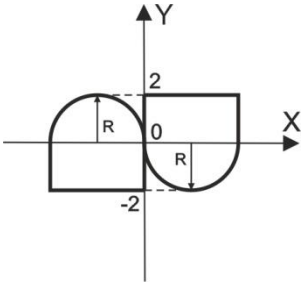
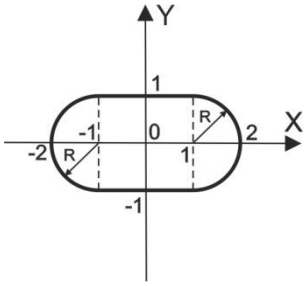
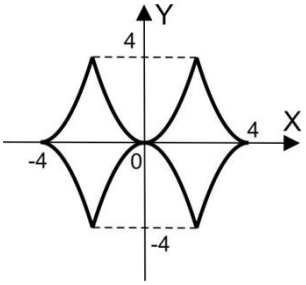
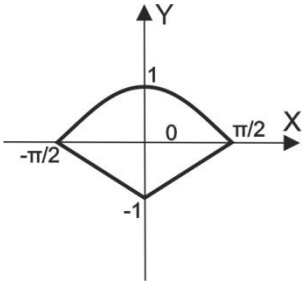
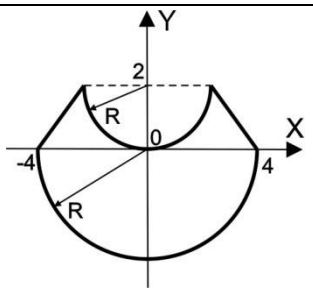
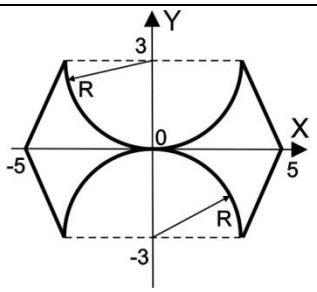
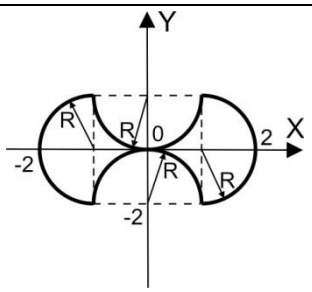
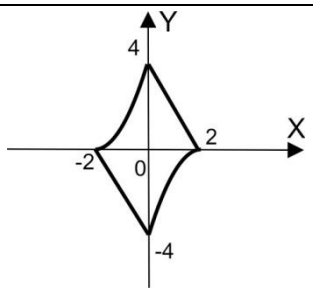
7. Построенное по записанным в текстовый файл данным изображение входного зрачка.

3.5. Варианты заданий для работы

Таблица 3.5.1. Варианты заданий

№ п/п	Форма входного зрачка	№ п/п	Форма входного зрачка
1		2	
3		4	
5		6	
7		8	
9		10	

№ п/п	Форма входного зрачка	№ п/п	Форма входного зрачка
11		12	
13		14	
15		16	
17		18	
19		20	
21		22	

№ п/п	Форма входного зрачка	№ п/п	Форма входного зрачка
23		24	
25		26	
27		28	
29		30	

4. Лабораторная работа №4. Работа с одномерными массивами. Обработка результатов измерений координат центра масс пятна рассеяния

4.1. Задание для работы

Написать программу, которая:

1. Считывает из файла в массив набор из 20 вещественных координат центров масс пятен рассеяния и выводит этот массив в результирующий файл. Среди координат должны быть отрицательные, положительные и равные нулю. Дополнительные параметры, если это необходимо, ввести с клавиатуры.
2. Вычисляет значения всех переменных, которые входят в состав выражения, и значение общего выражения согласно заданию в таблице 4.5.2.
3. Выводит значения всех полученных переменных и выражения с комментариями.
4. Оформить результаты, используя форматный вывод. Для проверки вычислить все указанные величины, используя любое программное обеспечение.

4.2. Теоретические сведения

4.2.1. Работа с массивами в языке C++

Массив - это структура данных, представленная в виде последовательной группы значений одного типа, объединенных под одним именем. Массивы используются для обработки большого количества однотипных данных. Отдельное значение данных массива называется элементом массива. Элементами массива могут быть данные любого типа. Массивы, имеющие одно измерение, называются одномерными.

Для создания массива используется оператор объявления. При объявлении массива необходимо указать тип значений элементов массива, имя массива и его размерность (количество элементов в массиве). Массивы бывают статическими и динамическими. Размерность статического массива указывается при его описании и вместе с его типом определяют объем памяти, необходимый для размещения массива. Размерность статического массива должна быть целочисленной константой или константным выражением, в котором известны все значения на момент компиляции. Объявление одномерного статического массива на языке C++ выглядит следующим образом:

```
double a [10]; //описание массива из 10 элементов вещественного типа
```

Размерность массивов лучше задавать с помощью именованных констант. При таком подходе для ее изменения размерности достаточно скорректировать значение константы в одной строке программы.

В отличие от статических массивов, у динамических размерность может быть переменной, то есть объем памяти, выделяемый под массив, определяется на этапе выполнения программы.

Элементы массива в языке C++ нумеруются с нуля, соответственно первый элемент массива будет иметь индекс 0, второй – 1, третий – 2 и т.д.

При инициализации статического массива все его элементы указываются по порядку в фигурных скобках при объявлении массива в программе:

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

Если при инициализации в фигурных скобках будет указано меньше элементов, чем размерность массива, то все оставшиеся элементы будут равны 0.

Для того, чтобы присвоить значение отдельному элементу массива, используют индексы (оператор []):

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

Если необходимо заполнить массив своими значениями с клавиатуры, из файла или случайными значениями, используются циклы. Пример заполнения одномерного массива случайными числами:

```
#include <iostream> // стандартный поток ввода/вывода
#include <cstdlib> /* заголовочный файл определяет несколько функций общего назначения, в том числе функции генерации случайных чисел*/
#include <ctime> // содержит функции для работы со временем и датой
using namespace std;

int main()
{
	int randomDigits[3] = {};// пустой массив из трех элементов
	srand(time(NULL)); /* устанавливает в качестве базы для генерации случайных чисел текущее время, таким образом, при каждом запуске генерируется новый набор случайных значений*/
	for (int i = 0; i < 3; i++)// цикл для заполнения массива
	{
		randomDigits[i] = rand(); /* функция rand() генерирует случайные числа*/
		cout << randomDigits[i] << endl;
	}
	return 0;
}
```

В приведенном выше примере массив из 3 элементов заполняется случайными целыми числами.

4.2.2. Средние значения действительных чисел

Средние значения n действительных чисел $a_1, a_2, a_3, \dots, a_n$ вычисляются по формулам, указанным в таблице 4.2.1.

Таблица 4.2.1. Средние значения действительных чисел

Среднее арифметическое	$A_n = \frac{a_1 + a_2 + a_3 + \cdots + a_n}{n}$
Среднее геометрическое	$G_n = \sqrt[n]{a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_n}$
Среднее квадратическое	$Q_n = \sqrt{\frac{1}{n}(a_1^2 + a_2^2 + \cdots + a_n^2)}$
Среднее гармоническое	$M_n = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3} + \cdots + \frac{1}{a_n}}$, где $a_i \neq 0$

Вычисление среднего гармонического должно производиться только для ненулевых элементов массива.

4.3. Пример программы

Вычислить характеристику координат центров масс пятен рассеяния по формуле $C = A + B$, где A – среднее арифметическое отрицательных координат (элементов массива), больших D ($D < 0$), B – номер минимальной координаты (элемента массива) из тех, что имеют нечетный номер.

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

using namespace std;

int main()
{
	const int size = 20; // размер массива
	double M[size], A = 0.0, D = 0.0, Mid = 0.0, C = 0.0;
	int B, i, n;
	ifstream inFile("In.txt"); // файл с исходным массивом
	ofstream outFile("Out.txt"); // результирующий файл
	// ввод массива
	for (i = 0; i < size; i++)
	{
		inFile >> M[i];
	}
	inFile.close();
	// вывод массива для контроля
	for (i = 0; i < size; i++)
	{
		cout << M[i] << endl;
	}
	// поиск минимального среди элементов с номерами 1,3,...11
	B = 1; // пусть минимальный - это первый элемент массива
	for (i = 3; i < size, i += 2)
	{
```

```
            if (M[i] < M[B])
                  B = i; // обновить B, если M[B] не минимум
      }
      outFile << "Номер минимального элемента с нечетным номером = "
<< B << endl;
      // поиск среднего арифметического
      cout << "Введите D";
      cin >> D;
      n = 0; // счетчик для среднего арифметического
      Mid = 0; // сумма элементов для среднего арифметического
      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 << " Среднее арифметическое > " << D <<
endl;
            C = A + B;
            outFile << "C = " << C;
      }
      else
      {
            outFile << "Отрицательных больших " << D << " нет";
      }
      outFile.close();
}
```

4.4. Содержание отчета

Отчет к лабораторной работе должен содержать:

1. Текст задания, вариант.
2. Блок-схему программы.
3. Листинг программы.
4. Скриншоты результатов выполнения программы.
5. Входные и выходные данные из файлов.
6. Проверку правильности вычисления значений программой.

4.5. Варианты заданий для работы

Таблица 4.5.2. Варианты заданий

№ п/п	Выражение	Определение переменных
1	$\frac{A+C}{B+C}$	A –среднее квадратичное положительных координат (элементов массива); B -количество ненулевых координат (элементов массива); C -произведение ненулевых координат (элементов массива).
2	$\frac{A}{A+1}+C+B$	A – среднее гармоническое отрицательных координат (элементов массива) больших D; B - сумма координат (элементов массива) с четным номером; C - номер последней положительной координаты (элемента массива).
3	$\left(1+\frac{1}{A}+\frac{1}{B}\right)C$	A –количество координат (элементов массива) меньших D с нечетным номером; B - среднее арифметическое первых M координат (элементов массива); C - модуль наименьшей координаты (элемента массива).
4	$\frac{1}{A+B+C}+3$	A – наибольшая четная координата (элемент массива); B - среднее квадратичное всех координат (элементов массива); C - количество координат (элементов массива) из интервала [A, B].
5	$\frac{A+B+C}{A\cdot B\cdot C}+1$	A – среднее гармоническое первых M координат (элементов массива) не равных 0; B - номер наименьшей по модулю ненулевой координаты (элемента массива); C -количество координат (элементов массива) меньших D.
6	$\frac{A+B+C}{100}+A$	A – наибольшая координата (элемент массива) с нечетным номером; B - среднее гармоническое координат (элементов массива) с четным номером; C - сумма положительных координат (элементов массива) с четными номерами.
7	$\frac{A\cdot B}{C+2}$	A –сумма отрицательных координат (элементов массива) больших D; B - количество положительных координат (элементов массива) с нечетным номером; C -сумма положительных координат (элементов массива).
8	$\frac{A\cdot B\cdot C}{A+B+C}$	A –сумма координат (элементов массива) из интервала [A, B]; B - среднее квадратичное координат (элементов массива) с четным номером; C -номер последней отрицательной координаты (элемента массива).

№ п/п	Выражение	Определение переменных
9	$\frac{A}{(B+1)(C+1)}$	A –номер наименьшей положительной координаты (элемента массива); B - количество нулевых координат (элементов массива); C -среднее гармоническое положительных координат (элементов массива).
10	$\frac{A}{10}+\frac{C+B}{10}$	A – количество отрицательных координат (элементов массива) с нечетным номером; B - сумма координат (элементов массива) с нечетным номером; C -номер первой положительной координаты (элемента массива).
11	$\frac{B}{A+1}+C$	A –среднее арифметическое отрицательных координат (элементов массива); B - произведение отрицательных координат (элементов массива); C -сумма модулей отрицательных координат (элементов массива).
12	$\frac{B\cdot C+A}{A\cdot B+2}$	A –наибольшая по модулю координата (элемент массива); B - сумма отрицательных координат (элементов массива); C -произведение модулей отрицательных координат (элементов массива).
13	$C+\frac{A}{100+B}$	A –сумма первых M координат (элементов массива); B - номер последней нулевой координаты (элемента массива); C -количество отрицательных координат (элементов массива).
14	$(A+1)(B+2)(C+3)$	A –сумма координат (элементов массива) больших 0 с нечетным номером; B - среднее арифметическое координат (элементов массива); C -произведение первых M координат (элементов массива) не равных 0.
15	$\frac{C}{(A+1)(B+1)}+C$	A –среднее арифметическое положительных координат (элементов массива); B - сумма всех координат (элементов массива); C -произведение координат (элементов массива) с нечетным номером.
16	$\frac{A\cdot C+B}{B\cdot C+A}$	A –сумма координат (элементов массива) больших D с нечетным номером; B - номер последней отрицательной координаты (элемента массива); C -количество положительных координат (элементов массива).

№ п/п	Выражение	Определение переменных
17	$A + B + \frac{1}{C}$	A –наименьшая координата (элемент массива) с четным номером; B - произведение отрицательных координат (элементов массива); C -среднее арифметическое координат (элементов массива).
18	$C + \frac{A}{10} + \frac{B}{100}$	A –наименьшая по модулю координата (элемент массива) не равная 0; B - среднее гармоническое отрицательных координат (элементов массива); C -номер первой отрицательной координаты (элемента массива).
19	$\frac{A^2}{B + C}$	A –произведение положительных координат (элементов массива); B - наибольшая отрицательная координата (элемент массива); C -среднее арифметическое положительных координат (элементов массива).
20	$\frac{A \cdot B}{C} + \frac{A}{B}$	A –среднее арифметическое координат (элементов массива) с нечетным номером; B - наибольшая по модулю координата (элемент массива); C -номер наименьшей координаты (элемента массива).
21	$\left(A + \frac{1}{A}\right) + B \cdot C$	A –произведение координат (элементов массива) с четными номерами; B - среднее арифметическое модулей координат (элементов массива) меньших 0; C -номер наибольшей координаты (элемента массива).
22	$C + \frac{1}{A + B} - 1$	A –среднее арифметическое координат (элементов массива) с четным номером; B - наименьшая положительная координата (элемент массива); C -количество координат (элементов массива) больших D.
23	$\frac{A}{A \cdot B \cdot C} + C^3$	A –номер наибольшей координаты (элемента массива); B - произведение координат (элементов массива) больших 0 с нечетным номером; C -сумма отрицательных координат (элементов массива).
24	$\frac{R}{Q + 1} + S$	A –сумма координат (элементов массива) с четным номером; B - среднее гармоническое координат (элементов массива) не равных 0; C -произведение ненулевых координат (элементов массива).
25	$(A + B)(C + A) - C$	A –номер наибольшей по модулю координаты (элемента массива); B–наименьшая координата (элемент массива); C -среднее квадратичное первых M элементов массива.

№ п/п	Выражение	Определение переменных
26	$C + \frac{A + B}{A \cdot B + 1}$	A –среднее арифметическое отрицательных координат (элементов массива), больших D; B - произведение координат (элементов массива) из интервала [A, B]; C -номер наибольшей отрицательной координаты (элемента массива);
27	$\frac{A + 1}{(C + 2)B}$	A –модуль наименьшей координаты (элемента массива); B - среднее гармоническое отрицательных координат (элементов массива); C -сумма положительных координат (элементов массива) с четными номерами.
28	$\frac{(A + B) \cdot C}{(B + C + 4)}$	A –среднее гармоническое первых M координат (элементов массива) не равных 0; B - сумма отрицательных координат (элементов массива) больших D; C -номер первого нулевого координат (элементов массива).
29	$\frac{C}{A \cdot B \cdot C + 1}$	A – сумма координат (элементов массива) из интервала [A, B]. B - наибольшая отрицательная координата (элемент массива); C -количество нулевых элементов массива.
30	$\frac{C}{C + 1} + A + B$	A –наибольшая координата (элемент массива); B - количество координат (элементов массива), меньших D; C -среднее квадратичное первых M координат (элементов массива).

5. Лабораторная работа №5. Работа с двумерными массивами. Попиксельная обработка цифровых изображений

5.1. Задание для работы

Написать программу, которая считывает из текстового файла значения интенсивностей пикселей цифрового изображения (значение интенсивности пикселя в диапазоне от 0 до 255), заполняет ими двумерный массив $A[m,n]$, производит его обработку согласно варианту задания в таблице 5.5.1 и выводит по формату в результирующий файл следующие данные: размер изображения (массива), исходные и полученные значения интенсивностей всех пикселей обработанного изображения (исходный и результирующие массивы).

Размер обрабатываемого изображения не должен превышать 20×20 пикселей.

5.2. Теоретические сведения

5.2.1. Двумерные массивы в C++

Двумерный массив — это одномерный массив одномерных массивов. Основные характеристики двумерного массива: количество строк и количество столбцов. В памяти двумерные массивы хранятся последовательно по строкам.

Элементы двумерного массива, как и одномерного нумеруются с нуля. Объявление массива должно содержать следу

- тип значений;
- имя массива;
- размерность (количество элементов в массиве, а именно количество строк и столбцов).

При указании размерности двумерного массива обе размерности указываются в квадратных скобках:

```
matr [6][8];    //описание массива из 6 строк и 8 столбцов
```

Если при инициализации двумерный массив задается как массив из массивов, тогда первую размерность, а именно количество строк, можно опустить:

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

Также при инициализации двумерного массива можно указать элементы списком через запятую в том порядке, в котором они располагаются в памяти. При этом обе размерности массива необходимо указать.

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

Для доступа к любому элементу двумерного массива указываются все его индексы:

```
matr [i][j]; // i – номер строки, j – номер столбца
matr [2][2] = 5; //элемент в 3 строке и 3 столбце будет равен 5.
```

5.3. Пример программы

Все пиксели черного цвета (значение интенсивности 0) в исходном изображении поменять на пиксели белого цвета (значение интенсивности 255).

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

using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian");
	// размеры двумерного массива
	const int rows = 4; // количество строк
	const int cols = 4; // количество столбцов
	int A[rows][cols];
	ifstream inFile("In.txt"); // файл с исходным массивом
	ofstream outFile("Out.txt"); // результирующий файл
	// Ввод в двумерный массив
	for (int i = 0; i < rows; i++) // цикл по строкам
	{
		for (int k = 0; k < cols; k++) // цикл по столбцам
		{
			inFile>>A[i][k];  // запись значения в элемент
массива
		}
	}
	inFile.close(); // закрываем файл на чтение
	outFile << "Исходная матрица:" << endl;
	// вывод массива в файл
	for (int i = 0; i < rows; i++) // цикл по строкам
	{
		for (int k = 0; k < cols; k++) // цикл по столбцам
		{
			outFile << A[i][k] << " "; // записываем значения
элемента массива в файл
		}
		outFile << endl;
	}
	// заменяем элементы массива, равные 0, на 255
	for (int i = 0; i < rows; i++) // цикл по строкам
	{
		for (int k = 0; k < cols; k++) // цикл по столбцам
		{
			if (A[i][k] == 0) // если элемент массива равен 0,
то заменяем значение на 255
			{
```

```
                    A[i][k] = 255;
                }
            }
        }
        outFile << "Результирующая матрица:" <<endl;
        for (int i = 0; i < rows; i++) // цикл по строкам
        {
            for (int k = 0; k < cols; k++) // цикл по столбцам
            {
                outFile << A[i][k] << " ";
            }
            outFile << endl;
        }
        outFile.close();
        return 0;
}
```

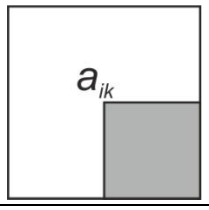
5.4. Содержание отчета

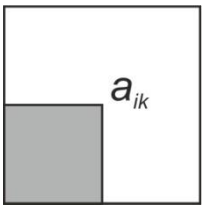
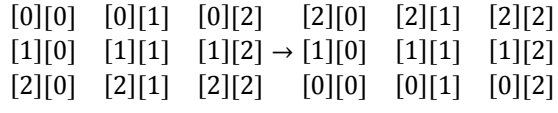
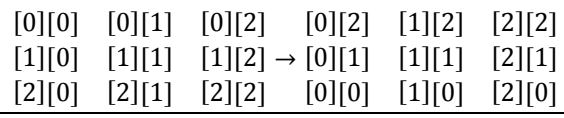
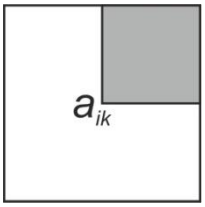
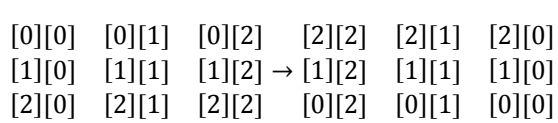
Отчет к лабораторной работе должен содержать:

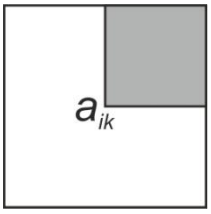
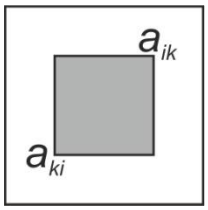
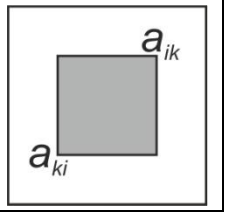
1. Текст задания, вариант.
2. Исходный и результирующий массивы.
3. Блок-схему программы
4. Листинг программы.
5. Скриншоты результатов выполнения программы.

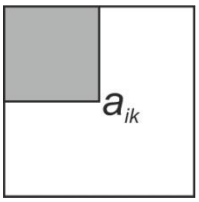
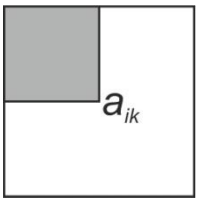
5.5. Варианты заданий для работы

Таблица 5.5.1. Варианты заданий

№ п/п	Задание
1	Отразить исходное изображение по горизонтали: в каждой строке исходного цифрового изображения значения интенсивности каждого пикселя переставить в обратном порядке [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	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен сумме элементов исходного цифрового изображения A из области, определяемой индексами i,j 
3	Повернуть изображение 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]

№ п/п	Задание
4	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен среднему арифметическому элементов исходного цифрового изображения A из области, определяемой индексами i,j. 
5	Посчитать количество пикселей на изображении с каждым возможным уровнем интенсивности. Сформировать одномерный массив, содержащий информацию о количестве элементов с каждым возможным уровнем интенсивности (0-255).
6	Выполнить бинаризацию изображения: все значения интенсивности меньше Dзаменить на 0, остальные на 255. Значение Dввести с клавиатуры по запросу.
7	Все значения интенсивности элементов исходного цифрового изображения больше D_{max}поменять на максимально возможное значение интенсивности (255), а все значения меньше D_{min} на минимальное значение интенсивности (0). Значения D_{max}и D_{min} ввести с клавиатуры по запросу.
8	Увеличить значения интенсивности каждого пикселя изображения на D. Если новое значение будет превышать максимальное значение интенсивности 255, принять его равным 255.
9	Отразить исходное изображение по вертикали: в каждом столбце исходного цифрового изображения все значения интенсивности переставить в обратном порядке 
10	Сформировать цифровое изображение, каждый элемент которого $b_{i,j}$ будет средним арифметическим элемента исходного цифрового изображения $a_{i,j}$и всех элементов окружающих его
11	Повернуть изображение на 90° влево 
12	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен сумме элементов исходного цифрового изображения A из области, определяемой индексами i,j 
13	Найти все значения интенсивности исходного цифрового изображения равные D, посчитать их количество, заменить их на 255, остальные значения заменить на 0 Значение D ввести с клавиатуры по запросу.
14	Отразить изображение относительно верхнего левого угла (переставить элементы исходного цифрового изображения в обратном порядке) 

№ п/п	Задание
15	Выполнить фильтрацию, в результате которой сформировать новое цифровое изображение, в котором каждый элемент $b_{i,j}$ будет серединой одномерного массива отсортированного по возрастанию и состоящего из элемента $a_{i,j}$ и всех элементов его окружающих
16	Сформировать массив интенсивностей пикселей изображения размерностью m×nявляющийся центральной частью исходного цифрового изображения
17	Сформировать цифровое изображение, каждый элемент которого $b_{i,j}$ будет средним арифметическим элемента исходного цифрового изображения $a_{i,j}$ и элементов стоящих в изображении справа и слева от него
18	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен среднему арифметическому элементов исходного цифрового изображения A из области, определяемой индексами i,j 
19	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен сумме элементов исходного цифрового квадратного изображения A из области, определяемой индексами i,j. 
20	Выполнить фильтрацию, в результате которой сформировать цифровое изображение, в котором каждый элемент $b_{i,j}$ будет серединой одномерного массива отсортированного по возрастанию и состоящего из элемента исходного цифрового изображения $a_{i,j}$ и всех элементов его окружающих
21	Выполнить обработку цифрового изображения: все значения элементов, лежащие в диапазоне от D_{min} до D_{max}, заменить на 255, все остальные элементы исходного изображения оставить без изменений. Значения D_{max}и D_{min} ввести с клавиатуры по запросу.
22	Преобразовать исходное изображение (массив значений интенсивностей пикселей) в негатив. Для этого каждый элемент цифрового изображения изменить по формуле: $b_{i,j} = 255 - a_{i,j}$
23	Считать из исходного файла ДВА цифровых изображения одинаковой размерности и сравнить их поэлементно: если значение элемента первого изображения больше или равно соответствующему элементу второго изображения, то записать в результирующее изображение 255, если меньше – 0
24	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен среднему арифметическому элементов исходного квадратного цифрового изображения A из области, определяемой индексами i,j. 

№ п/п	Задание
25	Выполнить обработку цифрового изображения: все значения элементов больше Dзаменить на 255, все остальные элементы исходного изображения оставить без изменений. Значение D ввести с клавиатуры по запросу.
26	Считать из исходного файла ДВА изображения одинаковой размерности и сравнить их поэлементно: если значение элемента первого изображения меньше или равно соответствующему элементу второго изображения, то записать в результирующее изображение 255, если больше – 0
27	Найти в исходном цифровом изображении все элементы равные 0 и заменить их на среднее арифметическое окружающих элементов
28	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен среднему арифметическому элементов исходного изображения A из области, определяемой индексами i,j 
29	Считать из исходного файла ДВА цифровых изображения одинаковой размерности и сформировать новое изображение, каждый элемент которого рассчитывается по формуле: $c_{i,j} = a_{i,j} + b_{i,j}$
30	Сформировать цифровое изображение B, элемент $b_{i,j}$ которого равен сумме элементов исходного цифрового изображения A из области, определяемой индексами i,j 

6. Лабораторная работа №6. Работа со строками в языке C++

6.1. Задание для работы

Написать программу, которая считывает из файла исходный текст, согласно заданию в таблице 6.5.2, и формирует новый текст. Полученный текст записать в файл.

Исходный текст состоит из строки, содержащей не более 200 символов. В конце предложения есть точка. Каждому слову, кроме первого, предшествует один пробел, внутри слова пробелов нет. Знаки препинания, если они есть, пишутся сразу после слова.

6.2. Теоретические сведения

6.2.1. Строки языке C++

Строка – массив символов, заканчивающийся нуль-символом.

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

Объявление строк аналогично одномерным массивам типа char. Инициализировать строку можно строковым литералом (константой), при инициализации строки при ее определении размерность можно опустить. В этом случае будет выделено необходимое количество байт для этой строки.

```
char str[10] = "Vasia"; // инициализация строки литералом
```

Для строки будет выделено 10 элементов от 0 до 9. Первые 6 элементов строки: 'V', 'a', 's', 'i', 'a', '\0'.

```
char str[] = "Vasia"; //при инициализации опущена размерность строки
```

Для строки будет выделено 6 элементов от 0 до 5: 'V', 'a', 's', 'i', 'a', '\0'. Символьные строки состоят из набора символьных констант заключённых в двойные кавычки. При объявлении строкового массива необходимо учитывать наличие в конце строки нуль-символа, и отводить дополнительный байт под него.

В языке C++ существует две возможности работы со строками:

1. Функции, унаследованные из библиотеки языка C (<cstring>) Библиотека <cstring> содержит функции копирования строк, сравнения, объединения строк, поиска подстроки, поиска вхождения символа, определения длины строки и другие.
2. Библиотечный класс языка C++ string.

6.2.2. Функции библиотеки <cstring>

Для определения длины строки в библиотеке <cstring> существует функция strlen, которая определяет длину указанной строки без учета нуль-символа.

```
strlen(имя_строки);
```

```
char str[10] = “Vasia”;
a = strlen(str); //переменной a будет присвоено значение 5
```

Для копирования строк реализованы две функции: strcpy и strncpy. Первая выполняет побайтное копирование символов из строки_2 в строку_1. Вторая функция выполняет побайтное копирование n символов из строки_2 в строку_1.

```
strcpy(строка_1,строка_2);
strncpy(строка_1,строка_2, n);

char str_1[] = "Быть или не быть";
char str_2[10];
char str_3[10];

strncpy(str_2, str_1);     //копирует строку str_1 в строку str_2
strncpy(str_3, str_1, 8);  /* копирует в строку str_3 8 символов из
str_1 */
```

Чтобы избежать переполнения, строка, на которую указывают str_2 и str_3 должны быть достаточно длинными, чтобы в них поместилась копируемая строка (включая завершающий нулевой символ). Копируемая строка и строка назначения не должны перекрываться в памяти.

Конкатенация строк осуществляется посредством функций strcat и strncat:

```
strcat(строка_1,строка_2); /*  объединяет  строку_2  со  строкой_1.
Результат сохраняется в строке_1 */
strncat(строка_1,строка_2,n);   /* объединяет n символов строки_2
сострокой_1. Результат сохраняется встроке_1 */
char str_1[] = "Hello ";
char str_2[] = "World!";
strcat(str_1,str_2); //в str_1 будет "Hello World!";
```

Функция добавляет копию строки_2 в конец строки_1. Нулевой символ конца строки_1 заменяется первым символом строки_2, и новый нуль-символ добавляется в конец уже новой строки, сформированной объединением символов двух строк в строке_1.

Библиотека <cstring> позволяет сравнивать строки между собой с учетом или без учета регистра. Функция strcmp сравнивает строку_1 со строкой_2 с учетом регистра и возвращает результат типа int: 0 – строки эквивалентны, больше 0 – строка_1 меньше строки_2, меньше 0 – строка_1 больше строки_2:

```
strcmp(строка_1, строка_2);
```

Функция strncmp сравнивает n символов строки_1 со строкой_2 с учетом регистра и возвращает результат типа int: 0 – строки эквивалентны, больше 0 – строка_1 меньше строки_2, меньше 0 – строка_1 больше строки_2:

```
strncmp(строка_1, строка_2, n);
```

С первого символа попарно сравнивает все символы двух строк, пока не буду найдены различные символы или не будет достигнут конец строки.

Функции для сравнения без учета регистра работают аналогично функциям сравнения строк с учетом регистра:

```
stricmp(строка_1, строка_2);
strnicmp(строка_1, строка_2, n);
```

В библиотеке <cstring> существуют следующие функции поиска:

```
strchr(строка, символ);
```

Функция осуществляет поиск первого вхождения символа в строке. В случае удачного поиска возвращает указатель на место первого вхождения символа. Если символ не найден, то возвращается ноль.

Завершающий нулевой символ считается частью строки. Таким образом, он также может быть найден для получения указателя на конец строки.

```
strcspn(строка_1, строка_2);
```

Определяет длину начального сегмента строки_1, содержащего те символы, которые не входят в строку_2. Поиск учитывает и завершающий нуль-символов, поэтому, если функция возвращает длину строки_1, это значит, что ни один из символов строки_2 не входит в состав строки_1.

```
strspn(строка_1, строка_2);
```

Функция strspn возвращает длину начального сегмента строки_1, содержащего только те символы, которые входят в строку_2. Поиск учитывает и завершающий нуль-символ, поэтому, если функция возвращает длину строки_1, это значит, что все символы строки_2 входят в состав строки_1.

```
strprbk(s1,s2);
```

Возвращает указатель первого вхождения любого символа строки_2 в строке_1.

При необходимости можно преобразовать значение строки в другой тип данных. Для преобразования строки с тип double необходимо использовать следующую функцию:

```
atof(строка_1);
```

Для преобразования в тип int:

```
atoi(строка_1);
```

Для преобразования в тип longint:

```
atol(строка_1);
```

6.2.3. Функции стандартной библиотеки ввода/вывода

Для работы со строками в языке C++ также используются функции стандартной библиотеки ввода/вывода:

```
getchar(c);
```

Функция getchar считывает символ с со стандартного потока ввода, возвращает символ в формате int.

```
gets(s);
```

Функция gets считывает поток символов со стандартного устройства ввода в строку s до тех пор, пока не будет нажата клавиша ENTER.

6.2.4. Функции библиотеки <cctupe> для обработки символов

В таблице 6.2.1 приведены функции, которые позволяют проверить принадлежность символа к тому или иному типу.

Таблица 6.2.1. Функции обработки символов

Функция	Описание функции
`isalnum(c);`	Возвращает значение true, если с является буквой или цифрой, и false в других случаях.
`isalpha(c)`	Возвращает значение true, если с является буквой, и false в других случаях
`isdigit(c)`	Возвращает значение true, если с является цифрой, и false в других случаях
`islower(c)`	Возвращает значение true, если с является буквой нижнего регистра, и false в других случаях
`isupper(c)`	Возвращает значение true, если с является буквой верхнего регистра, и false в других случаях
`isspace(c)`	Возвращает значение true, если с является пробелом, и false в других случаях
`toupper(c)`	Если символ с, является символом нижнего регистра, то функция возвращает преобразованный символ с в верхнем регистре, иначе символ возвращается без изменений.

6.3. Пример программы

В исходной строке длиной 70 символов - предложение, состоящее не менее чем из двух слов. Сформировать новую строку, в которой из последнего слова предложения удалена первая буква.

```
#include <iostream>
#include <cstdio>

int main()
{
	setlocale(LC_ALL, "Russian");
	using namespace std;
	char StIn[70], StOut[70]; // входная и выходная строки
	int Nend, N1; // позиции в тексте
	cout << "Введите текст: ";
	gets_s(StIn); // функция gets() считывает все введённые символы
с пробелами до тех пор, пока не будет нажата клавиша Enter
	Nend = strlen(StIn); // вернёт длину строки
	char * pch = strrchr(StIn, ' '); // Возвращает указатель на
последнее вхождение символа в строке
```

```
    N1 = (pch - StIn + 1); // количество символов до последнего
пробела
    // скопировать в результирующую строку N1 символов
    strncpy_s(StOut, StIn, N1);
    // добавить последнее слово без первой буквы
    strncat_s(StOut, &StIn[N1 + 1], (Nend - N1 - 1));
    cout << StOut;
    cin.get();
}
```

6.4. Содержание отчета

Отчет к лабораторной работе должен содержать:

1. Текст задания, вариант.
2. Исходный и результирующий тексты;
3. Листинг программы.
4. Скриншоты результатов выполнения программы.

6.5. Варианты заданий для работы

Таблица 6.5.2. Варианты заданий

№ п/п	Исходный текст	Задание
1	Для удобства чтения оптических схем и компьютерных расчетов в оптике приняты единые правила знаков. Положительным направлением света считается распространение слева направо.	Поменять предложения местами
2	Волновая аберрация пропорциональна отклонениям оптических длин лучей пучка.	Слова разделить запятыми
3	Если разложить поле на монохроматические составляющие (каждая с определенной длиной волны), то вся энергия некоторым образом распределится между ними.	Текст в скобках перенести в конец предложения (перед точкой)
4	В идеальной оптической системе все лучи, исходящие из точки, пересекаются в сопряженной с ней точке.	Удалить вторую запятую
5	Кома появляется при смещениях точки предмета с оси. Кома добавляется.	Слова второго предложения поменять местами
6	Поэтому влияние волновой аберрации на качество изображения не зависит от типа изображения а определяется тем, сколько длин волн она составляет.	Вставить недостающую запятую перед союзом "а"
7	Апланатизм может выполняться только для какой-то части предмета. В окрестности оси.	Второе и третье слова второго предложения поменять местами

№ п/п	Исходный текст	Задание
8	Спектральная плотность потока излучения –показывает распределение энергии по спектру излучения. Поверхностная плотность потока энергии – это величина потока, приходящегося на единицу площади.	Первые слова предложений поменять местами
9	Световой поток, распространяясь в оптической среде, частично поглощается.	Вывести на печать часть текста между запятыми
10	Монохроматические аберрации: сферическая, кома, астигматизм и кривизна изображения, дисторсия.	Третье и последнее слова предложения поменять местами
11	Поверхности: плоские, сферические, асферические.	Каждое слово с новой строки
12	Абсолютно черное тело – это тело, которое полностью поглощает падающую на него энергию.	Слова до и после дефиса поменять местами
13	Энергетический коэффициент пропускания.	Новый порядок слов: 2,3,1
14	Дисперсия оптических материалов.	Среднее слово - в круглые скобки
15	Оптическая поверхность – это гладкая регулярная поверхность точно известной формы. Оптическая среда – это прозрачная однородная среда с точным значением показателя преломления.	Вторые слова предложений поменять местами
16	Идеальная оптическая система.	Новый порядок слов: 3,2,1
17	Угол между лучом и оптической осью считается положительным, если для совмещения оси с лучом ось нужно вращать по часовой стрелке.	Заменить слово “ось” словом “ее”
18	Геометрическая теория оптических изображений.	Новый порядок слов: 4,3,2,1
19	В названии “неизопланатизм” присутствуют корни греческих слов: изо – одинаковый, равный, планета – блуждающее тело.	Удалить слово вместе с кавычками
20	Если квадратный предмет изображается в виде подушки – это дисторсия положительная. Если изображение квадрата имеет выпуклые стороны (в виде бочки), то это дисторсия отрицательная.	Последние слова предложений поменять местами
21	Полевая диафрагма – это диафрагма, ограничивающая размеры поля.	Третье слово - в кавычки (“)

№ п/п	Исходный текст	Задание
22	Можно выделить одну (наименьшую) диафрагму, и считать, что остальные не ограничивают ход лучей. Такая диафрагма называется апертурной.	Удалить слово, начинающееся на букву “а”, из текста
23	Верхний луч внеосевого пучка – это луч, проходящий через верхний край апертурной диафрагмы и соответствующие ему сопряженные точки входного и выходного зрачков.	Третье и седьмое слова предложения заключить в круглые скобки
24	Изображение суммы объектов равно сумме изображений каждого объекта. При смещении точки ее изображение только смещается на пропорциональную величину.	Точки в конце предложений заменить знаком “?”
25	Хроматизм положения – это аберрация, при которой изображения одной точки предмета расположены на разном расстоянии от оптической системы для разных длин волн (разные положения плоскости изображения).	Удалить третье слово
26	Апланатизм – нет ни комы, ни сферической аберрации. Изображение разных точек предмета идеальное. Апланатизм может выполняться только для какой-то части предмета, например в окрестности оси.	Первое и последнее предложения поменять местами
27	Если поток излучается площадкой, то поверхностная плотность потока энергии будет иметь смысл энергетической светимости.	Заменить слово “поток” символом “Ф”
28	Амплитуда поля не может непосредственно наблюдаться или измеряться.	Заключить это слово в апострофы.
29	Если угол падения невелик, то часть поля отражается, а часть преломляется.	Вместо второй запятой символ “;”
30	Оптический диапазон состоит из следующих видов излучения: рентгеновское, ультрафиолетовое (УФ), видимое, инфракрасное (ИК).	Исключить две первые запятые

7. Лабораторная работа №7. Использование функций в языке С++. Решение оптических задач

7.1. Задание для работы

Написать программу, содержащую функцию для решения оптической задачи согласно варианту в таблице 7.5.1.

Программа должна содержать функции для:

- ввода исходных данных;
- расчета необходимых оптических величин (в главной программе необходимо предусмотреть возможность обращения к функции с различными исходными данными);
- вывода результатов решения задачи с комментариями.

Вывод всей информации в результирующий файл осуществлять по формату. Организовать проверку правильности ввода данных пользователем.

7.2. Теоретические сведения

7.2.1. Функции в языке С++

Функция – это именованная последовательность описаний и операторов, обычно предназначенная для решения определенной задачи. Функция может принимать параметры и возвращать значение.

Любая программа на С++ состоит из функций, одна из которых должна иметь имя main. С функции mainначинается выполнение программы. Выполнение функции начинается в момент ее вызова. Как и переменная, функция должна быть объявлена и определена, при этом объявление функции должно находиться в тексте программы раньше ее вызова.

При объявлении функции указывается ее имя, тип возвращаемого значения и список параметров, передаваемых в нее:

```
int sum (int a, int b); //объявление функции
```

Определение функции содержит, кроме объявления, тело функции, представляющее собой последовательность описаний и операторов в фигурных скобках:

```
int sum (int a, int b) { //определение функции
     return (a+b);
}
```

В объявлении, определении и при вызове одной и той же функции типы и порядок следования параметров должны совпадать. На имена параметров ограничения не накладываются.

Для того, чтобы вызвать функцию, необходимо указать ее имя, после которого в круглых скобках через запятую перечисляются имена передаваемых в функцию аргументов:

```
#include <iostream>
int sum (int a, int b); //объявление функции

int main(){
	int a = 2, b = 3, c, d;
	c = sum(a, b); /*вызов функции и передача в качестве входных параметров значений переменных a и b*/
	cin >> d;
	cout << sum(c, d); /*вызов функции и передача в качестве входных параметров значений переменных c и d*/
	return 0;
}
```

Все величины, описанные в функции, и ее параметры являются локальными, областью их действия является функция. При выходе из функции значения локальных переменных не сохраняются, так как участок стека освобождается. Чтобы этого избежать используется параметр static.

При совместной работе функции должны обмениваться информацией. Это можно осуществить с помощью:

- глобальных переменных;
- через параметры;
- через возвращаемое функцией значение.

Возврат из функции в вызвавшую ее функцию реализуется оператором return:

```
return [выражение];
```

Функция может содержать несколько операторов return, однако ее работа прекращается при выполнении первого возможного оператора return. Если функция описана как void, выражение не указывается. Оператор return можно опускать для функций типа void, если возврат из неё происходит перед фигурной скобкой, и для функции main.

```
int function_1 {
	return 1;
}
double function_2 {
	return 1; //1 преобразуется к типу double
}
```

ВАЖНО: Нельзя возвращать из функции указатель на локальные переменные, поскольку память, выделенная локальным переменным при входе в функцию, освобождается после возврата из неё.

7.2.2. Параметры функции

В функциях есть два типа параметров: формальные и фактические. Формальные параметры (параметры) – это параметры, перечисленные в заголовке описания функции, их еще называют просто параметрами. Фактические параметры (аргументы) – это параметры записанные в операторе вызова функции.

Существует два способа передачи параметров функции:

- по значению;

- по адресу:
  - по указателю;
  - по ссылке.

При передаче параметров по значению в память заносятся копии значений аргументов, и операторы функции работают с этими копиями. Доступа к исходным значениям параметров у функции нет, а, следовательно, нет возможности их изменить.

При передаче по адресу в память заносятся копии адресов аргументов, функция осуществляет доступ к ячейкам памяти по этим адресам и может изменить исходные значения аргументов.

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

В примере выше параметр i передается по значению. Его изменение функцией не повлияет на исходное значение. Параметр j передается по адресу с использованием указателя, при этом для передачи в функцию адреса фактического параметра используется операция взятия адреса, а для получения его значения функции требуется операция разыменования.

Параметр k передается по адресу с помощью ссылки. При передаче по ссылке в функцию передается адрес указанного при вызове параметра, внутри функции все обращения к параметру неявно разыменовываются. При использовании ссылок, а не указателей читаемость программы лучше, также это избавляет от использования операций получения адреса и разыменования.

7.3. Пример программы

Написать функцию для расчета линейного увеличения β. Входные параметры для функции: размер изображения y', размер предмета y.

```
#include <iostream>

using namespace std;

double CalcLinMagnification(double objectSize, double imageSize)
{
      return imageSize / objectSize;
}
```

```
int main()
{
	setlocale(LC_ALL, "Russian"); /* подключение русского языка в
консоли */
	double y, y_;
	cout << "Введите размер предмета ";
	cin >> y;
	cout << "Введите размер изображения ";
	cin >> y_;
	cout << "Линейное увеличение = " << CalcLinMagnification(y, y_)
<< endl;
	cin.get();
	cin.get();
}
```

7.4. Содержание отчета

Отчет к лабораторной работе должен содержать:

1. Текст задания, вариант.
2. Блок-схема функции.
3. Листинг программы.
4. Скриншоты результатов работы программы.

7.5. Варианты заданий для работы

Таблица 7.5.1. Варианты заданий

№п/п	Задание	Формулы для расчета
1	Написать функцию для определения передней апертуры A объектива для предмета дальнего типа. Входные параметры для функции: относительное отверстие объектива $1:k$, фокусное расстояние объектива f'	$\frac{D}{f'} = 1:k$ $A = \frac{D}{2}$
2	Написать функцию для определения диаметра выходного зрачка объектива. Входные параметры для функции: относительное отверстие объектива $1:k$, фокусное расстояние объектива f в мм, линейное увеличение объектива β	$1:k = \frac{D}{f'}$ $\beta = \frac{D}{D'}$
3	Свет падает нормально на границу раздела двух сред. Написать функцию для вычисления показателя преломления второй среды. Входные параметры для функции: показатель преломления первой среды n_1 и коэффициент отражения ρ	$\rho = \left(\frac{n_2 - n_1}{n_2 + n_1}\right)^2$

№п/п	Задание	Формулы для расчета
4	Написать функцию для вычисления яркости квадратного рассеивателя. Входные параметры для функции: поток рассеивателя Ф в лм, сторона квадрата a в м, степень белизны поверхности k	$L = \frac{kE}{\pi}$ $Ф = ES$
5	Написать функцию по вычислению оптической силы D тонкой линзы в воздухе с показателем преломления n_0. Входные параметры для функции: показатель преломления линзы n, радиусы передней R_1 и задней R_2 поверхностей линзы	$D = \frac{1}{F} = \left(\frac{n_л}{n_{ср}} - 1\right)\left(\frac{1}{R_1} + \frac{1}{R_2}\right)$
6	В опыте Ллойда световая волна, исходящая непосредственно из источника, интерферирует с волной, отраженной от зеркала. В результате на экране образуется система интерференционных полос. Написать функцию для вычисления ширины интерференционной полосы. Входные параметры для функции: расстояние от источника до экрана l, высотаот источника до плоскости зеркала $d/2$, длина волны света λ	$\Delta x = \frac{\lambda}{d} l$
7	Световая волна падает на зеркала Френеля. Написать функцию по определению длины волны света λ. Входными параметрами для функции: угол между зеркалами φ, ширина интерференционной полосы Δl, расстояние от источника до ребра зеркал r, расстояние от ребра до экрана L_0	$\lambda = 2\varphi r \cdot \frac{\Delta l}{L_0 + r}$
8	Написать функцию для вычисления угла преломленного луча α_2. Входные параметры для функции: угол падения луча α_1, показатели преломления первой n_1 и второй n_2 среды	$n_1 \sin \alpha_1 = n_2 \sin \alpha_2$
9	Написать функцию для вычисления диаметра полевой диафрагмы $D_{пд}$ в плоскости предмета. Входные параметры для функции: величина изображения y' в мм, угловое увеличение системы β	$D_{пд} = 2y$ $\beta = \frac{y'}{y}$

№п/п	Задание	Формулы для расчета
10	Написать функцию для вычисления оптической силы D двухкомпонентной оптической системы. Входные параметры для функции: оптические силы первого D_1 и второго D_2 компонентов оптической системы, расстояние между компонентами d	$D = D_1 + D_2 - D_1 D_2 d$
11	Написать функцию для вычисления коэффициента пропускания τ при нормальном падении света. Входные параметры для функции: показатель преломления первой среды n_1 и второй среды n_2	$\tau = \frac{4n_2 n_1}{n_2 + n_1}$
12	Написать функцию для вычисления радиуса линзы шара. Входные параметры для функции: фокусное расстояние линзы шара f', показатель преломления n	$D = \frac{1}{f'}$ $D = \frac{2(n-1)}{nr}$
13	Написать функцию для вычисления положения входного зрачка a. Входные параметры для функции: расстояние от тонкой линзы до апертурной диафрагмы a', фокусное расстояние линзы f'	$\frac{1}{a'} - \frac{1}{a} = \frac{1}{f}$
14	Написать функцию для расчета радиуса круглой площадки. Входные параметры для функции: светимость круглой площадки M, поток, излучаемый площадкой Ф	$M = \frac{\Phi}{S}$ $S = \pi R^2$
15	Написать функцию для расчета полного потока от плоского ламбертовского излучателя. Входные данные для функции: сила света плоского ламбертовского излучателя I, телесный угол образован вращением плоского угла σ	$\Phi = \frac{\pi I \sin^2 \sigma}{2}$
16	Написать функцию для расчета показателя преломления n плосковогнутой линзы. Входные параметры для функции: оптическая сила линзы D, радиус поверхности линзы r_2	$D = (n-1)\left(-\frac{1}{r_2}\right)$
17	Свет падает нормально на границу раздела двух сред. Написать функцию для вычисления коэффициента пропускания τ. Входные параметры для функции: показатель преломления первой среды n_1 и второй среды n_2	$\rho = \left(\frac{n_2 - n_1}{n_2 + n_1}\right)^2$ $\rho + \tau = 1$

№п/п	Задание	Формулы для расчета
18	Написать функцию для вычисления коэффициента виньетирования сверху $K_{\text{в}}$. Входные параметры для функции: диаметр апертурной диафрагмы $D_{\text{АД}}$, высота верхнего луча внеосевого пучка на апертурной диафрагме $h_{\text{в}}$	$a_{\text{в}} = \frac{D_{\text{АД}}}{2 - h_{\text{в}}}$ $K_{\text{в}} = \frac{2a_{\text{в}}}{D_{\text{АД}}}$
19	Луч света падает на плоскопараллельную стеклянную пластину. Написать функцию для вычисления величины смещения луча Δx , прошедшего через эту пластину. Входные параметры для функции: толщина стеклянной пластинки d, угол падения θ , показатель преломления n	$\Delta x = d \sin\theta \left(1 - \frac{1 - \sin^2\theta}{n^2 - \sin^2\theta}\right)$
20	Написать функцию для расчета силы света сферического ламбертовского излучателя. Входные данные для функции: полный поток Ф, телесный угол образован вращением плоского угла σ	$I = \frac{\Phi}{\Omega}$ $\Omega = 4\pi \sin^2\left(\frac{\sigma}{2}\right)$
21	Написать функцию для вычисления оптической силы D двухкомпонентной оптической системы. Входные параметры для функции: фокусные расстояния первой f_1 и второй f_2 линз, расстояние между линзами d	$D = \frac{1}{f_1} + \frac{1}{f_2} - \frac{d}{f_1 f_2}$
22	Написать функцию для вычисления диаметра входного зрачка телескопической системы. Входные параметры для функции: угловое поле в пространстве предмета ω , угловое поле в пространстве изображения ω' диаметр выходного зрачка D' телескопической системы	$\Gamma = \frac{D}{D'}$ $\Gamma = \frac{\operatorname{tg}\omega'}{\operatorname{tg}\omega}$
23	Написать функцию для вычисления размера изображения y' . Входные параметры для функции: величина предмета y в мм, угловое увеличение системы W	$\beta = \frac{y'}{y}$ $\beta = \frac{1}{W}$
24	Написать функцию для вычисления расстояния d между элементами двухкомпонентной оптической системы. Входные параметры для функции: оптические силы первого D_1 и второго D_2 компонентов оптической системы, оптическая сила оптической системы D	$D = D_1 + D_2 - D_1 D_2 d$

№п/п	Задание	Формулы для расчета
25	Написать функцию для расчета полного потока Ф сферического ламбертовского излучателя. Входные данные для функции: сила света I, телесный угол образован вращением плоского угла σ	$I = \frac{\Phi}{\Omega}$ $\Omega = 4\pi \sin^2\left(\frac{\sigma}{2}\right)$
26	Написать функцию для вычисления показателя преломления n_2. Входные параметры для функции: показатель преломления n_2, угол полного отраженияε (в градусах)	$\sin\varepsilon = \frac{n_2}{n_1}$
27	Написать функцию для вычисления фокусного расстояния линзы f'. Входные параметры для функции: расстояние от тонкой линзы до апертурной диафрагмы a', положение входного зрачка a	$\frac{1}{a'} - \frac{1}{a} = \frac{1}{f}$
28	Луч света падает на плоскопараллельную стеклянную пластину. Написать функцию для вычисления толщины стеклянной пластинки d. Входные параметры для функции: смещение луча Δx, прошедшего через эту пластину, угол падения θ, показатель преломления n	$\Delta x = d\sin\theta\left(1 - \frac{1-\sin^2\theta}{n^2-\sin^2\theta}\right)$
29	Написать функцию для вычисления угла полного отражения (в градусах). Входные параметры для функции: показатели преломления n_1 и n_2	$\sin\varepsilon = \frac{n_2}{n_1}$
30	Написать функцию для вычисления полного потока Ф сферического ламбертовского излучателя. Входные данные для функции: сила света I, телесный угол образован вращением плоского угла σ	$I = \frac{\Phi}{\Omega}$ $\Omega = 4\pi \sin^2\left(\frac{\sigma}{2}\right)$

8. Лабораторная работа №8. Пользовательские типы данных в языке С++. Структуры. Работа с каталогом оптических стекол

8.1. Задание для работы

На основе данных из файла разработать и создать структуру, содержащую данные об оптических стеклах. Создать массив структур и заполнить его данными из файла. Выполнить индивидуальное задание согласно заданию в таблице 8.5.1.

8.2. Теоретические сведения

8.2.1. Структуры в языке С++

В языке С++ предусмотрена возможность создания, так называемых, пользовательских типов данных, т.е. типов данных, состав которых определяется программистом. Одним из инструментов для создания пользовательских типов данных является использование структур.

В отличие от массива, все элементы которого однотипны, структура объединяет элементы разных типов.

В языке С++ структуры являются видом класса и обладают всеми его свойствами, но во многих случаях достаточно использовать структуры так, как они определены языке С:

```
struct [имя_типа] {
      тип_1 элемент_1;
      тип_2 элемент_2;
…
      тип_n  элемент_n;
} [список_описателей];
```

Элементы структуры называется полями структуры и могут иметь любой тип, кроме типа этой же структуры, но могут быть указателями на него.

Если отсутствует имя типа, должен быть указан список описателей переменных, указателей или массивов. В этом случае описание структуры служит определением элементов этого списка.

```
//определение массива структур и указателя на структуру
struct {
      char fio[30];
      int date, code;
      double salary;
} staff[100], *ps;
```

Если список отсутствует, описание структуры определяет новый тип, имя которого можно использовать в дальнейшем наряду со стандартными типами, например:

```
struct Worker{ //описание нового типа Worker
      char fio[30];
```

```
      int date, code;
      double salary;
}; //конец определения структуры
Worker staff[100], *ps; /* определение массива типа Worker и
указателя на него */
```

Имя структуры можно использовать сразу после его объявления в тех случаях, когда компилятору не требуется знать размер структуры:

```
struct List; //объявление структуры List
struct Link{
      List *p; //указатель на структуру List
      Link *prev, *succ; //указатель на структуру Link
}
struct List {
      определение_структуры_List
};
```

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их и описания:

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

При инициализации массивов структур следует заключать в фигурные скобки каждый элемент массива:

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

Для переменных одного и того же структурного типа определена операция присваивания, при этом происходит поэлементное копирование. Структуры можно передавать функции, возвращать в качестве значения функции. Другие операции со структурами могут быть определены пользователям.

Доступ к полям структуры выполняется с помощью операции выбора . (точка) при обращении к полю через имя структуры и -> при обращении через указатель:

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

8.2.2. Битовые поля

Битовые поля – это особый вид полей структуры. Битовые поля удобно использовать для хранения флажков типа "да/нет". Таким флажкам достаточно одного бита, в то время как минимальная адресуемая ячейка памяти 1 байт.

При описании битового поля в структуре после имени через двоеточие указывается длина поля в битах:

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

Битовые поля могут быть любого целого типа. Доступ к полю осуществляется обычным способом – по имени.

8.3. Пример программы

Разработать и создать структуру, содержащую данные об объективах. Создать массив структур и заполнить его данными из файла. Найти все объективы с фокусом в диапазоне от 40 до 60 мм или угловым полем больше 60 градусов.

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

using namespace std;

struct Objective // структура объектива
{
     char name[30]; // название объектива
     double focus; // фокусное расстояние
     double w2; // угловое поле
     double k; // относительное отверстие
};

int main()
{
     setlocale(LC_ALL, "Russian"); /* подключение русского языка в
консоли */
     Objectivecatalog[4] = {    { "Индустар-7", 104, 24, 3.5 },
                                { "Индустар-50", 52, 46, 2.9 },
                                { "Гелиос-10", 14, 53, 1.9 },
                                { "Юпитер-6", 180, 14, 3.2}};

     cout << "Введите фокусное расстояние ";
     double Focus;
     cin >>Focus;
     cout << "Список объективов, у которых фокусное расстояние
больше " << Focus<<endl;
     // название колонок
     cout << "Название\t";
     cout << "F\'\t";
     cout << "2w\t";
     cout << "1:k" << endl;
     // цикл по массиву 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();
}
```

8.4. Содержание отчета

Отчет к лабораторной работе должен содержать:

1. Текст задания, вариант.
2. Текст программы.
3. Содержание исходного текстового файла.
4. Скриншоты результатов работы программы.

8.5. Варианты заданий для работы

Таблица 8.5.1. Варианты заданий

№ п/п	Задание
1	Организовать поиск по марке стекла. Вывести на экран данные по запрашиваемой марке стекла
2	Вывести на экран марки стекол, которые входят в группу флинтов
3	Вывести в файл данные о стеклах, у которых число Аббе равно введенному значению
4	Вывести на экран марку стекла с минимальным показателем преломления
5	Вывести на экран марки стекол с показателем преломления в диапазоне от n_1 до n_2
6	Отсортировать массив структур по возрастанию значения показателя преломления и вывести его в файл
7	Организовать ввод с клавиатуры для добавления данных о марках стекол. Добавить данные в массив, используя ввод с клавиатуры и вывести все данные в файл
8	Вывести в файл марки стекол, у которых число Аббе меньше значения, введенного с клавиатуры
9	Рассчитать для каждой марки стекла угол полного внутреннего отражения относительно воздуха и вывести в файл данные: марка стекла и угол ПВО

№ п/п	Задание
10	Создать второй файл с данными по маркам стекол и сравнить его с данными, записанными в массив структур. При обнаружении совпадения вывести на экран сообщение: «Найдено совпадение!», иначе вывести сообщение: «Совпадений не найдено!»
11	Вывести на экран марки стекол, которые являются кронами
12	Организовать поиск марки стекла по введенному значению показателя преломления и вывести на экран данные найденной марки стекла
13	Для каждой записи структуры рассчитать общую дисперсию и вывести на экран данные: марка стекла, общая дисперсия
14	Вывести на экран марку стекла с показателем преломления больше чем у стекла К8
15	Вывести на экран марки стекол, у которых значение числа Аббе находится в диапазоне от V_1 до V_2
16	Рассчитать среднее арифметическое показателей преломления всех марок стекол, данные о которых записаны в массив структур, и вывести полученное значение на экран
17	Введите с клавиатуры показатель преломления и выведите на экран данные о марках стекол, которые близкий к введенному показателю преломления (отклонение не больше 10%)
18	Вывести на экран стекла, которые входят в группу тяжелых кронов
19	Вывести марки стекол, у которых разница показателей преломления для длин волн e и d минимальна
20	Отсортируйте массив структур по возрастанию значения числа Аббе и выведите отсортированные данные о марках стекол в файл
21	Ввести с клавиатуры значение показателя преломления и выведите на экран данные о марках стекол, у которых показатель преломления больше или равен введенному значению
22	Ввести с клавиатуры значения показателя преломления и числа Аббе. Выведите на экран данные о марках стекол, которые имеют значение числа Аббе близкое к введенному (отклонение не больше 5%) и показатель преломления больше введенного значения
23	Ввести названия двух марок стекол и вывести на экран данные о стеклах, у которых значение показателя преломления больше показателя преломления одной из введенных марок стекла и меньше показателя преломления второй введенной марки
24	Создать второй файл с данными по маркам стекол и сравнить его с данными, записанными в массив структур. Если данных о такой марке стекла в массиве структур нет, то добавить такую запись
25	Вывести марки стекол, которые являются легкими кронами
26	Расшифровать название каждой марки стекла и вывести в файл данные: название марки стекла, расшифровка

№ п/п	Задание
27	Ввести с клавиатуры значение показателя преломления и вывести на экран марки стекол, у которых показатель преломления меньше или равен введенному значению
28	Создать новую структуру, содержащую следующие поля: марка стекла, общая дисперсия. Рассчитать для каждой марки стекла общую дисперсию и записать название марки стекла и вычисленную общую дисперсию в массив с новой структурой. Вывести полученный массив структур в файл
29	Отсортировать массив структур по возрастанию показателя преломления и вывести на экран три первые записи, имеющие минимальное значение показателя преломления
30	Вывести на экран марки стекла, у которых показатель преломления для длины волны e меньше 1,5

Для заполнения массива структур данными использовать каталог оптических стекол по ГОСТ РФ (glassbank.ifmo.ru).

Приложение

1. Начало работы с Microsoft Visual Studio

Microsoft Visual Studio — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности. Ниже описан процесс создания проекта и написания простой программы в Microsoft Visual Studio 2010.

1.1. Создание пустого проекта

После запуска среды программирования перед пользователем открывается начальная страница (рисунок П1.1) где представлены последние разработанные проекты, предусмотрена возможность открытия уже существующих проектов и создания нового проекта.

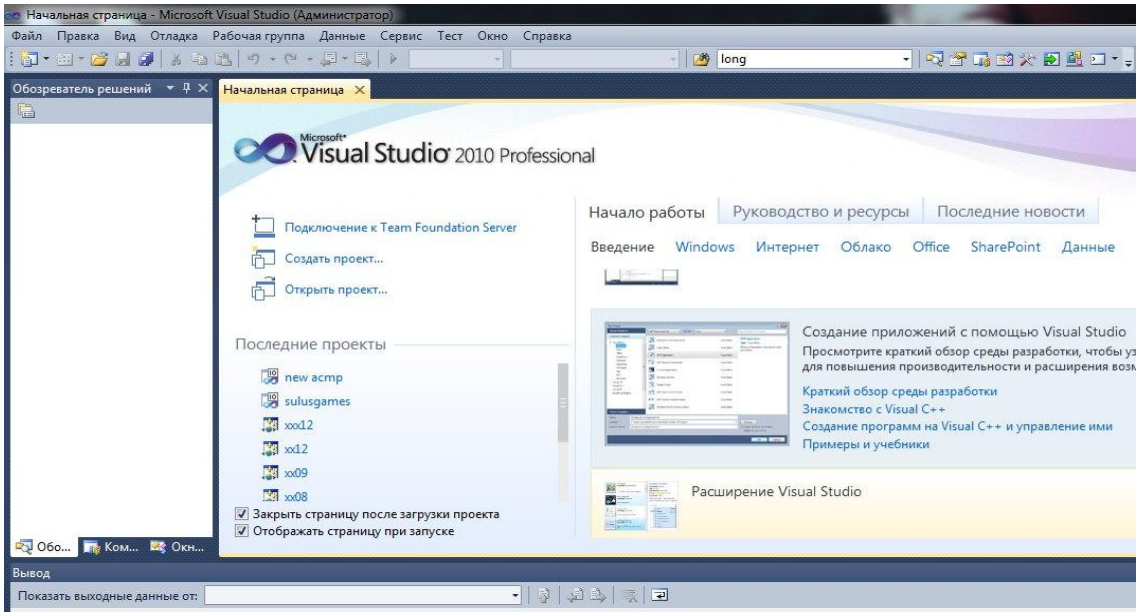


Рисунок П1.1. Начальная страница среды программирования Microsoft Visual Studio

Для создания проекта необходимо выбрать в меню Файл→Создать→Проект, (рисунок П1.2) Также для создания проекта могут быть использованы «горячие» клавиши «Ctrl+Shift+N».

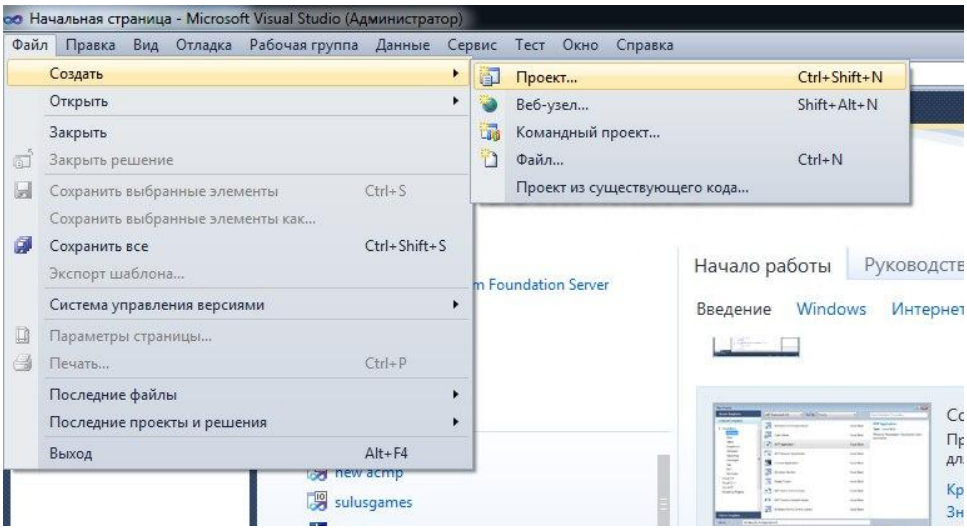


Рисунок П1.2. Создание проекта

Далее открывается окно свойств проекта, состоящее из нескольких частей (рисунок П1.3) В правой части окна необходимо выбрать Visual C++, при этом может быть выбран тип создаваемого проекта. В рамках курса "Основы программирования на C++" рекомендуется использовать «Пустой проект». В нижней части окна свойств указывается имя проекта и выбирается папка, в которой проект будет сохранен. Завершается процесс создания проекта нажатием кнопки «ОК».

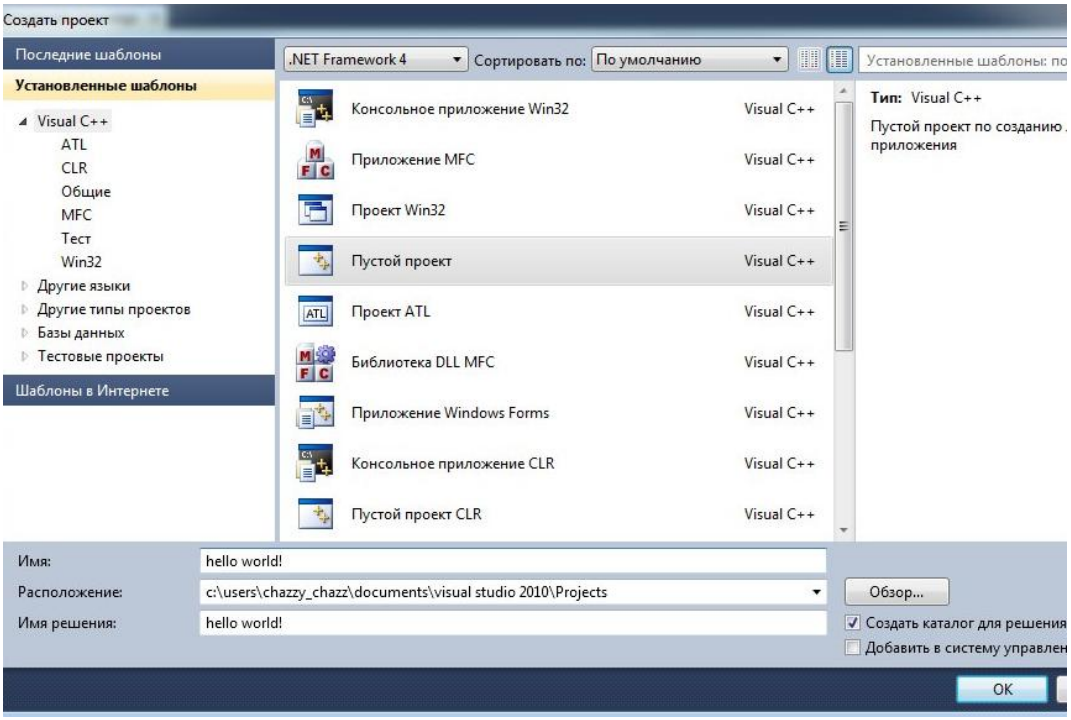


Рисунок П1.3. Окно свойств проекта

Проект создан. Для начала работы необходимо добавить в проект файл с кодом программы.

1.2. Добавление файла с кодом программы

Существует три способа добавления файла в проект:

- использование обозревателя решений (рисунок П1.4);
- выбор в меню Проект→Добавить новый элемент→Создать элемент;
- использование «горячих» клавиш «Ctrl+Shift+A».

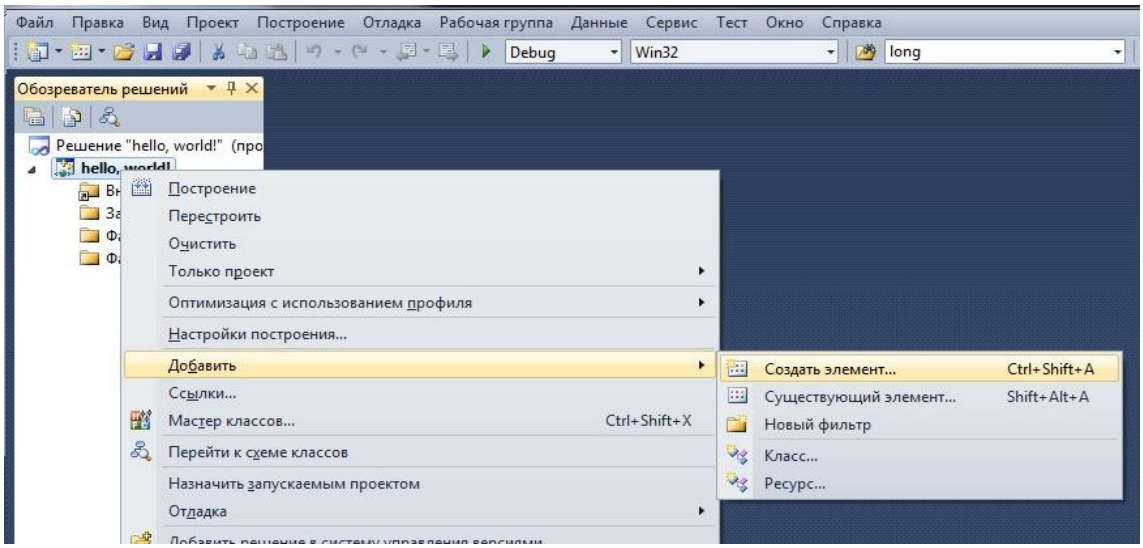


Рисунок П1.4. Добавление файла в проект через обозреватель решений

Вне зависимости от выбранного способа открывается окно добавления нового элемента. В открывшемся окне необходимо выбрать пункт «Файл C++(.cpp)» и написать имя файла в нижней части окна (рис. П1.5).

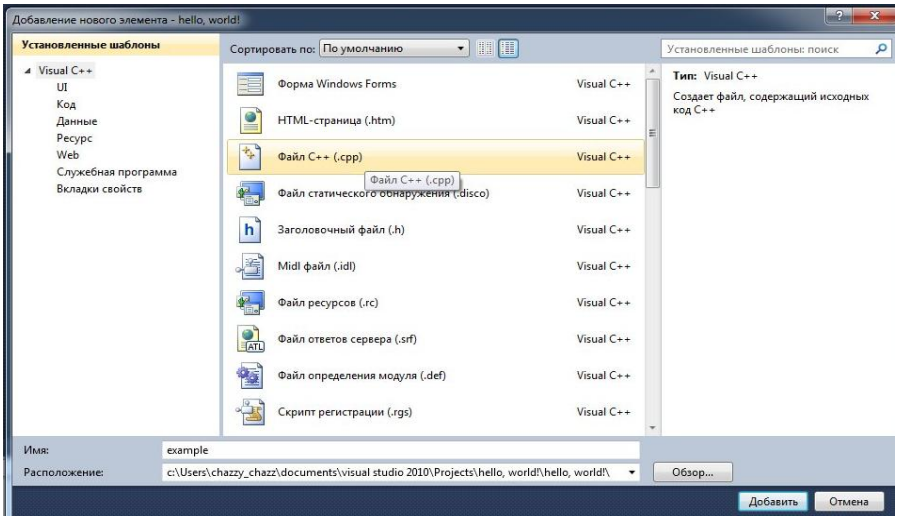


Рисунок П1.5. Добавление нового элемента

После нажатия кнопки «Добавить» открывается страница редактора, предназначенная для написания кода программы.

1.3. Запуск программы на выполнение

После окончания написания программы, необходимо провести ее компиляцию с проверкой на возможные синтаксические ошибки, сборку и, в случае успешной компиляции, запустить программу для выполнения. Для этого можно выбрать на панели инструментов кнопку «Запуск» или воспользоваться клавишей «F5», и при успешной компиляции программы открывается окно консоли, в котором вводятся/выводятся на экран данные, если это предусмотрено программой. На рисунке П1.6 в результате успешного запуска программы в консоли выведено сообщение «Hello, world!».

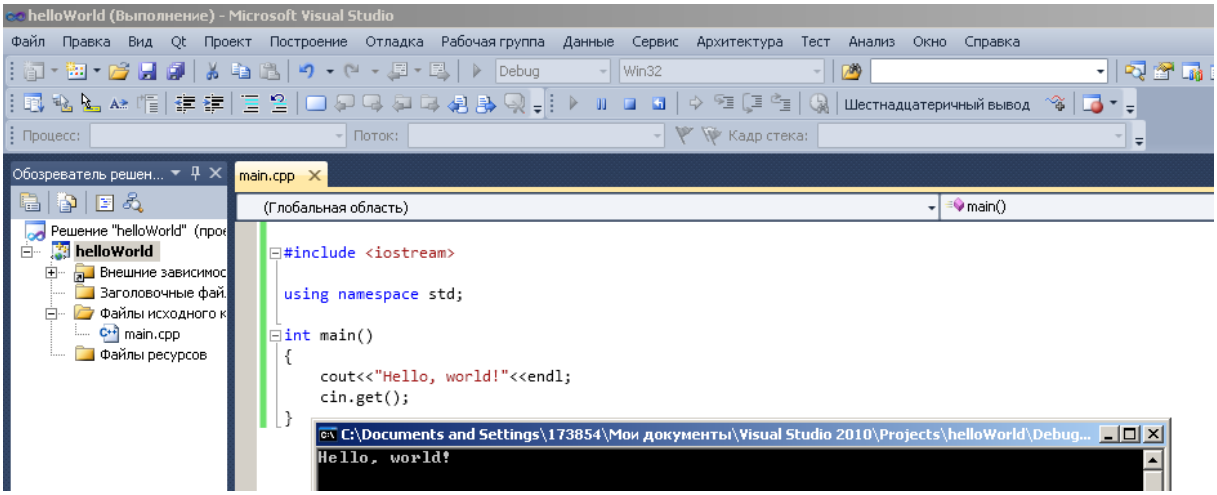


Рисунок П1.6. Результат успешного выполнения программы

В том случае, когда в программном коде есть ошибки, программа запущена не будет. Посмотреть список ошибок можно выбрав в меню «Вид→Список ошибок». На рисунке П1.7 видно, что в коде присутствует синтаксическая ошибка, а именно, отсутствует «;» перед идентификатором cin.

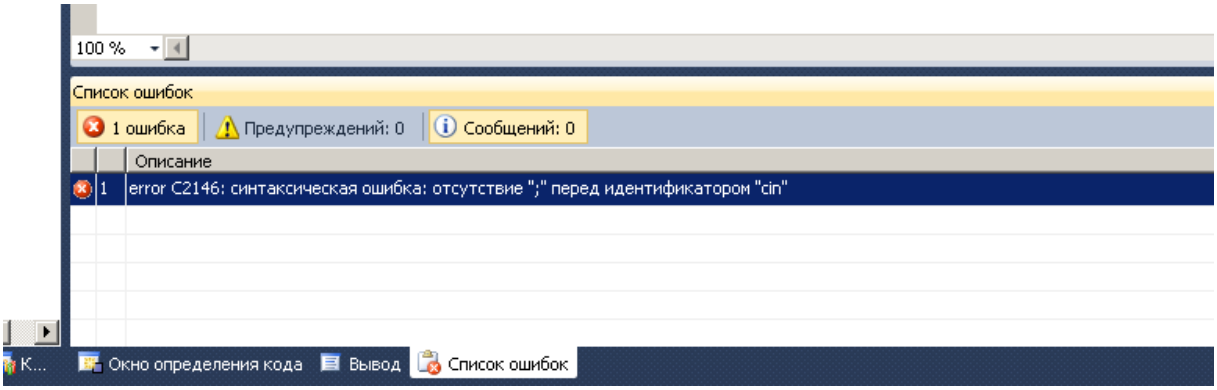


Рисунок П1.7. Список ошибок

1.4. Использование русского языка в консоли

Как видно на рисункеП1.6 в консоли выводится текст на английском языке. В Visual Studio предусмотрена возможность подключить для использования в консоли русский язык. Для этого необходимо просто добавить в код программы строку:

```
setlocale(LC_ALL, "Russian");
```

Теперь можно выводить в консоль текст на русском языке. Это проиллюстрировано на рисунке П1.8.

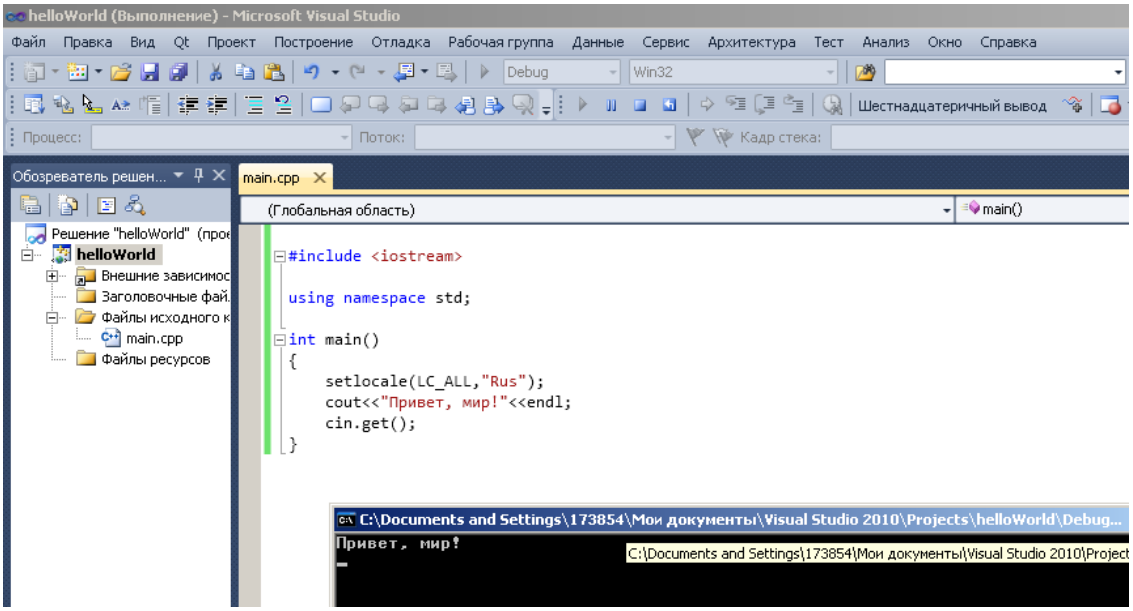


Рисунок П1.8. Подключение русского языка в консоли

Литература

1. Павловская Т.А., Программирование на языке высокого уровня – СПб.: Питер, 2005. – 461 с.
2. Павловская Т.А., Щупак Ю.А., С++. Структурное программирование. Практикум - СПб.: Питер, 2003. – 240 с..
3. С. Прата. Язык программирования С++. 6 издание. – М. Вильямс, 2011. – 1244 с.
4. Егерев В.К., Зайцев В.В., Кордемский Б.А. и др. Под ред. Сканави М.И., Сборник задач по математике для поступающих во ВТУЗы – 6-е изд. – М.: ООО «Издательство «Мир и Образование»: ООО «Издательство «ОНИКС-ЛИТ», 2013. – 608 с.
5. Ежова К.В. Моделирование и обработка изображений - СПб: НИУ ИТМО, 2011. - 93с.
6. Родионов С.А. Основы оптики. - СПб: СПб ГИТМО (ТУ), 2000. - 167с.
7. Толстоба Н.Д., Карпова Г.В., Багдасарова О.В. Основы оптики. Методические рекомендации по организации самостоятельной работы студентов. Часть 1 - СПб: СПбГУ ИТМО, 2009. - 110с.
8. Толстоба Н.Д., Карпова Г.В., Багдасарова О.В. Основы оптики. Методические рекомендации по организации самостоятельной работы студентов. Часть 2 - СПб: СПбГУ ИТМО, 2009. - 95с.

УНИВЕРСИТЕТ ИТМО

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ПРИКЛАДНОЙ И КОМПЬЮТЕРНОЙ ОПТИКИ

Кафедра прикладной и компьютерной оптики - одна из крупнейших кафедр российских ВУЗов, занимающихся задачами современной оптической науки.

Кафедра возникла при слиянии двух кафедр оптического факультета: теории оптических приборов и кафедры оптических приборов и компьютерной оптики. Поэтому на кафедре учат специалистов, имеющих самое широкое представление об оптике в целом, от проектирования оптических систем самого разного назначения до компьютерной обработки изображений и интерферограмм.

Овладение такими разнообразными знаниями невозможно без практической работы с приборами, и кафедра имеет в своем составе несколько учебно-исследовательских лабораторий.

В лаборатории оптических измерений и контрольно-измерительных приборов студенты получают знания и навыки в области метрологии, учатся измерять характеристики оптических систем и параметры деталей и материалов.

Лаборатория микроскопов и медицинских оптических приборов знакомит с различными типами микроскопов (поляризационными, биологическими, металографическими), методами наблюдения микрообъектов и т.п., а также с приборами, применяемыми офтальмологами для диагностики зрения.

Лаборатория геодезических приборов позволяет получить начальные навыки работы с теодолитами, дальномерами и другими приборами, применяемыми в геодезии и картографии, узнать особенности проектирования различных их узлов и конструкции.

В лабораториях компьютерных средств контроля оптики и исследования качества оптического изображения занимаются проблемами контроля качества оптических поверхностей оптической системы в целом, а также компьютеризации и автоматизации этих процессов.

В учебном процессе используются научный потенциал и лабораторная база крупнейшего в России научного центра в области оптики - ВНЦ ГОИ

им. С.И.Вавилова, ведущего оптического предприятия - ОАО "ЛОМО".

Достижения кафедры отмечены двумя Ленинскими премиями, пятью Государственными премиями, премией Совета Министров, премией французской Академии Наук. Кроме того, работы, выполненные на кафедре, отмечались многочисленными медалями и дипломами международных и российских выставок, медалями С.П.Королева, Ю.А.Гагарина, премиями Минвуза.

За период существования кафедры было подготовлено более 150 кандидатов наук, из них 30 иностранцев, а также 16 докторов наук. Большинство научных и производственных подразделений в области прикладной оптики в России, а также многие в США, Израиле и Китае возглавляют ученики нашей научной школы.

В настоящее время кафедра прикладной и компьютерной оптики факультета Лазерной и световой инженерии является одним из крупнейших подразделений Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики, ориентированным на выпуск высококвалифицированных специалистов в области оптотехники.

С информацией о кафедре можно ознакомиться на сайте: aco.ifmo.ru

Ежова Ксения Викторовна
Бурцева Анастасия Александровна
Данцаранов Руслан Олегович

Основы программирования на C++

Методические указания к лабораторным работам

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати 27.12.2016
Заказ № 3814
Тираж 60 экз.
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49