**Министерство образования и науки Российской Федерации**

**ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»**

**Кафедра «УНЦИБ»**

**ОТЧЁТ**

О прохождении летней практике по дисциплине «Языки и методы программирования»

Выполнил:

Студент: Черников Святослав Сергеевич

Специальность (направление подготовки) Информационно-аналитические системы безопасности (10.05.04)

Группа РИ-221055

Проверил: преподаватель Кужбанова Елена Александровна

Екатеринбург, 2024

**ОГЛАВЛЕНИЕ**

# 1. Основные операции языка С++. Часть 1

Написать программу, которая запрашивает у пользователя длины сторон прямоугольника, рассчитывает длину его диагонали и выводит результат вычисления на экран. При составлении программы учесть, что длины сторон прямоугольника – положительные величины.

## 1.1 Реализация

#include <iostream>

using namespace std;

// Функция для нахождения квадратного корня методом Ньютона

double sqrt\_newton(double num) {

// Начальное приближение: берем само число num как x и единицу как y

double x = num;

double y = 1.0;

// Значение точности, до которой хотим вычислить квадратный корень

double epsilon = 0.000001; // точность

// В каждой итерации улучшаем приближения x и y

while (x - y > epsilon) {

// Новое приближение x является средним арифметическим текущих x и y

x = (x + y) / 2;

// Новое приближение y вычисляется как num, деленное на текущее значение x

y = num / x;

}

// Итоговое значение. которое равняется корню из числа, которое поступило на вход

return x;

}

int main()

{

double side1, side2 ;

setlocale(LC\_ALL, "Russian") ;

// Ввод длин сторон a и b

wcout << L"Введите длину стороны a" << endl;

cin>>side1;

wcout << L"Введите длину стороны b" << endl;

cin>>side2;

// Проверка на корректность ввода (больше нуля и числовой тип)

if (side1 <= 0 || side2 <=0)

{

wcout << L"Длины сторон должны быть положительными и не должны равняться 0" << endl;

return -1;

}

// √(a\*\*2+b\*\*2)

// Значение диагонали без корня

double diagonal\_withno\_sqtr = side1\*side1+side2\*side2 ;

// Значение диагонали с корнем (вызываем функцию, которая отвечает за это)

double diagonal\_with\_sqtr = sqrt\_newton(diagonal\_withno\_sqtr) ;

wcout << L"Длина диагонали : " << diagonal\_with\_sqtr << endl ;

return 0;

}

## 1.2 Инструкция пользователя

Введите длины сторон прямоугольника. Вводимые значения должны быть положительными, не равняться 0 и соответствовать числовому типу. После ввода каждого значения необходимо нажать Enter. В случае корректного ввода значений будет вычислена длина диагонали прямоугольника. Результат представлен на рисунке 1.

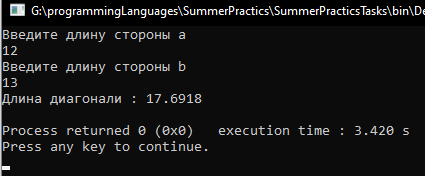


Рисунок 1. Вычисленная диагональ прямоугольника

В случае неправильного ввода в консоль будет выведено сообщение, информирующее об этом. Оно представлено на рисунке 2.

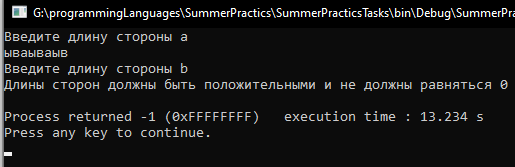


Рисунок 2. Сообщение, информирующее о некорректном вводе значений

# 2. Основные операции языка С++. ЧАсть 2

Пользователь вводит натуральное число n. Программа находит и выводит на экран все совершенные числа на отрезке от 1 до n. Совершенным называется число, сумма делителей которого равна самому числу. Делителем числа считается такое натуральное число, на которое исходное число делится нацело без остатка. При поиске совершенных чисел само число не считается делителем, а единица – считается.

## 2.1 Реализация

#include <iostream>

#include <vector>

using namespace std;

// Функция для проверки, является ли число совершенным

// Число является совершенным, если сумма его делителей равна самому числу

bool isPerfect(int num) {

int sum = 0; // иизначальное значениее суммы

// Находим все числа, на которые число делится без остатка

for (int i = 1; i < num ;++i) {

if (num % i == 0) {

sum += i;

}

}

// Возвращаем логический аргумент, совершенное ли число

return sum == num;

}

int main() {

int n;

setlocale(LC\_ALL, "Russian") ;

// Запрашиваем у пользователя ввод натурального числа

wcout << L"Введите натуральное число n: ";

cin >> n;

if (n <= 0) {

wcout << L"Пожалуйста, введите натуральное число больше 0." << endl;

return 1;

}

// Массив, в котором храним совершенные числа с целью последующего вывода

vector<int> perfectNumbers;

// Ищем все совершенные числа на отрезке от 1 до n

for (int i = 1; i <= n; ++i) {

// Если совершенное - добавляем в массив

if (isPerfect(i)) {

perfectNumbers.push\_back(i);

}

}

// Выводим все совершенные числа

wcout << L"Совершенные числа на отрезке от 1 до " << n << ": ";

// Если пустой

if (perfectNumbers.empty()) {

wcout << L"Нет совершенных чисел.";

} else {

for (int num : perfectNumbers) {

wcout << num << " ";

}

}

cout << endl;

return 0;

}

## 2.2 Инструкция пользователя

Введите натуральное положительное число, которое является крайним в интересующим вас интервале. В случае правильного ввода в консоль будут выведены все совершенные числа от 1 до введённого числа. Результат выполнения программы в случае корректного ввода значений представлен на рисунке 3.

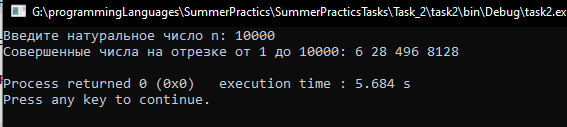


Рисунок 3. Вывод всех совершенных чисел

В случае неправильного ввода в консоль выведется сообщение, информирующее об этом, представленное на рисунке 4.

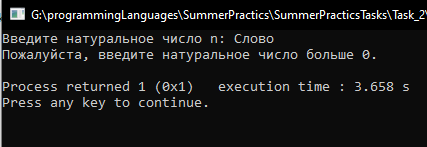


Рисунок 4. Некорректный ввод

# 3. Функции

Написать функцию, которая по номеру дня недели определяет название дня недели и возвращает указатель на соответствующую строку.

## 3.1 Реализация

#include <iostream>

using namespace std;

// Функция, которая возращает указатель на строку по номеру дня

const wchar\_t\* WhatDayOfWeek(int day\_num)

{

// Массив с названиями дней недели

const wchar\_t\* days[] = {

L"Понедельник",

L"Вторник",

L"Среда",

L"Четверг",

L"Пятница",

L"Суббота",

L"Воскресенье"};

// Номер дня недели должен быть числом не меньше 1 и не больше 7

// Возвращаем указатель

return days[day\_num-1] ;

}

int main()

{

int day\_of\_week\_num ;

setlocale(LC\_ALL, "Russian") ;

// Ввод номера дня недели

wcout << L"Введите номер дня недели (от 1 до 7)" << endl;

cin >> day\_of\_week\_num ;

if (day\_of\_week\_num <=0 || day\_of\_week\_num > 7)

{

wcout << L"Введите корректный номер дня недели" << endl ;

return -1 ;

}

// Вызов функции и вывод результатов

const wchar\_t\* day = WhatDayOfWeek(day\_of\_week\_num) ;

wcout << L"День с номером " << day\_of\_week\_num << " " << day << endl ;

return 0;

}

## 3.2 Инструкция пользователя

Введите номер дня (вводимое число должно быть не меньше 1 и не больше 7). В случае корректного ввода в консоль будет выведено название дня недели по введённому номеру. Результат представлен на рисунке 5.

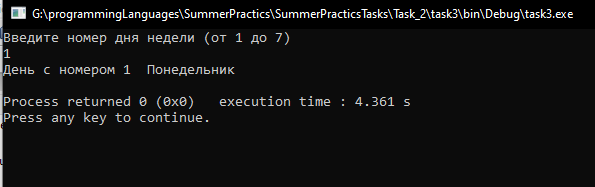


Рисунок 5. Корректное выполнение программы

В случае неправильного ввода номера дня в консоль будет выведено сообщение, информирующее об этом. Оно представлено на рисунке 6.

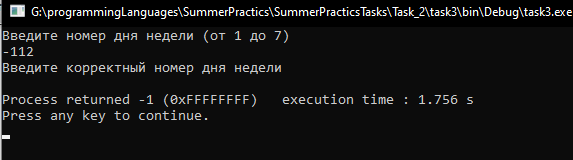


Рисунок 6. Некорректный ввод номера дня недели

# 4. ОДномерные массивы

С одномерным массивом, состоящим из 15 элементов, произвести следующие операции:

1) вычислить произведение положительных элементов массива;

2) вычислить сумму элементов массива, расположенных до минимального элемента. При наличии нескольких минимальных элементов (равных друг другу) выбирается последний из них;

3) упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

## 4.1 Реализация

#include <iostream>

#include <cstdlib>

using namespace std;

// Размер массива. Помещаем его в глобальную область видимости

const int ARRAY\_SIZE = 15 ;

// Функция для сортировки массива методом пузырька

void sortArray(int\* arr, int sizear) {

// Перебираем все элементы массива

for (int i = 0; i < sizear; ++i) {

// Так как в первую итерацию один из элементов "всплывёт",

// то диапазон на 1 меньше

for (int j = 0; j < sizear-1; ++j) {

// Если предшествующий элемент меньше - двигаем вперёд

if (arr[j] > arr[j+1]) {

// Временное значение предыдущего элемента

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

// Функция для разделения массива по чётным и нечётным индексам

// И их последующей сортировки

void sortEvenAndOdd(int\* arr, int sizear)

{

// Массивы для чётных и нечётных элементов

int EvenArr[ARRAY\_SIZE/2 +1], OddArr[ARRAY\_SIZE/2] ;

// Индексы для чётных и нечётных элементов

int EvenIndex = 0, OddIndex = 0 ;

// Разделяем чётные и нечётные элементы массива

for (int index = 0; index < sizear; index++)

{

if (index % 2 == 0) {EvenArr[EvenIndex++] = arr[index];}

else {OddArr[OddIndex++] = arr[index];}

}

// Сортируем массивы чётных и нечётных элементов

sortArray(EvenArr, EvenIndex) ;

sortArray(OddArr, OddIndex) ;

wcout << L"Отсортированные по чётным и нечётным индексам элементы : " << endl ;

EvenIndex = OddIndex = 0 ;

// Перезаписываем исходный массив отсортированными элементами

for (int i = 0; i < sizear; ++i) {

if (i % 2 == 0) {

arr[i] = EvenArr[EvenIndex++];

} else {

arr[i] = OddArr[OddIndex++];

}

}

}

// функция для вычисления суммы до минимального элемента

int SumPerMinimalElem(const int\* arr, int sizear)

{

int sum = 0 ;

// Изначально минимальный элемент - самый первый

int minimal\_elem = arr[0];

// Индекс минимального элемента

int min\_elem\_ind = 0 ;

for (int index = 0 ; index < sizear; index++)

{

// Если элемент меньше того, что ранее был минимальным - перезаписываем

if (arr[index] <= minimal\_elem)

{

minimal\_elem = arr[index] ;

min\_elem\_ind = index ;

}

}

wcout << L"\nМинимальный элемент массива : " << minimal\_elem << endl ;

wcout << L"\nИндекс минимального элемента : " << min\_elem\_ind << endl ;

// Вычисляем сумму до минимального элемента

for (int index = 0 ; index < min\_elem\_ind; index++)

{

sum+= arr[index] ;

}

return sum ;

}

// Функция для вычисления произведения положительных элементов массива

int ProductOfPositive(const int\* arr, int sizear)

{

// Изначальное значение суммы

int product = 1;

// Логическая переменная для проверки наличия хотя бы одного положительного элемента массива

bool PositiveInArray = false;

// Перебираем массив

for (int index; index < sizear; index++)

{

// Если положительное - умножаем

if (arr[index] > 0)

{

product \*= arr[index] ;

PositiveInArray = true;

}

}

// Если положительных элементов нет - возвращаем 0

if (!PositiveInArray) {product = 0 ;}

return product ;

}

int main()

{

setlocale(LC\_ALL, "Russian") ;

// Размер массива

int array\_of\_digits[ARRAY\_SIZE] ;

// Заполняем массив случайными числами

for (int index = 0; index < ARRAY\_SIZE; index++)

{

array\_of\_digits[index] = rand() % (10+10+1) - 10 ;

}

// Выводим

wcout << L"Содержимое массива : ";

for (int index = 0 ; index < ARRAY\_SIZE ; index++)

{

wcout << array\_of\_digits[index] << " , " ;

}

wcout << endl ;

wcout << L"Произведение положительных элементов массива : " << ProductOfPositive(array\_of\_digits, ARRAY\_SIZE) << endl ;

int sum\_per\_min\_elem = SumPerMinimalElem(array\_of\_digits, ARRAY\_SIZE) ;

wcout << L"Сумма до минимального элемента: " << sum\_per\_min\_elem << endl ;

sortEvenAndOdd(array\_of\_digits, ARRAY\_SIZE) ;

for (int index = 0 ; index < ARRAY\_SIZE; index++)

{

wcout << array\_of\_digits[index] << " , " ;

}

wcout << endl ;

return 0;

}

## 4.2 Руководство пользователя

При запуске программы массив заполнится 15 случайными числами в диапазоне от -10 до 10. После чего будет посчитано произведение всех положительных элементов массива, определён минимальный элемент, его индекс, а также посчитана сумма элементов, предшествующих ему. Также будут отдельно упорядочены элементы с чётными и нечётными индексами. Результат представлен на рисунке 7.

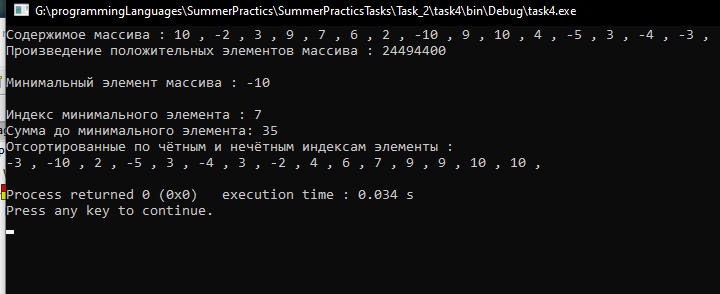


Рисунок 7. Результат работы программы

# 5. Двумерные массивы

Дана целочисленная прямоугольная матрица 5х5. Определить количество отрицательных элементов в тех строках, которые содержат хотя бы один нулевой элемент.

## 5.1 Реализация

#include <iostream>

#include <cstdlib>

#include <iomanip>

using namespace std;

// Число строк и столбцов

const int ROWS\_COUNT = 5 ;

const int COLUMNS\_COUNT = 5 ;

void HowManyMinusElems(int matrix[ROWS\_COUNT][COLUMNS\_COUNT])

{

// Есть ли в матрице 0

bool isZeroInMatrix = false;

// Проходимся по матрице

for (int row\_index = 0 ; row\_index < ROWS\_COUNT ; row\_index++ )

{

// Счётчик отрицательных элементов

int minuselemscounter = 0 ;

// Проходимся в диапазоне количества столбцов

for (int col\_index = 0 ; col\_index < COLUMNS\_COUNT ; col\_index++ )

{

// Если в строке есть 0 - перебираем её и считаем отрицательные элементы

if (matrix[row\_index][col\_index] == 0 )

{

isZeroInMatrix = true;

for (int index\_counter = 0 ; index\_counter < COLUMNS\_COUNT ; index\_counter++)

{

// Если в строке есть отрицательные элементы - считаем их количество

if (matrix[row\_index][index\_counter] < 0)

{

minuselemscounter++ ;

}

}

wcout << L"Количество отрицательных элементов в строке " << row\_index+1 << " : " << minuselemscounter << endl ;

}

}

}

// Если в матрице нет отрицательных элементов

if (!isZeroInMatrix) {wcout << L"В матрице нет отрицательных элементов" << endl ; }

}

// Функция для генерации матрицы

void GenerateMatrix(int matrix[ROWS\_COUNT][COLUMNS\_COUNT])

{

for (int row\_ind = 0 ; row\_ind < ROWS\_COUNT; row\_ind++)

{

for (int col\_ind = 0 ; col\_ind < COLUMNS\_COUNT ; col\_ind++)

{

matrix[row\_ind][col\_ind] = rand() % 21 - 10 ;

}

}

}

// Функция для вывода матрицы на экран

void PrintMatrix(int matrix[ROWS\_COUNT][COLUMNS\_COUNT])

{

for (int row\_index = 0 ; row\_index < ROWS\_COUNT; row\_index++)

{

for (int col\_index = 0 ; col\_index < COLUMNS\_COUNT ; col\_index++ )

{

wcout << setw(5) << matrix[row\_index][col\_index] << "\t" ;

}

wcout << endl ;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int matrix[ROWS\_COUNT][COLUMNS\_COUNT] ;

GenerateMatrix(matrix) ;

// Принудительно зануляем некоторые элементы

matrix[0][3] = 0 ;

matrix[2][4] = 0 ;

matrix[4][4] = 0 ;

wcout << L"Исходная матрица : " << endl ;

PrintMatrix(matrix) ;

// Вызываем функцию подсчёта отрицательных элементов

HowManyMinusElems(matrix) ;

return 0;

}

## 5.2 Руководство пользователя

При запуске программы будет создан двумерный массив 5х5, содержащий в себе элементы в диапазоне от -10 до 10. После будет произведён подсчёт количества отрицательных элементов в строках, содержащий нулевые элементы. Результат представлен на рисунке 8.

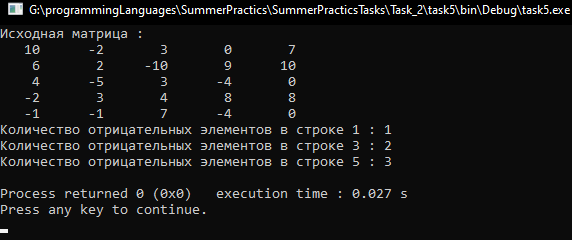


Рисунок 8. Подсчёт отрицательных элементов

# 6. Динамически распределяемая память

С одномерным динамическим массивом, состоящим из n элементов, произвести следующие операции:

1) найти количество отрицательных элементов массива;

2) вычислить сумму модулей элементов массива, расположенных после минимального по модулю элемента. при наличии нескольких минимальных по модулю элементов (равных друг другу) выбирается первый из них;

3) заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

## 6.1 Реализация

#include <iostream>

#include <cstdlib>

using namespace std;

// Размер массива (число букв имени и фамилии)

const int ARRAY\_SIZE = 17 ;

// Функция для сортировки массива методом пузырька

void sortArray(int\* arr, int sizear) {

// Перебираем все элементы массива

for (int i = 0; i < sizear; ++i) {

// Так как в первую итерацию один из элементов "всплывёт",

// то диапазон на 1 меньше

for (int j = 0; j < sizear-1; ++j) {

// Если предшествующий элемент меньше - двигаем вперёд

if (arr[j] > arr[j+1]) {

// Временное значение предыдущего элемента

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

// Функция для вычисления модуля

int abs\_func(int digit)

{

// Модуль - расстояние, поэтому возвращаем обратное по знаку значение

if (digit < 0) {return -digit;}

return digit ;

}

// Функция для вывода содержимого массива на экран

void PrintArray(int \*arr)

{

for (int index = 0; index < ARRAY\_SIZE ; index++)

{

wcout << arr[index] << " , " ;

}

wcout << endl ;

}

// Функция для замены всех отрицательных элементов их квадратами

// И последующей сортировки получившегося массива

void ReplaceAndSort(int \*arr)

{

for (int index = 0; index < ARRAY\_SIZE ; index++)

{

// Если элемент отрицательный - заменяем значение квадратом

if (arr[index] < 0) {arr[index] \*= arr[index] ;}

}

wcout << L"Массив с квадратами отрицательных чисел" << endl ;

PrintArray(arr) ;

// Сортируем массив

sortArray(arr, ARRAY\_SIZE) ;

wcout << L"Отсортированный массив : " << endl ;

PrintArray(arr) ;

}

// Функция для вычисления суммы элементов до минимального элемента

void SumBeforeMinElem(int arr[ARRAY\_SIZE])

{

// Сумма

int sum = 0 ;

// Изначальный индекс минимального элемента

int min\_elem\_index = 0;

// Изначальное значение минимального элемента

int min\_elem = abs\_func(arr[0]) ;

// Проходимся по массиву

for (int index =0 ; index < ARRAY\_SIZE; index++)

{

// Если модуль числа меньше минимального элемента - переприсваеваем значение и индекс

if (abs\_func(arr[index]) < min\_elem & abs\_func(arr[index]) != min\_elem)

{

min\_elem = abs\_func(arr[index]) ;

min\_elem\_index = index ;

}

}

wcout << L"Минимальный элемент: " << min\_elem << endl << L"Индекс минимального элемента " << min\_elem\_index << endl ;

// Вычисляем сумму до минимального элемента

for (int index = 0 ; index < min\_elem\_index ; index++) {sum+=abs\_func(arr[index]);}

wcout << L"Сумма до минимального элемента : " << sum << endl ;

}

// Функция для вычисленния количества отрицательных элементов

int HowManyMinusElems (int arr[ARRAY\_SIZE])

{

// Счётчик

int minuselemscounter = 0 ;

for (int index = 0 ; index < ARRAY\_SIZE ; index++)

{

if (arr[index] < 0) {minuselemscounter++;}

}

return minuselemscounter ;

}

int main()

{

setlocale(LC\_ALL, "Russian") ;

// Создаём динамический массив

int \*array\_of\_digits = new int[ARRAY\_SIZE] ;

// Заполняем массив случайными числами

for (int index = 0 ; index < ARRAY\_SIZE; index++)

{

array\_of\_digits[index] = rand() % 21 - 10 ;

}

wcout << L"Изначальное содержимое массива" << endl ;

PrintArray(array\_of\_digits) ;

wcout << L"Количество отрицательных элементов : " << HowManyMinusElems(array\_of\_digits) << endl ;

SumBeforeMinElem(array\_of\_digits) ;

ReplaceAndSort(array\_of\_digits) ;

// Очищаем память, удаляя массив

delete array\_of\_digits ;

return 0;

}

## 6.2 Руководство пользователя

При запуске программы будет создан динамический массив размерностью 17 (сумма букв имени и фамилии) и заполнен случайными числами в диапазоне от -10 до 10. После будет найдено количество отрицательных элементов массива, вычислена сумма модулей элементов массива, стоящих до первого минимального по модулю элемента. Также все отрицательные элементы будут заменены квадратами их значений, а сам полученный вследствие проделанных действий массив отсортирован. Результат представлен на рисунке 9.

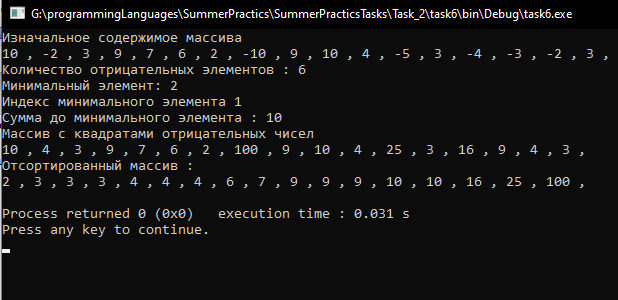


Рисунок 9. Результат работы с динамическим массивом