МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

КАФЕДРА ЭЛЕКТРОННЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

**ПРИЛОЖЕНИЕ К ОТЧЕТУ**

**ОБ УЧЕБНОЙ ПРАКТИКЕ**

**студента 145 учебной группы**

Мелёхина Дениса Владимировича

*(Ф.И.О. студента)*

Направление подготовки:09.03.01 «Информатика и вычислительная техника»

Основная образовательная программа: Электронно-вычислительные машины

Трудоемкость практики – зачетных единицы, часов

Сроки проведения практики: « » 2022 г. по « » 2022 г.

Место прохождения практики Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

*(официальное название)*

Руководитель практики

Панина Ирина Сергеевна, ассистент кафедры ЭВМ ФГБОУ ВО РГРТУ *(фамилия, имя, отчество (полностью), должность, степень, звание)*

Оглавление

[Введение 3](#_Toc122906469)

[1. Общее задание 5](#_Toc122906470)

[1.1 Анализ программных средств и среды разработки 5](#_Toc122906471)

[1.2 Обзор существующих языков программирования для разработки 6](#_Toc122906472)

[1.3 Анализ регламентирующих документов 7](#_Toc122906473)

[2. Индивидуальное задание 8](#_Toc122906474)

[2.1 Программирование линейных программ на языке Паскаль 8](#_Toc122906475)

[2.2 Программирование линейных программ на С++ 11](#_Toc122906476)

[2.3 Разветвляющиеся вычислительные процессы 14](#_Toc122906477)

[2.4 Табулирование функции 21](#_Toc122906478)

[2.5 Одномерные массивы 27](#_Toc122906479)

[2.6 Структуры 41](#_Toc122906480)

[2.7 Файлы 46](#_Toc122906481)

[2.8 Разработка простейшего CLR-приложения 52](#_Toc122906482)

[Заключение 55](#_Toc122906483)

[Библиографический список 56](#_Toc122906484)

# Введение

Целью освоения практики получение первичных профессиональных умений и навыков посредством обеспечения этапов формирования компетенций, предусмотренных ФГОС ВО.

Для достижения поставленной цели будут решены следующие задачи:

* анализ программных средств и среды разработки;
* обзор существующих языков для разработки;
* основная концепция и содержание;
* анализ регламентирующих документов.

Учебная практика проходила в институте РГРТУ. В ходе выполнения учебной практики изучался язык программирования С++, на котором были написаны почти все программы. Для разработки кода была выбрана среда программирования Visual Studio 2022.

Язык программирования С++ был выбран из-за ряда следующих достоинств:

* Производительность. Благодаря своей вычислительной мощности язык обеспечивает высокую скорость исполнения кода.
* Отсутствие существенной нагрузки. Язык не утяжеляет программы, позволяет использовать их даже на старых устройствах.
* Универсальность. Возможно, этот главный плюс C++. Он кроссплатформенный и подходит для любых целей (не является узкоспециализированным, как другие языки).
* Популярность. Это традиционный язык для разработки ПО и различного софта. Компиляторы с ним есть на любой ОС. А программы, написанные на C++, обычно легко переносятся с одной платформы на другую.
* Хорошая основа для изучения других языков программирования. Зная C++, вам будет проще освоить другие ЯП. Хотя у него сложный синтаксис, язык является эталонным, на примере которого разработаны более современные ЯП (С#, JavaScript, Java), имеющие более простую структуру.
* Постоянные обновления. Регулярно появляются новые версии стандарта, с доработанными и улучшенными элементами. Свежая версия языка – С++20, которая появилась в декабре 2020 года.

Есть у С++ и недостатки:

* Сложный синтаксис. Быстро обучиться языку не получится, нужен опыт и знание теории. Язык существует с 1983 года и постоянно обновляется (новая версия выходит каждые три года). Появляются новые возможности, шаблоны, а конструкции становятся сложнее, что соответственно усложняет и обучение. А чтобы язык действительно обеспечил производительность программ, требуется его знание на высоком уровне.
* Важность подготовки правильного кода. Ошибки при написании коды приводят к неполадкам при работе программ. К тому же ошибки в готовом коде довольно сложно определять, поэтому проверять нужно при написании.
* Сложность написания кода. Дело не только в синтаксисе. При работе с языком нужно выполнять мониторинг типов данных, контроль выделения и освобождения памяти и т. д.

Несмотря на все минусы язык программирования C++ является популярным и интересным языком программирования, в который поместили очень много разных функций, на этом языке написано множество современных программ, поэтому он и был выбран для программирования заданных задач.

# Общее задание

Необходимо разработать программы по следующим темам: программирование линейных программ, разветвляющиеся вычислительные процессы, табулирование функций, одномерные массивы, структуры, файлы, а также разработка простейшего CLR-приложения. К каждой программе следует приложить блок-схемы алгоритмов, оформленных по ГОСТ 19.701-90 ЕСПД.

## Анализ программных средств и среды разработки

В данной практике для языка программирования С++ была выбрана среда разработки Microsoft Visual Studio 22. Visual Studio 22 – интегрированная среда разработки C++, которая позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms. Microsoft Visual Studio 22 является удобной IDE для разработки как простого ПО, так и сложных программных систем. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Она также подходит для создания веб-сайтов, веб-приложений и веб-служб для всех поддерживаемых платформ: Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone, .NET Compact Framework и Silverlight.

Любое программное средство можно подразделить на частные последовательные процессы, которые, дополняют друг друга, совершаются непрерывно, последовательно один за другим во времени и пространстве и образуют единое целое. Минимальные технические требования для оптимальной работы Microsoft Visual Studio 22 перечислены ниже.

* 64-разрядный процессор 1,8 ГГц или более мощный. Рекомендуется четырехъядерный или с большим количеством ядер. Процессоры ARM не поддерживаются.
* Не менее 4 ГБ ОЗУ. На используемые ресурсы влияет множество факторов; для стандартных профессиональных решений рекомендуется 16 ГБ ОЗУ.
* Windows 365: не менее 2 виртуальных ЦП и 8 ГБ ОЗУ. Рекомендуется 4 виртуальных ЦП и 16 ГБ ОЗУ.
* Место на жестком диске: от 850 ГБ до 210 ГБ свободного места в зависимости от установленных компонентов, обычно для установки требуется от 20 до 50 ГБ свободного места. Для повышения производительности рекомендуется устанавливать Windows и Visual Studio на твердотельном накопителе (SSD).
* Видеоадаптер с минимальным разрешением WXGA (1366 на 768 пикселей); для оптимальной работы Visual Studio рекомендуется разрешение 1920 на 1080 пикселей или выше.

## Обзор существующих языков программирования для разработки

В настоящее время существует много разный языков программирования, их можно разделить на две группы по зависимости от аппаратных средств: языки низкого уровня и языки высокого уровня. Быстрый ответ. Низкоуровневое программирование применяется в основном для создания компактного программного обеспечения, например, для программирования систем реального времени (когда недопустимы задержки в работе), встраиваемых систем (микроконтроллеров), а также драйверов, управляющих внешними устройствами. Из самых популярных низкоуровневых языков программирования можно выделить следующие: Assembler, С, FASM, NASM, MASM. Высокоуровневый язык программирования – средство записи компьютерных программ, обеспечивающее высокую скорость и удобство работы. Его отличительной чертой является абстракция. Другими словами, высокоуровневый язык программирования обеспечивает возможность введения смысловых конструкций, способных коротко описать форматы данных и операции с ними в тех случаях, когда описания на низкоуровневом языке (например, на машинном коде) будут сложными для восприятия и очень длинными. Из самых популярных высокоуровневых языков программирования можно выделить следующие: Python, JavaScript, Visual Basic, Delphi, Perl, PHP, ECMAScript, Ruby, C #, Java. Язык программирования C++ не был включён ни в один из этих списков, так как он сочетает в себе свойства как высокоуровневых, так и низкоуровневых языков программирования. Этот язык программирования и был выбран для разработки поставленных задач.

## Анализ регламентирующих документов

Данное приложение к отчету об учебной практике, задание и отчёт были оформлены по ГОСТ 7.32-2017. Настоящий стандарт устанавливает общие требования к структуре и правилам оформления отчетов различных направлений, а также для тех случаев, когда единая процедура оформления будет содействовать обмену информацией, совершенствуя обработку отчета в информационной системе. Блок-схемы алгоритмов разработаны в соответствии с ГОСТ 19.701-90 ЕСПД. Данный стандарт определяет правила обозначения графических символов, размерную характеристику, правила применения и расположения в схеме.

## Индивидуальное задание

## Программирование линейных программ на языке Паскаль

**Вариант 3**

Для заданного по варианту выражения (таблица 1):

1. Разработать алгоритм вычисления выражения.
2. Представить алгоритм в виде БСА.
3. Подготовить программную реализацию алгоритма.
4. Выполнить отладку и экспериментальную проверку программы на 3-4 контрольных примерах.

Таблица 1 ‒ Задание

|  |  |
| --- | --- |
| Вариант | Задание |
| 3 | https://lh5.googleusercontent.com/O8lk06a89wyI3f7aeWOh_Xj_2k0jJvnt2F3omUpBhnhwgzJEEK7o4xiGfe2UVoQ5dO9vq-UFuwxvEprg1ONCxvcg5u08t10MgjnFAnuzRwIK4DMWvXFsYjt9ESTquDA_ZRiV_CDffZr86AS9rT8u4Ajpv9ScF4pJuqxRzcI-UFeUJcCI3Mhi6q2LilB1SxO1fNWHGbDDlw |

**Код программы:**

**Program** zadanie1;

**var** p, x, y, z, min, max: real;

**begin**

writeln('Введите x');

readln(x);

writeln('Введите y');

readln(y);

writeln('Введите z');

readln(z);

min:=x;

**if** y<x **then**

min:=y;

writeln('min (x, y) = ',min);

max:=y;

**if** z>y **then**

max:=z;

writeln('max (y, z) = ',max);

p:=(abs(min-max))/2;

writeln('p = ',p);

**end**.

**Блок-схема алгоритма изображена на рисунке 1:**

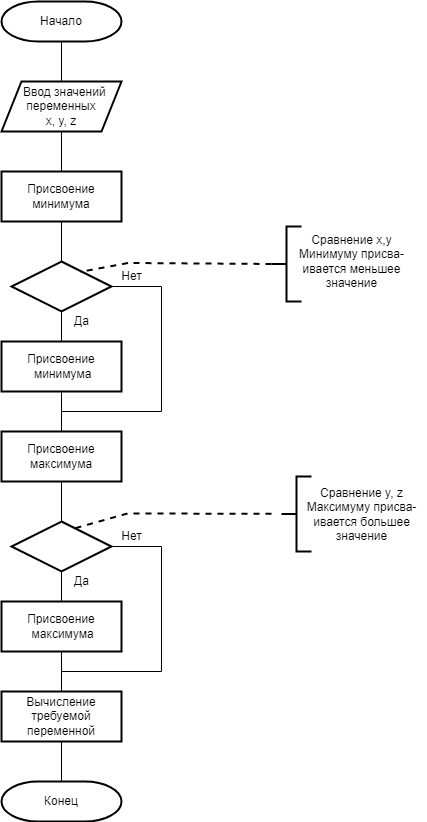


Рисунок 1 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

В начале программы производится ввод переменных x, y, z. Следующим шагом за минимум берется переменная x, а затем происходит сравнение переменных x, y; минимуму присваивается значение меньшей переменной. Далее максимум становится равным y, после чего сравниваются y, z; максимуму присваивается значение большей переменной. Подставляем полученные переменные в выражение, вычисляем и выводим ответ на экран.

**Окно выполнения программы (рисунок 2):**

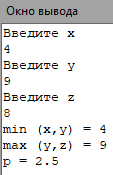


Рисунок 2 ‒ Окно выполнения

## Программирование линейных программ на С++

Цель работы: получение навыков программирования линейных вычислительных процессов.

**Вариант 3**

**Часть 1**

В квадрат вписана окружность (рис.3). Определить площадь заштрихованной части фигуры, если известна сторона квадрата.

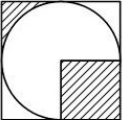


Рисунок 3 ‒ Окружность

**Код программы:**

#include <iostream>

#include <stdio.h>

int main() {

setlocale(LC\_ALL, "Russian");

float x;

float s;

puts("Введите сторону квадрата");

printf("x = ");

scanf\_s("%f", &x);

s = ((x\*x) - (0.5\*x\*0.5\*x) - (3.14\*0.5\*x\*0.5\*x\*3/4)) / 3;

printf("Площадь фигуры равна: %4.3f", s);

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 4:**



Рисунок 4 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

В программу вводятся данные о длине стороны квадрата, после чего по формуле производится расчёт площади заштрихованной части фигуры и вывод полученного значения на экран.

**Окно выполнения программы (рисунок 5):**



Рисунок 5 ‒ Окно выполнения

**Часть 2**

Напишите программу для расчета по двум формулам (таблица 2). Предварительно подготовьте тестовые примеры для второй формулы с помощью калькулятора (результаты вычисления по обеим формулам должны совпадать).

Таблица 2 ‒ Задание

|  |  |  |
| --- | --- | --- |
| № Варианта | Формула 1 | Формула 2 |
| 3 |  |  |

**Код программы:**

#include <iostream>

#include <stdio.h>

int main() {

setlocale(LC\_ALL, "Russian");

float z1;

float z2;

float a;

float b;

puts("Введите угол a");

printf("a = ");

scanf\_s("%f", &a);

puts("Введите угол b");

printf("b = ");

scanf\_s("%f", &b);

z1 = (sin(2 \* a) + sin(5 \* a) - sin(3 \* a)) / (cos(a) + 1 - 2 \* (sin(2 \* a)\*sin(2 \* a)));

z2 = pow(2\*sin(a), 2);

printf("z1 = %f ", z1);

printf("z2 = %f ", z2);

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 6:**



Рисунок 6 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

В программу вводятся данные о величине углов a и b. Далее производится расчёт значений z1 и z2 по двум формулам, после чего следует вывод результата на экран.

**Окно выполнения программы (рисунок 7):**

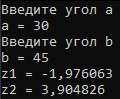


Рисунок 7 ‒ Окно выполнения

## Разветвляющиеся вычислительные процессы

**Вариант 3**

**Часть 1**

Разработать алгоритм и программу с использованием оператора выбора switch.

Вывести на экран следующую информацию:

ОВОЩИ И ФРУКТЫ

1. Яблоко 2. Груша 3. Огурец 4. Дыня

5. Томат 6. Картофель 7. Манго 8. Лук

Предоставьте пользователю возможность выбора элемента и определите, какой элемент выбрал пользователь: овощ или фрукт.

**Код программы:**

#define \_CRT\_NO\_WARNINGS\_

#include <iostream>

#include <stdio.h>

int main() {

setlocale(LC\_ALL, "Russian");

int a;

puts("Овощи и фрукты");

*std*::cout << "1. Яблоко 2. Груша 3. Огурец 4. Дыня" << *std*::endl;

*std*::cout<<"5. Томат 6. Картофель 7. Манго 8. Лук"<<*std*::endl;

scanf\_s("%d", &a);

switch (a) {

case 1: printf("Яблоко (фрукт)"); break;

case 2: printf("Груша (фрукт)"); break;

case 3: printf("Огурец (овощ)"); break;

case 4: printf("Дыня (фрукт)"); break;

case 5: printf("Томат (овощ)"); break;

case 6: printf("Картофель (овощ)"); break;

case 7: printf("Манго (фрукт)"); break;

case 8: printf("Лук (овощ)"); break;

default: printf("Ошибка");

}

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 8:**

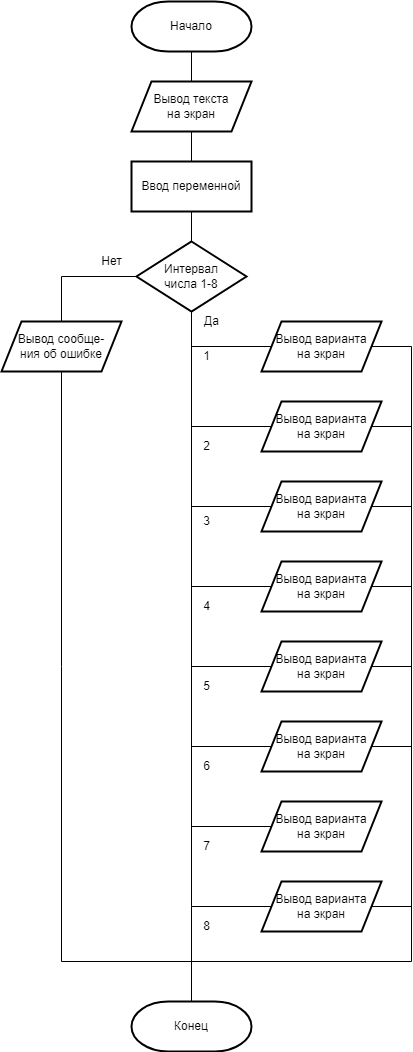


Рисунок 8 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Вначале программа выводит на экран информацию об овощах и фруктах. Пользователь вводит требуемое значение, после чего оператор case выводит на экран соответствующую информацию.

**Окно выполнения программы (рисунок 9):**

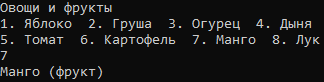


Рисунок 9 ‒ Окно выполнения

**Часть 2**

Разработать алгоритм и программу вычисления значения функции по заданному графику. Каждый вариант использует свой метод записи команд ветвления:

|  |  |
| --- | --- |
| **Номер**  **варианта** | **Метод** |
| 1-4 | 1) метод, когда проверяются оба признака наступления ситуации;  2) метод последовательного выделения ситуаций, начиная с одного из концов цепочки ситуаций |

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика (рисунок 10).

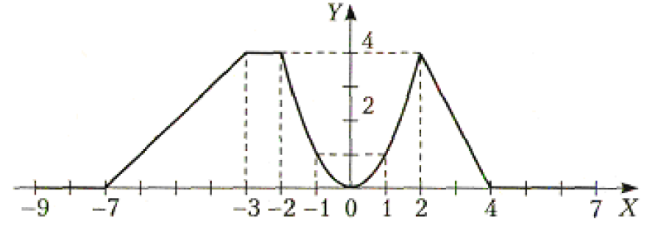


Рисунок 10 ‒ График

1. Метод, когда проверяются оба признака наступления ситуации.

**Код программы:**

#define \_CRT\_NO\_WARNINGS\_

#include <iostream>

#include <stdio.h>

int main() {

setlocale(LC\_ALL, "Russian");

float x;

float y1, y2, y3, y4, y5;

std::cout << "Введите значение аргумента от -9 до 7" << std::endl;

scanf\_s("%f", &x);

y1 = x + 7;

y2 = 4;

y3 = pow(x, 2);

y4 = -2 \* x + 8;

y5 = 0;

if ((x > -9) && (x < 7)) {

if ((x >= -9) && (x <= -7))

printf("y = %3.2f", y5);

if ((x > -7) && (x <= -3))

printf("y = %3.2f", y1);

if ((x > -3) && (x <= -2))

printf("y = %3.2f", y2);

if ((x > -2) && (x <= 2))

printf("y = %3.2f", y3);

if ((x > 2) && (x <= 4))

printf("y = %3.2f", y4);

if ((x > 4) && (x <= 7))

printf("y = %3.2f", y5);

}

else printf("Ошибка");

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 11:**

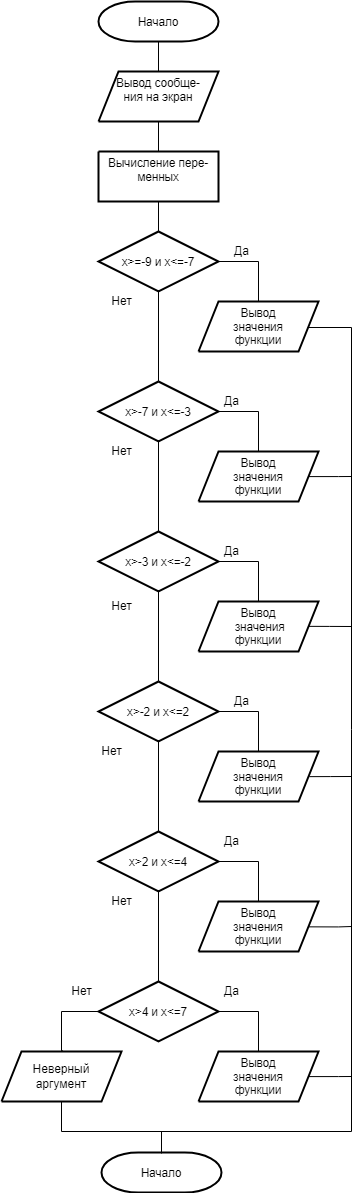
****

Рисунок 11 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит значение аргумента в интервале [-9; 7], после чего программа попарно анализирует дальнейшее вычисление функции по определённым формулам, в зависимости от введённых данных, выводя на экран полученное значение.

**Окно выполнения программы (рисунок 12):**



Рисунок 12 ‒ Окно выполнения

1. Метод последовательного выделения ситуаций, начиная с одного из концов цепочки ситуаций.

**Код программы:**

#define \_CRT\_NO\_WARNINGS\_

#include <iostream>

#include <stdio.h>

int main() {

setlocale(LC\_ALL, "Russian");

float x;

float y1, y2, y3, y4, y5;

*std*::cout << "Введите значение аргумента от -9 до 7" << *std*::endl;

scanf\_s("%f", &x);

y1 = x + 7;

y2 = 4;

y3 = pow(x, 2);

y4 = -2 \* x + 8;

y5 = 0;

if ((x > -9) && (x < 7)) {

if ((x >= -9) && (x <= -7))

printf("y = %3.2f", y5); else

if ((x > -7) && (x <= -3))

printf("y = %3.2f", y1); else

if ((x > -3) && (x <= -2))

printf("y = %3.2f", y2); else

if ((x > -2) && (x <= 2))

printf("y = %3.2f", y3); else

if ((x > 2) && (x <= 4))

printf("y = %3.2f", y4); else

if ((x > 4) && (x <= 7))

printf("y = %3.2f", y5);

}

else printf("Ошибка");

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 13:**

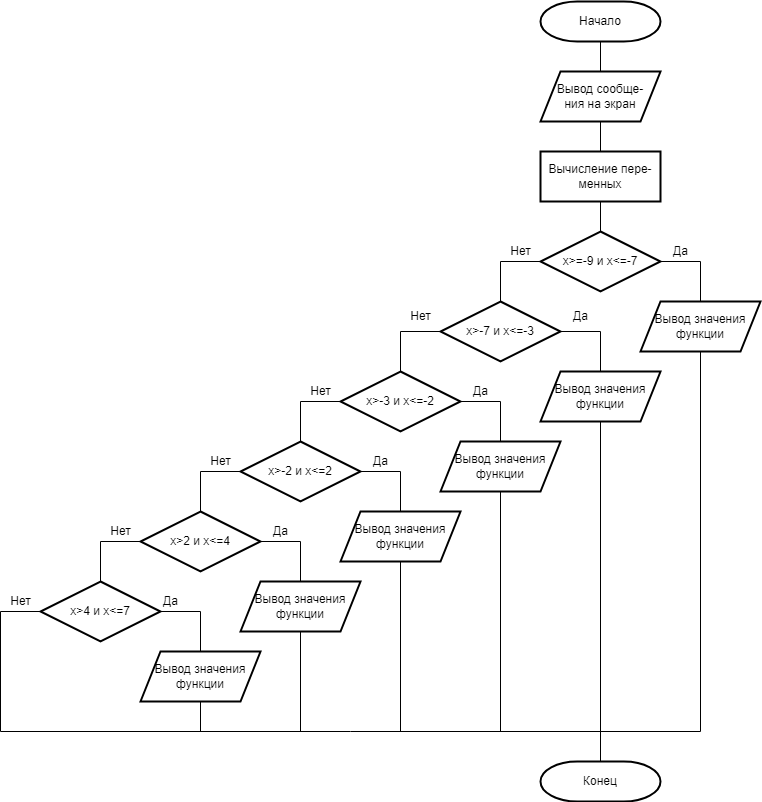


Рисунок 13 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит значение аргумента в интервале [-9; 7], после чего программа последовательно анализирует дальнейшее вычисление функции по определённым формулам, в зависимости от введённых данных, выводя на экран полученное значение.

**Окно выполнения программы (рисунок 14):**



Рисунок 14 ‒ Окно выполнения

## Табулирование функции

**Вариант 3**

Вычислить и вывести на экран значение функции, заданной графически (см. лабораторную работу № 3 часть 2), на интервале от ***Хнач*** до ***Хкон*** шагом ***h***. Таблицу снабдить заголовком и шапкой.

Разработку алгоритма построения таблицы выполнить с помощью цикла с известным числом повторений **и** с помощью цикла с заранее неизвестным числом повторений.

Используемый метод указан в таблице 3.

Таблица 3 ‒ Методы

|  |  |
| --- | --- |
| **Номер**  **варианта** | **Метод** |
| 1-4 | Метод половинного деления цепочки ситуаций |

1. Программа, содержащая цикл с известным числом повторений

**Код программы:**

#define \_CRT\_NO\_WARNINGS\_

#include <iostream>

#include <math.h>

#include <stdio.h>

int main() {

setlocale(LC\_ALL, "Russian");

float x, x1, xn, hx;

float y1, y2, y3, y4, y5;

for (x = -9; x <= 7; x++) {

*std*::cout << "\t------------------------------------\n";

*std*::cout << "\t| x = " << x << "\t|";

if (x < -2)

if (x <= -3)

if (x <= -7) {

*std*::cout << " y = " << 0 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << x+7 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << 4 << "\t |" << *std*::endl;

}

else if (x >= -2)

if (x >= 2)

if (x >= 4) {

*std*::cout << " y = " << 0 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << -2 \* x + 8 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << pow(x,2) << "\t |" << *std*::endl;

}

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 15:**

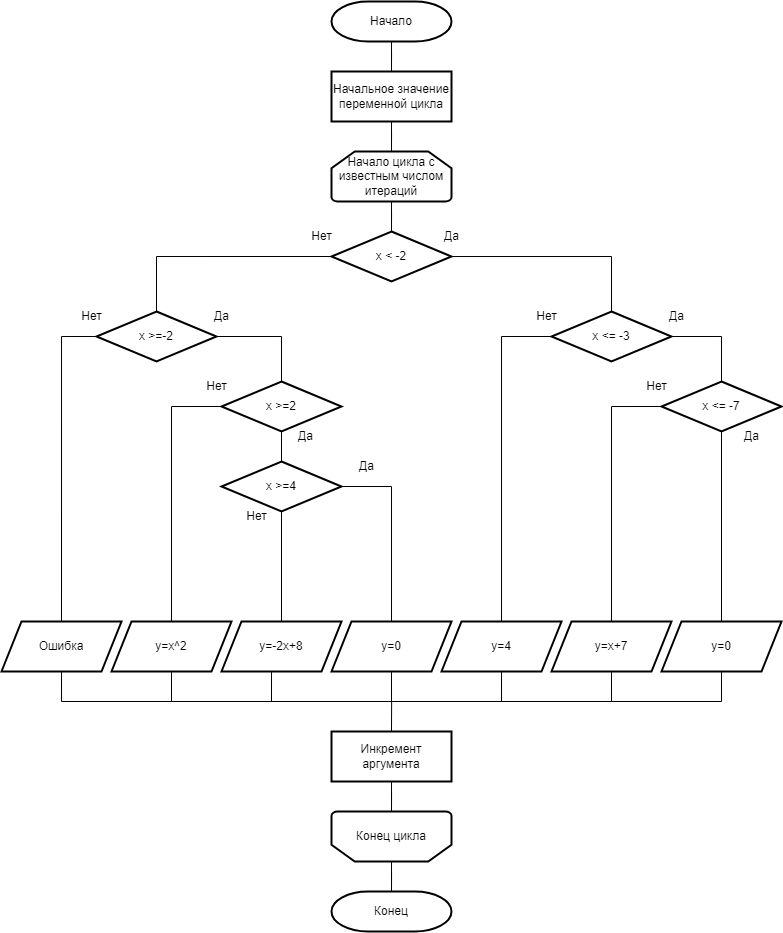


Рисунок 15 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит значение аргумента в интервале [-9; 7], после чего программа, содержащая цикл с известным числом итераций (шаг равен 1), реализует метод половинного деления цепочки ситуаций. Осуществляется вычисление функции по определённым формулам и вывод на экран таблицы полученных значений.

**Окно выполнения программы (рисунок 16):**

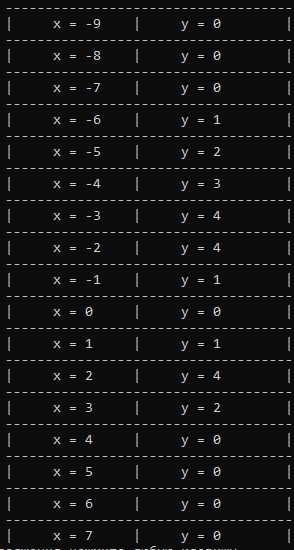


Рисунок 16 ‒ Окно выполнения

1. Программа, содержащая цикл с заранее неизвестным числом повторений.

**Код программы:**

#define \_CRT\_NO\_WARNINGS\_

#include <iostream>

#include <math.h>

#include <stdio.h>

int main() {

setlocale(LC\_ALL, "Russian");

float x, x1, xn, hx;

float y1, y2, y3, y4, y5;

*std*::cout << "Введите начальное значение аргумента от -9 до 7" << *std*::endl;

scanf\_s("%f", &x);

if (x < -9) {

*std*::cout << "Неверный аргумент функции" << *std*::endl;

system("pause");

return 0;

}

if (x > 7) {

*std*::cout << "Неверный аргумент функции" << *std*::endl;

system("pause");

return 0;

}

*std*::cout << "Введите конечное значение аргумента от -9 до 7" << *std*::endl;

scanf\_s("%f", &xn);

if (xn < -9) {

*std*::cout << "Неверный аргумент функции" << *std*::endl;

system("pause");

return 0;

}

if (xn > 7) {

*std*::cout << "Неверный аргумент функции" << *std*::endl;

system("pause");

return 0;

}

*std*::cout << "Введите шаг приращения функции" << *std*::endl;

scanf\_s("%f", &hx);

*std*::cout << "\t------------------------------------\n";

*std*::cout << "\t| X | Y(X) |\n";

if (x < -9 && x > 7) {

*std*::cout << "Неверный аргумент функции" << *std*::endl;

system("pause");

return 0;

}

while (x < xn + hx / 2) {

*std*::cout << "\t------------------------------------\n";

*std*::cout << "\t| x = " << x << "\t|";

if (x < -2)

if (x <= -3)

if (x <= -7) {

*std*::cout << " y = " << 0 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << x+7 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << 4 << "\t |" << *std*::endl;

}

else if (x >= -2)

if (x >= 2)

if (x >= 4) {

*std*::cout << " y = " << 0 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << -2 \* x + 8 << "\t |" << *std*::endl;

}

else {

*std*::cout << " y = " << pow(x,2) << "\t |" << *std*::endl;

}

x = x + hx;

}

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 17:**

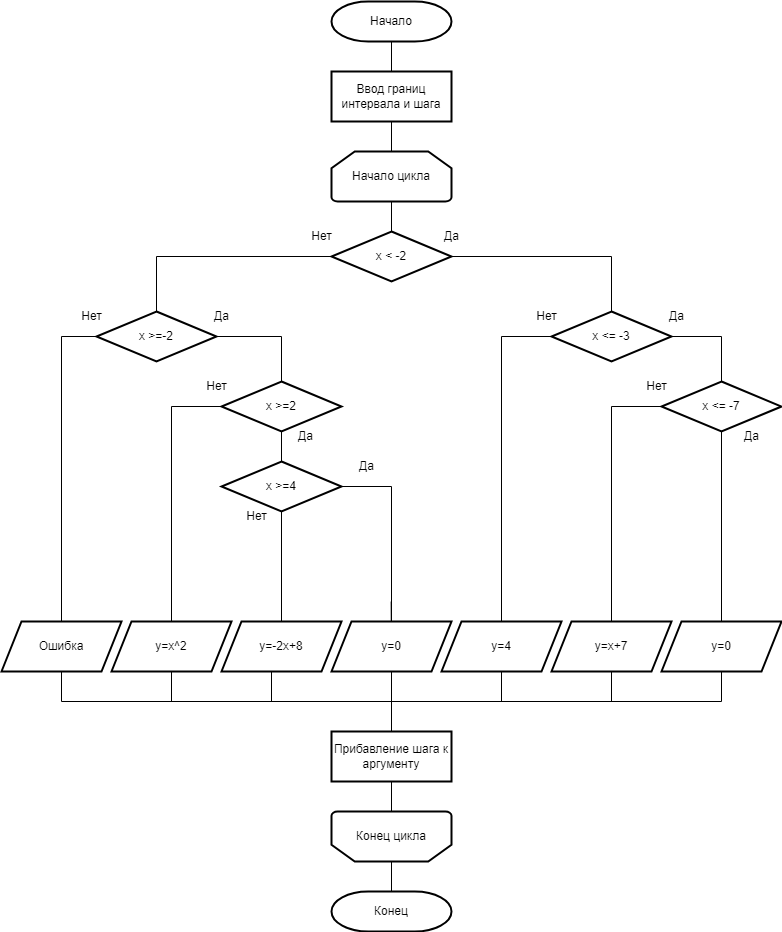
****

Рисунок 17 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит границы интервала в пределах [-9;7] и шаг приращения аргумента. Программа, содержащая цикл с заранее неизвестным числом итераций, реализует метод половинного деления цепочки ситуаций. Осуществляется вычисление функции по определённым формулам и вывод на экран таблицы полученных значений.

**Окно выполнения программы (рисунок 18):**

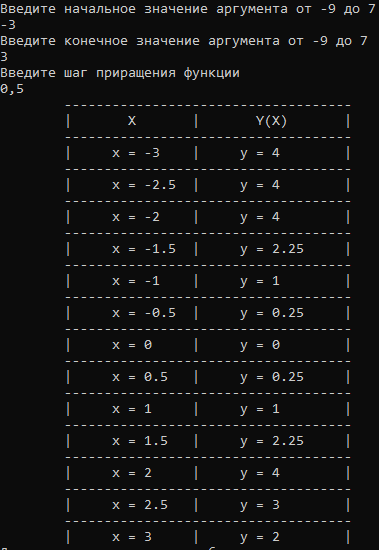


Рисунок 18 ‒ Окно выполнения

## Одномерные массивы

**Вариант 3**

Цель работы: получение навыков обработки одномерных массивов.

В одномерном массиве, состоящем из n целых элементов

1. вычислить сумму элементов массива, расположенных между первым и последним нулевыми элементами;

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <math.h>

int main()

{

setlocale(LC\_CTYPE, "Russian");

int a, n, i, j,k,p;

float q, s;

float\* A = new float[a];

printf("Введите количество элементов массива");

scanf("%d", &n);

for (i = 0; i <= n; i++)

{

A[i] = rand()% 10-5;

}

printf("Исходный массив:\n");

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

{

printf("%2.2f", A[i]);

printf("\n");

}

for(i=0; i<n; i++)

{

if(A[i]==0)

{

k=i;

break;

}

}

for (i=n-1; i>=0; i--)

if (A[i]==0)

{

p=i;

break;

}

for (i=k; i<=p; i++)

{

s=s+A[i];

}

printf("Сумма:\n");

printf("%2.2f", s);

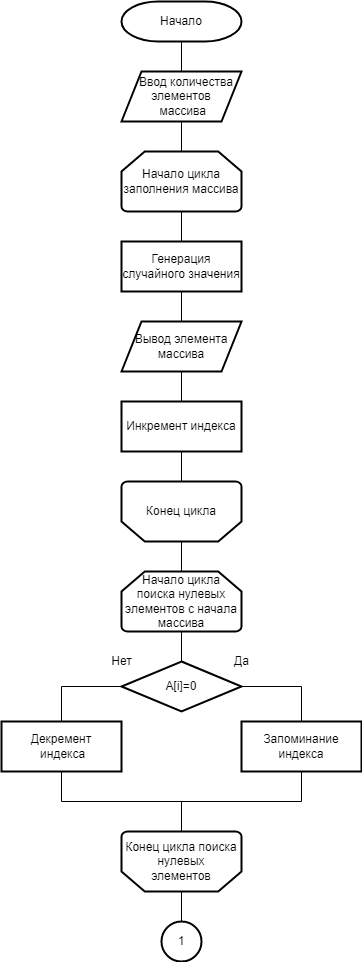
printf("\n");

return 0;

system("pause");

}

**Блок-схема алгоритма изображена на рисунке 19:**



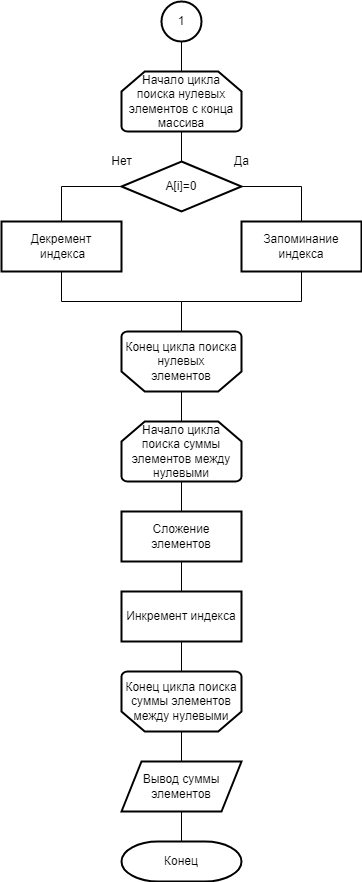


Рисунок 19 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит требуемое количество элементов массива, после чего в цикле производится заполнение массива случайными значениями и вывод на экран. В следующем цикле осуществляется поиск нулевых элементов с начала массива, а далее ‒ с конца массива. В полученном интервале цикл находит сумму всех элементов, после чего выводит её на экран.

**Окно выполнения программы (рисунок 20):**

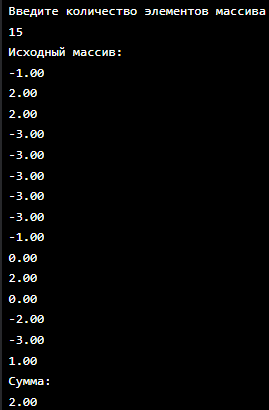


Рисунок 20 ‒ Окно выполнения

1. преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные 0, считать положительными);

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <math.h>

int main()

{

setlocale(LC\_CTYPE, "Russian");

int a, n, i, j;

float q;

float\* A = new float[a];

printf("Введите количество элементов массива");

scanf("%d", &n);

for (i = 0; i <= n; i++)

{

A[i] = rand()% 130-50;

}

printf("Исходный массив:\n");

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

{

printf("%2.2f", A[i]);

printf("\n");

}

j = 1;

while (j != n)

{

for (i = j; i <= n; i++)

{

if (A[j] < 0)

{

q = A[j];

A[j] = A[i];

A[i] = q;

}

}

j += 1;

}

printf("Преобразованный массив:\n");

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

{

printf("%2.2f", A[i]);

printf("\n");

}

system("pause");

}

**Блок-схема алгоритма изображена на рисунке 21:**



Рисунок 21 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит требуемое количество элементов массива, после чего в цикле производится заполнение массива случайными значениями и вывод на экран. В следующем цикле осуществляется перестановка элементов таким образом, чтобы сначала располагались положительные, а затем отрицательные элементы. В крайнем цикле производится печать преобразованного массива.

**Окно выполнения программы (рисунок 22):**

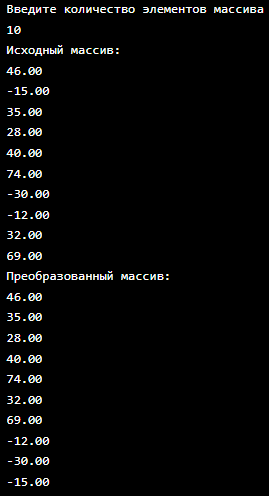


Рисунок 22 ‒ Окно выполнения

1. упорядочить элементы массива по убыванию, используя пузырьковую сортировку;

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <math.h>

int main()

{

setlocale(LC\_CTYPE, "Russian");

int a, n, i, j;

float q;

float\* A = new float[a];

printf("Введите количество элементов массива");

scanf("%d", &n);

for (i = 0; i <= n; i++)

{

A[i] = rand()% 201-100;

}

printf("Исходный массив:\n");

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

{

printf("%2.2f", A[i]);

printf("\n");

}

j = 1;

while (j != n)

{

for (i = j; i <= n; i++)

{

if (A[j] < A[i])

{

q = A[j];

A[j] = A[i];

A[i] = q;

}

}

j += 1;

}

printf("Отсортированный массив:\n");

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

{

printf("%2.2f", A[i]);

printf("\n");

}

system("pause");

}

**Блок-схема алгоритма изображена на рисунке 23:**

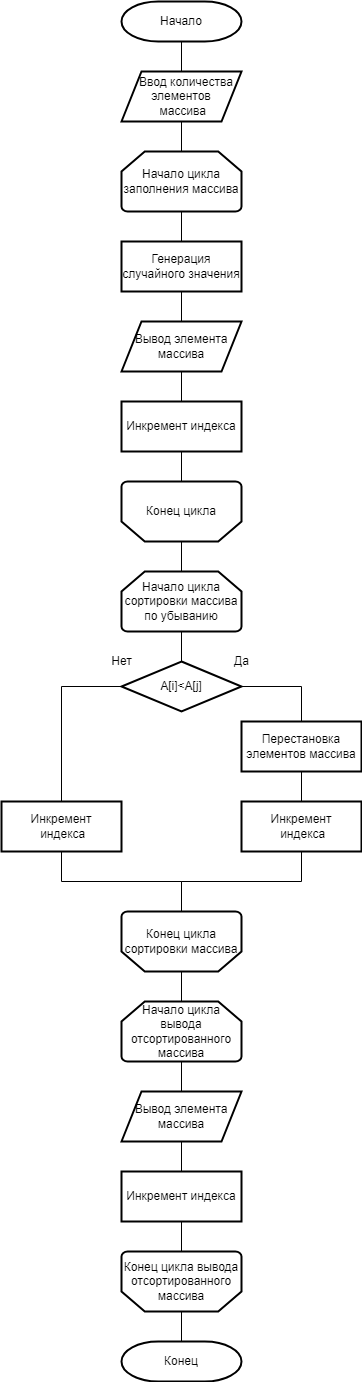
****

Рисунок 23 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит требуемое количество элементов массива, после чего в цикле производится заполнение массива случайными значениями и вывод на экран. В следующем цикле осуществляется пузырьковая сортировка массива по убыванию, идея которой состоит в сравнении соседних элементов и их перестановке. В крайнем цикле производится печать преобразованного массива.

**Окно выполнения программы (рисунок 24):**

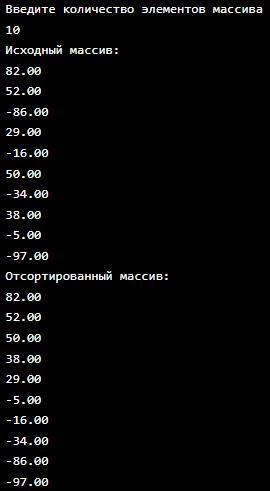


Рисунок 24 ‒ Окно выполнения

1. найти заданный элемент в упорядоченном массиве, используя бинарный метод поиска.

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <math.h>

int main()

{

setlocale(LC\_CTYPE, "Russian");

int a, n, i, j, m, l, r;

float q,g;

float\* A = new float[a];

printf("Введите количество элементов массива");

scanf("%d", &n);

for (i = 0; i <= n; i++)

{

A[i] = rand()% 10-5;

}

printf("Исходный массив:\n");

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

{

printf("%2.2f", A[i]);

printf("\n");

}

j = 1;

while (j != n)

{

for (i = j; i <= n; i++)

{

if (A[j] < A[i])

{

q = A[j];

A[j] = A[i];

A[i] = q;

}

}

j += 1;

}

printf("Отсортированный массив:\n");

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

{

printf("%2.2f", A[i]);

printf("\n");

}

printf("Введите элемент, который требуется найти:");

scanf("%f", &g);

m = 0;

l = 1;

r = n;

while ((l + 1) < r)

{

m = (l + r) / 2;

if (A[m] < g)

r = m;

else l = m;

}

l++;

if (A[l] == g)

printf("Найденный элемент: A[%d] = %2.2f ", l, A[l]);

r--;

if (A[r] == g)

printf("Найденный элемент: A[%d] = %2.2f ", r, A[r]);

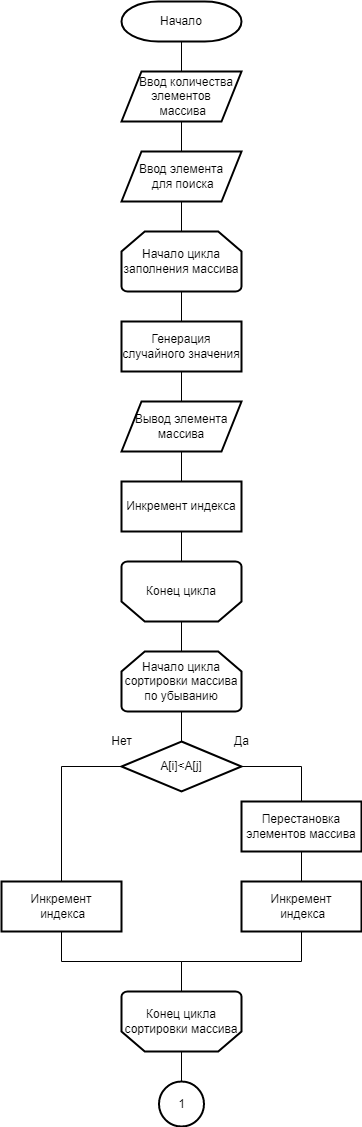
else printf("Элемент отсутствует\n");

return 0;

system("pause");

}

**Блок-схема алгоритма изображена на рисунке 25:**

****

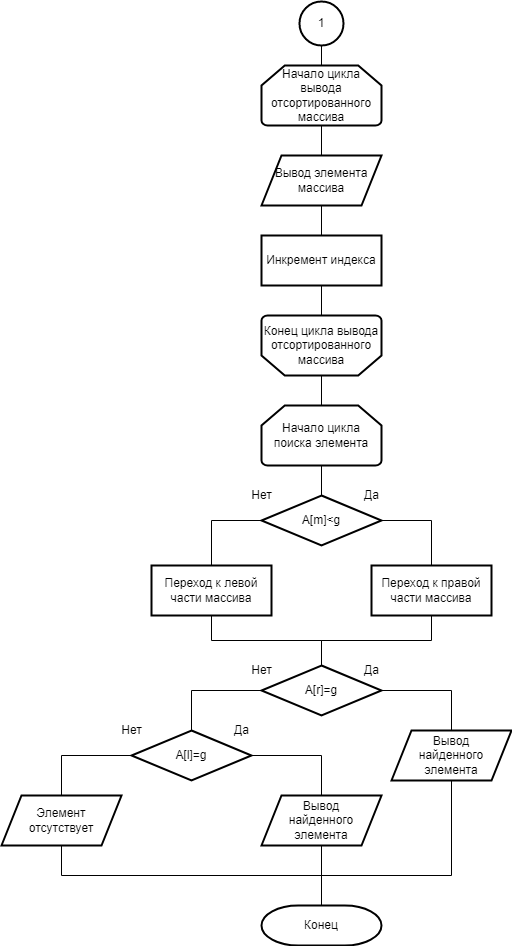
****

Рисунок 25 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Пользователь вводит требуемое количество элементов массива, после чего в цикле производится заполнение массива случайными значениями и вывод на экран. В следующем цикле осуществляется пузырьковая сортировка массива по убыванию, идея которой состоит в сравнении соседних элементов и их перестановке. После этого другой цикл выводит на экран отсортированный массив. Далее пользователь вводит элемент, который необходимо найти. В первом условии сравнивается требуемый элемент и элемент из середины отсортированного массива; если он меньше, то рассматривается левая половина массива и наоборот, соответственно. Далее все элементы левой или правой половины сравниваются с исходным значением; если элемент найден, то производится его печать на экран вместе с номером, иначе ‒ вывод на экран сообщения о том, что элемент отсутствует.

**Окно выполнения программы (рисунок 26):**

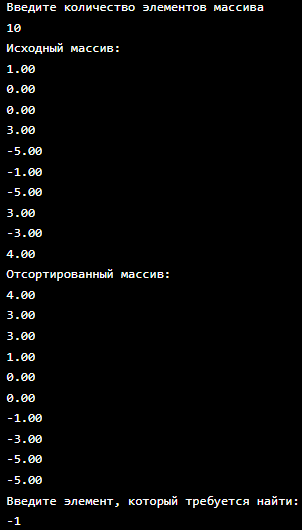


Рисунок 26 ‒ Окно выполнения

## Структуры

**Вариант 3**

Цель работы: приобретение навыков разработки алгоритмов и программ с использованием структур.

Описать структуру с именем WORKER, содержащую следующие поля:

* фамилия и инициалы работника;
* название занимаемой должности;
* год поступления на работу.

Написать программу, выполняющую следующие действия:

* ввод с клавиатуры данных в массив, состоящий из десяти структур типа WORKER;
* вывод таблицы на экран;
* записи упорядочить по алфавиту;
* вывод отсортированной таблицы на экран;
* вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
* если таких работников нет, вывести на дисплей соответствующее сообщение.

**Код программы:**

#include<iostream>

#include<iomanip>

using namespace *std*;

struct WORKER

{

char surname[13];

char initials[14];

char post[15];

int date;

};

int main()

{

setlocale(LC\_CTYPE, "Russian");

int const N = 2;

int staj;

int j;

bool f = true;

WORKER mass[N];

for (int i = 0; i < N; ++i)

{

cout << "Введите информацию о сотруднике №" << i + 1 << ":" << endl;

cout << "1) Фамилия: "; cin >> mass[i].surname;

cout << "2) Инициалы: "; cin >> mass[i].initials;

cout << "3) Должность: "; cin >> mass[i].post;

cout << "4) Год поступления на работу: "; cin >> mass[i].date;

}

cout << "Введите требуемый стаж работы в организации (текущий год 2022): ";

cin >> staj;

WORKER b;

int t = 1;

for (int j = 0; j < N; j++)

{

for (int i = j; i < N; i++) t = mass[i].surname[0] < mass[t].surname[0] ? i : t;

b = mass[j];

mass[j] = mass[t];

mass[t] = b;

}

printf(" |-------------------------------------------------------------------------|\n");

printf(" | Работники |\n");

printf(" |-------------------------------------------------------------------------|\n");

printf(" | Фамилия | Инициалы | Должность | Год поступления на работу|\n");

printf(" |-------------------------------------------------------------------------|\n");

for (int i = 0; i<N; ++i)

{

cout << " |" << setw(13) << mass[i].surname << "|" << setw(14) << mass[i].initials << "|" << setw(15) << mass[i].post << "|" << setw(28) << mass[i].date << "|" << endl;

printf(" |-------------------------------------------------------------------------|\n");

}

cout << "Фамилии работников, чей стаж работы в организации превышает " << staj << " лет:" << endl;

for (int i = 0; i < N; ++i)

{

if ((2022 - mass[i].date) > staj)

{

cout << mass[i].surname << endl;

f = false;

}

}

if (f)

cout << "Отсутствуют." << endl;

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 27:**

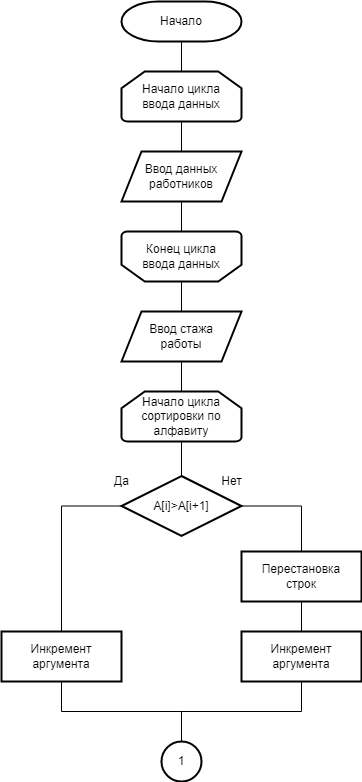




Рисунок 27 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

В первом цикле пользователь вводит данные работников (фамилия, инициалы, должность и год поступления на работу) и требуемый стаж. Следующий цикл осуществляет сортировку строк по фамилии в алфавитном порядке. После этого в другом цикле производится поиск работников, стаж которых превышает введённое пользователем значение (сравнивается год поступления на работу и текущий год, определённый в программе заранее). Далее в программе осуществляется печать таблицы с данными и вывод фамилии работника, чей стаж превышает заданное значение.

**Окно выполнения программы (рисунок 28):**

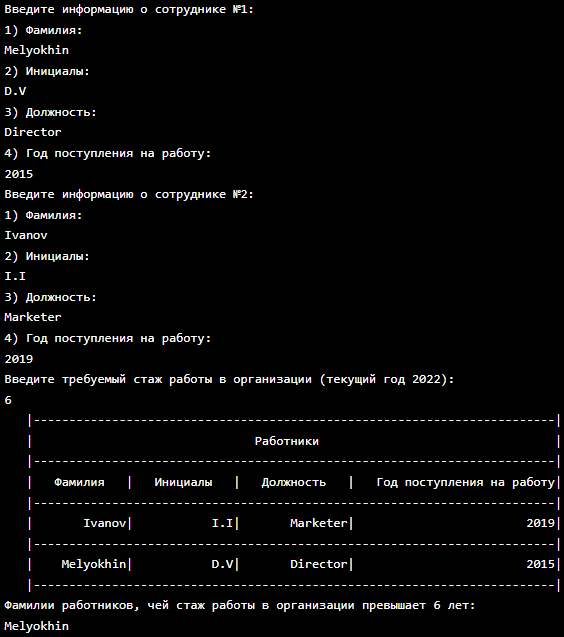


Рисунок 28 ‒ Окно выполнения

## Файлы

Цель работы: приобретение навыков работы с файлами.

Варианты заданий те же, что и по теме «Структуры».

Доработать:

1. вызов запросов оформить в виде меню;
2. добавить пункты: сохранить таблицу в файл и прочитать таблицу из файла.

Типы файлов:

* текстовый файл;
* двоичный файл, блочный ввод-вывод.

1. запросы оформить в виде подпрограмм.

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<iomanip>

#include <stdio.h>

using namespace std;

struct WORKER

{

char surname[13];

char initials[14];

char post[15];

int date;

};

int const N = 2;

int staj, num;

int j, i;

bool f = true;

WORKER mass[N];

struct WORKER work;

FILE\* fp;

void vvodtabl()

{

for (int i = 0; i < N; ++i)

{

cout << "Введите информацию о сотруднике №" << i + 1 << ":" << endl;

cout << "1) Фамилия: "; cin >> mass[i].surname;

cout << "2) Инициалы: "; cin >> mass[i].initials;

cout << "3) Должность: "; cin >> mass[i].post;

cout << "4) Год поступления на работу: "; cin >> mass[i].date;

}

cout << "Введите требуемый стаж работы в организации (текущий год 2022): ";

cin >> staj;

}

void vyvodtabl()

{

for (int i = 0; i < N; i++)

{

for (int j = i + 1; j < N; j++)

{

if (strcmp(mass[i].surname, mass[j].surname) > 0)

{

WORKER b = mass[i];

mass[i] = mass[j];

mass[j] = b;

}

}

}

printf(" |-------------------------------------------------------------------------|\n");

printf(" | Работники |\n");

printf(" |-------------------------------------------------------------------------|\n");

printf(" | Фамилия | Инициалы | Должность | Год поступления на работу|\n");

printf(" |-------------------------------------------------------------------------|\n");

for (int i = 0; i < N; ++i)

{

cout << " |" << setw(13) << mass[i].surname << "|" << setw(14) << mass[i].initials << "|" << setw(15) << mass[i].post << "|" << setw(28) << mass[i].date << "|" << endl;

printf(" |-------------------------------------------------------------------------|\n");

}

cout << "Фамилии работников, чей стаж работы в организации превышает " << staj << " лет:" << endl;

for (int i = 0; i < N; ++i)

{

if ((2022 - mass[i].date) > staj)

{

cout << mass[i].surname << endl;

f = false;

}

}

if (f)

cout << "Отсутствуют." << endl;

}

void outputtxt()

{

if ((fp = fopen("symbol.txt", "rt")) == NULL) printf("\nФайл не найден");

else

{

while (fscanf(fp, "%s %s %s %d", mass[i].surname, &(mass[i].initials), &(mass[i].post), &(mass[i].date)) != EOF) i++;

{

vyvodtabl();

fclose(fp);

}

}

}

void inputtxt()

{

if ((fp = fopen("symbol.txt", "wt")) == NULL) printf("\nФайл не найден");

else

{

for (int i = 0; i < N; i++)

{

fprintf(fp, "%s %s %s %d\n", mass[i].surname, mass[i].initials, mass[i].post, mass[i].date);

}

fclose(fp);

cout << endl;

cout << "Таблица была сохранена в текстовый файл" << endl;

}

}

void inputbi()

{

if ((fp = fopen("symbol.txt", "wb")) == NULL) printf("\nФайл не найден");

else

{

for (int i = 0; i < 7; i++)

{

work = mass[i];

fwrite(&work, sizeof(WORKER), 1, fp);

}

fclose(fp);

cout << endl;

cout << "Таблица была сохранена в двоичный файл" << endl;

}

}

void outputbi()

{

if ((fp = fopen("symbol.txt", "rb")) == NULL) printf("\nФайл не найден");

else

{

while (fread(&work, sizeof(WORKER), 1, fp)) i++;

vyvodtabl();

fclose(fp);

}

}

int main()

{

setlocale(LC\_CTYPE, "Russian");

int const N = 2;

int staj, num, odin;

int j;

bool f = true;

WORKER mass[N];

FILE\* fp;

while (true)

{

cout << "1 - Заполнить таблицу" << endl;

cout << "2 - Вывести таблицу" << endl;

cout << "3 - Записать таблицу в файл" << endl;

cout << "4 - Считать таблицу из файла" << endl;

cout << "5 - Записать таблицу в двоичный файл" << endl;

cout << "6 - Считать таблицу из двоичного файла" << endl;

cin >> num;

switch (num)

{

case 1:

{

vvodtabl();

break;

}

case 2:

{

vyvodtabl();

break;

}

case 3:

{

inputtxt();

break;

}

case 4:

{

outputtxt();

break;

}

case 5:

{

inputbi();

break;

}

case 6:

{

outputbi();

break;

}

default:

{

cout << endl;

cout << "Неверное значение";

break;

}

}

}

system("pause");

return 0;

}

**Блок-схема алгоритма изображена на рисунке 29:**

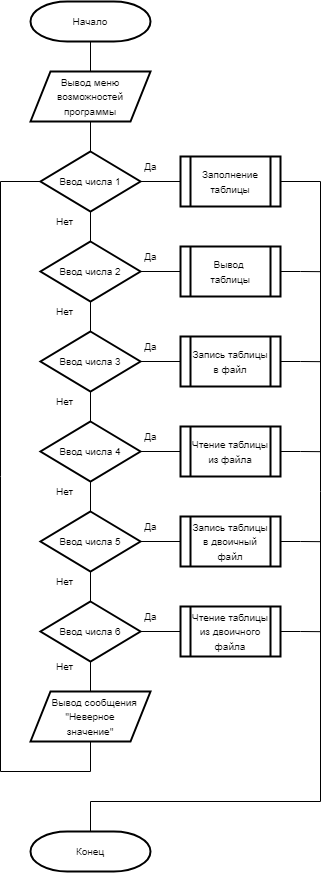


Рисунок 29 ‒ Блок-схема алгоритма

**Словесное описание алгоритма**

Программа выводит на экран меню, состоящее из 6 пунктов: заполнение таблицы, вывод таблицы, запись таблицы в файл, чтение таблицы из файла, запись таблицы в двоичный файл, чтение таблицы из двоичного файла. Пользователь вводит требуемое значение, после чего программа, с помощью оператора case переходит к соответствующей подпрограмме.

**Окно выполнения программы (рисунок 30):**

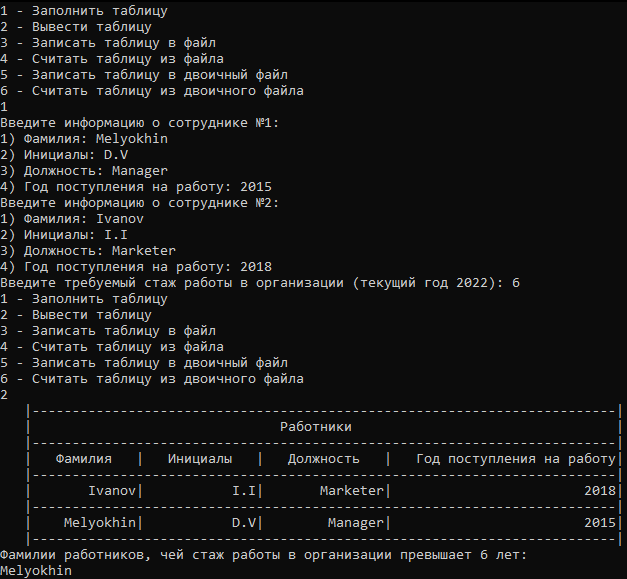


Рисунок 30 ‒ Окно выполнения

## Разработка простейшего CLR-приложения

**Задание 1:** разработайте приложение «Калькулятор».

**Фрагмент кода программы:**

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

// Считывание операнда 1

String^ S1 = textBox1->Text;

// Перевод значения в число

double operand1 = Convert::ToDouble(S1);

// Считывание операнда 2

String^ S2 = textBox2->Text;

// Перевод значения в число

double operand2 = Convert::ToDouble(S2);

// Вычисление суммы

double itog = operand1 + operand2;

// Вывод

label4->Text = itog.ToString();

}

private: System::Void label1\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void label2\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void textBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

// Считывание операнда 1

String^ S1 = textBox1->Text;

// Перевод значения в число

double operand1 = Convert::ToDouble(S1);

// Считывание операнда 2

String^ S2 = textBox2->Text;

// Перевод значения в число

double operand2 = Convert::ToDouble(S2);

// Вычисление разности

double itog = operand1 - operand2;

// Вывод

label4->Text = itog.ToString();

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

// Считывание операнда 1

String^ S1 = textBox1->Text;

// Перевод значения в число

double operand1 = Convert::ToDouble(S1);

// Считывание операнда 2

String^ S2 = textBox2->Text;

// Перевод значения в число

double operand2 = Convert::ToDouble(S2);

// Вычисление произведения

double itog = operand1 \* operand2;

// Вывод

label4->Text = itog.ToString();

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

// Считывание операнда 1

String^ S1 = textBox1->Text;

// Перевод значения в число

double operand1 = Convert::ToDouble(S1);

// Считывание операнда 2

String^ S2 = textBox2->Text;

// Перевод значения в число

double operand2 = Convert::ToDouble(S2);

// Вычисление частного

double itog = operand1 / operand2;

// Вывод

label4->Text = itog.ToString();

}

private: System::Void label4\_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

}

**Окно выполнения программы (рисунок 31):**

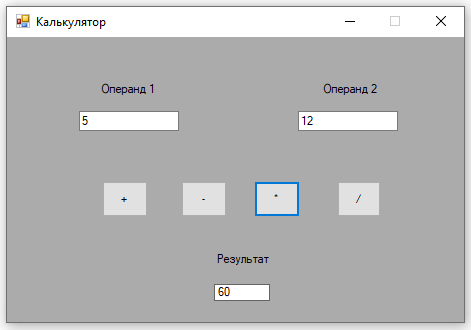
****

Рисунок 31 ‒ Окно выполнения

**Задание 2:** создать программу определения жирности тела.

**Фрагмент кода программы:**

int age; // возраст

int height; // рост

int weight; // вес

// Определение жирности тела

private: System::Double gir() {

double result;

// мужской пол

if (radioButton1->Checked) result = (weight - ((height \* 3 - 450 + age) \* 0.25 + 45)) / weight \* 100 + 15;

// женский пол

if (radioButton2->Checked) result = (weight - ((height \* 3 - 450 + age) \* 0.225 + 40.4)) / weight \* 100 + 22;

return result;

**Окно выполнения программы (рисунок 32):**

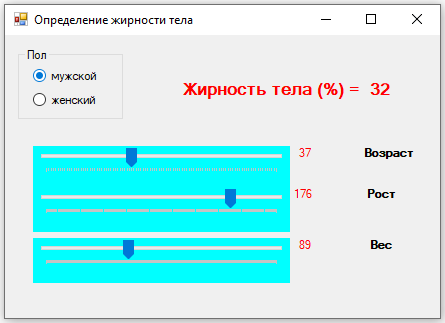


Рисунок 32 ‒ Окно выполнения

# Заключение

В ходе прохождения практики прошло ознакомление с языком программирования C++, были изучены такие темы как: программирование линейных программ, разветвляющиеся вычислительные процессы, табулирование функций, одномерные массивы, структуры, файлы, а также разработка простейшего CLR-приложения. Был изучен ГОСТ 7.32-2017, с помощью которого оформлено данное приложение, отчёт и задание на учебную практику. К индивидуальным заданиям были разработаны блок-схемы алгоритмов в соответствии с ГОСТ 19.701-90 ЕСПД.

# Библиографический список

1. Рейзлин, В.И. Язык С++ и программирование на нём: учебное пособие / В.И. Рейзлин ; Томский политехнический университет. — 3-е изд., перераб. — Томск: Изд-во Томского политехнического университета, 2021. — 208 с. ISBN 978-5-4387-0975-6
2. Шилдт, Герберт. Полный справочник по C + + , 4-е издание. Пер. с англ. — М.: Издательский дом “Вильямс”, 2006. — 800 с. ISBN 5-8459-0489-7
3. Зырянов, К. И. Программирование на C++ : учеб. пособие / К. И. Зырянов, Н. П. Кисленко ; Новосиб. гос. архитектур.-строит. ун-т (Сибстрин). — Новосибирск: НГАСУ (Сибстрин), 2017. — 128 с. ISBN 978-5-7795-0817-9
4. Пышкин, Е.В. Структурное проектирование: основание и развитие методов. С примерами на языке C++: Учеб. пособие. — СПб.: Изд-во Политехнического ун-та, 2005. — 324 с. ISBN 978-5-4567-0823-7
5. Пахомов, Б. И. C/C++ и MS Visual C++ 2010 для начинающих. — СПб.: БХВ-Петербург, 2011. — 736 с. ISBN 978-5-9775-0599-4