МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Отчёт по лабораторным работам за 2 семестр**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Вариант 7.

Проверил: доц. Шакин В. Н.

Москва, 2023 г.

**Оглавление**

[**Лабораторная работа №8 2**](#_Toc137626573)

[**Лабораторная работа №9 8**](#_Toc137626574)

[**Лабораторная работа №10 17**](#_Toc137626575)

[**Лабораторная работа №11 25**](#_Toc137626576)

[**Лабораторная работа №12 33**](#_Toc137626577)

[**Лабораторная работа №13 39**](#_Toc137626578)

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

# Лабораторная работа №8

**«Интегрированная среда разработки Visual Studio .NET**

**и технология визуального событийного**

**программирования»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Проверил: доц. Шакин В. Н.

Москва, 2023 г.

**1) Задание на разработку приложения**

Создать приложение Windows Forms на C++, которое

должно вычислять периметр и площадь треугольника по заданным значени-

ям координат трех его вершин.

Проект-ЛР1 должен иметь простой интерфейс пользователя:

• две командные кнопки для запуска и завершения работы приложения;

• шесть текстовых полей для ввода и отображения входных данных;

• два текстовых поля, отображающих выходные данные;

• поясняющие текстовые надписи.

**2) Формализация и уточнение задания**

Задано: x1, y1; x2, y2; x3, y3 – значения координат вершин треугольника;

Требуется определить: Р = A+B+C – периметр треугольника;

S = – площадь треугольника;

где A, B, C – длины сторон треугольника (расстояния между заданными

точками вершин треугольника), определяемые по формулам:

A = ;

B = ;

C = ;

р = Р/2 – полупериметр.

Должны выполняться следующие ограничения на значения входных данных и соотношения между ними: A>0 и B>0 и C>0 и A+B>C и A+C>B и B+C>A

**3) Разработка приложения**

3.1) Создание проекта

Запустить VS, и, выбрав шаблон CLR приложения WinForms, создать проект с именем lab\_work\_8.

3.2) Разработка графического интерфейса пользователя

Изменить масштаб формы, добавить в нее кнопки, текст и элементы ввода текста. Рисунок 8.1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 8.1 – Макет формы

Изменив свойства объекта button1, переключиться на вторую кнопку button2 при помощи Окна Свойств. Рисунок 8.2.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 8.2 – Все объекты формы

Далее изменить свойства второй кнопки. Рисунок 8.3.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 8.3 – Свойства второй кнопки

После подробного изучения окна свойств разработка пользовательского интерфейса окончена. Рисунок 8.4.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 8.4 – Пользовательский интерфейс

3.3) Программный код проекта

Создадим файл Resh.cpp, куда необходимо добавить функции для успешного расчета формы. Код файла представлен ниже.

#include "Resh.h"

double LenS(double Xa, double Ya, double Xb, double Yb) {

return sqrt(pow((Xa - Xb), 2) + pow((Ya - Yb), 2));

}

double CalcP(double a, double b, double c) {

return a + b + c;

}

double CalcS(double a, double b, double c) {

double p = CalcP(a, b, c) / 2;

return sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

double PS(double x1, double y1, double x2, double y2,

double x3, double y3, double& p) {

double a, b, c, s;

a = LenS(x1, y1, x2, y2);

b = LenS(x3, y3, x2, y2);

c = LenS(x1, y1, x3, y3);

p = CalcP(a, b, c);

s = CalcS(a, b, c);

return s;

}

Создадим заголовочный файл Resh.h, чтобы хранить необходимые директивы using namespace, директивы include для подключения необходимых заголовочных файлов, а также прототипы используемых функций пользователя.

#pragma once

using namespace System;

using namespace System::Windows::Forms;

#include <cmath>

double PS(double, double, double, double, double, double, double&);

double LenS(double, double, double, double);

double CalcP(double, double, double);

double CalcS(double, double, double);

Код конструктора формы представлен ниже.

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

this->Close();

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

double x1 = Convert::ToDouble(txtX1->Text);

double y1 = Convert::ToDouble(txtY1->Text);

double x2 = Convert::ToDouble(txtX2->Text);

double y2 = Convert::ToDouble(txtY2->Text);

double x3 = Convert::ToDouble(txtX3->Text);

double y3 = Convert::ToDouble(txtY3->Text);

double s, p;

// Расчет

s = PS(x1, y1, x2, y2, x3, y3, p);

// Вывод

this->txtPttr->Text = p.ToString();

this->txtPltr->Text = s.ToString();

}

**4) Результаты выполнения программы.**

Результат представлен на рисунке 8.5. Вычисления произведены верно.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 8.5 – Результат выполнения программы

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

# Лабораторная работа №9

**«Создание приложений Windows Forms для формирования и обработки автоматических одномерных массивов»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Вариант 7.

Проверил: доц. Шакин В. Н.

Москва, 2023 г.

**1) Индивидуальное задание на разработку проекта**

Создать проект решения следующей задачи:

* Сформировать автоматический массив заданного размера из вещественных случайных целых чисел в заданном диапазоне;
* Если в сформированном массиве среднее значение чисел, не превышающих заданное значение z, положительно, то создать новый массив, записав в него отрицательные элементы исходного массива в обратном порядке;
* Вывести оба массива в списки формы.

Разработать схему алгоритма и программный код функции task – решения задачи вычисления среднего значения и нового массива.

**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

• формирование исходного массива из случайных целых чисел – функция input;

• формирование нового массива из простых чисел исходного массива – функция task;

• вывод нового массива из простых чисел в заданный список (элемент управления ListBox) формы – функция output.

Вызов функций формирования массивов и вывода нового массива производится в событийной процедуре по нажатию кнопки Вычислить.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 9.1.

Форма содержит текстовые поля TextBox для ввода исходных данных, списки ListBox для вывода значений массивов, а также кнопки Button для вычисления и выхода из приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 9.1 – Графический интерфейс проекта

**4) Разработка схемы алгоритма функции task**

Схема алгоритма изображена на рисунке 9.2.

Входными параметрами функции task являются массив mas и число элементов в массиве n. Функция реализует алгоритм вычисления среднего значения и создания нового массива из элементов другого массива (при условии, что среднее значение положительно).

Сначала в цикле подсчитываются сумма и количество элементов, удовлетворяющих условию <= z. Затем, если среднее значение положительно, то в новом цикле с конца проверяется каждый элемент на отрицательность и, если это так, то это число добавляется в новый массив на j-е место (тем самым в обратном порядке записывая отрицательные числа).

Сформированный массив new\_mas является выходным параметром функции, а количество его элементов j – возвращаемым ею значением.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 9.2 – Схема алгоритма функции task

**5) Разработка программного кода приложения**

**5.1) Файлы с функциями решения поставленной задачи**

В данном проекте в соответствии с заданием создается два файла с программным кодом функций:

* Файл inoutput.cpp, содержащий функции GetInt, GetDouble для ввода из текстового поля формы и преобразования в целое число и в вещественное число соответственно, функцию input формирования массива из случайных вещественных чисел, и функцию output для ввода одномерного массива вещественных чисел в элемент управления ListBox формы;
* Файл task.cpp, содержащий функцию task, которая рассчитывает среднее значение, а также формирует новый массив из отрицательных чисел исходного массива, необходимые для решения поставленной задачи.

Ниже приведен программный код файла inoutput.cpp.

#include "Resh.h"

using namespace System;

using namespace System::Windows::Forms;

void input(double mas[], int n, double a, double b) {

Random ^rnd = gcnew Random();

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

mas[i] = a+(b-a)\*rnd->NextDouble();

}

}

void output(double mas[], int n, ListBox^ Lb) {

Lb->Items->Clear();

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

String ^s = mas[i].ToString();

Lb->Items->Add(s);

}

}

bool GetInt(const int max\_val, TextBox^ tn, String^ s, int& n) {

if (tn->Text->Length == 0) {

MessageBox::Show(s, "Ошибка вводa", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

n = Convert::ToInt32(tn->Text);

if (n < 1 || n > max\_val) {

MessageBox::Show(s, "Ошибка ввода", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

return true;

}

bool GetDouble(const int max\_val, TextBox^ tn, String^ s, double& n) {

if (tn->Text->Length == 0) {

MessageBox::Show(s, "Ошибка вводa", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

n = Convert::ToDouble(tn->Text);

if (n > max\_val) {

MessageBox::Show(s, "Ошибка ввода", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

return true;

}

Функция input заполняет массив случайными натуральными числами и имеет 4 формальных параметра: сам массив mas[], количество элементов в массиве n, диапазон случайных чисел a,b. В теле функции для генерации случайных чисел создается объект класса Random, а затем в цикле с помощью метода NextDouble этого объекта генерируется случайное число и присваивается очередному элементу массива.

Следующая функция в этом файле GetInt служит для ввода целого числа из текстового поля формы и тоже имеет 4 формальных параметра. Первый параметр - const int MaxVal типа целая константа – это максимальное число, которое пользователь имеет право ввести в текстовое поле, второй – само текстовое поле (вернее, указатель на него). Третий параметр – это строка, которая выводится в окно сообщений MessageBox в случае ошибочного ввода, т.е. в том случае, когда текстовое поле пусто, либо если введенное в него значение превышает заданное в параметре MaxVal максимально возможное число, или, наоборот, введенное число меньше 1. Четвертый параметр функции передается по ссылке, и в него записывается введенное число в случае успешного ввода. Функция 11 возвращает значение логического типа (true или false), показывающее, был ли произведен успешный ввод значения или нет. Аналогично с функцией GetDouble.

Функция output выводит массив в заданный список на форме и имеет 3 формальных параметра: массив, число выводимых элементов и указатель на объект типа ListBox, в который требуется произвести вывод.

Ниже приведен программный код файла task.cpp.

Функция task соответствует схеме алгоритма на рисунке 9.2.

#include "Resh.h"

int task(double mas[], double new\_mas[], double z, int n) {

int count = 0;

double sum = 0;

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

if (mas[i] <= z) {

sum += mas[i];

count++;

}

}

double sred = sum / count;

if (sred > 0) {

int j = 0;

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

if (mas[i] < 0) {

new\_mas[j] = mas[i];

j++;

}

}

return j;

}

return 0;

}

5.2) Программный код заголовочного файла Resh.h.

#pragma once

using namespace System;

using namespace System::Windows::Forms;

void input(double mas[], int n, double a, double b);

void output(double mas[], int n, ListBox^ Lb);

bool GetInt(const int max\_val, TextBox^ tn, String^ s, int& n);

bool GetDouble(const int max\_val, TextBox^ tn, String^ s, double& n);

int task(double mas[], double new\_mas[], double z, int n);

5.3) Событийные процедуры

В данной работе создаются две событийные процедуры, вызываемые при нажатии кнопок button1\_Click (Вычисление) и button2\_Click (Выход). На рисунке 6 приведен код этих процедур в файле MyForm.h.

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

this->Close();

}

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

const int N = 100;

double mas1[N], mas2[N];

int n1, n2;

double a, b, z;

if (!GetInt(N, text\_n, "Введите n:", n1)) return;

if (!GetDouble(1000, text\_b, "Введите b:", b)) return;

if (!GetDouble(b, text\_a, "Введите a:", a)) return;

if (!GetDouble(b, text\_z, "Введите z:", z)) return;

input(mas1, n1, a, b);

output(mas1, n1, lstMas1);

n2 = task(mas1, mas2, z, n1);

if (n2)

output(mas2, n2, lstMas2);

else

MessageBox::Show("В исходном массиве либо нет отрицательных чисел, либо среднее значение неположительно", "Сообщение",

MessageBoxButtons::OK, MessageBoxIcon::Information);

}

**6) Результаты выполнения приложения**

На рисунке 9.3 приведены результаты выполнения приложения при различных исходных данных.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 9.4 – Результаты выполнения проекта

**7) Доказательство правильности результатов**

В первом случае в исходном массиве среднее значение не положительно, поэтому в окно MessageBox выведено соответствующее сообщение. Во втором случае элементы нового массива являются отрицательными, записанными в обратном порядке.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

# Лабораторная работа №10

**«Создание приложений Windows Forms для формирования и обработки динамических одномерных массивов» по теме «Указатели и динамические массивы»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Вариант 7.

Проверил: доц. Шакин В. Н.

Москва, 2023 г.

**1) Индивидуальное задание на разработку проекта**

Создать проект решения следующей задачи:

* Создать динамический одномерный массив из чисел, записанных в многострочном текстовом поле формы;
* Если в сформированном массиве среднее значение чисел, не превышающих заданное значение z, положительно, то создать новый массив, записав в него отрицательные элементы исходного массива в обратном порядке;
* Вывести массив в поле формы.

**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

* формирование исходного массива чтением чисел из поля формы – функция input;
* определение количества отрицательных чисел в исходном массиве – функция analys;
* формирование нового массива с отрицательными числами, записанными в обратном порядке из исходного массива – функция task;
* вывод нового массива из отрицательных чисел в заданный список (элемент управления ListBox) формы – функция output.

Вызов функций формирования массивов и вывода нового массива производится в событийной процедуре по нажатию кнопки Вычислить.

**3) Разработка графического интерфейса приложения**

Графический интерфейс представлен на рисунке 10.1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 1 – Графический интерфейс проекта

Форма содержит текстовые поля TextBox для ввода исходных данных, список ListBox для вывода значений массивов, а также кнопки Button для вычисления и выхода из приложения.

**4) Разработка схем алгоритмов функций**

Схемы алгоритмов функций analys и task приведены на рисунках 10.2,3.

Изображение выглядит как диаграмма, текст, План, снимок экрана

Автоматически созданное описание

Рисунок 10.2 – Функция analys

Функция analys подсчитывает количество отрицательных чисел в массиве, а также среднее значение чисел, меньших z.

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Автоматически созданное описание

Рисунок 10.3 – Функция task

Функция task создает новый динамический массив из отрицательных чисел исходного массива.

Функция прежде всего вызывает функцию analys, возвращающую количество отрицательных чисел в исходном массиве, и записывает это количество в выходной параметр m. Если функция analys вернула 0, то новый массив не создается, а функция task возвращает ноль и прекращает свою работу.

**5) Разработка программного кода приложения**

В данном проекте в соответствии с заданием создается два файла с программным кодом функций:

* Файл inoutput.cpp, содержащий функцию input чтения массива из случайных вещественных чисел, и функцию output для ввода одномерного массива вещественных чисел в элемент управления ListBox формы;
* Файл task.cpp, содержащий функцию analys, которая рассчитывает среднее значение и количество отрицательных чисел массива, и функцию task, которая формирует новый массив из отрицательных чисел исходного массива, необходимые для решения поставленной задачи.

Код файла inoutput.cpp представлен ниже.

#include "Resh.h"

using namespace System;

using namespace System::Windows::Forms;

double\* input(int& n, TextBox^ TB) {

n = TB->Lines->Length;

if (n == 0) {

MessageBox::Show("Заполните числами Текстовое поле",

"Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

TB->Focus();

return 0;

}

double\* mas = new double[n];

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

String^ s = TB->Lines[i];

\*(mas + i) = Convert::ToDouble(s);

}

return mas;

}

void output(double\* mas, int n, ListBox^ Lb) {

Lb->Items->Clear();

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

String^ s = Convert::ToString(mas[i]);

Lb->Items->Add(s);

}

if (!n) Lb->Items->Add("Массив пуст");

}

Код файла task.cpp представлен ниже.

#include "Resh.h"

int analys(double\* a, int n, double z, double& avg) {

int k = 0;

double sum = 0;

int count = 0;

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

if (\*(a + i) <= z) {

sum += \*(a + i);

count++;

}

if (\*(a + i) < 0) {

k++;

}

}

avg = sum / count;

return k;

}

double\* task(double\* mas, int n, int& m, double z) {

double avg;

m = analys(mas, n, z, avg);

if (avg <= 0) return 0;

double\* newmas = new double[m];

int k = 0;

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

if (\*(mas + i) < 0) {

\*(newmas + k) = \*(mas + i);

k++;

}

}

return newmas;

}

Программные коды заголовочных файлов Resh.h, MyForm.h приведены ниже.

/\*Файл Resh.h\*/

#pragma once

using namespace System;

using namespace System::Windows::Forms;

void output(double\* mas, int n, ListBox^ Lb);

double\* input(int& n, TextBox^ TB);

double\* task(double\* mas, int n, int& m, double z);

int analys(double\* a, int n, double z, double& avg);

/\*Файл MyForm.h\*/

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

this->Close();

}

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

lstMas2->Items->Clear();

int n = 0, k = 0;

double\* mas = input(n, textBox1);

if (!mas) return;

double\* newmas = task(mas, n, k, Convert::ToDouble(text\_z->Text));

if (!newmas) MessageBox::Show("Среднее значение отрицательно",

"Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error); return;

output(newmas, k, lstMas2);

if (mas) delete[] mas;

if (newmas) delete[] newmas;}

**6) Результаты выполнения приложения**

На рисунках 10.4,5 приведены результаты выполнения приложения при различных исходных данных.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 10.4 – Результат выполнения при наличии отрицательных чисел и положительного среднего значения в исходном массиве

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 10.5 – Результат выполнения при наличии отрицательных чисел и отрицательного среднего значения в исходном массиве

**7) Доказательство правильности результата**

В первом случае все записанные в новый массив элементы являются отрицательными числами из исходного массива. Во втором случае в исходном массиве среднее значение отрицательно, поэтому в список Новый массив функцией output выведено соответствующее сообщение.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

# Лабораторная работа №11

**«Создание приложений Windows Forms для формирования и обработки двумерных массивов»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Вариант 7.

Проверил: доц. Шакин В. Н.

Москва, 2023 г.

**1) Индивидуальное задание на разработку проекта**

Создать проект решения следующей задачи:

* Сформировать и вывести массив заданного размера из вещественных случайных чисел в заданном диапазоне в список формы;
* Если в сформированном массиве среднее значение положительных чисел не превышает заданное значение R, то создать новый двумерный массив, записав в него строки исходного массива в обратном порядке;
* Вывести оба массива в списки формы.

**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

* формирование исходного массива из случайных вещественных чисел – функция input;
* вывод исходного массива в заданный список формы – функция output.
* формирование нового массива с обратными строками исходного массива – функция task;
* освобождение памяти сформированных массивов – функция del.

Вызов функций формирования и вывода динамических массивов производится в событийной процедуре по нажатию кнопки Вычислить.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 11.1.

Форма содержит текстовые поля TextBox для ввода исходных данных, списки ListBox для вывода значений массивов, а также кнопки Button для вычисления и выхода из приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 11.1 – Графический интерфейс приложения

**4) Разработка схем алгоритмов функций**

Схема алгоритма функции mean представлена на рисунке 11.2.

Функция mean вычисляет среднее значение положительных чисел двумерного массива и возвращает bool-значение меньше ли среднее значение введенного R или нет.

Изображение выглядит как диаграмма, снимок экрана, зарисовка, текст

Автоматически созданное описание

Рисунок 11.2 – Схема алгоритма функции mean

Схема алгоритма функции task представлена на рисунке 11.3.

Функция task сначала вызывает функцию mean. В случае если mean вернула true, то формируется новый двумерный массив newmatr, строки которого обратны строкам исходного matr. В ином случае task возвращает 0.

Изображение выглядит как текст, диаграмма, зарисовка, План

Автоматически созданное описание

Рисунок 11.3 – Схема алгоритма функции task

**5) Разработка программного кода приложения.**

В данном проекте в соответствии с заданием создается два файла с программным кодом функций:

* Файл inoutput.cpp, содержащий функции GetInt, GetDouble для ввода из текстового поля формы и преобразования в целое число и в вещественное число соответственно, функцию input формирования двумерного массива из случайных вещественных чисел, и функцию output для вывода двумерного массива вещественных чисел в элемент управления ListBox формы;
* Файл task.cpp, содержащий функцию task, которая формирует новый массив из строк, обратных исходному массиву, и функцию mean, которая рассчитывает среднее значение положительных чисел.

Ниже приведен программный код файла inoutput.cpp.

#include "Resh.h"

using namespace System;

using namespace System::Windows::Forms;

double\*\* input(int m, int n, int a, int b) {

double\*\* matr = new double\* [m];

if (a > b)

{double t = a; a = b; b = t;}

Random^ rnd = gcnew Random();

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

\*(matr + i) = new double [n];

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

\*(\*(matr+i) + j) = a + (b - a) \* rnd->NextDouble();

}

return matr;

}

bool GetInt(int& x, TextBox^ tn, String^ s) {

if (tn->Text->Length == 0) {

MessageBox::Show(s, "Ошибка вводa", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

x = Convert::ToInt32(tn->Text);

if (x < 1 || x > 100) {

MessageBox::Show(s, "Ошибка ввода", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

return true;

}

bool GetDouble(double& x, TextBox^ tn, String^ s) {

if (tn->Text->Length == 0) {

MessageBox::Show(s, "Ошибка вводa", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

x = Convert::ToDouble(tn->Text);

if (x > 1e3) {

MessageBox::Show(s, "Очень большое double", MessageBoxButtons::OK, MessageBoxIcon::Error);

tn->Focus();

return false;

}

return true;

}

void output(double\*\* mas, int m, int n, ListBox^ Lb) {

Lb->Items->Clear();

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

String^ s = "";

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

s = s + String:: Format("{0,8:F2}", \*(\*(mas + i) + j));

//s += System::Convert::ToString(\*(\*(mas + i) + j)) + " ";

Lb->Items->Add(s);

}

}

void del(double\*\* a, int m) {

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

delete[] \*(a + i);

}

delete[] a;

}

Ниже приведен программный код файла task.cpp.

#include "Resh.h"

bool mean(double\*\* a, int m, int n, double R) {

double sum = 0;

int count = 0;

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

for (int j = 0; j < n; j++) {

double cur = \*(\*(a + i) + j);

if (cur > 0) {

sum += cur;

count++;

}

}

return ((sum / count) < R);

}

void copy(double\* b, double\* a, int n) {

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

\*(b + i) = \*(a + i);

}

double\*\* task(double\*\* matr, int m, int n, double R) {

if (!mean(matr, m, n, R)) return 0;

double\*\* newmatr = new double\*[m]; //?

int k = m-1;

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

\*(newmatr + i) = new double[n];

copy(\*(newmatr + i), \*(matr + k), n);

k--;

}

return newmatr;

}

Ниже представлены программные коды заголовочных файлов Resh.h, MyForm.h.

/\*Файл Resh.h\*/

#pragma once

using namespace System;

using namespace System::Windows::Forms;

void del(double\*\* a, int m);

void output(double\*\* mas, int m, int n, ListBox^ Lb);

bool GetDouble(double& x, TextBox^ tn, String^ s);

bool GetInt(int& x, TextBox^ tn, String^ s);

double\*\* input(int m, int n, int a, int b);

double\*\* task(double\*\* matr, int m, int n, double R);

/\*Файл MyForm.h\*/

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

this->Close();

}

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

lstMas1->Items->Clear();

lstMas2->Items->Clear();

int m, n;

double a, b, R;

if (!GetInt(m, text\_m, "Введите число строк матрицы (m)")) return;

if (!GetInt(n, text\_n, "Введите число столбцов матрицы (n)")) return;

if (!GetDouble(a, text\_a, "Введите левую границу диапазона (a)")) return;

if (!GetDouble(b, text\_b, "Введите правую границу диапазона (b)")) return;

if (!GetDouble(R, text\_z, "Введите R")) return;

double\*\* matr = input(m, n, a, b);

output(matr, m, n, lstMas1);

double\*\* matr2 = task(matr, m, n, R);

if (matr2) {

output(matr2, m, n, lstMas2);

del(matr2, m);

}

else MessageBox::Show("Новый массив не создан");

del(matr, m);

}

**6) Результаты выполнения приложения**

На рисунках 11.5,6 представлены результаты выполнения приложения при различных исходных данных.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11.5 – Результат выполнения проекта при среднем значении, меньшем числа R

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 11.6 - Результат выполнения проекта при среднем значении, меньшем числа R

**7) Доказательство правильности результатов**

В первом случае все записанные в новый массив строки обратны исходному массиву. Во втором среднее значение больше заданного R, поэтому в окно MessageBox выведено соответствующее сообщение.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

# Лабораторная работа №12

**«Создание приложений Windows Forms**

**для обработки строк»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Вариант 7.

Проверил: доц. Шакин В. Н.

Москва, 2023 г.

**1) Индивидуальное задание на разработку проекта**

Создать проект решения следующей задачи:

В заданной строке, состоящей из слов на русском языке, выяснить, можно ли из символов заданной строки составить мою фамилию.

**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

* Ввод исходной строки;
* Поиск в строке символов, состоящих в фамилии;
* Вывод найденного слова.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 12.1. Форма содержит 6 объектов:

• текстовое поле txtString для ввода обрабатываемой строки;

• текстовое поле txtAns для вывода найденного слова;

• кнопку cmdFind для запуска процедуры поиска;

• кнопку cmdExit для выхода из приложения;

• надписи lblString и lblWord, комментирующие назначение текстовых полей.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 12.1 – Графический интерфейс формы

**4) Разработка схем алгоритмов функций**

Схема алгоритма функции get\_sur, решающая поставленную задачу, представлена на рисунке 12.2.

Изображение выглядит как диаграмма, текст, снимок экрана, линия

Автоматически созданное описание

Рисунок 12.2 – Схема алгоритма функции get\_sur

Функция get\_sur получает входную строку str, формирует сроку из фамилии, в цикле по символам фамилии ищет каждый символ в str и, в случае нахождения удаляет его из str, иначе возвращает пустую строку. Если цикл завершился без возвращения пустой строки, возвращается положительный ответ.

**5) Разработка программного кода приложения**

В данном проекте в соответствии с заданием создается файл с программным кодом функции: task.cpp. Ниже представлен код этого файла.

#include "Resh.h"

using namespace System;

using namespace System::Windows::Forms;

String^ get\_sur(String^ str) {

str = str->ToLower();

String^ surname = "васильев";

String^ ch;

int index;

String^ ans = "Да, в строке есть необходимые символы";

for (int i = 0; i < 8; i++) {

ch = surname->Substring(i, 1);

if ((index = str->IndexOf(ch)) >= 0) {

str = str->Remove(index, 1);

}

else return "";

}

return ans;

}

В файл Resh.h помещаются директивы using и записывается прототип используемой функции.

#pragma once

using namespace System;

using namespace System::Windows::Forms;

String^ get\_sur(String^ str);

Событийные процедуры проекта приведены в коде ниже. В процедуре MyForm\_Load события загрузка формы на всякий случай очищаются оба текстовых поля. В процедуре button2\_Click события нажатие кнопки Выйти выполняется выход из приложения. Выход можно выполнить также закрытием окна с формой. Процедура txtString\_TextChanged служит для очистки текстового поля с найденным словом при изменении введенной строки перед повторным поиском. Основные действия выполняются в процедуре button1\_Click.

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

String^ source = txtString->Text;

if (source->Length == 0) {

MessageBox::Show("Введите строку", "Ошибка",

MessageBoxButtons::OK, MessageBoxIcon::Error);

txtString->Focus();

return;

}

String^ res = get\_sur(source);

if (res->Length > 0)

txtAns->Text = res;

else

MessageBox::Show("Необходимые символы не найдены", "Сообщение",

MessageBoxButtons::OK, MessageBoxIcon::Information);

txtString->Focus();

txtString->SelectionStart = 0;

}

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

this->Close();

}

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

txtAns->Text = "";

}

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

txtString->Text = "";

txtAns->Text = "";

}

**6) Результаты выполнения приложения**

На рис. 12.3-5 приведены результаты работы приложения для случаев пустой исходной строки, найденных символов фамилии и отсутствия таких символов.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 - Результаты работы приложения с пустой строкой

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 6 – Результаты работы приложения с найденными символами

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 - Результаты работы приложения с ненайденными символами

**7) Доказательство правильности результата**

Правильность работы приложения подтверждается приведенными выше результатами его выполнения.

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

# Лабораторная работа №13

**«Создание приложений Windows Forms**

**для работы с потоковыми файлами»**

**по дисциплине**

**«Алгоритмизация и программирование»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Вариант 7.

Проверил: доц. Шакин В. Н.

Москва, 2023 г.

**1) Индивидуальное задание на разработку проекта**

Создать проект решения следующей задачи:

Записать в новый файл те строки исходного текстового файла, состоящего из слов на русском языке, из символов которой можно составить мою фамилию.

**2) Формализация задачи**

Из условия задания следует, что его выполнение распадается на следующие этапы, реализуемые соответствующими функциями приложения:

* открытие входного файла для чтения строк;
* открытие выходного файла для записи строк;
* чтение из входного файла очередной строки;
* вывод очередной строки в соответствующий объект формы;
* обработка очередной строки с использованием функции, разработанной в предыдущей лабораторной работе;
* вывод результата обработки в выходной файл;
* вывод результата обработки в соответствующий объект формы.

**3) Разработка графического интерфейса пользователя приложения**

Графический интерфейс проекта изображен на рисунке 13.1. Форма содержит 6 объектов:

* текстовое поле txtIn для отображения полного пути к входному файлу;
* текстовое поле txtOut для отображения полного пути к входному файлу;
* список lstIn для отображения строк входного файла;
* список lstOut для отображения строк выходного файла;
* кнопку butInp для перехода в окно диалога выбора входного файла;
* кнопку butOut для перехода в окно диалога выбора выходного файла;
* кнопку butExec для запуска процедуры чтения и обработки входного файла;
* кнопку butExit для выхода из приложения;
* четыре надписи, комментирующие назначение текстовых полей и списков.

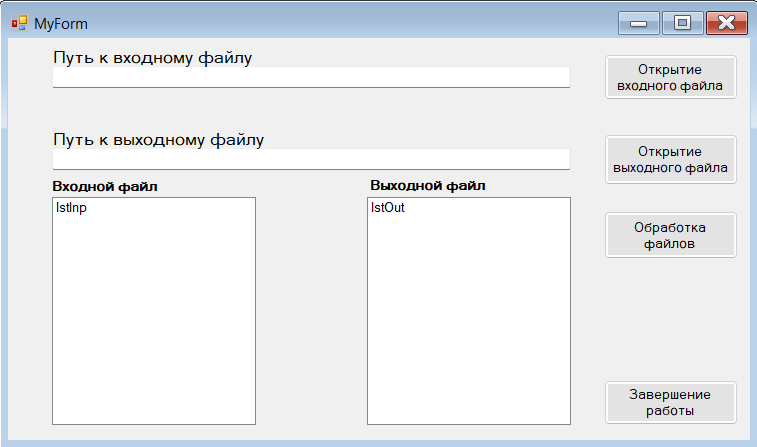


Рисунок 13.1 – Графический интерфейс программы

**4) Разработка схем алгоритмов функций**

Схема алгоритма новой функции FileProc представлена на рисунке 13.2. Функция get\_sur берется из предыдущей лабораторной работы.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 13.2 – Схема алгоритма FileProc

Функция получает в качестве входных параметров входной и выходной потоки sr и sw, а также списки для отображения строк входного и выходного файлов in\_file и out\_file. В итеративном цикле с предусловием (пока не достигнут конец входного потока) выполняется чтение строк из входного потока; вывод прочитанной строки в соответствующий список формы; вызов функции get\_sur, возвращающей первое слово с одинаковым количеством гласных и согласных букв; вывод найденной строки в выходной поток, вывод найденной строки в соответствующий список формы.

**5) Разработка программного кода приложения**

В данном проекте в соответствии с заданием создается файл с программным кодом функции: files.cpp. На рисунке 3 приведен программный код функции FileProc, вызываемой из событийной процедуры. Функция get\_sur копируется из предыдущей работы (представлена ниже).

void FileProc(StreamReader^ sr, StreamWriter^ sw,

ListBox^ in\_file, ListBox^ out\_file) {

String^ in\_buffer = "";

String^ out\_buffer = "";

try {

while (in\_buffer = sr->ReadLine()) {

if (in\_buffer->Length == 0) continue;

in\_file->Items->Add(in\_buffer);

out\_buffer = get\_sur(in\_buffer);

if (out\_buffer->Length < 0) continue;

sw->WriteLine(out\_buffer);

out\_file->Items->Add(out\_buffer);

}

}

catch (Exception^ ex) {

MessageBox::Show(ex->Message, "Ошибка",

MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

В файл Resh.h помещаются директивы using и записывается прототипы используемых функций. Данный файл представлен на рисунке 4.

#pragma once

using namespace System;

using namespace System::IO;

using namespace System::Windows::Forms;

String^ get\_sur(String^ str);

void FileProc(StreamReader^ sr, StreamWriter^ sw,

ListBox^ in\_file, ListBox^ out\_file);

В процедуре Form1\_Load события загрузка формы очищаются списки формы и текстовые поля, а также делается недоступной кнопка Обработка файлов. В процедуре cmdExit\_Click события нажатие кнопки Завершение работы 10 выполняется выход из приложения. Выход можно выполнить также закрытием окна с формой. В событийной процедуре кнопки Открытие входного файла производится выбор текстового файла. В событийной процедуре кнопки Открытие выходного файла выполняется установка полного пути к текстовому файлу, в который будут выводиться результаты обработки строк входного файла. Событийная процедура кнопки Обработка файлов подготавливает и запускает процесс обработки.

Все событийные процедуры представлены в коде ниже.

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

lstInp->Items->Clear();

lstOut->Items->Clear();

txtIn->Clear();

txtOut->Clear();

butExec->Enabled = false;

}

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

this->Close();

}

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

OpenFileDialog^ openFileDialog = gcnew OpenFileDialog;

openFileDialog->Title = "Открытие входного файла";

openFileDialog->InitialDirectory = "C:\\";

openFileDialog->Filter = "Текстовые файлы (\*.txt)|\*.txt";

openFileDialog->FilterIndex = 1;

openFileDialog->ShowReadOnly = true;

openFileDialog->ReadOnlyChecked = true;

openFileDialog->RestoreDirectory = true;

if (openFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)

txtIn->Text = openFileDialog->FileName;

else

txtIn->Text = "";

butExec->Enabled = txtIn->Text->Length > 0 && txtOut->Text->Length > 0;

}

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

SaveFileDialog^ saveFileDialog = gcnew SaveFileDialog;

saveFileDialog->Title = "Открытие выходного файла";

saveFileDialog->Filter = "Текстовые файлы (\*.txt)|\*.txt";

saveFileDialog->FilterIndex = 1;

saveFileDialog->OverwritePrompt = true;

if (saveFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)

txtOut->Text = saveFileDialog->FileName;

else

txtOut->Text = "";

butExec->Enabled = txtIn->Text->Length > 0 && txtOut->Text->Length > 0;

}

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

lstInp->Items->Clear();

lstOut->Items->Clear();

String^ in\_file = txtIn->Text;

String^ out\_file = txtOut->Text;

StreamReader^ sr = nullptr;

StreamWriter^ sw = nullptr;

try {

sr = gcnew StreamReader(in\_file);

sw = gcnew StreamWriter(out\_file, false);

}

catch (Exception^ ex) {

MessageBox::Show(ex->Message, "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

if (sr) sr->Close();

if (sw) sw->Close();

return;

}

FileProc(sr, sw, lstInp, lstOut);

sr->Close();

sw->Close();

}

**6) Результаты выполнения приложения**

На рис. 13.3,4 приведены результаты работы приложения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 6 – Результаты работы приложения

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Файл тест.txt

**7) Доказательство правильности результата**

Правильность работы приложения подтверждается приведенными выше результатами его выполнения.