Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Санкт-Петербургский государственный морской технический университет»**

**(СПБГМТУ)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Факультет морского приборостроения

Кафедра систем автоматического управления и бортовой вычислительной техники

Направление подготовки 09.03.01 Информатика и вычислительная техника

Пояснительная записка к курсовой работе на тему «Табулирование функции»

По дисциплине «Программирование»

Курсовая работа студента группы №3270 II курса ФМП Горбарчук А.П.

Дата выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель: доцент кафедры систем автоматического управления и бортовой вычислительной техники Сакович С. Ю.

Оценка “\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ “\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

(подпись) (дата проверки)

Санкт Петербург

2021

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc89902555)

[1. Общие сведения 5](#_Toc89902556)

[2. Функциональное назначение 5](#_Toc89902557)

[3. Объектно-ориентированный анализ предметной области 5](#_Toc89902558)

[4. Описание логической структуры программы 6](#_Toc89902559)

[5. Входные данные 6](#_Toc89902560)

[6. Выходные данные 6](#_Toc89902561)

[Глава 1. Анализ решаемой задачи 7](#_Toc89902562)

[1.1 Математическое исследование 7](#_Toc89902563)

[1.2 Аналитическая модель решения задачи 7](#_Toc89902564)

[Глава 2. Парадигма структурного программирования в терминах C++ 9](#_Toc89902565)

[Глава 3. Концепция объектно-ориентированного программирования 13](#_Toc89902566)

[Глава 4. Разработка интерфейса 16](#_Toc89902567)

[Заключение 18](#_Toc89902568)

[Список литературы 19](#_Toc89902569)

[Приложение А. 20](#_Toc89902570)

[Приложение Б. 22](#_Toc89902571)

[Приложение В. 24](#_Toc89902572)

# ВВЕДЕНИЕ

*Парадигма* – совокупность принципов, методов и понятий, определяющих организацию вычислений, структурирование работы и стиль написания программы. Парадигма программирования не определяется однозначно языком. Многие современные языки программирования допускают использование различных парадигм. Так в решении представленной задачи на языке Си++, который является объектно-ориентированным, помимо объектно-ориентированной парадигмы используются парадигмы структурного и событийно-ориентированного программирования.

*Структурное программирование* воплощает принципы системного подхода в процессе создания и эксплуатации программного обеспечения ЭВМ. В основу структурного программирования положены следующие положения: алгоритм и программа должны составляться поэтапно; сложная задача должна разбиваться на достаточно простые части, каждая из которых имеет один вход и выход; логика алгоритма и программы должна опираться на минимальное число простых базовых управляющих структур.

Фундаментом структурного программирования является *теорема о структурировании*. Эта теорема устанавливает, что схема любой, даже сложной, задачи всегда может быть представлена с использованием ограниченного числа элементарных управляющих структур. Базовыми элементарными структурами являются следование, ветвление и цикл. Любой алгоритм может быть реализован в виде композиции этих конструкций.

Структурное программирование делает текст программы более понятным, т.к. алгоритм решения виден из исходного текста.

Основой *объектно-ориентированного программирования* является понятие об объекте. Объект объединяет в себе структуры данных и характерные для него методы их обработки. Такой подход меняет стиль программирования. Он заключается в отображении объектов реального мира на программную среду.

Основные принципы ООП:

1. Инкапсуляция – объединение данных и свойственных им процедур обработки в одном объекте. Детальная реализация остаётся скрытой для пользователей.
2. Наследование – механизм использования данных и методов одного объекта в объектах, построенных на его основе.
3. Полиморфизм позволяет в иерархии наследников и базовых классов изменять работу методов базовых объектов.

*Событийно-ориентированное программирование* основано на событиях – действиях пользователя (клавиатура, мышь, сенсорный экран), сообщениями других программ и потоков, в ответ на которые в системе генерируется подходящее сообщение, которое отсылается окну соответствующей программы.

Событийно-ориентированное программирование применяется, как правило, при построении пользовательских интерфейсов, создании серверных приложений и при программировании игр.

Исследование различных процессов начинается с их моделирования, т.е. с отражения реального процесса через математическое соотношение. С математической стороны разработка данной программы несёт большую практическую ценность для обучающегося программиста, т.к. программирование и вычислительная работа компьютера основывается на математических принципах, от простейших операций до сложных вычислений. Математическое моделирование различных процессов появляется во многих областях человеческой деятельности.

Актуальность темы данной работы заключается в необходимости анализа возможностей и средств разработки программных приложений с помощью языков программирования высокого уровня.

Объект исследования – язык программирования высокого уровня, особенности его состава и практическое применение.

Предмет исследования – специфика разработки программных приложений с помощью различных парадигм программирования.

Целью настоящей работы является расширение, закрепление и систематизация знаний по изучаемой дисциплине, приобретение необходимых практических навыков для работы с функциями путём проведения анализа высокоуровневого языка программирования.

Задача: составить программы табулирования заданных функций в заданном интервале с заданной точностью, используя различные парадигмы программирования.

В пояснительной записке описано функциональное назначение данного программного продукта, его логическая структура и принцип работы программы.

1. Общие сведения

Разработка программного продукта была произведена с использованием средств языка C++ в среде визуального программирования Visual Studio 2019 в операционной системе Windows 10.

1. Функциональное назначение

Программа предназначена для табулирования функции в заданном интервале с заданной точностью и имеет следующие возможности:

1. Загрузка данных из файла;
2. Расчёт значений заданной функции с помощью классов;
3. Сохранение результатов вычисления в файл.
4. Объектно-ориентированный анализ предметной области

В абстрактном классе Base объявлена чистая виртуальная функция F.

Класс Function – дочерний класс Base – содержит массивы переменных и дочерний метод для расчёта значений. Эта переопределённая функция объявляется в открытой части класса.

Класс File предназначен для работы с исходным файлом. Содержит имя файла, функцию для определения количества переменных и функцию создания объекта класса Function.

В программе создаётся объект класса File. Затем с помощью метода String\_Counter класса File через объект класса Function производятся вычисления.

1. Описание логической структуры программы

В программе предусмотрены следующие компонентные функции:

1. String\_Counter считает количество строк в исходном файле для определения количества диапазонов табулирования.
2. Create\_arrays создаёт массивы для хранения переменных. В программе, реализованной с помощью концепции ООП, заменена на конструктор внутри класса Function.
3. F – функция для произведения расчётов. Принимает количество строк и записывает результаты в файл “result.txt”.
4. CreateFunction – функция, создающая объект класса Function, заполняющая массивы значений и передающая количество диапазонов табулирования в функцию F.
5. Входные данные

Входными данными данной программы является информация, загруженная из текстового файла. Типы вводимой информации должны соответствовать типам, представленным в Таблице 1.

Таблица 1. Типы входных данных

| Входные данные | Тип данных |
| --- | --- |
| Начало табулирования start | Число двойной точности double |
| Конец диапазона end | Число двойной точности double |
| Точность t | Число двойной точности double |

1. Выходные данные

Выходной информацией являются данные, сохраняемые в файл.

# Глава 1. Анализ решаемой задачи

Решение задач с помощью компьютера включает в себя несколько основных этапов, часть из которых осуществляется без участия компьютера.

## Математическое исследование

Перед началом разработки программы необходимо провести математическое исследование заданной функции.

Дана функция:

, (1)

Для корректной работы программы определим область определения функции.

Разделим функцию на части. Областью определения числа в степени являются все вещественные числа. В область определения натурального логарифма входят только положительные числа. Область определения квадратного корня – неотрицательные вещественные числа. Значит, чтобы логарифм не принял отрицательное значение, x должен быть больше или равен двум.

Таким образом, получаем область определения для всей функции:

, (2)

где – множество всех вещественных чисел.

* 1. Аналитическая модель решения задачи

В исследовании операций широко используются аналитические модели. Они учитывают меньшее число факторов и упрощают решаемую задачу до простых и идеализированных задач и объектов. Они обладают высокой общностью и отражают основные закономерности и результаты расчёта.

Для выполнения поставленной задачи в первую очередь необходимо передать данные из файла в функцию. Для возможности инициализации нескольких диапазонов табулирования можно создать массивы для каждого значения, рассчитанные по шагам значения так же следует помещать в массив и записывать в файл.

Таким образом задача разбивается на простые действия:

1. Считывание данных из файла, заполнение массивов;
2. Произведение и помещение расчётов в массив;
3. Запись полученных значений в файл.

# Глава 2. Парадигма структурного программирования в терминах C++

Распространённый способ описания алгоритма – блок-схема. Блок-схема позволяет представить сложный алгоритм в виде последовательностей простых структур. По сути блок-схема является схематичным представлением процесса или компьютерного алгоритма. На рисунке 2.1 представлена общая блок-схема программы.

int main()

int n = String\_Counter()

Function p = new Function

Create\_arrays(p, n)

ifstream n

in.open(“file.txt”)

i от 0 до n

p->start[i]

p->start[i]

p->start[i]

in.close()

F(p, n)

err.what()

end

exception err

Рисунок 2.1 – Блок-схема программы

Такая схема содержит подпрограммы, для каждой из которых представлены следующие блок-схемы:

struct Function

double\* start

double\* start

double\* start

double\* start

Рисунок 2.2 – Схема структуры Function

int String\_Counter()

int n =0

fstream f(“file.txt”)

string str = “”

f.close()

n++

return n

getline(f, str)

нет

да

Рисунок 2.3 – Схема функции String\_Counter

void Create\_arrays(struct Function\* p, int n)

p->start = new double[n]

p->end = new double[n]

p->t = new double[n]

p->y = new double[n]

Рисунок 2.4 – Схема функции Create\_arrays

В блок-схеме функции F цикл for для i расписан более подробно для наглядности.

i==2

void F(struct Function\* p, int n)

j от 0 до n

double i = p->start[j]

I < p->end[j]

p->y[j] = pow(i, 2) \* sqrt(log(i/2))

throw exception

p->y[j]

i += p->t[j]

нет

да

нет

да

Рисунок 2.5 – Схема функции F

Программа завершается выводом полученных значений на экран:

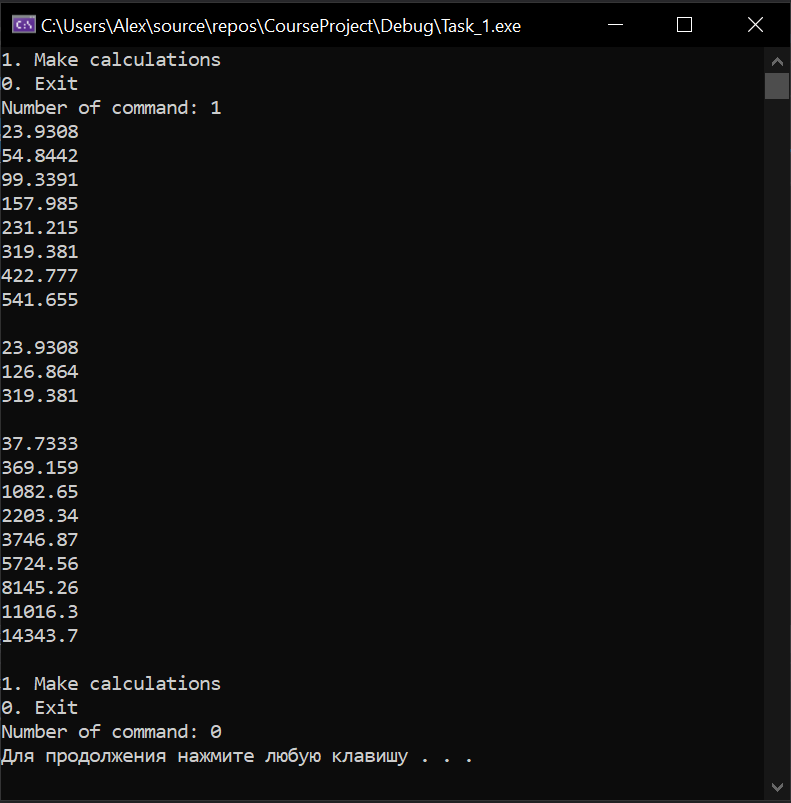


Рисунок 2.6 – Пример работы программы

# Глава 3. Концепция объектно-ориентированного программирования

Процесс разработки программы с использованием объектно-ориентированного подхода включает несколько этапов:

1. Анализ;
2. Проектирование;
3. Эволюция;
4. Модификация.

Цель анализа – максимально развёрнутое описание задачи. На этом этапе выполняются анализ предметной области, составление объектной композиции разрабатываемой программы и определяются особенности поведения объектов.

По результатам разрабатывается структурная схема программного продукта, на которой отображаются основные структурные компоненты и их связи.

Файл

Программа

Файл

Рисунок 3.1 – Структурная схема программы

Программа

Ввод данных

Запись результатов

Рисунок 3.2 – Функциональная схема программы

Проектирование заключается в разработке структуры классов. На этом этапе определяются поля для хранения объектов и алгоритмы методов. Далее классы объединяются в модули, производится выбор схем их подключения, определяются способы взаимодействия с оборудованием, с операционной системой или другим программным обеспечением.

В данной программе в производном классе Function определена виртуальная функция F из класса Base. Класс File использует объект класса Function в своём методе CreateFunction. Такой вид классовых отношений моделируется как зависимость.

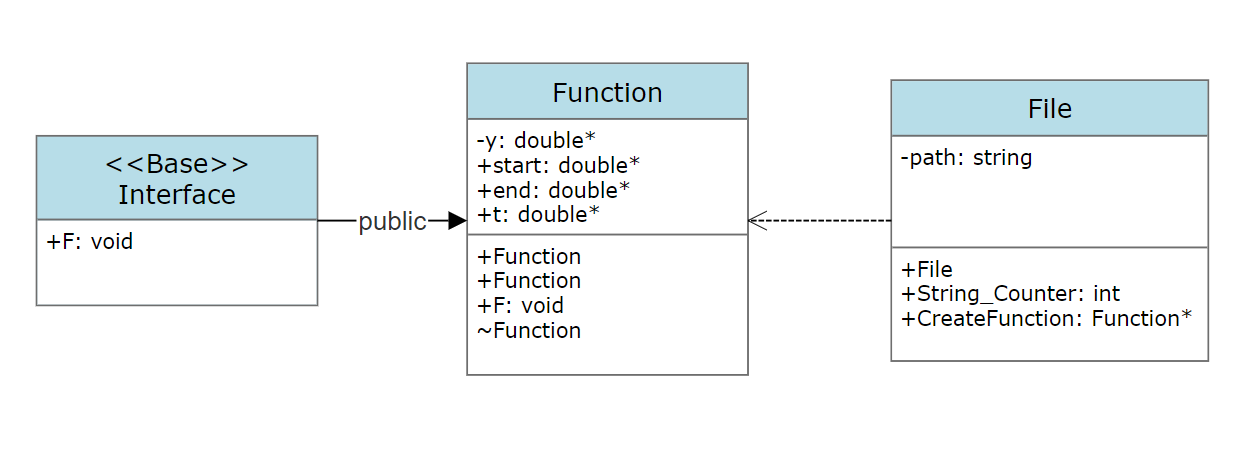


Рисунок 3.3 – Диаграмма классов

Эволюция системы – это процесс поэтапной реализации классов и подключения их к проекту.

Модификация – это процесс добавления новых функциональных возможностей или изменение существующих. При использовании ООП процесс обычно обходится без существенных проблем, т.к. изменения затрагивают локальную область программного кода.

Результатом работы данной программы является файл «result.txt», в который записываются расчёты.

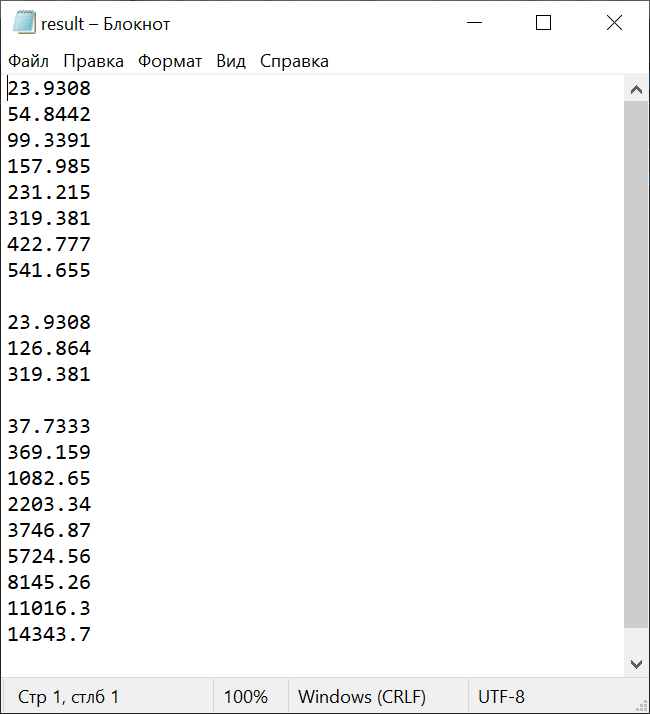


Рисунок 3.4 – Файл «result.txt»

# Глава 4. Разработка интерфейса

Пользовательский интерфейс – совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером.

Разработка пользовательского интерфейса включает те же основные этапы, что и разработка программы.

1. Определение типа интерфейса и общих требований к нему;
2. Определение сценариев использования и пользовательской модели интерфейса;
3. Проектирование;
4. Реализация.

Первой и самой главной задачей любого интерфейса является понятность. Назначение интерфейса должно быть очевидным для пользователя, пользователь должен понимать, с чем он взаимодействует через интерфейс.

В данной программе интерфейс содержит описание функции, поля для ввода данных, поле для графика и кнопки разных цветов. В основе программы лежит одно основное действие – расчёт значений функции. Второстепенное действие – запись в файл – имеет меньший визуальный вес. После нажатия основной копки значения выводятся на экран, в специальном поле строится график.

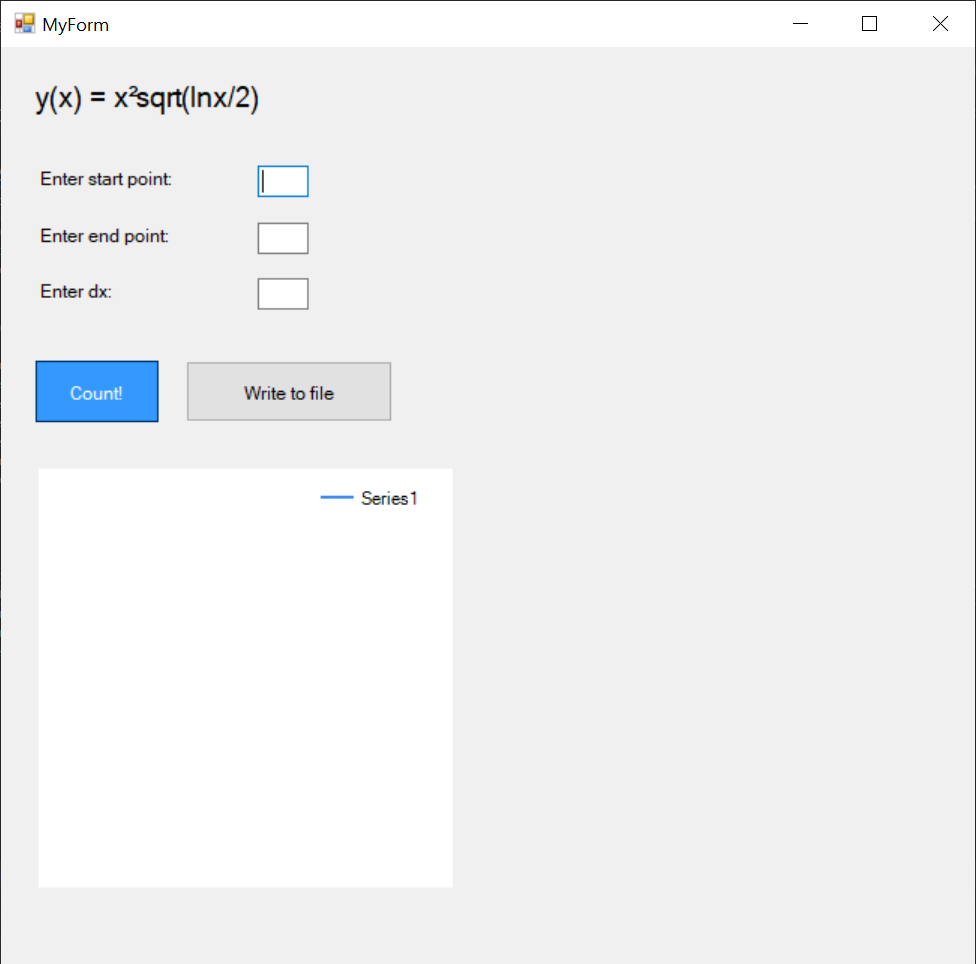


Рисунок 4.1 – Интерфейс приложения

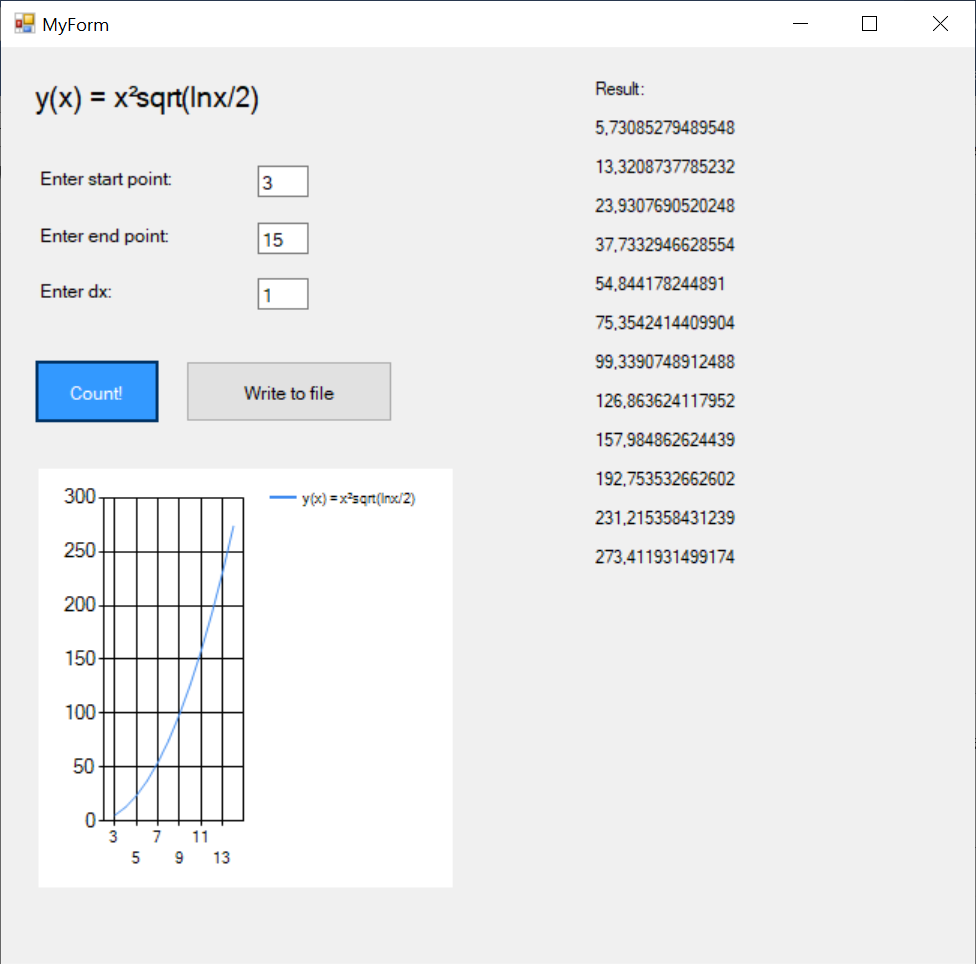


Рисунок 4.2 – Вид интерфейса после произведения расчётов

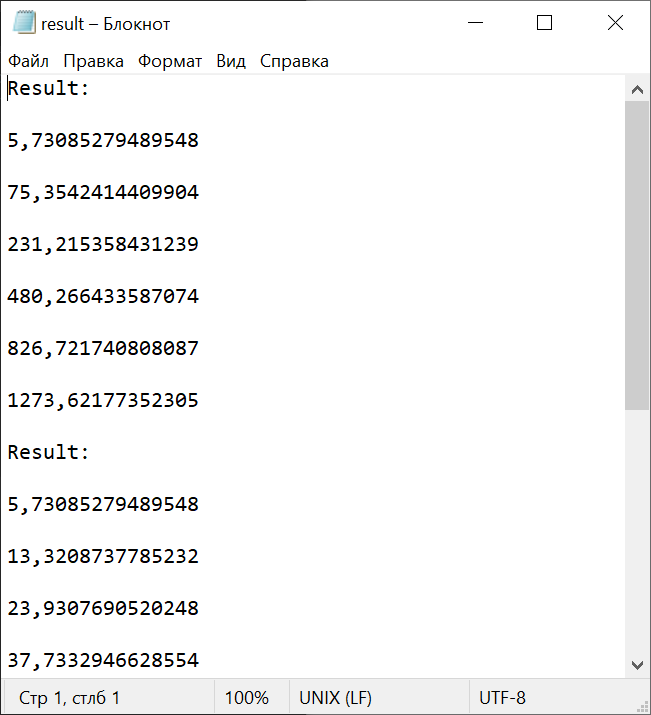


Рисунок 4.3 – Результат работы второй кнопки

# Заключение

В рамках данной курсовой работы была поставлена задача составить программы табулирования заданных функций в заданных интервалах с заданной точностью.

В результате проектирования был составлен алгоритм для решения поставленной задачи. Далее он был конкретизирован для реализации с помощью различных парадигм программирования. В конце работы был произведён анализ полученных результатов и сделаны необходимые выводы.

Итогом работы можно считать созданный программный продукт, позволяющий решить данную задачу. Стоит указать, что программа имеет интуитивно понятный интерфейс, что помогает пользователю использовать программу с наибольшей результативностью. Разработанный алгоритм может служить органической частью решения более сложных задач.

Цель, поставленная в начале работы, достигнута, задачи выполнены.

# Список литературы

1. Бочарова Т. А., Бегункова Н. О. Основы алгоритмизации. - Хабаровск: Издательство ТОГУ, 2011. - 64 с.
2. Сергеев С. Ф., Падерно П. И., Назаренко Н. А. Введение в проектирование интеллектуальных интерфейсов: Учебное пособие. – СПб.: СПбГУ ИТМО, 2011. - 108 с.
3. Спицина И. А., Аксёнов К. А. ПРИМЕНЕНИЕ СИСТЕМНОГО АНАЛИЗА ПРИ РАЗРАБОТКЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ИНФОРМАЦИОННЫХ СИСТЕМ: Учебное пособие. - Екатеринбург: Издательство Уральского университета, 2018. - 100 с.
4. Леоненков А. В. Самоучитель UML 2. — СПб.: БХВ-Петербург, 2007. — 576 с.
5. Назаров С.В., Белоусова С.Н., Бессонова И.А., Гиляревский Р.С., Гудыно Л.П. Введение в программные системы и их разработку. - М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2012. - 456 с.
6. Наместников С.М. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ С++: Учебное пособие. - Ульяновск: УлГТУ, 2007. - 136 с.
7. Радченко, Г.И. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. Конспект лекций / Г.И. Радченко, Е.А. Захаров. – Челябинск : Издательский центр ЮУрГУ, 2013. – 167 с.
8. C++ OOP // W3Schools.com URL: <https://www.w3schools.com>
9. Difference between Structured Programming and Object Oriented Programming // GEEKSFORGEEKS URL: <https://www.geeksforgeeks.org>
10. Structured Programming // Rebus community URL: <https://press.rebus.community>
11. UML для самых маленьких: диаграмма классов // Habr URL: <https://habr.com/ru/post/511798/>
12. UML: первое знакомство / Бабич А.В. – М.: Национальный Открытый Университет «ИНТУИТ», 2016 (Основы информационных технологий).

# Приложение А.

Содержание файла Function.cpp

#include <iostream>

#include <cmath>

#include <fstream>

#include <string>

#include <exception>

using namespace std;

struct Function {

double\* start;

double\* end;

double\* t;

double\* y;

};

int String\_Counter() {

int n = 0;

ifstream f("file.txt");

string str = "";

while (getline(f, str)) {

n++;

}

f.close();

return n;

}

void Create\_arrays(struct Function\* p, int n) {

p->start = new double[n];

p->end = new double[n];

p->t = new double[n];

p->y = new double[n];

}

void F(struct Function\* p, int n) {

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

for (double i = p->start[j]; i < p->end[j]; i += p->t[j]) {

if (i <= 2) {

throw exception("Wrong value");

break;

}

p->y[j] = pow(i, 2) \* sqrt(log(i / 2));

cout << p->y[j] << '\n';

}

cout << endl;

}

}

int main() {

Function\* p = new Function();

int n = String\_Counter();

Create\_arrays(p, n);

ifstream in;

in.open("file.txt");

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

in >> p->start[i] >> p->end[i] >> p->t[i];

}

in.close();

int number;

do {

cout << "1. Make calculations" << endl;

cout << "0. Exit" << endl;

cout << "Number of command: "; cin >> number;

switch (number) {

case 1:

try {

F(p, n);

}

catch (exception err) {

cout << err.what() << endl;

}

break;

case 0: break;

default: cout << "Command undefined\n\n";

break;

}

} while (number != 0);

system("pause");

}

# Приложение Б.

Содержание файла Function.h

#pragma once

#include <iostream>

#include <cmath>

#include <fstream>

#include <string>

#include <exception>

using namespace std;

class Base {

public:

virtual void F(int) = 0;

};

class Function: public Base

{

private:

double\* y;

public:

double\* start;

double\* end;

double\* t;

Function(int n) {

start = new double[n];

end = new double[n];

t = new double[n];

y = new double[n];

}

Function() {

}

virtual void F(int n) {

ofstream out;

out.open("result.txt");

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

for (double i = start[j]; i < end[j]; i += t[j]) {

if (i <= 2) {

throw exception("Wrong value");

}

y[j] = pow(i, 2) \* sqrt(log(i / 2));

out << y[j] << '\n';

}

out << endl;

}

out.close();

}

};

class File {

private:

string path;

public:

File(string path) {

this->path = path;

}

int String\_Counter() {

int n = 0;

ifstream f(path);

string str = "";

while (getline(f, str)) {

n++;

}

f.close();

return n;

}

Function\* CreateFunction() {

int n = this->String\_Counter();

Function\* p = new Function(n);

ifstream in;

in.open(path);

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

in >> p->start[i] >> p->end[i] >> p->t[i];

}

in.close();

return p;

}

};

Содержание файла Function.cpp

#include "Function.h"

#include <iostream>

using namespace std;

int main() {

File f("file.txt");

try {

f.CreateFunction()->F(f.String\_Counter());

}

catch (exception err) {

cout << err.what() << endl;

}

}

# Приложение В.

Файл MyForm.h

#pragma once

#include <cmath>

#include <fstream>

#include <string>

#include <vector>

namespace CourseProjectWinForms {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Windows::Forms::DataVisualization::Charting;

/// <summary>

/// Сводка для MyForm

/// </summary>

private value class MyFunction{

private:

double y;

public:

double start;

double end;

double t;

MyFunction(double Start, double End, double T) {

start = Start;

end = End;

t = T;

}

void F(std::vector<double>\* vecX, std::vector<double>\* vecY) {

for (double i = start; i < end; i += t) {

if (i <= 2) {

break;

}

vecX->push\_back(i);

vecY->push\_back(Get\_Value(i));

}

}

double Get\_Value(double x) {

return (pow(x, 2) \* sqrt(log(x / 2)));

}

};

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::DataVisualization::Charting::Chart^ chart1;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Legend^ legend1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Legend());

System::Windows::Forms::DataVisualization::Charting::Series^ series1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

this->label1 = (gcnew System::Windows::Forms::Label());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->label2 = (gcnew System::Windows::Forms::Label());

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->label4 = (gcnew System::Windows::Forms::Label());

this->label5 = (gcnew System::Windows::Forms::Label());

this->button2 = (gcnew System::Windows::Forms::Button());

this->chart1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->BeginInit();

this->SuspendLayout();

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(23, 79);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(142, 22);

this->label1->TabIndex = 0;

this->label1->Text = L"Enter start point:";

//

// textBox1

//

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(171, 79);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(34, 28);

this->textBox1->TabIndex = 1;

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label2->Location = System::Drawing::Point(23, 117);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(137, 22);

this->label2->TabIndex = 2;

this->label2->Text = L"Enter end point:";

//

// textBox2

//

this->textBox2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox2->Location = System::Drawing::Point(171, 117);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(34, 28);

this->textBox2->TabIndex = 3;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(23, 154);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(82, 22);

this->label3->TabIndex = 4;

this->label3->Text = L"Enter dx:";

//

// textBox3

//

this->textBox3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox3->Location = System::Drawing::Point(171, 154);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(34, 28);

this->textBox3->TabIndex = 5;

//

// button1

//

this->button1->BackColor = System::Drawing::SystemColors::MenuHighlight;

this->button1->Cursor = System::Windows::Forms::Cursors::Hand;

this->button1->FlatAppearance->BorderColor = System::Drawing::SystemColors::MenuHighlight;

this->button1->FlatAppearance->BorderSize = 0;

this->button1->FlatStyle = System::Windows::Forms::FlatStyle::Popup;

this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->ForeColor = System::Drawing::SystemColors::ButtonHighlight;

this->button1->Location = System::Drawing::Point(23, 209);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(82, 41);

this->button1->TabIndex = 6;

this->button1->Text = L"Count!";

this->button1->UseVisualStyleBackColor = false;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

//

// label4

//

this->label4->AutoSize = true;

this->label4->Location = System::Drawing::Point(393, 21);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(0, 13);

this->label4->TabIndex = 7;

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label5->Location = System::Drawing::Point(19, 21);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(240, 32);

this->label5->TabIndex = 8;

this->label5->Text = L"y(x) = x²sqrt(lnx/2)";

//

// button2

//

this->button2->BackColor = System::Drawing::SystemColors::GradientActiveCaption;

this->button2->Cursor = System::Windows::Forms::Cursors::Hand;

this->button2->FlatAppearance->BorderColor = System::Drawing::SystemColors::MenuHighlight;

this->button2->FlatAppearance->BorderSize = 0;

this->button2->FlatStyle = System::Windows::Forms::FlatStyle::System;

this->button2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button2->ForeColor = System::Drawing::SystemColors::ButtonHighlight;

this->button2->Location = System::Drawing::Point(123, 209);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(138, 41);

this->button2->TabIndex = 9;

this->button2->Text = L"Write to file";

this->button2->UseVisualStyleBackColor = false;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

//

// chart1

//

chartArea1->Name = L"ChartArea1";

this->chart1->ChartAreas->Add(chartArea1);

legend1->Name = L"Legend1";

this->chart1->Legends->Add(legend1);

this->chart1->Location = System::Drawing::Point(25, 281);

this->chart1->Name = L"chart1";

this->chart1->Palette = System::Windows::Forms::DataVisualization::Charting::ChartColorPalette::None;

series1->ChartArea = L"ChartArea1";

series1->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;

series1->Legend = L"Legend1";

series1->Name = L"Series1";

this->chart1->Series->Add(series1);

this->chart1->Size = System::Drawing::Size(276, 279);

this->chart1->TabIndex = 10;

this->chart1->Text = L"chart1";

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(649, 611);

this->Controls->Add(this->chart1);

this->Controls->Add(this->button2);

this->Controls->Add(this->label5);

this->Controls->Add(this->label4);

this->Controls->Add(this->button1);

this->Controls->Add(this->textBox3);

this->Controls->Add(this->label3);

this->Controls->Add(this->textBox2);

this->Controls->Add(this->label2);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->label1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;

this->Name = L"MyForm";

this->Text = L"MyForm";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

std::vector<double>\* vecY = new std::vector<double>();

std::vector<double>\* vecX = new std::vector<double>();

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

double start;

start = Convert::ToInt32(this->textBox1->Text);

double end;

end = Convert::ToInt32(this->textBox2->Text);

double dx;

dx = Convert::ToInt32(this->textBox3->Text);

MyFunction^ func = gcnew MyFunction(start, end, dx);

func->F(vecX, vecY);

this->label4->Text = "";

String^ str = "Result:\n\n";

for (std::vector<double>::iterator iter = vecY->begin(); iter != vecY->end(); iter++) {

str += (\*iter).ToString() + "\n\n";

}

this->label4->Text = str;

chart1->Series->Clear();

Series^ series = gcnew Series(L"y(x) = x²sqrt(lnx/2)");

series->IsVisibleInLegend = true;

series->IsXValueIndexed = true;

series->ChartType = SeriesChartType::Line;

chart1->Series->Add(series);

for (std::vector<double>::iterator iter = vecX->begin(); iter != vecX->end(); iter++) {

series->Points->AddXY((\*iter), func->Get\_Value((\*iter)));

}

}

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

IO::StreamWriter^ sw = gcnew IO::StreamWriter("result.txt", true);

sw->Write(this->label4->Text);

sw->Close();

delete sw;

}

};

}

Файл MyForm.cpp

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void Main(array<String^>^ arg)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

CourseProjectWinForms::MyForm form;

Application::Run(% form);

}