МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тульский государственный университет»

Институт *прикладной математики и компьютерных наук*

Кафедра *вычислительной техники*

Курсовая работа по дисциплине

**СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ**

на тему: Реализация нерекурсивного алгоритма построения кривой Гильберта.

Студент группы 220681 Шайхаттаров Д.В. \_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (Подпись, дата)

Руководитель работы Доц. Афанасьева С.М. \_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О., должность) (Подпись, дата)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тула, 2020

**Содержание**

[Введение 3](#_Toc23950232)

[1. Постановка задачи на проектирование 3](#_Toc23950233)

[2. Проектирование 3](#_Toc23950234)

[3. Математическое обеспечение 4](#_Toc23950235)

[4. Алгоритмическое обеспечение 5](#_Toc23950236)

[5. Программная реализация 5](#_Toc23950237)

[6. Тестирование 6](#_Toc23950238)

[7. Инструкция программисту 9](#_Toc23950239)

[8. Инструкция пользователю 10](#_Toc23950240)

[9. Инструкция по установке 12](#_Toc23950241)

[Заключение 16](#_Toc23950242)

[Список использованных источников 17](#_Toc23950243)

[Приложения 18](#_Toc23950244)

# Введение

Целью выполнения курсовой работы является закрепление теоретических знаний по курсу «Структуры и алгоритмы обработки информации» и получения практических навыков самостоятельного проектирования ирактических я курсового проекта реализации основных элементов.

Задачами курсовой работы являются:

* приобретение навыков разработки и отладки на языке высокого уровня программных средств;
* практическое овладение основными принципами создания программных средств;
* практическое освоение технологий программирования на языке высокого уровня с использованием системных вызовов и библиотек стандартных подпрограмм;
* приобретение практических навыков оформления и выпуска документации в соответствии с ГОСТ.

# 1. Постановка задачи на проектирование

Необходимо разработать программное обеспечение, которое реализует кодирование строки с помощью алгоритма Хаффмана. При этом в работе должны быть разработаны:

* интерфейс пользователя;
* система авторизации;
* инсталляционный пакет;
* функциональная часть.

# 2. Проектирование

Структуру программы можно представить в виде диаграммы классов (рисунок 1).



Рисунок 1. Диаграмма классов

# 3. Математическое обеспечение

# Эти кривые связаны с понятием теории функций, а именно — всюду плотными кривыми . Кривая на плоскости называется всюду плотной в некоторой области, если она проходит через любую сколь угодно малую окрестность каждой точки этой области. Несколько упрощенно можно считать, что всюду плотные кривые целиком заполняют указанную область. Известные математики Гильберт и Серпинский построили примеры всюду плотных кривых. Хотя эти примеры различны, схема получения соответствующих кривых одинакова. По определенному правилу строятся кривые (соответственно Гильберта и Серпинского) первого, второго, ..., n-го порядка, вписанные в заданный квадрат. При неограниченном возрастании n они стремятся к некоторой предельной кривой, которая является всюду плотной в заданном квадрате.

# 4. Алгоритмическое обеспечение

Нерекурсивный алгоритм построения кривой Гильберта реализуется следующим образом:

1. Вычисляем длину шага исходя из размера нашего поля и порядка кривой.
2. Записываем последовательно 4 точки для построения кривой первого порядка в список.
3. Транспонируем и перемещаем точки из этого списка во второй список для получения кривой второго порядка.
4. Перезаписываем точки первого списка точками из второго .
5. Повторяем пункт 3,4 в зависимости от нужного нам порядка кривой.
6. Соединяем последовательно точки из списка.

# 5. Программная реализация

Основные компоненты программы реализованы следующим образом:

1. Метод построения кривой гильберта, который находится внутри класса Graph:

List<Point> p = **new** List<Point>();

**float** l = h;

p.Add(**new** Point(x\_start, y\_start));

p.Add(**new** Point(x\_start, y\_start - h));

p.Add(**new** Point(x\_start + h, y\_start - h));

p.Add(**new** Point(x\_start + h, y\_start));

**for** (**int** i = 0; i < level - 1; i++)

{

List<Point> new\_p = **new** List<Point>();

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(-p[j].y + y\_start, -p[j].x + y\_start));

}

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(p[j].x, p[j].y - l - h));

}

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(p[j].x + l + h, p[j].y - l - h));

}

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(p[j].y + 2 \* l + h - y\_start, p[j].x + y\_start - l));

}

p = new\_p;

l = l \* 2 + h;

}

**for** (**int** i = 1; i < p.Count; i++)

{

DrawLine(graphics, pen, p[i - 1], p[i]);

}

1. Класс Point:

**public** **class** Point

{

**public** **float** x;

**public** **float** y;

**public** Point(**float** x, **float** y)

{

**this**.x = x;

**this**.y = y;

}

}

1. Получение из формы порядок кривой, заданный пользователем, и последующий запуск отрисовки кривой гильберта:

**private** **void** start\_button\_Click(**object** sender, EventArgs e)

{

**int** i = Convert.ToInt32(txt\_level.Text);

Graph\_space space = **new** Graph\_space(**this**);

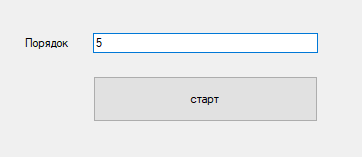
Graph graph = **new** Graph(i,space);

graph.Draw();

}

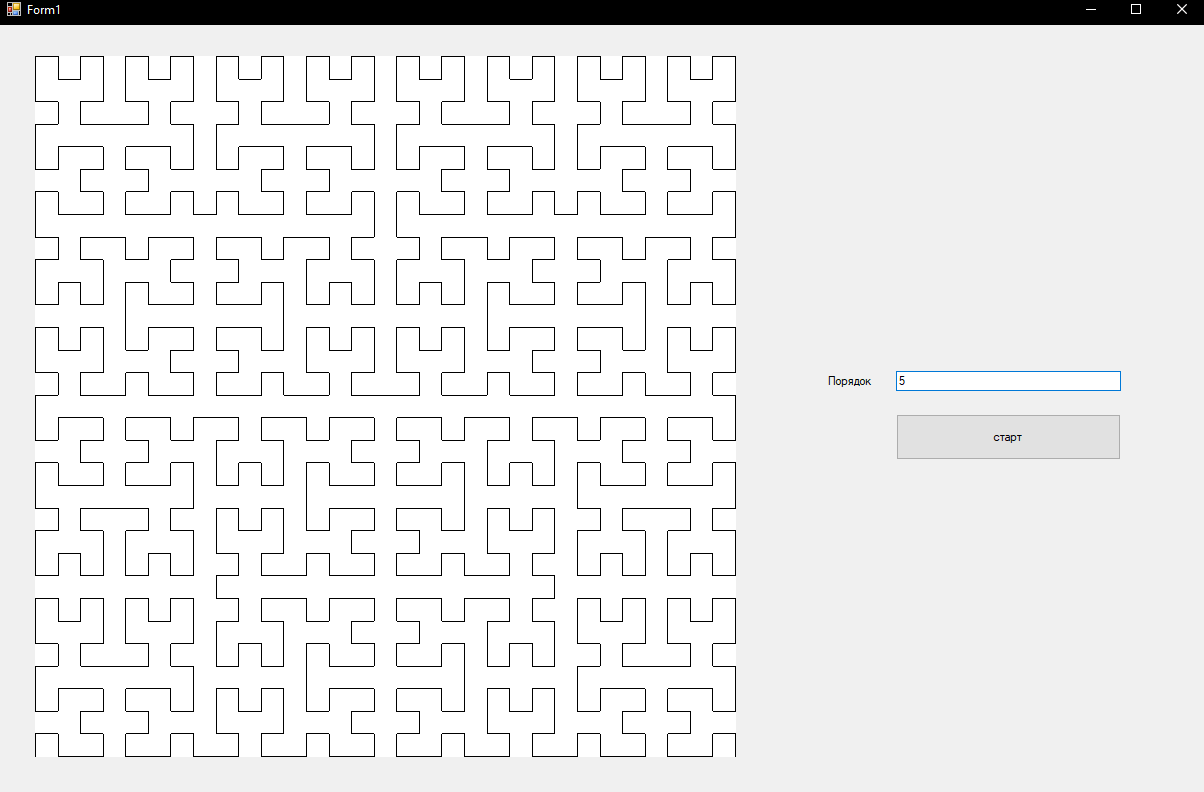
# 6. Тестирование

Введём в поле ввода данных число 5 (Рисунок 2).



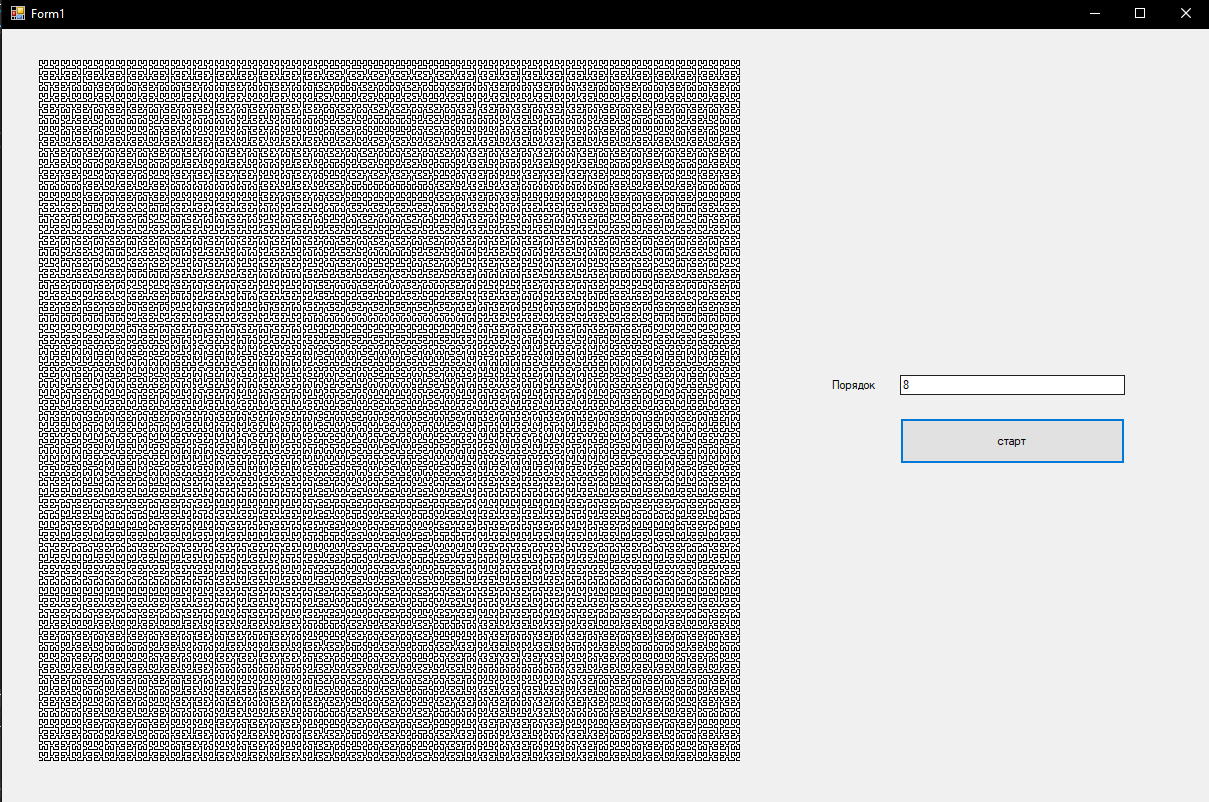
## Рисунок 2. Поле ввода и кнопка, запускающая построение кривой гильберта

Запустим программу, которая выполнит кодирование (Рисунок 3).



## Рисунок 3. Результат работы программы

Теперь введём число 8 и снова нажмём на кнопку «старт» (Рисунок 4).

 Рисунок 4. Результат работы программы

# 7. Инструкция программисту

1. Содержимое программы

После установки программы в её папке будет содержаться три файла: WindowsFormsApp5.exe – исполнительный файл, который запускает программу и unins.exe, необходимый для удаления программы.

1. Условия применения программы

Для работы программы необходимо наличие на компьютере библиотек .Net Framework 4 версии и выше. Компьютер должен иметь следующие минимальные системные требования:

* 32-разрядный (x86) или 64-разрядный (x64) процессор с тактовой частотой 1 ГГц или выше.
* 1 ГБ (для 32-разрядного процессора) или 2 ГБ (для 64-разрядного процессора) ОЗУ.
* Графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии.

1. Характеристики программы

Установленная программа занимает на диске 2.4 МБ. Время работы программы составляет не более одной секунды.

1. Обращение к программе

Программу можно запустить как с помощью ярлыка на рабочем столе, так и с помощью исполнительного файла WindowsFormsApp5.exe, находящегося в папке программы.

1. Удаление программы

Для удаления программы необходимо зайти в панель управления и выбрать пункт «Удаление программы». Далее из списка программ необходимо выбрать необходимую и нажать кнопку «Удалить». Далее возникнет окно, где необходимо подтвердить удаление программы.

# 8. Инструкция пользователю

При запуске программы возникнут текстовые поля авторизации (Рисунок 5), куда нужно ввести логин и пароль. После ввода данных нужно нажать на кнопку «Вход» или «Зарегистрироваться». Остальные элементы управления при этом будут недоступны.

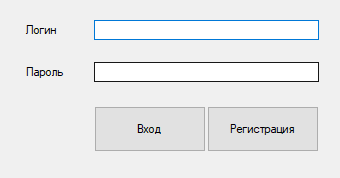


Рисунок 5. Поля авторизации

После входа поля авторизации скроются, покажется кнопка, запускающая работу алгоритма, и станет доступно для редактирования текстовое поле для входных данных (Рисунок 6).

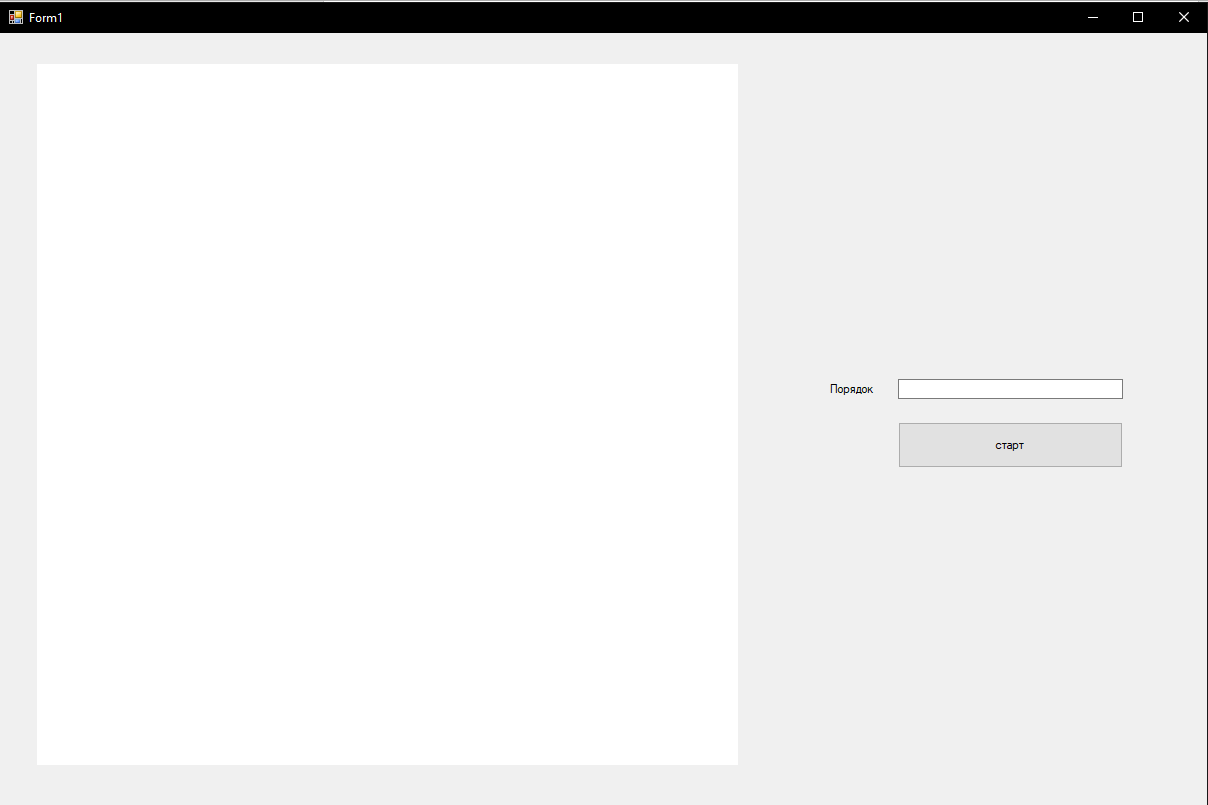


Рисунок 6. Основное окно программы

Чтобы выполнить построение кривой, необходимо ввести порядок и нажать кнопку «старт». После этого кривая будет построена (Рисунок 7).

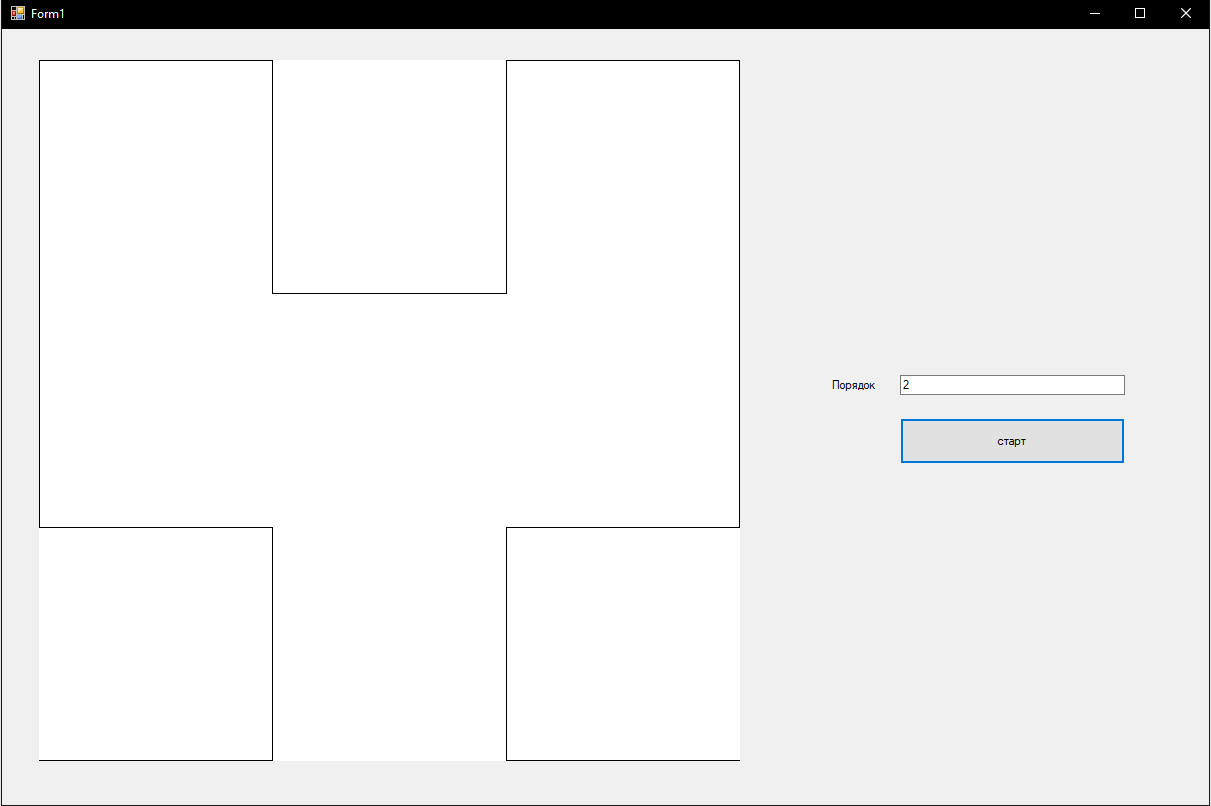


Рисунок 7. Пример работы программы

# 9. Инструкция по установке

Для установки программы необходимо запустить инсталлятор, который называется «S&A.exe». При его запуске возникнет следующее окно (Рисунок 8):

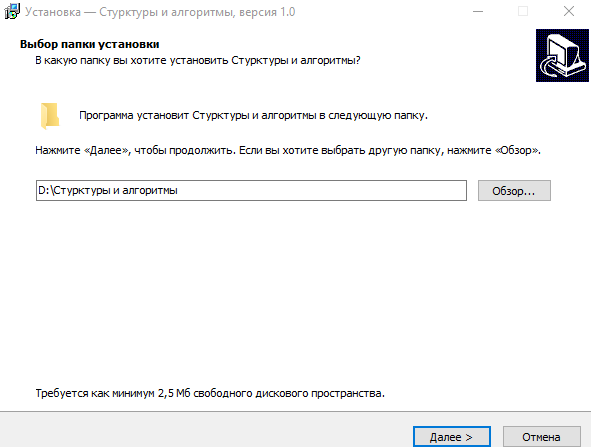


Рисунок 8. Окно, возникающее при запуске инсталлятора

На данном этапе есть возможность отменить установку с помощью кнопки «Отмена». Необходимо ввести путь куда будет установлена программа.

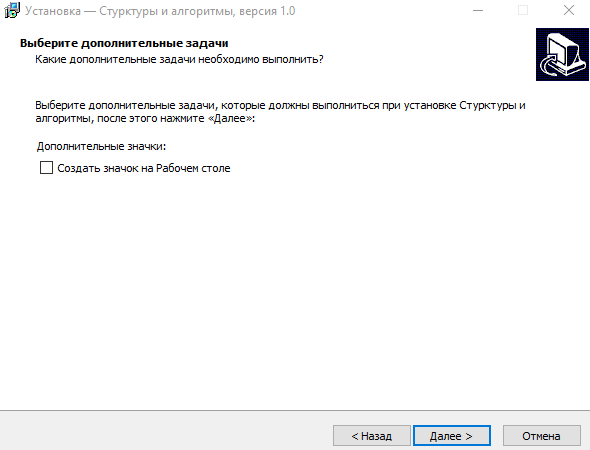


Рисунок 9. Окно, где можно указать необходимость создания ярлыка на рабочем столе.

На данном этапе есть возможность отменить установку с помощью кнопки «Отмена» или вернуться к прошлому окну с помощью кнопки «Назад». Здесь вы можете указать необходимость создания ярлыка на рабочем столе, после чего нажмите далее для того, чтобы перейти к следующему окну.(Рисунок 10):

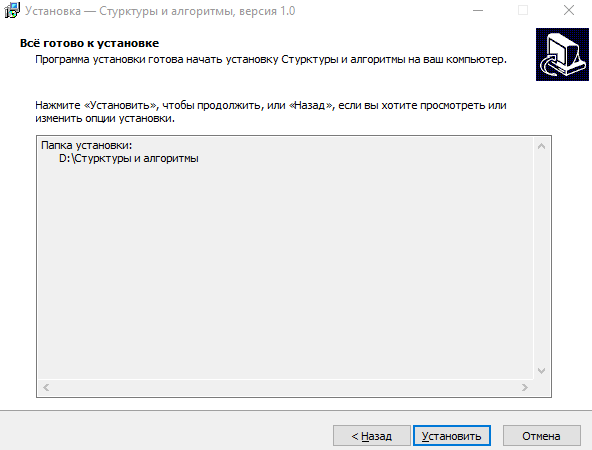


Рисунок 10. Окно подтверждения установки

На данном этапе есть возможность отменить установку с помощью кнопки «Отмена» или вернуться к прошлому окну с помощью кнопки «Назад». Для дальнейшей установки необходимо нажать на кнопку «Установить». После этого начнётся установка программы по заданному вами пути по завершении которой вы перейдёте к следующему окну (Рисунок 11):

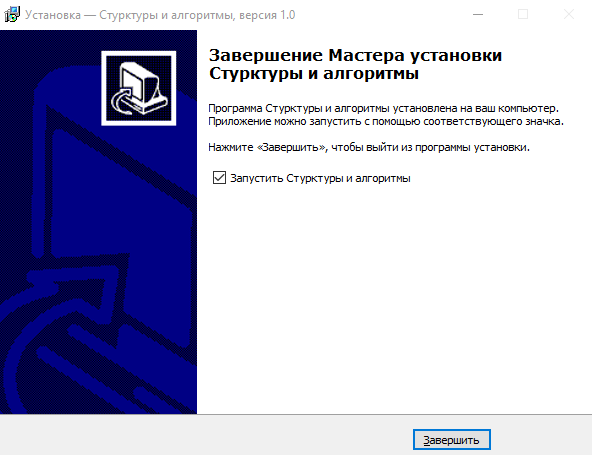


Рисунок 11. Окно, сообщающее об успешной установке программы

Теперь нужно нажать на кнопку «Завершить», чтобы закрыть это окно, также в этом окне вы можете выбрать запускать ли программу после закрытия окна. После завершения установки на рабочем столе возникнет ярлык «Структуры и алгоритмы», при нажатии на который запустится установленная программа.

# Заключение

В результате выполнения курсовой работы все поставленные цели были достигнуты, задачи решены в полном объёме, профессиональные навыки по разработке и отладке на языке высокого уровня программных средств были приобретены. Также на практике освоены основные принципы создания программных средств и технологии программирования на языке высокого уровня с использованием системных вызовов и библиотек стандартных подпрограмм.

За время выполнения курсовой работы были реализованы: интерфейс пользователя, система авторизации, инсталляционный пакет, функциональная часть.

# Список использованных источников

1. Metanit, Руководство по программированию в Windows Forms [Электронный ресурс]. - URL: <https://www.metanit.com/sharp/windowsforms/>
2. Кривые Гильберта и Серпинского [Электронный ресурс]. - URL: <https://inf.1sept.ru/1999/art/zlat1.htm>

# Приложения

**Приложение 1**

**Исходный код файла Form1.cs**

**using** System;

**using** System.IO;

**using** System.Windows.Forms;

**namespace** WindowsFormsApp5

{

**public** partial **class** Form1 : Form

{

**public** Form1()

{

InitializeComponent();

}

**private** **void** Form1\_Load(**object** sender, EventArgs e)

{

txt\_pass.PasswordChar = '\*';

}

**private** **void** reg\_button\_Click(**object** sender, EventArgs e)

{

**if** (txt\_login.Text != "" && txt\_pass.Text != "")

{

**using** (StreamWriter sw = **new** StreamWriter("log.txt", **true**, System.Text.Encoding.Default))

{

sw.Write(txt\_login.Text + " " + txt\_pass.Text + "\n");

}

}

}

**private** **void** login\_button\_Click(**object** sender, EventArgs e)

{

**if** (txt\_login.Text != "" && txt\_pass.Text != "")

{

**using** (StreamReader sr = **new** StreamReader("log.txt", System.Text.Encoding.Default))

{

**string** line;

**while** ((line = sr.ReadLine()) != **null**)

{

**string**[] words = line.Split(' ');

**if** (words[0] == txt\_login.Text && words[1] == txt\_pass.Text)

{

login\_button.Hide();

reg\_button.Hide();

txt\_login.Hide();

txt\_pass.Hide();

l\_pass.Hide();

l\_log.Hide();

l\_level.Show();

txt\_level.Show();

start\_button.Show();

}

}

}

}

}

**private** **void** start\_button\_Click(**object** sender, EventArgs e)

{

**int** i = Convert.ToInt32(txt\_level.Text);

Graph\_space space = **new** Graph\_space(**this**);

Graph graph = **new** Graph(i,space);

graph.Draw();

}

**private** **void** txt\_level\_KeyPress(**object** sender, KeyPressEventArgs e)

{

**if** (!((e.KeyChar >= 49 && e.KeyChar <= 56) || e.KeyChar == 8))

{

e.Handled = **true**;

}

}

**private** **void** txt\_login\_KeyPress(**object** sender, KeyPressEventArgs e)

{

**if** (!((e.KeyChar >= 97 && e.KeyChar <= 122) || (e.KeyChar >= 65 && e.KeyChar <= 90) || (e.KeyChar >= 48 && e.KeyChar <= 57) || e.KeyChar == 8))

{

e.Handled = **true**;

}

}

**private** **void** txt\_pass\_KeyPress(**object** sender, KeyPressEventArgs e)

{

**if** (!((e.KeyChar >= 97 && e.KeyChar <= 122) || (e.KeyChar >= 65 && e.KeyChar <= 90) || (e.KeyChar >= 48 && e.KeyChar <= 57) || e.KeyChar == 8))

{

e.Handled = **true**;

}

}

}

}

**Приложение 2**

**Исходный код файла Graph\_space.cs**

**using** System.Drawing;

**namespace** WindowsFormsApp5

{

**class** Graph\_space

{

**public** Form1 formMain;

**public** Graphics graphics;

Bitmap bmp;

**public** Graph\_space(Form1 p\_formMain)

{

formMain = p\_formMain;

bmp = **new** Bitmap(formMain.graph\_space.Width, formMain.graph\_space.Height);

graphics = Graphics.FromImage(bmp);

formMain.graph\_space.Image = bmp;

}

}

}

**Приложение 3**

**Исходный код файла Graph.cs**

**using** System;

**using** System.Collections.Generic;

**using** System.Drawing;

**namespace** WindowsFormsApp5

{

**class** Graph

{

Pen pen;

Graphics graphics;

Form1 form;

**float** h;

**float** x;

**float** y;

**int** i;

**public** Graph(**int** i, Graph\_space graph\_space)

{

**this**.i = i;

graphics = graph\_space.graphics;

pen = **new** Pen(Color.Black);

form = graph\_space.formMain;

h = (form.graph\_space.Width - 1) / (Convert.ToSingle(Math.Pow(2, i)) - 1);

x = 0;

y = (form.graph\_space.Width - 1);

}

**public** **void** Draw()

{

Hilbert(i, h, x, y, graphics, pen);

}

**private** **void** Hilbert(**int** level, **float** h, **float** x\_start, **float** y\_start, Graphics graphics, Pen pen)

{

List<Point> p = **new** List<Point>();

**float** l = h;

p.Add(**new** Point(x\_start, y\_start));

p.Add(**new** Point(x\_start, y\_start - h));

p.Add(**new** Point(x\_start + h, y\_start - h));

p.Add(**new** Point(x\_start + h, y\_start));

**for** (**int** i = 0; i < level - 1; i++)

{

List<Point> new\_p = **new** List<Point>();

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(-p[j].y + y\_start, -p[j].x + y\_start));

}

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(p[j].x, p[j].y - l - h));

}

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(p[j].x + l + h, p[j].y - l - h));

}

**for** (**int** j = 0; j < p.Count; j++)

{

new\_p.Add(**new** Point(p[j].y + 2 \* l + h - y\_start, p[j].x + y\_start - l));

}

p = new\_p;

l = l \* 2 + h;

}

**for** (**int** i = 1; i < p.Count; i++)

{

DrawLine(graphics, pen, p[i - 1], p[i]);

}

}

**private** **void** DrawLine(Graphics graphics, Pen pen, Point p1, Point p2)

{

graphics.DrawLine(pen, p1.x, p1.y, p2.x, p2.y);

}

}

}

**Приложение 4**

**Исходный код файла Point.cs**

**namespace** WindowsFormsApp5

{

**public** **class** Point

{

**public** **float** x;

**public** **float** y;

**public** Point(**float** x, **float** y)

{

**this**.x = x;

**this**.y = y;

}

}

}