МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет) \_\_\_\_\_\_\_\_\_Институт информационных технологий\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_МПО ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КУРСОВАЯ РАБОТА

по дисциплине Структурное программирование

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

на тему Программирование на языке высокого уровня

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

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

Выполнил студент группы

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

1ПИб-02-1оп-22

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

направления подготовки (специальности)

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

Программная инженерия

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

*шифр, наименование*

\_\_\_\_Микуцких Григорий Андреевич\_\_\_\_

*фамилия, имя, отчество*

Руководитель

Пышницкий Константин Михайлович

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

*фамилия, имя, отчество*

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

старший преподаватель

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

*должность*

Дата представления работы

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г.

Заключение о допуске к защите

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

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

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

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

количество баллов

Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Череповец, \_2023\_

*год*

# АННОТАЦИЯ

Курсовая работа посвящена программированию на языке высокого уровня и реализации шифрования по методу построения блочных шифров сети Фейстеля. В работе описывается метод шифрования и его реализация на языке программирования C++. Для разработки программы использовалась интегрированная среда разработки Visual Studio. Были получены консольное приложение для кодирования и декодирования сообщения, а также результаты тестирования, которые подтверждают корректность его работы.

Блок – строка бит определённой длины [1].

Блочный шифр – шифр из класса симметричных криптографических методов, в котором алгоритм зашифрования применяется к блокам открытого текста для получения блоков шифртекста [1].

Зашифрование – обратимое преобразование данных с помощью шифра, которое формирует шифртекст из открытого текста [1].

Ключ – изменяемый параметр в виде последовательности символов, определяющий криптографическое преобразование [1].

Криптография – наука о методах обеспечения конфиденциальности информации, её защиты путём преобразования (шифрования).

Криптограф – специалист в области криптографии.

Криптоанализ – наука о методах дешифровки, исследующая возможности взлома шифрованной информации без ключа, а также сам процесс дешифровки.

Криптоаналитик – специалист криптоанализа.

Открытый текст – незашифрованная информация [1].

Расшифрование – операция, обратная зашифрованию [1].

Раунд – один из последовательных шагов обработки данных в алгоритме блочного шифрования. В шифрах Фейстеля (построенных в соответствии с архитектурой сети Фейстеля) и близких ему по архитектуре шифрах – один шаг шифрования, в ходе которого одна или несколько частей шифруемого блока данных подвергается модификации путём применения круговой функции [3].

Шифр – криптографический метод, используемый для обеспечения конфиденциальности данных, включающий алгоритм зашифрования и алгоритм расшифрования [1].

Шифртекст – данные, полученные в результате зашифрования открытого текста с целью скрытия его содержания [1].

Расчётно-пояснительная записка содержит 8 таблиц, 4 иллюстраций, 3 приложения и 8 использованных первоисточников.

ОГЛАВЛЕНИЕ

[АННОТАЦИЯ 2](#_Toc138076286)

[ВВЕДЕНИЕ 4](#_Toc138076287)

[1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc138076288)

[2. ОПИСАНИЕ КЛАССОВ GRAPHICS, PEN И BRUSH 7](#_Toc138076289)

[3. ОПИСАНИЕ СОЗДАННОГО ПРИЛОЖЕНИЯ 13](#_Toc138076290)

[3.1 Постановка задачи 13](#_Toc138076291)

[3.2 Логическое проектирование – алгоритм работы программы 13](#_Toc138076292)

[3.3 Физическое проектирование – структуры данных и спецификация функций 15](#_Toc138076293)

[3.4 Тестирование 18](#_Toc138076294)

[3.5 Результаты работы 21](#_Toc138076295)

[ЗАКЛЮЧЕНИЕ 22](#_Toc138076296)

[СПИСОК ЛИТЕРАТУРЫ 23](#_Toc138076297)

[ПРИЛОЖЕНИЕ 1. Техническое задание 24](#_Toc138076298)

[ПРИЛОЖЕНИЕ 2. Руководство пользователя 30](#_Toc138076299)

[ПРИЛОЖЕНИЕ 3. Программный код 32](#_Toc138076300)

# ВВЕДЕНИЕ

В наше время программирование на языках высокого уровня является одной из наиболее распространенных и востребованных специализаций. С помощью языков высокого уровня можно написать web-приложения и сайты, мобильные приложения, разрабатывать игры, работать с базами данных и искусственным интеллектом, управлять системами, сетями и прочее. В рамках курсовой работы будет разработана программа, которая реализует метод блочного шифрования сеть Фейстеля.

Цель: разработка программы на языке высокого уровня.

Задача: написать программу для шифрования сообщения сетью Фейстеля.

План решения: реализовать метод сети Фейстеля для шифрования сообщения в визуальном приложении Windows Forms в среде CLR, используя язык программирования C++.

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Сеть Фейстеля — это метод блочного шифрования, разработанный Хорстом Фейстелем в лаборатории IBM в 1971 году. Метод обладает свойством инволюции – шифратор является дешифратором. Данное свойство определило его популярность на несколько десятилетий – сегодня сеть Фейстеля лежит в основе большого количества криптографических протоколов.

Схема работы метода представлена на рис. 1.

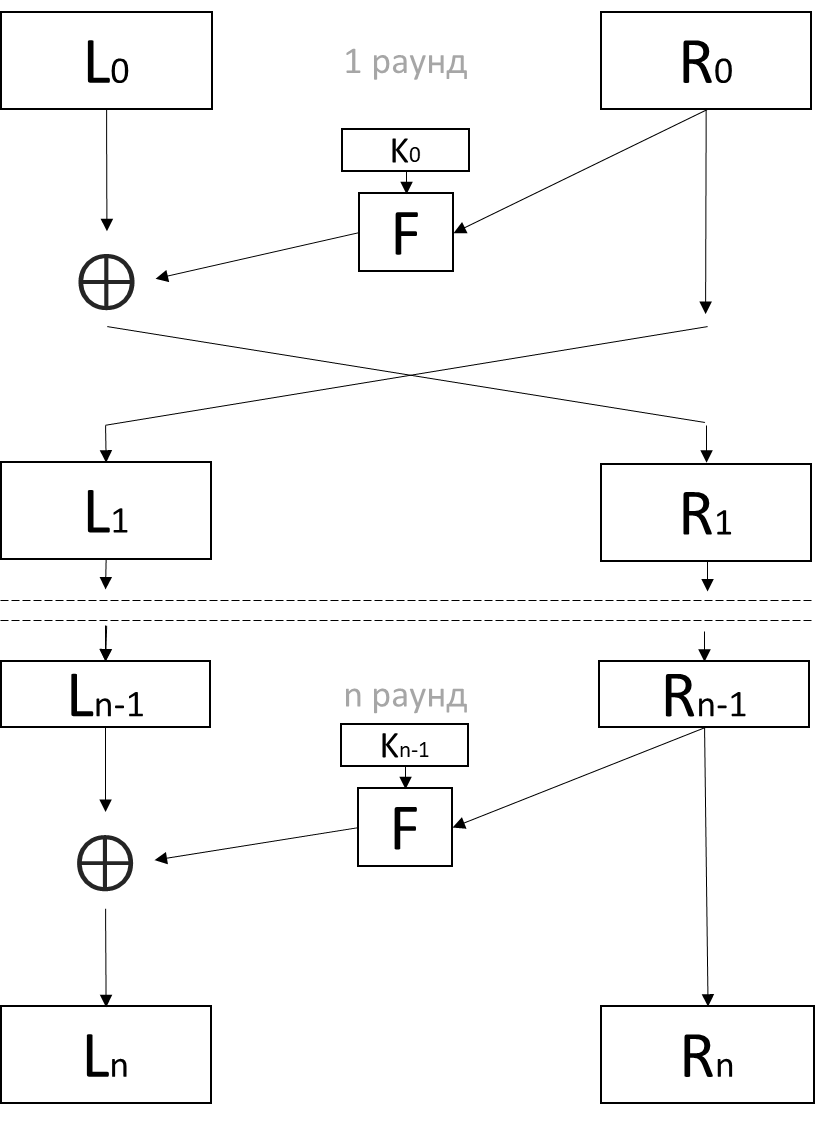


Рис. 1. Схема работы метода.

Такая схема называется ячейкой Фейстеля. Полученные на выходе первой ячейки подблоки поступают на вход второй ячейки, результирующие подблоки из второй ячейки попадают на вход третьей ячейки и так далее в зависимости от количества раундов [4]. В каждом таком раунде применяется заранее определенный раундовый ключ, который чаще всего получен из основного ключа K (например ключ 256-битный, а подблок 32-битный: ключ делится на 8 частей, каждая из которых применяется в определённом порядке).

Алгоритм метода:

* вся информация разбивается на блоки фиксированной длины. В случае, если длина входного блока меньше, чем размер, который шифруется заданным алгоритмом, то блок удлиняется каким-либо способом. Как правило длина блока является степенью двойки, например: 32 бита, 64 бита, 128 бит и т.д.;
* каждый блок делится на 2 равные части – и ;
* проводятся вычисления:

(1.1)

(1.2)

где – некоторая функция преобразования, – ключ первого раунда, – значение правого блока после первого раунда, – значение левого блока после первого раунда;

* вычисления проводятся n-1 раундов, в n раунде смена блоков местами не производится:

(1.3)

(1.4)

где – некоторая функция преобразования, – ключ n раунда.

Процесс, при котором предпринимается попытка узнать Х (сообщение), K (ключ) или и то, и другое, называется криптоанализом. Одной из возможных атак на алгоритм шифрования является лобовая атака, т.е. простой перебор всех возможных ключей. Если множество ключей достаточно большое, то подобрать ключ нереально. При длине ключа n бит количество возможных ключей равно 2n. Таким образом, чем длиннее ключ, тем более стойким считается алгоритм для лобовой атаки.

Существуют различные типы атак, основанные на том, что противнику известно определенное количество пар незашифрованное сообщение – зашифрованное сообщение. При анализе зашифрованного текста противник часто применяет статистические методы анализа текста. При этом он может иметь общее представление о типе текста, например, английский или русский текст, исходный текст на некотором конкретном языке программирования и т.д. Во многих случаях криптоаналитик имеет достаточно много информации об исходном тексте. Криптоаналитик может иметь возможность перехвата одного или нескольких незашифрованных сообщений вместе с их зашифрованным видом. Или криптоаналитик может знать основной формат или основные характеристики сообщения. Говорят, что криптографическая схема абсолютно безопасна, если зашифрованное сообщение не содержит никакой информации об исходном сообщении. Говорят, что криптографическая схема вычислительно безопасна, если:

* цена расшифровки сообщения больше цены самого сообщения;
* время, необходимое для расшифровки сообщения, больше срока жизни сообщения [5].

Сети Фейстеля были широко изучены криптографами в силу их обширного распространения. В 1988 году Майкл Люби (Michael Luby) и Чарльз Ракофф (Charles Rackoff) провели исследования сети Фейстеля и доказали, что если раундовая функция является криптостойкой псевдослучайной, и используемые ключи независимы в каждом раунде, то 3-х раундов будет достаточно для того, чтобы блочный шифр являлся псевдослучайной перестановкой, тогда как четырёх раундов будет достаточно для того чтобы сделать сильную псевдослучайную перестановку.

Псевдослучайной перестановкой Люби и Ракофф назвали такую, которая устойчива к атаке с адаптивным выбором открытого текста, а сильной псевдослучайной перестановкой — псевдослучайную перестановку устойчивую к атаке с использованием выбранного шифрованного текста.

Иногда сеть Фейстеля в западной литературе называют «Luby-Rackoff block cipher» в честь Люби и Ракоффа, которые проделали большой объём теоретических исследований в этой области.

В дальнейшем, в 1997 году, Мони Наор (Moni Naor) и Омер Рейнголд (Omer Reingold) предложили упрощённый вариант конструкции Люби – Ракоффа из четырёх раундов, где в качестве первого и последнего раунда используются две попарно-независимые перестановки. Два средних раунда конструкции Наора – Рейнголда идентичны раундам в конструкции Люби – Ракоффа [6].

Достоинства метода:

* простота аппаратной реализации на современной электронной базе;
* простота программной реализации в силу того, что значительная часть функций поддерживается на аппаратном уровне в современных компьютерах (например, сложение по модулю 2, сложение по модулю 2^n, умножение по модулю 2^n, и т. д.);
* хорошая изученность алгоритмов на основе сетей Фейстеля.

Из недостатков отмечают то, что за один раунд шифруется только половина входного блока [6].

# ОПИСАНИЕ КЛАССОВ GRAPHICS, PEN И BRUSH

1) Класс graphics.

Объект Graphics – это указатель на место, где будут рисоваться примитивы. Пусть мы хотим рисовать в форме Windows. Синтаксис задания ссылки на нее – Graphics g = Graphics.FromHwnd(this.Handle), где:

* Graphics – тип объекта;
* g – имя переменной;
* Graphics.FromHwnd(this.Handle) - используемый метод FromHwnd из класса Graphics, который задает ссылку Handle на форму Windows [7].

В С# инструменты рисования определены в пространстве имен System.Drawing. Там находятся классы Pen (перо) и Brush (кисть).

Объекты пера используются в методах рисования линий и контуров геометрических фигур.

Объекты кисти используются в методах заливки областей, ограниченных контурами [8].

Класс Graphics представляет GDI+ поверхность рисования. Графический объект поддерживает для поверхности рисования: масштаб, единицы, ориентацию поверхности рисования.

В С# инструменты рисования определены в пространстве имен System.Drawing.

При создании любого GDI-проекта:

* должно быть определено свойство g класса Graphics;
* необходимо создать обработчик события onPaint, в котором будет вызываться метод рисования Draw().

В методе Draw():

* методом CreateGraphics() необходимо создать объект g;
* методом Clear() очистить поверхность рисования;
* что-нибудь нарисовать;
* освободить память от объекта g.

Класс Graphics содержит много свойств. Часто используемые свойства перечислены в таблице ниже (табл. 1), большинство которых будет демонстрироваться позже. Все они позволяют не только читать себя, но и изменять [2].

*Таблица 1*

**Свойства Graphics**

|  |  |  |
| --- | --- | --- |
| Свойство | Класс | Описание |
| Clip | Region | Возвращает или задает объект Region, ограничивающий область рисования данного объекта |
| DpiX / DpiY | Float / single | Задает значение горизонтального и вертикального разрешения в точках на дюйм |
| PageScale | Float / single | Задает значение для масштабирования между универсальными единицами и единицами страницы |
| PageUnit | GraphicsUnit | Возвращает или задает единицу измерения для координат страницы данного объекта |

Свойство PageScale устанавливает масштабирование между мировыми единицами и единицами страницы [8].

Методы класса представлены в табл. 2.

*Таблица 2*

**Методы Graphics**

|  |  |  |
| --- | --- | --- |
| Имя метода | Описание | Число перегрузок |
| 1 | 2 | 3 |

*Продолжение табл. 2*

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Clear(Color) | Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона. | 1 |
| CopyFromScreen(Point, Point, Size) | Выполняет передачу данных о цвете, соответствующих прямоугольной области пикселей, блоками битов с экрана на поверхность рисования объекта Graphics. | 4 |
| Dispose() | Освобождает все ресурсы, используемые данным объектом Graphics. | 1 |
| DrawArc(Pen, Rectangle, Single, Single) | Рисует дугу, которая является частью эллипса, заданного структурой Rectangle. | 4 |
| DrawBezier(Pen, Point, Point, Point, Point) | Рисует кривую Безье, определяемую четырьмя структурами Point. | 3 |
| DrawBeziers(Pen, Point[]) | Рисует несколько (N) кривых Безье, определяемых массивом из (3N+1) структур Point. | 2 |
| DrawCloseCurve(Pen, Point[ ]) | Рисует замкнутый фундаментальный сплайн | 4 |
| DrawEllipse(Pen, Rectangle) | Рисует эллипс | 4 |
| DrawIcon(Icon, Rectangle) | Рисует значок | 2 |
| DrawImage(Image image, int x, int y) | Рисует заданное изображение image, используя его фактический размер в месте с координатами (x,y) | 30 |
| DrawLine(Pen, Point, Point) | Проводит линию, соединяющую две структуры Point. | 4 |
| DrawLines(Pen, Point[ ]) | Рисует набор сегментов линий, которые соединяют массив структур Point. | 2 |
| DrawPath(Pen, gp) | Рисует пером Pen объект GraphicsPath gp. | 1 |
| DrawPie(Pen, Rectangle, Single, Single) | Рисует сектор, который определяется эллипсом, заданным структурой Rectangle и двумя радиалtьными линиями. | 4 |
| DrawPolygon(Pen, Point[]) | Рисует многоугольник, определяемый массивом структур Point. | 2 |
| DrawRectangle(Pen, Rectangle) | Рисует прямоугольник, определяемый структурой Rectangle. | 3 |

*Продолжение табл. 2*

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| DrawRectangles(Pen, Rectangle[]) | Рисует набор прямоугольников, определяемых структурами Rectangle. | 2 |
| DrawString(String, Font, Brush, PointF) | Создает указываемую текстовую строку в заданном месте с помощью определяемых объектов Brush и Font. | 6 |
| Equals(Object) | Определяет, равен ли заданный объект текущему объекту. (Унаследовано от Object.) | 1 |
| ExcludeClip(Rectangle) | Обновляет вырезанную область данного объекта Graphics, чтобы исключить из нее часть, определяемую структурой Rectangle. | 1 |
| ExcludeClip(Region) | Обновляет вырезанную область данного объекта Graphics, чтобы исключить из нее часть, определяемую структурой Region. | 1 |
| FillClosedCurve(Brush, Point[]) | Заполняет внутреннюю часть замкнутой фундаментальной кривой, определяемой массивом структур Point. | 6 |
| FillEllipse(Brush, Rectangle) | Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, который задан структурой Rectangle. | 4 |
| FillPath(Brush, GraphicsPath) | Заполняет внутреннюю часть объекта GraphicsPath. | 1 |
| FillPie(Brush, Rectangle, Single, Single) | Заполняет внутреннюю часть сектора, определяемого эллипсом, который задан структурой RectangleF, и двумя радиальными линиями. | 3 |
| FillPolygon(Brush, Point[]) | Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами Point. | 4 |
| FillRectangle(Brush, Rectangle) | Заполняет внутреннюю часть прямоугольника, определяемого структурой Rectangle. | 4 |
| FillRegion(Brush, Region) | Заполняет внутреннюю часть объекта Region. | 1 |
| Flush() | Вызывает принудительное выполнение всех отложенных графических операций | 2 |

*Продолжение табл. 2*

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
|  | и немедленно возвращается, не дожидаясь их окончания. |  |
| IntersectClip(Region) | Обновляет вырезанную область данного объекта, включая в нее пересечение текущей вырезанной области и указанной структуры. | 3 |
| ResetClip() | Сбрасывает выделенную область g, делая ее бесконечной | 1 |

Свойство PageScale устанавливает масштабирование между мировыми единицами и единицами страницы.

2) Класс pen.

Объекты пера используются в методах рисования линий и графических фигур.

Объекты Pen выбираются из класса Pens (перья). Класс Pens содержит набор объектов для выбора. У них толщина линии (1 пиксель), стиль линии – сплошная. У каждого объекта свой цвет линии, имя которого идентифицирует объект. Такой объект нельзя редактировать, его можно только применять [8].

В таблице ниже (табл. 3) приведены классы перьев [2].

*Таблица 3*

**Классы перьев**

|  |  |
| --- | --- |
| Класс | Описание |
| Pen | Определяет объект, используемый для рисования линий |
| Pens | Класс для стандартных цветов |
| SystemPens | Свойства этого класса являются объектами Pen, то есть цветом элемента изображения Windows толщиной 1 |

Класс Pen имеет 4 конструктора:

1. Pen (Brush brush);
2. Pen (Color color);
3. Pen (Brush brush, float width);
4. Pen (Color color, float width).

Экземпляр класса после завершения работы с ним необходимо уничтожить методом Dispose(), который освобождает все ресурсы, используемые данным объектом Pen.

Перо обладает следующими свойствами:

* Brush – получает или задает объект Brush;
* Color – получает или задает цвет пера;
* float Width – получает или устанавливает толщину пера Pen в единицах объекта;
* PenType – получает или задает стиль линий;
* DashStyle – задает пользовательский стиль пунктирных линий.

Свойство PenType перечислимого типа задает вид заполнения, используемый объектом Pen, и может принимать следующие значения:

* SolidColor – задает сплошное заполнение;
* HatchFill – задает заполнение штриховкой;
* TextureFill – задает заполнение с текстурой точечного рисунка;
* PathGradient – задает градиентное заполнение контура;
* LinearGradient – задает линейное градиентное заполнение.

Свойство перечислимого типа DashStyle может принимать следующие значения:

* Solid – сплошная линия;
* Dash – линия, состоящая из штрихов;
* Dot – линия, состоящая из точек;
* DashDot – штрихпунктирная линия;
* DashDotDot – линия, состоящая из шаблонов «штрих-две точки»;
* Custom – задает произвольный тип пунктирных линий.

В следующем примере создается черное перо [2]:

* черное перо толщиной 1 – Pen myPen = new Pen(Color.Black);
* черное перо толщиной 5 – Pen myPen = new Pen(Color.Black, 5);

Ниже показывается создание пера, основанное на существующей кисти с именем myBrush [2]:

* перо с такими же свойствами, как у кисти myBrush толщиной 1 – SolidBrush myBrush = new SolidBrush(Color.Red); Pen myPen = new Pen(myBrush);
* перо с такими же свойствами, как у кисти myBrush толщиной 5 – Pen myPen = new Pen(myBrush, 5).

3) Класс brush.

Объекты кисти используются в методах заливки графических фигур. Объекты Brush выбираются из класса Brushes, который содержит кисти со сплошной заливкой. Класс Brushes содержит набор объектов для выбора, у которых по умолчанию определен цвет. Кисти используются для рисования сплошных фигур и отображения текста.

В таблице ниже (табл. 4) приведены классы кистей, каждый из которых наследуется от класса Brush [2].

*Таблица 4*

**Классы кистей**

|  |  |
| --- | --- |
| Класс Brush | Описание |
| 1 | 2 |
| SolidBrush | Сплошная заливка |
| HatchBrush | Сплошная заливка, позволяет выбрать из представленных шаблонов |
| TextureBrush | Кисть, использующая изображение |

*Продолжение табл. 4*

|  |  |
| --- | --- |
| 1 | 2 |
| LinearGradientBrush | Рисует двумя цветами, смешивая цвета вдоль указанной линии. В начале линии назначается один цвет, в конце — другой |
| PathGradientBrush | Задается полигон, в каждой вершине которого назначается свой цвет |
| LinearGradientBrush | Рисует двумя цветами, смешивая цвета вдоль указанной линии. В начале линии назначается один цвет, в конце — другой |

Экземпляры класса Brush после завершения работы необходимо уничтожать методом Dispose().

# ОПИСАНИЕ СОЗДАННОГО ПРИЛОЖЕНИЯ

## Постановка задачи

* Программа принимает данные любого типа и количества.
* Зашифрование и расшифрование проводится по формулам (1.3) и (1.4).
* Блок F состоит из нескольких функций преобразования.
* Раскодированное сообщение соответствует исходному.
* Приложение работает на операционной системе Windows 7 и выше.

## Логическое проектирование – алгоритм работы программы

Программа будет работать по следующему алгоритму:

1. сообщение преобразуется в двоичный код. Определяется длина преобразованного сообщения, после чего проводятся операции:

* сообщение дополняется справа единицами, если длина сообщения больше или меньше 64-х символов, тем самым длина сообщения увеличивается до ближайшего значения, являющегося степенью двойки (26=64, 27=128, …);
* вычисляется количество 64-х битных блоков сообщения;

1. проводится зашифрование сообщения:

* 64-х битный блок разбивается на два 32-х битных – правый (далее именуемый П) и левый (далее именуемый Л);
* блок П преобразуется посимвольно функциями преобразования: значение символа = нулю, значение символа = номер раунда + 1 – значение символа, значение символа = номер раунда + 2 – значение символа;
* блоки Л и П суммируются по модулю два, результат присваивается блоку Л, а блоку П присваивается предыдущее значение блока Л;
* цикл шифрования производится ещё 31 раз. В последнем смена не проводится;
* возвращается зашифрованная последовательность с длиной исходного сообщения, добавленной справа.

1. проводится расшифрование зашифрованного сообщения:

* читается длина сообщения, добавленной справа;
* вычисляется длина зашифрованной последовательности и количество 64-х битных блоков;
* остальные действия аналогичны 2).

1. исходное сообщение, его зашифрованная и расшифрованная последовательности, преобразованная в символы строка записываются в отдельные окна для удобства дальнейшей работы.

Алгоритм работы программы представлен в виде блок-схем (рис. 2, 3)

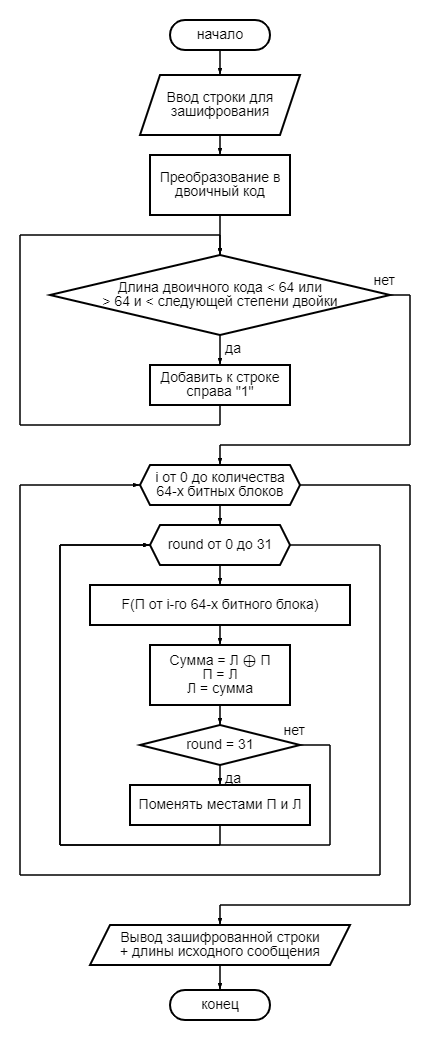


Рис. 2. Схема работы алгоритма зашифрования

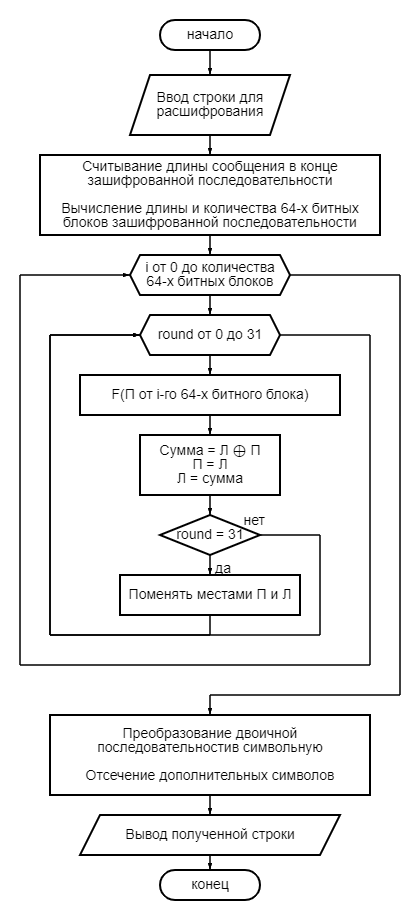
**

Рис. 3. Схема работы алгоритма расшифрования

## Физическое проектирование – структуры данных и спецификация функций

В программе используются типы данных, представленные в табл. 5.

*Таблица 5*

**Структуры данных**

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| 1 | 2 | 3 |
| Строка, введённая пользователем (открытый текст) | str | String^ |
| Длина введённой строки | strlen | int |

*Продолжение табл. 5*

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Число 4-х символьных блоков строки str | block\_count | int |
| Десятичное представление символа строки str | n | int |
| Строка, хранящая инвертированное двоичное представление символа строки str | strl | String^ |
| Двоичное представление символа строки str | istr | String^ |
| Число 64-х битных блоков строки istr | two | int |
| 64-х битный блок двоичной строки | block64 | String^ |
| Правый 32-х символьный подблок | Rstr | String^ |
| Левый 32-х символьный подблок | Lstr | String^ |
| Символ из правого подблока | r | Int |
| Символ из левого подблока | l | int |
| Левого и правый подблоки вместе | sum | String^ |
| Зашифрованная двоичная последовательность стоки istr (закрытый текст) | res | String^ |
| Длина сообщения в двоичной последовательности кратная степени 2 | meslen | int |
| Реальная длина сообщения в строковом представлении без дополнительных символов | x | int |
| Вспомогательные переменные позиции символа в строке | pos, p | int |
| Расшифрованная двоичная последовательность стоки res | code | String^ |
| Десятичное представление двоичного кода | decode | int |
| Расшифрованное сообщение (открытый текст) | dec | String^ |

Используемые в программе модули представлены в табл. 6.

*Таблица 6*

**Спецификации модулей**

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры  или функции | Формальные  параметры | Выполняемое  действие |
| 1 | 2 | 3 | 4 |
| Перевод | Transformation | String^ str, int strlen | Преобразует строку в двоичный код |
| Дополнение | Addition | String^ istr, int \*block\_count | Дополняет код до степени двойки |
| Кодирование | Encoding | String^ istr, int block\_count | Возвращает зашифрованную |

*Продолжение табл. 6*

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
|  |  |  | двоичную последовательность |
| Декодирование | Decoding | String^ res, int block\_count, int strlen | Возвращает расшифрованный двоичный код, преобразованный в строку |
| Сумма по модулю 2 | String^ sum2 | String^ sum | Суммирует по модулю 2 символы левого и правого блоков и меняет местами их значения на протяжении 32-х раундов. В 32-ой раунд дополнительно меняет местами левый и правый блоки. |
| Функция преобразования F | int F | int r, int l, int n | Проводит указанные пользователем преобразования символа из блока П |
| Зашифрование | button1\_Click | System::Object^ sender, System::EventArgs^ e | Принимает введённую пользователем строку, переводит её в двоичный код, зашифровывает по формулам (1.3), (1.4) |
| Расшифрование | button2\_Click | System::Object^ sender, System::EventArgs^ e | Принимает введённую зашифрованную строку, расшифровывает по формулам (1.3) и (1.4), переводит из двоичного кода в символьное представление |

*Продолжение табл. 6*

|  |  |  |  |
| --- | --- | --- | --- |
| Копирование | button3\_Click | System::Object^ sender, System::EventArgs^ e | Копирует зашифрованную последовательность в буфер обмена |
| Очистка | button4\_Click | System::Object^ sender, System::EventArgs^ e | Очищает все текстовые поля |
| Вставка | button5\_Click | System::Object^ sender, System::EventArgs^ e | Вставляет содержимое буфера обмена в поле для расшифрования сообщения |
| Смена цвета | button6\_Click | System::Object^ sender, System::EventArgs^ e | Меняет цвет формы на красный |

## Тестирование

Для тестирования программы были подобраны тестовые данные в табл. 7.

*Таблица 7*

**Тестовые данные**

|  |  |  |
| --- | --- | --- |
| Исходные данные | Тестируемый модуль или подпрограмма | Ожидаемый результат |
| 1 | 2 | 3 |
| Мир; Программирование | Перевод | 000001000001110000000100  001110000000010001000000;  000001000001111100000100  010000000000010000111110  000001000011001100000100  010000000000010000110000  000001000011110000000100  001111000000010000111000  000001000100000000000100  001111100000010000110010  000001000011000000000100  001111010000010000111000  0000010000110101 |
| Ожидаемый результат Перевода | Дополнение | 000001000001110000000100  001110000000010001000000 |

*Продолжение табл. 7*

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
|  |  | 1111111111111111; ожидаемый результат Перевода |
| Ожидаемый результат Дополнения | Кодирование | Зашифрованное сообщение |
| Зашифрованное сообщение | Декодирование | Исходные данные Перевода |
| Ожидаемый результат Дополнения | Функция преобразования F | Преобразованная строка |
| Ожидаемый результат Функции преобразования F | Сумма по модулю 2 | Зашифрованное сообщение |
| Исходное сообщение | Зашифрование | 000001000001110000000100  001110000000000001011100  111110111100011111111111  111111111111111111111111  000000000000000000000000  000000003;  000001000001111100000100  010000000000000000100001  000000000111001100000100  010000000000010000110000  000000000111110000000000  000011000000010000111000  000001000100000000000000  000001100000000001110010  000001000011000000000100  001111010000000000001000  000000000000100016 |
| Ожидаемый результат зашифрования | Расшифрование | Мир; Программирование |
| Зашифрованное сообщение | Копирование | Содержимое буфера обмена соответствует скопированному |
| Все поля заполнены | Очистка | Все поля очищены |
| Зашифрованное сообщение | Вставка | Вставка соответствует содержимому буфера обмена |
| Форма | Смена цвета | Форма стала красного цвета |

Результаты тестирования представлены в табл. 8.

*Таблица 8*

**Результаты тестирования**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время  тестиро-вания | Тестируемый  модуль или  подпрограмма | Кто  проводил  тестирова-ние | Описание  теста | Результаты  тестирования |
| 25.02.23 | Перевод | Микуцких  Г. А. | Вводится сообще-ние. В результате должен получиться двоичный код сообщения | Результаты соответствуют ожидаемым |
| 01.03.23 | Дополнение | Микуцких  Г. А. | Двоичный код дополнится до длины равной ближайшей степени двойки | Результаты соответствуют ожидаемым |
| 05.03.23 | Кодирование | Микуцких  Г. А. | Двоичный код зашифруется | Результаты соответствуют ожидаемым |
| 10.03.23 | Декодирова-ние | Микуцких  Г. А. | Двоичный код расшифруется и преобразуется в строку | Результаты соответствуют ожидаемым |
| 20.03.23 | Функция преобразова-ния F | Микуцких  Г. А. | Двоичный код преобразуется функцией | Результаты соответствуют ожидаемым |
| 02.04.23 | Зашифрова-ние | Микуцких  Г. А. | Двоичный код зашифруется | Результаты соответствуют ожидаемым |
| 02.04.23 | Расшифрова-ние | Микуцких  Г. А. | Зашифрованное сообщение расшифруется и переведётся в строку символов, равной исходной | Результаты соответствуют ожидаемым |
| 02.04.23 | Копирование | Микуцких  Г. А. | Нажатие на кнопку копирования скопирует зашифрованное сообщение в буфер обмена | Результаты соответствуют ожидаемым |

*Продолжение табл. 8*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 02.04.23 | Очистка | Микуцких  Г. А. | Нажатие на кнопку очистки очистит все поля | Результаты соответствуют ожидаемым |
| 02.04.23 | Вставка | Микуцких  Г. А. | Нажатие на кнопку вставки выгрузит зашифрованное сообщение из буфера обмена в окно для расшифрования | Результаты соответствуют ожидаемым |
| 02.04.23 | Смена цвета | Микуцких  Г. А. | Нажатие на кнопку сменит цвет формы на красный | Результаты соответствуют ожидаемым |

## Результаты работы

В результате работы было получено визуальное приложение для шифрования сообщение по методу сети Фейстеля (рис. 4).

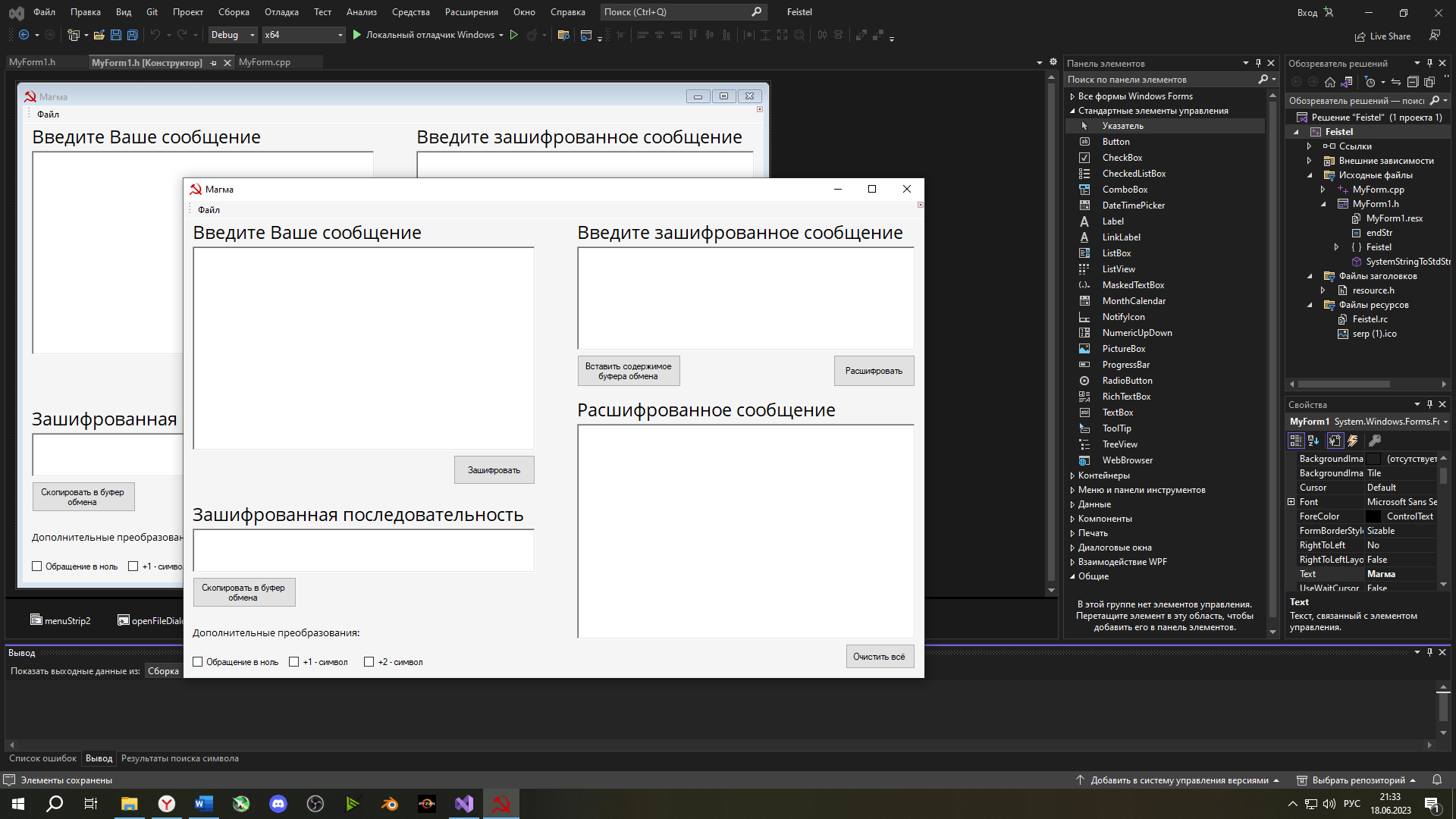


Рис. 4. Общий вид программы «Магма»

Также есть возможность не писать сообщение непосредственно в приложении. Для этого нужно написать сообщение в текстовом файле и сохранить в кодировке ANSI, а после вставить содержимое файла в строку для зашифрования с помощью кнопки «Файл». Путь для шифрования данным способом: «Файл → Открыть для: → Зашифровки».

Чтобы сохранить сообщение в файл: «Файл → Сохранить».

Для добавления зашифрованного сообщения в файле в строку для расшифрования: «Файл → Открыть для: → Расшифровки».

Программе решено дать имя «Магма» в честь одноимённого блочного шифра из стандарта ГОСТ Р 34.12-2015, так как разработка вдохновлялась данным стандартом.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана программа для шифрования данных методом сети Фейстеля. Программа может быть использована в различных сферах, связанных с передачей и хранением конфиденциальной информации, и не только. Одним из основных её достоинств является открытый исходный код, который позволяет пользователю изучать алгоритм работы программы и вносить какие-либо изменения в её структуре – отключать или подключать дополнительные блоки преобразования, изменять количество раундов преобразований и так далее.

# СПИСОК ЛИТЕРАТУРЫ

[1] ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры»

[2] Тюкачев Н. А. C#. Программирование 2D и 3D векторной графики: учебное пособие / Н. А. Тюкачев, В. Г. Хлебостроев. — 4-е изд., стер. — СанктПетербург: Лань, 2020. — 320 с.

[3] Раунд (в криптографии) | это... . URL: <https://dic.academic.ru/dic.nsf/ruwiki/1857488> (дата доступа: 01.06.23).

[4] Погружение в крипту. Часть 3: отечественные шифры — Хакер.

URL: <https://xakep.ru/2016/04/21/crypto-part3/#toc04> (дата доступа: 01.06.23).

[5] НОУ ИНТУИТ | Лекция |Алгоритмы симметричного шифрования. Часть1. URL: https://intuit.ru/studies/courses/28/28/lecture/20412?page=2 (дата доступа: 01.06.23).

[6] Сеть Фейстеля — Википедия. URL: https://web.archive.org/web/201301 17044805/http://ru.wikipedia.org/wiki/ (дата доступа: 01.06.23).

[7] 23.0. Графика в c#. URL: <https://studfile.net/preview/4602998/> (дата доступа: 01.06.23).

[8] Двумерная графика на C#, классы Graphics, Pen и Brush — C# ~ Си шарп для начинающих. URL: https://c-sharp.pro/graphics-pen-и-brush/ (дата доступа: 01.06.23).

# ПРИЛОЖЕНИЕ 1

Техническое задание.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Институт информационных технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование института (факультета)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_МПО ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование кафедры

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Структурное программирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ,

д.т.н., профессор \_\_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

Техническое задание на курсовую работу

Листов \_6\_

Руководитель Пышницкий К. М.

Ф.И.О. преподавателя

Исполнитель

студент студент гр. 1ПИб-02-1оп-22

группа

\_\_\_\_\_\_Микуцких Г.А.\_\_\_\_

Фамилия, имя, отчество

\_2023\_ год

Введение

Проблема передачи сообщений, исключающей прочтения посторонним лицом, появилась несколько тысяч лет до н.э. Для того, чтобы сообщение смог прочесть только тот, кому оно адресовано, необходимо применить какой-либо способ его преобразования. В эпоху автоматизации процессов достаточно ввести сообщение в шифратор, и он выдаст зашифрованную комбинацию буквально через несколько мгновений, которую можно отправлять без опаски перехвата и прочтения сторонним лицом. В то же время, адресат должен иметь возможность без проблем расшифровать полученное сообщение.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине "Структурное программирование", выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 21 февраля 2023 года.

Наименование темы разработки: «Сеть Фейстеля».

1. Назначение разработки

Программа, реализующая кодирование и декодирование сообщений по методу сети Фейстеля, обеспечит надежную защиту конфиденциальной информации.

1. Требования к программе
   1. Требования к функциональным характеристикам

В процессе разработки были утверждены обязательные функции: у программы должен быть удобный и интуитивно понятный интерфейс, не должна имеет значения длина сообщения, время ответа системы не должно превышать пяти секунд.

* 1. Требования к надежности

Работа программы должна соответствовать следующим требованиям:

* пользователь, введя сообщение, должен не только получить зашифрованную последовательность, но и быть уверенным, что расшифрованное сообщение будет идентичным исходному;
* программа должна работать бесперебойно, длина сообщения не должна «ломать» процесс шифрования;
* программа должна работать даже на относительно слабых системах;

3.3. Условия эксплуатации

Пользователь может воспользоваться программой сразу после установки.

3.4. Требования к составу и параметрам технических средств

ЭВМ должна соответствовать следующим параметрам:

* исходные данные и результаты представляются в виде текста и графических изображений;
* время отклика системы не более 1 секунды;
* минимальные системные требования:
* Windows 7 или выше;
* 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, 16 ГБ (для 32-разрядной системы) или 20 ГБ (для 64-разрядной системы) свободного места на жестком диске;
* графическое устройство DirectX 9 с драйвером WDDM 1.0 или более поздней версии;
* скорость подключения к сети Интернет не менее 512 Кбит;
* поддержка клавиатуры, компьютерной мыши, монитора разрешением 1920×1080 пикселей.

3.5. Требования к информационной и программной совместимости

Программа должна корректно работать «из коробки».

3.6. Требования к маркировке и упаковке

Архив с ПО будет отправляться в социальной сети «ВКонтакте» в личных чатах, а также беседах групп первого курса направления «Программная инженерия» Череповецкого государственного университета.

3.7. Требования к транспортированию и хранению

Всем пользователям запрещается осуществлять действия, нарушающие статьи УК РФ 228.1. «Незаконные производство, сбыт или пересылка наркотических средств, психотропных веществ или их аналогов, а также незаконные сбыт или пересылка растений, содержащих наркотические средства или психотропные вещества, либо их частей, содержащих наркотические средства или психотропные вещества», а также под видом оригинального ПО статью 273 «Создание, использование и распространение вредоносных компьютерных программ».

3.8. Специальные требования

Для эффективной работы с программой рекомендуется наличие опыта нахождения в информационной среде и умение в ней ориентироваться.

4. Требование к программной документации

4.1. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку, которая содержит:

Титульный лист

Аннотацию

Оглавление

Введение

1. Описание предметной области

2. Описание классов graphics, pen и brush

3. Описание созданного приложения

Заключение

Список литературы

Приложения.

4.2. Технико-экономические показатели

Требования не предъявляются.

4.3. Требования к оформлению

Элементы курсовой работы оформлены в соответствии с параметрами ниже (табл. П1.1).

*Таблица П1.1*

**Требования к оформлению**

|  |  |
| --- | --- |
| Элемент | Требования |
| 1 | 2 |
| Документ | 1. Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б.  2. Файлы предъявляются на компакт-диске: РПЗ с ТЗ; программный код.  3. Листы и диск в конверте вложены в пластиковую папку скоросшивателя. |
| Страницы | 1. Ориентация – книжная; отдельные страницы, при необходимости, альбомная.  2. Поля: верхнее, нижнее – по 2 см, левое – 3 см, правое – 1 см. |
| Абзацы | Межстрочный интервал – 1, перед и после абзаца – 0. |
| Шрифты | Кегль – 14. В таблицах шрифт 12. Шрифт листинга – 8 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: Рис. Х. Название  В приложениях: Рис. П1.3. Название. |

*Продолжение табл. П1.1*

|  |  |
| --- | --- |
| 1 | 2 |
| Таблицы | 1. Подписывается: над таблицей, выравнивание по правому: «Таблица Х».  2. В следующей строке по центру Название  3. Надписи в «шапке» (имена столбцов, полей) – по центру.  4. В теле таблицы (записи) текстовые значения – выровнены по левому краю, числа, даты – по правому. |

5. Стадии и этапы разработки

Программа будет разрабатываться в течение следующих этапов (табл. П1.2):

*Таблица П1.2*

**Стадии и этапы разработки**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Прототипирование базовых преобразований сообщения | 22 февраля  2023 г. | Сообщение преобразуется в последовательность нулей и единиц |  |
| Создание блока шифрования | 23 февраля –  1 марта 2023 г. | Последовательность преобразуется для передачи |  |
| Создание функции дополнительного преобразования | 2 марта –  2 апреля 2023 г. | Функция преобразования F |  |
| Доработка функционала | 3 апреля –  4 мая 2023 г. | Визуальное приложение |  |

6. Порядок контроля и приемки

Курсовая работа будет оформляться в течение этапов в табл. П1.3.

*Таблица П1.3*

**Порядок контроля и приемки**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  контрольного этапа  выполнения  курсовой работы | Сроки  контроля | Результат выполнения | Отметка о приемке  результата  контрольного этапа |
| 1 | 2 | 3 | 4 |
| 1 | 2 | 3 | 4 |
| Оформление технического задания | 21 февраля 2023 г. | Техническое задание |  |
| Создание программы | 22 февраля –  4 мая 2023 г. | Программа |  |
| Оформление расчётно-пояснительной записки | 5 мая –  5 июня 2023 г. | Расчётно-пояснительная записка |  |

# ПРИЛОЖЕНИЕ 2

Руководство пользователя.

Общие сведения о программе

«Магма» – программа для шифрования и дешифрования сообщений любой длины. Алгоритм шифрования базируется на методе сети Фейстеля.

Описание установки

Установки не требует. Программа работает «из коробки».

Описание запуска

Для запуска необходимо нажать на ярлык с именем «Магма». После запуска программа встречает окном, сверху которого можно увидеть два сообщения: *«Введите Ваше сообщение» и «Введите зашифрованное сообщение».*

Инструкции по работе

Для зашифрования сообщения необходимо ввести сообщение с клавиатуры в окно для зашифрования или открыть текстовый файл для зашифровки, сохранённый в кодировке ANSI (Файл → Открыть для: → Зашифровки), затем нажать на кнопку «Зашифровать». Введённое сообщение преобразуется в зашифрованную последовательность и отобразиться в окне «Зашифрованная последовательность». Его можно скопировать в буфер обмена соответствующей кнопкой или сохранить в виде файла («Файл → Сохранить).

Для расшифрования сообщения необходимо ввести сообщение с клавиатуры или открыть полученный файл (Файл → Открыть для: → Расшифровки) в поле расшифровки, затем нажать на кнопку «Расшифровать». Введённая последовательность преобразуется в исходное сообщение и отобразиться в окне «Расшифрованное сообщение».

Сообщения пользователю

*Введите Ваше сообщение* – в окно ниже этого сообщения пользователь должен ввести сообщение для зашифрования.

*Зашифровать* – кнопка, нажатие на которую запускает процесс зашифрования сообщения.

*Зашифрованная последовательность* – в окне ниже будет отображён двоичный зашифрованный код, который можно отправить адресату.

*Скопировать в буфер обмена* – кнопка, позволяющая скопировать двоичный зашифрованный код в буфер обмена.

*Введите зашифрованное сообщение* – в окно ниже этого сообщения пользователь должен ввести зашифрованную последовательность.

*Вставить содержимое буфера обмена* – кнопка, позволяющая вставить содержимое буфера обмена в окно для расшифровки.

*Расшифровать* – кнопка, нажатие на которую запускает процесс расшифрования сообщения и преобразования.

*Расшифрованное сообщение* – в окне ниже будет отображено расшифрованное и преобразованное в пригодный для чтения вид сообщение.

Если окно для зашифровки или расшифровки не содержит преобразуемой последовательности, будет получено сообщение: «Нет исходных данных, заполните поле сообщения!!!».

Если окно для расшифровки содержит символы, непригодные для расшифровки, будет получено сообщение: «Неправильный тип исходных данных!!!».

# ПРИЛОЖЕНИЕ 3

Программный код.

П3.1. Файл «MyForm1.h»

#pragma once

#include <cmath>

#include <string>

#include <fstream>

#include <Windows.h>

std::string SystemStringToStdString(System::String^ str)

{

char\* pstr = static\_cast<char\*>(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(str).ToPointer());

std::string resultString(pstr);

System::Runtime::InteropServices::Marshal::FreeHGlobal(System::IntPtr(pstr));

return resultString;

}

const char endStr[] = { 13,10,0 };

namespace Feistel {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

public ref class MyForm1 : public System::Windows::Forms::Form

{

public:

MyForm1(void)

{

InitializeComponent();

}

protected:

~MyForm1()

{

if (components)

{

delete components;

}

}

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

protected:

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

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

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

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

private: System::Windows::Forms::CheckBox^ checkBox3;

private: System::Windows::Forms::CheckBox^ checkBox2;

private: System::Windows::Forms::CheckBox^ checkBox1;

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

private: System::Windows::Forms::Button^ button3;

private: System::Windows::Forms::Button^ button4;

private: System::Windows::Forms::Button^ button5;

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

private: System::Windows::Forms::Label^ label6;

private: System::Windows::Forms::ToolStripMenuItem^ сохранитьВФайлToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ сохранитьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ открытьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ дляЗашифровкиToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ расшифровкиToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выходToolStripMenuItem;

private: System::Windows::Forms::MenuStrip^ menuStrip2;

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::SaveFileDialog^ saveFileDialog1;

private: System::Windows::Forms::RichTextBox^ Message;

private: System::Windows::Forms::RichTextBox^ Decrypted;

private: System::Windows::Forms::RichTextBox^ Ciphertext;

private: System::Windows::Forms::RichTextBox^ Encrypted;

private: System::Windows::Forms::Button^ button6;

private:

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

void InitializeComponent(void){

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm1::typeid));

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

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

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

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

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

this->checkBox3 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());

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

this->button3 = (gcnew System::Windows::Forms::Button());

this->button4 = (gcnew System::Windows::Forms::Button());

this->button5 = (gcnew System::Windows::Forms::Button());

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

this->label6 = (gcnew System::Windows::Forms::Label());

this->сохранитьВФайлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->сохранитьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->открытьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->дляЗашифровкиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->расшифровкиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выходToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->menuStrip2 = (gcnew System::Windows::Forms::MenuStrip());

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->saveFileDialog1 = (gcnew System::Windows::Forms::SaveFileDialog());

this->Message = (gcnew System::Windows::Forms::RichTextBox());

this->Decrypted = (gcnew System::Windows::Forms::RichTextBox());

this->Ciphertext = (gcnew System::Windows::Forms::RichTextBox());

this->Encrypted = (gcnew System::Windows::Forms::RichTextBox());

this->button6 = (gcnew System::Windows::Forms::Button());

this->menuStrip2->SuspendLayout();

this->SuspendLayout();

//

// button2

//

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

this->button2->Location = System::Drawing::Point(857, 203);

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

this->button2->Size = System::Drawing::Size(108, 42);

this->button2->TabIndex = 31;

this->button2->Text = L"Расшифровать";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm1::button2\_Click);

//

// button1

//

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

this->button1->Location = System::Drawing::Point(356, 335);

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

this->button1->Size = System::Drawing::Size(108, 39);

this->button1->TabIndex = 30;

this->button1->Text = L"Зашифровать";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm1::button1\_Click);

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Open Sans", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label4->Location = System::Drawing::Point(513, 258);

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

this->label4->Size = System::Drawing::Size(356, 33);

this->label4->TabIndex = 28;

this->label4->Text = L"Расшифрованное сообщение";

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Open Sans", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label3->Location = System::Drawing::Point(513, 24);

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

this->label3->Size = System::Drawing::Size(445, 33);

this->label3->TabIndex = 23;

this->label3->Text = L"Введите зашифрованное сообщение\r\n";

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Open Sans", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label2->Location = System::Drawing::Point(6, 396);

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

this->label2->Size = System::Drawing::Size(452, 33);

this->label2->TabIndex = 21;

this->label2->Text = L"Зашифрованная последовательность\r\n";

//

// checkBox3

//

this->checkBox3->AutoSize = true;

this->checkBox3->Location = System::Drawing::Point(310, 600);

this->checkBox3->Name = L"checkBox3";

this->checkBox3->Size = System::Drawing::Size(148, 17);

this->checkBox3->TabIndex = 20;

this->checkBox3->Text = L" +2 - символ";

this->checkBox3->UseVisualStyleBackColor = true;

//

// checkBox2

//

this->checkBox2->AutoSize = true;

this->checkBox2->Location = System::Drawing::Point(161, 600);

this->checkBox2->Name = L"checkBox2";

this->checkBox2->Size = System::Drawing::Size(138, 17);

this->checkBox2->TabIndex = 19;

this->checkBox2->Text = L" +1 - символ";

this->checkBox2->UseVisualStyleBackColor = true;

//

// checkBox1

//

this->checkBox1->AutoSize = true;

this->checkBox1->Location = System::Drawing::Point(12, 600);

this->checkBox1->Name = L"checkBox1";

this->checkBox1->Size = System::Drawing::Size(121, 17);

this->checkBox1->TabIndex = 18;

this->checkBox1->Text = L"Обращение в ноль";

this->checkBox1->UseVisualStyleBackColor = true;

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Open Sans", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label1->Location = System::Drawing::Point(6, 24);

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

this->label1->Size = System::Drawing::Size(317, 33);

this->label1->TabIndex = 16;

this->label1->Text = L"Введите Ваше сообщение";

//

// button3

//

this->button3->Location = System::Drawing::Point(12, 496);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(137, 40);

this->button3->TabIndex = 37;

this->button3->Text = L"Скопировать в буфер обмена";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm1::button3\_Click);

//

// button4

//

this->button4->Location = System::Drawing::Point(873, 584);

this->button4->Name = L"button4";

this->button4->Size = System::Drawing::Size(92, 33);

this->button4->TabIndex = 38;

this->button4->Text = L"Очистить всё";

this->button4->UseVisualStyleBackColor = true;

this->button4->Click += gcnew System::EventHandler(this, &MyForm1::button4\_Click);

//

// button5

//

this->button5->Location = System::Drawing::Point(519, 203);

this->button5->Name = L"button5";

this->button5->Size = System::Drawing::Size(137, 42);

this->button5->TabIndex = 39;

this->button5->Text = L"Вставить содержимое буфера обмена";

this->button5->UseVisualStyleBackColor = true;

this->button5->Click += gcnew System::EventHandler(this, &MyForm1::button5\_Click);

//

// label5

//

this->label5->AutoSize = true;

this->label5->Font = (gcnew System::Drawing::Font(L"Open Sans", 9.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label5->Location = System::Drawing::Point(9, 560);

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

this->label5->Size = System::Drawing::Size(228, 18);

this->label5->TabIndex = 40;

this->label5->Text = L"Дополнительные преобразования:";

//

// label6

//

this->label6->AutoSize = true;

this->label6->Font = (gcnew System::Drawing::Font(L"Open Sans", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label6->Location = System::Drawing::Point(12, 9);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(0, 33);

this->label6->TabIndex = 41;

//

// сохранитьВФайлToolStripMenuItem

//

this->сохранитьВФайлToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->сохранитьToolStripMenuItem,

this->открытьToolStripMenuItem, this->выходToolStripMenuItem

});

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

this->сохранитьВФайлToolStripMenuItem->Size = System::Drawing::Size(48, 20);

this->сохранитьВФайлToolStripMenuItem->Text = L"Файл";

//

// сохранитьToolStripMenuItem

//

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

this->сохранитьToolStripMenuItem->Size = System::Drawing::Size(146, 22);

this->сохранитьToolStripMenuItem->Text = L"Сохранить";

this->сохранитьToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm1::сохранитьToolStripMenuItem\_Click);

//

// открытьToolStripMenuItem

//

this->открытьToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->дляЗашифровкиToolStripMenuItem,

this->расшифровкиToolStripMenuItem

});

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

this->открытьToolStripMenuItem->Size = System::Drawing::Size(146, 22);

this->открытьToolStripMenuItem->Text = L"Открыть для:";

//

// дляЗашифровкиToolStripMenuItem

//

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

this->дляЗашифровкиToolStripMenuItem->Size = System::Drawing::Size(153, 22);

this->дляЗашифровкиToolStripMenuItem->Text = L"Зашифровки";

this->дляЗашифровкиToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm1::дляЗашифровкиToolStripMenuItem\_Click);

//

// расшифровкиToolStripMenuItem

//

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

this->расшифровкиToolStripMenuItem->Size = System::Drawing::Size(153, 22);

this->расшифровкиToolStripMenuItem->Text = L"Расшифровки";

this->расшифровкиToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm1::расшифровкиToolStripMenuItem\_Click);

//

// выходToolStripMenuItem

//

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

this->выходToolStripMenuItem->Size = System::Drawing::Size(146, 22);

this->выходToolStripMenuItem->Text = L"Выход";

this->выходToolStripMenuItem->Click += gcnew System::EventHandler(this, &MyForm1::выходToolStripMenuItem\_Click);

//

// menuStrip2

//

this->menuStrip2->BackColor = System::Drawing::Color::WhiteSmoke;

this->menuStrip2->GripStyle = System::Windows::Forms::ToolStripGripStyle::Visible;

this->menuStrip2->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->сохранитьВФайлToolStripMenuItem });

this->menuStrip2->Location = System::Drawing::Point(0, 0);

this->menuStrip2->Name = L"menuStrip2";

this->menuStrip2->RenderMode = System::Windows::Forms::ToolStripRenderMode::System;

this->menuStrip2->Size = System::Drawing::Size(977, 24);

this->menuStrip2->TabIndex = 43;

this->menuStrip2->Text = L"menuStrip2";

//

// openFileDialog1

//

this->openFileDialog1->FileName = L"openFileDialog1";

//

// Message

//

this->Message->Location = System::Drawing::Point(12, 60);

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

this->Message->Size = System::Drawing::Size(452, 269);

this->Message->TabIndex = 44;

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

//

// Decrypted

//

this->Decrypted->Location = System::Drawing::Point(519, 294);

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

this->Decrypted->Size = System::Drawing::Size(446, 284);

this->Decrypted->TabIndex = 45;

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

//

// Ciphertext

//

this->Ciphertext->Location = System::Drawing::Point(519, 60);

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

this->Ciphertext->Size = System::Drawing::Size(446, 137);

this->Ciphertext->TabIndex = 46;

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

//

// Encrypted

//

this->Encrypted->Location = System::Drawing::Point(12, 432);

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

this->Encrypted->Size = System::Drawing::Size(452, 58);

this->Encrypted->TabIndex = 47;

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

//

// button6

//

this->button6->BackColor = System::Drawing::Color::Red;

this->button6->ForeColor = System::Drawing::SystemColors::ButtonFace;

this->button6->Location = System::Drawing::Point(967, 0);

this->button6->Name = L"button6";

this->button6->Size = System::Drawing::Size(10, 10);

this->button6->TabIndex = 48;

this->button6->UseVisualStyleBackColor = false;

this->button6->Click += gcnew System::EventHandler(this, &MyForm1::button6\_Click);

//

// MyForm1

//

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

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

this->BackColor = System::Drawing::Color::WhiteSmoke;

this->ClientSize = System::Drawing::Size(977, 629);

this->Controls->Add(this->button6);

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

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

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

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

this->Controls->Add(this->label6);

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

this->Controls->Add(this->button5);

this->Controls->Add(this->button4);

this->Controls->Add(this->button3);

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

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

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

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

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

this->Controls->Add(this->checkBox3);

this->Controls->Add(this->checkBox2);

this->Controls->Add(this->checkBox1);

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

this->Controls->Add(this->menuStrip2);

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->Name = L"MyForm1";

this->Text = L"Магма";

this->Load += gcnew System::EventHandler(this, &MyForm1::MyForm1\_Load);

this->menuStrip2->ResumeLayout(false);

this->menuStrip2->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void MyForm1\_Load(System::Object^ sender, System::EventArgs^ e) {

} //Преобразование кода строки в двоичный код

String^ Transformation(String^ str, int strlen) {

int n; String^ istr;

for (int len = 0; len < strlen; len++) {

String^ strl;

n = str[len] - 0;

do {

strl += n % 2; n /= 2;

} while (n > 0);

while (strl->Length < 16) strl += 0;

for (int i = 0; i < strl->Length; i++) istr += strl[15 - i];

}

return istr;

}

//Дополнение кода до степени двойки

String^ Addition(String^ istr, int\* block\_count) {

int two = 1;

while (two < \*block\_count) two \*= 2;

if (two > \*block\_count) \*block\_count = two;

for (int i = istr->Length; i < \*block\_count \* 64; i++) istr += 1;

return istr;

}

//Зашифрование

String^ Encoding(String^ istr, int block\_count) {

String^ res;

for (int n = 0; n < block\_count; n++) {

String^ block64;

for (int i = 0; i < 64; i++) block64 += istr[n \* 64 + i];

res += sum2(block64);

block64 = "";

}

return res;

}

//Расшифрование

String^ Decoding(String^ res, int block\_count, int strlen) {

String^ dec; String^ code; String^ block64;

for (int n = 0; n < block\_count; n++) {

for (int i = 0; i < 64; i++) block64 += res[n \* 64 + i];

code += sum2(block64);

block64 = "";

}

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

int pos = 0; int decode = 0;

for (int r = 15; r >= 0; r--) {

if (code[i \* 16 + pos] == 49) decode += pow(2, r);

pos++;

}

dec += Convert::ToChar(decode);

}

return dec;

}

int F(int r, int n) {

if (checkBox1->Checked)r= 0;

if (checkBox2->Checked)r= n+1-r;

if (checkBox3->Checked)r = n + 2 - r;

return r;

}

//Сумма по модулю 2 левого и правого блоков

String^ sum2(String^ sum) {

for (int round = 0; round < 32; round++) {

String^ Lstr; String^ Rstr;

//sum = F(sum);

for (int i = 0; i < 32; i++) {

int l = sum[i];

int r = sum[i + 32];

Rstr += (l + F(r, i)) % 2;

Lstr += r - 48;}

sum = Lstr + Rstr;

if (round == 31) sum = Rstr + Lstr;

}

return sum;

}

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

if (String::IsNullOrEmpty(Message->Text)) {

MessageBox::Show("Нет исходных данных, заполните поле сообщения!!!",

"Критическая ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

};

String^ str = Message->Text;

int strlen = str->Length;

int block\_count = (strlen + 3) / 4;

String^ istr = Transformation(str, strlen);

istr = Addition(istr, &block\_count);

Encrypted->Text = Encoding(istr, block\_count) + strlen;

}

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

if (String::IsNullOrEmpty(Ciphertext->Text)) {

MessageBox::Show("Нет исходных данных, заполните поле сообщения!!!",

"Критическая ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

};

String^ res;

res = Ciphertext->Text;

int strlen = res->Length, block\_count, meslen = 0;

for (int i = 0; meslen == 0; i++) if (strlen - pow(2, 7 + i) < 0) meslen = pow(2, 7 + i - 1);

for (int len = 0; len < meslen; len++) {

if (res[len] != 48 && res[len] != 49) {

MessageBox::Show("Неправильный тип исходных данных!!!",

"Критическая ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

};

block\_count = meslen / 64;

int x=0, p = 0;

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

int k = res[meslen + p] - 48;

x += k \* pow(10, n - 1);

p++;

}

strlen = x;

Decrypted->Text = Decoding(res, block\_count, strlen);

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

if (String::IsNullOrEmpty(Encrypted->Text)) {

MessageBox::Show("Нет исходных данных, заполните поле для копирования!!!",

"Критическая ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

};

Clipboard::SetText(Encrypted->Text);

}

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

Encrypted->Text = "";

Ciphertext->Text = "";

Decrypted->Text = "";

Message->Text = "";

BackColor = Color::WhiteSmoke;

}

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

Ciphertext->Text = Clipboard::GetText();

}

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

Close();

}

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

if (saveFileDialog1->ShowDialog() != System::Windows::Forms::DialogResult::OK) return;

std::string fileName = SystemStringToStdString(saveFileDialog1->FileName);

if (fileName.empty()) return;

std::ofstream out(fileName.c\_str());

std::string saveText = SystemStringToStdString(Encrypted->Text);

out << saveText;

}

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

if (openFileDialog1->ShowDialog() != System::Windows::Forms::DialogResult::OK) return;

std::string fileName = SystemStringToStdString(openFileDialog1->FileName);

if (fileName.empty()) return;

std::ifstream in(fileName.c\_str());

String^ systemEndStr = gcnew String(endStr);

while (!in.eof()) {

std::string s;

getline(in, s);

String^ myInfoDisp = gcnew String(s.c\_str());

Message->Text += (myInfoDisp + systemEndStr);

}

}

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

if (openFileDialog1->ShowDialog() != System::Windows::Forms::DialogResult::OK) return;

std::string fileName = SystemStringToStdString(openFileDialog1->FileName);

if (fileName.empty()) return;

std::ifstream in(fileName.c\_str());

while (!in.eof()) {

std::string s;

getline(in, s);

String^ myInfoDisp = gcnew String(s.c\_str());

Ciphertext->Text = myInfoDisp;

}

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

BackColor = Color::Red;

}

};

}

П3.2. Файл «MyForm.cpp»

#include "MyForm1.h"

using namespace System;

using namespace System::Windows::Forms;

[STAThread]

void main(cli::array<String^> ^ args) {

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Feistel::MyForm1 form;

Application::Run(% form);

}