Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра вычислительной техники

А.И. Сердюк, О.Н. Яркова

КРИПТОГРАФИЯ. РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ ШИФРОВАНИЯ ИНФОРМАЦИИ

Рекомендовано к изданию Редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Оренбургский государственный университет» в качестве методических указаний для студентов, обучающихся по программам высшего профессионального образования по специальности 090104.65 Комплексная защита объектов информатизации, 230101.65 Вычислительные машины, комплексы, системы и сети, направлению подготовки 090900.62 Информационная безопасность, профиль «Комплексная защита объектов информатизации»

Оренбург
2012

УДК 004.056.5:003.26(076.5)
ББК 32.973.2я7+32.811я7
С 32

Рецензент - доктор технических наук, профессор Т.З. Аралбаев

Сердюк, А.И.
С 32 Криптография. Разработка приложений для шифрования информации: методические указания / А.И. Сердюк, О.Н. Яркова; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2012. – 98 с.

Методические указания содержат теоретический материал и задания к лабораторным работам, посвященным разработке компьютерных программ на основе конкретных шифрсистем. В качестве среды программирования использована интегрированная среда Delphi 7.

Методические указания предназначены студентам специальностей 090104.65 Комплексная защита объектов информатизации, 230101.65 Вычислительные машины, комплексы, системы и сети, направлениюю подготовки 090900.62 Информационная безопасность профиль «Комплексная защита объектов информатизации», изучающим дисциплины «Математические основы криптологии», «Криптографические методы и средства обеспечения информационной безопасности» и родственные дисциплины, а так же для научных работников и инженеров, занятых вопросами создания и внедрения компьютерно управляемых комплексов оборудования.

УДК 004.056.5:003.26(076.5)
ББК 32.973.2я7+32.811я7

Содержание

Введение

Развитие локальных и глобальных компьютерных сетей выдвигает новые проблемы по обеспечению безопасности информации от раскрытия ее конфиденциальности, нарушения целостности. Одним из инструментов обеспечения информационной безопасности является криптография.

В переводе с греческого языка слово криптография означает тайнопись. Смысл этого термина выражает основное предназначение криптографии - защитить или сохранить в тайне необходимую информацию. Отличием криптографии от других способов обеспечения безопасности является то, что криптография не "прячет" передаваемые сообщения, а преобразует их в форму, недоступную для понимания противником. Традиционной задачей криптографии является обеспечение конфиденциальности информации при передаче сообщений по контролируемому противником каналу связи. Прогресс в развитии вычислительной техники сделал возможными программные реализации криптографических алгоритмов, которые все увереннее вытесняют во многих сферах традиционные аппаратные средства.

Целью предлагаемого пособия является практическое обучение пользователей некоторым приемам разработки компьютерных программ для автоматизированного шифрования информации с использованием различных шифров – от классических шифров перестановки и замены до современных многопользовательских шифрсистем типа RSA и цифровой подписи. Пособие содержит теоретический материал и задания к лабораторным работам, посвященным разработке компьютерных программ на основе конкретных шифрсистем. Наиболее сложные вопросы разбиты на последовательные этапы, облегчающие восприятие материала. В качестве среды программирования использована интегрированная среда Delphi 7.

1 Символы и их коды. Системы исчисления

1.1 Лабораторная работа Символы и их коды

Постановка задачи: Написать приложение, которое:

а) выводит на экран ASCII-коды символов русского алфавита;

б) для ASCII-кодов заданного диапазона выводит соответствующие символы.

Порядок выполнения: пример экранной формы приложения представлен на рисунке 1.1

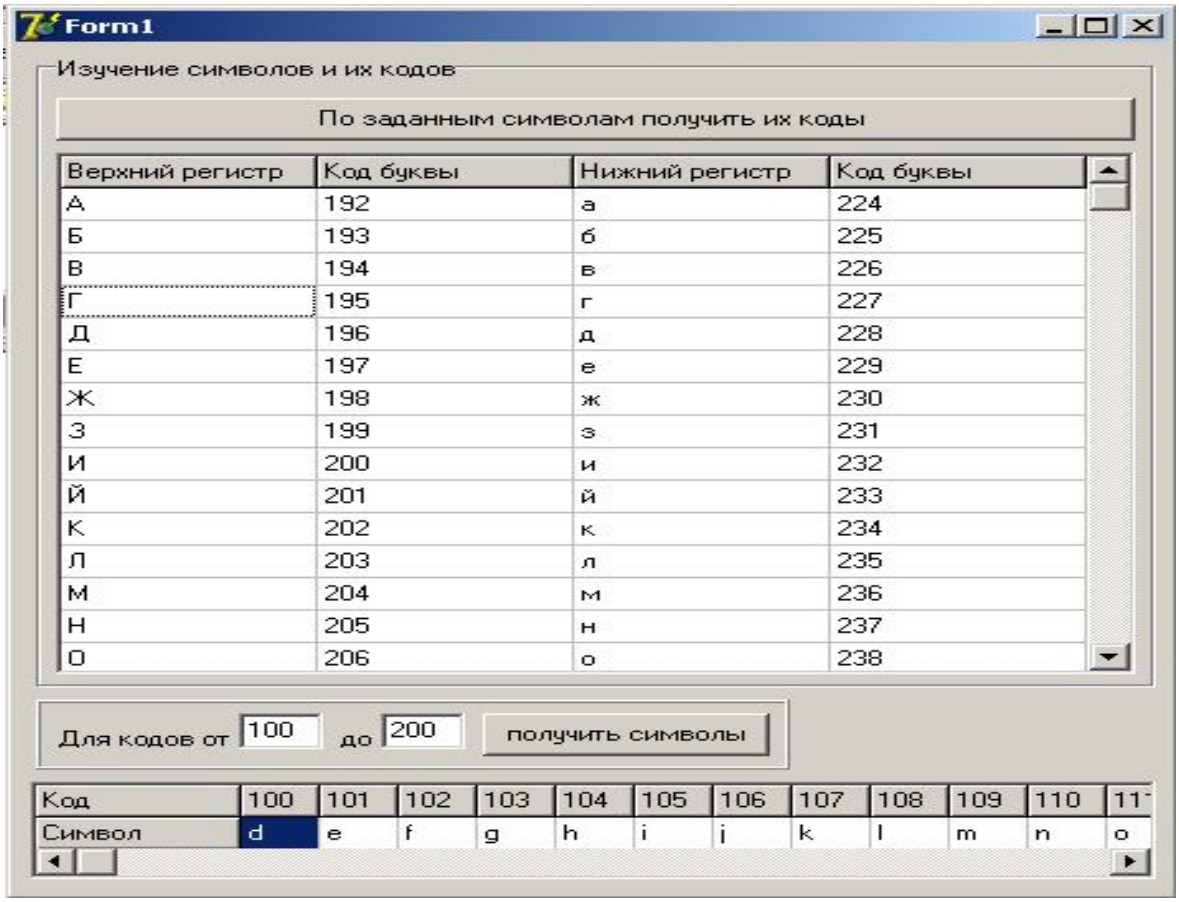


Рисунок 1.1 - Экранная форма приложения

Листинг кода программы с комментариями представлен ниже:

```
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
  With StringGrid1 do                  // разметка таблицы 1
  begin
    Cells[0,0]:='Верхний регистр';
    Cells[1,0]:='Код буквы';
    Cells[2,0]:='Нижний регистр';
    Cells[3,0]:='Код буквы';
  end;
  StringGrid2.ColWidths[0]:=80;    // разметка таблицы 2
  StringGrid2.Cells[0,0]:='Код';
  StringGrid2.Cells[0,1]:='Символ';
end;
procedure TForm1.Button1Click(Sender: TObject);
Const
 ABC_1: array[1..33] of char='АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ ';
 ABC_2: array[1..33] of char='абвгдежзийклмнопрстуфхцчшщъыьэюя ';
  Var
    Kod,i: integer;
begin
for i:=1 to 33 do                  // с первого по 33-й символ …
begin
 Kod:=Ord(ABC_1[i]);               // определяем числовой код символа
 StringGrid1.Cells[0,i]:=ABC_1[i];
 StringGrid1.Cells[1,i]:=IntToStr(Kod);
 StringGrid1.Cells[2,i]:=ABC_2[i];
 Kod:=Ord(ABC_2[i]);
 StringGrid1.Cells[3,i]:=IntToStr(Kod);
end;
end;
procedure TForm1.Button2Click(Sender: TObject);
Var i,j: integer;
begin
StringGrid2.ColCount:= StrToInt(Edit2.Text)- StrToInt(Edit1.Text)+2;
j:=0;
 For i:=StrToInt(Edit1.Text) to StrToInt(Edit2.Text) do
  begin
```

```
    j:=j+1;
     StringGrid2.Cells[j,0]:=IntToStr(i);
     StringGrid2.Cells[j,1]:=Chr(i);      // по числовому коду i
                                             определяем символ
  end;
end;
end.
```

1.1.1 Дополнительное задание

Напишите программу, которая определяет код нажатой клавиши клавиатуры, в том числе и служебных клавиш (**Enter**, **Esc**, **Alt** и др.).

1.1.2 Контрольные вопросы

1) Для чего предназначена функция **ORD()**?
2) Для чего предназначена функция **CHR()**?
3) В чем различие переменной символьного типа и переменной строкового типа? Как они объявляются?
4) В чем назначение выражения **Kod:=Ord(ABC_1[i])**?
5) Расскажите о функциях **IntToStr()** и **StrToInt()**
6) Расскажите о компоненте **StringGrid**. Как разметить таблицу с помощью данного компонента?

1.2 Системы исчисления

Совокупность приемов наименования и обозначения чисел называется ***системой исчисления***.

В качестве условных знаков для записи чисел используются цифры. Понятно, что большинство людей в жизни не воспользуются другой системой исчисления, кроме десятичной. Но знать о других системах исчисления надо, хотя бы для того,

чтобы не пугаться чисел типа **0xDEADBEEF** или **100100100100b**.

1.2.1 Непозиционные и позиционные системы исчисления

Система исчисления, в которой значение каждой цифры в произвольном месте последовательности цифр, обозначающей запись числа, не изменяется, называется *непозиционной*.

В непозиционной системе каждый знак в записи независимо от места означает одно и то же число. Хорошо известным примером непозиционной системы исчисления является римская система, в которой роль цифр играют буквы алфавита: I - один, V - пять, X - десять, C - сто, L - пятьдесят, D -пятьсот, M - тысяча.

Например, **324 = CCCXXIV**.

В непозиционной системе исчисления арифметические операции выполнять неудобно и сложно.

Система исчисления, в которой значение каждой цифры зависит от места в последовательности цифр в записи числа, называется *позиционной*.

Общепринятой в современном мире является *десятичная* позиционная система исчисления, которая из Индии через арабские страны пришла в Европу. Основой системы является число десять. Основой системы исчисления называется число, означающее, во сколько раз единица следующего разряда больше чем единица предыдущего. Выбор десятеричной системы как основной системы исчисления был обусловлен фактом наличия 10 пальцев на руках.

Другие системы исчисления имеют не меньше прав на существование, - в частности, двенадцатеричная, использовавшаяся до недавнего времени в Англии, потому что 12 делится нацело на 2, 3, 4 и 6, тогда как 10 - только на 2 и 5.

Основой представления чисел в памяти компьютера является *двоичная система исчисления*. Для изображения чисел в этой системе необходимо две цифры: 0 и 1, то есть достаточно двух стойких состояний физических элементов. Поскольку $2^3=8$, а $2^4=16$, то каждых три двоичных разряда числа образовывают один восьмеричный, а каждых четыре двоичных разряда - один шестнадцатеричный. Поэтому для сокра-

щения записи адресов и содержимого оперативной памяти компьютера используют шестнадцатеричную и восьмеричную системы исчисления.

В таблице 1.1 приведены первые 16 натуральных чисел записанных в десятичной, двоичной, восьмеричной и шестнадцатеричной системах исчисления.

Таблица 1.1 – Натуральные числа в разных системах исчисления

число в десятичной системе счисления	число в двоичной системе счисления	число в восьмеричной системе счисления	число в шестнадцатеричной системе счисления
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.2.2 Обозначение чисел в нестандартной системе исчисления

Чтобы отличить число, записанное в нестандартной, то есть не в десятеричной системе исчисления, обычно основание системы пишется маленькими цифрами после числа.

Например, $\mathbf{101_2}$ записано в двоичной системе исчисления, $\mathbf{101_{10}}$ - в десятеричной, а $\mathbf{101_{16}}$ - в шестнадцатеричной.

Кроме того, другая часто встречающаяся система индикации гласит: число в десятеричной системе не обозначается никак, после числа в двоичной записи ставится буква **b** (обозначающая **binary** - "двоичная запись"), а перед числом в ше-

стнадцатеричной системе ставится обозначение **0x**. Например, 101 - это число "сто один" в десятеричной системе, **101b** записано в двоичной, а **0x101** - в шестнадцатеричной системе.

Для отладки программ и в других ситуациях в программировании актуальной является проблема перевода чисел из одной позиционной системы исчисления в другую.

1.2.3 Алгоритмы перевода чисел из одной позиционной системы исчисление в другую

Общеупотребительной формой записи числа является сокращенная форма записи разложения по степеням основы системы исчисления, например

$$130678_{10} = 1*10_{10}^{5}+3*10_{10}^{4}+0*10_{10}^{3}+6*10_{10}^{2}+7*10_{10}^{1}+8*10_{10}^{0}$$

$$92C8_{16} = 9*10_{16}^{3}+2*10_{16}^{2}+C*10_{16}^{1}+8*10_{16}^{0},$$

$$110100101_{2} = 1*10_{2}^{8}+1*10_{2}^{7}+0*10_{2}^{6}+1*10_{2}^{5}+0*10_{2}^{4}+0*10_{2}^{3}+1*10_{2}^{2}+0*10_{2}^{1}+1*10_{2}^{0}.$$

В примерах индексы 10, 16 и 2 служат основой системы исчисления, а показатель степени - это номер позиции цифры в записи числа (нумерация ведется слева направо, начиная с нуля).

Существуют два правила (алгоритма) перевода чисел из одной системы исчисления в другую:

1) с использованием новой системы исчислений;
2) с использованием старой системы исчислений.

Правило 1. Для перевода чисел из системы исчисления с основой ***p*** в систему исчисления с основой ***q***, используя арифметику новой системы исчисления с основой ***q***, нужно записать коэффициенты разложения, основы степеней и показатели степеней в системе с основой ***q*** и выполнить все действия в этой самой системе.

Этим правилом удобно пользоваться при переводе из любой системы в десятичную систему исчисления.

Например:

из шестнадцатеричной в десятичную:

$92C8_{16}=9*10_{16}^{3}+2*10_{16}^{2}+C*10_{16}^{1}+8*10_{16}^{0}=$

$9*16_{10}^{3}+2*16_{10}^{2}+12*16_{10}^{1}+8*16_{10}^{0}=37576$.

из восьмеричной в десятичную:

$735_8=7*10_8^{2}+3*10_8^{1}+5*10_8^{0}= 7*8_{10}^{2}+3*8_{10}^{1}+5*8_{10}^{0}=477_{10}$

из двоичной в десятичную:

$110100101_2=1*10_2^{8}+1*10_2^{7}+0*10_2^{6}+1*10_2^{5}+0*10_2^{4}+0*10_2^{3}+1*10_2^{2}+0*10_2^{1}+1*10_2^{0}$

$= 1*2_{10}^{8}+1*2_{10}^{7}+0*2_{10}^{6}+1*2_{10}^{5}+ 0*2_{10}^{4}+0*2_{10}^{3}+1*2_{10}^{2}+0*2_{10}^{1}+ 1*2_{10}^{0} = 421_{10}$.

Правило 2. Для перевода чисел из системы исчисления с основой ***p*** в систему исчисления с основой ***q*** с использованием арифметики старой системы исчисления с основой ***p*** нужно

а) для перевода целой части:

последовательно число, записанное в системе основой делить на основу новой системы исчисления, выделяя остатки. Остатки, записанные в обратном порядке, будут образовывать число в новой системе исчисления;

б) для перевода дробной части:

последовательно дробную часть умножать на основу новой системы исчисления, выделяя целые части, которые и будут образовывать запись дробной части числа в новой системе исчисления.

Этим же правилом удобно пользоваться в случае перевода из десятичной системы исчисления, поскольку ее арифметика для нас привычна.

Пример: $999,35_{10}=1111100111,01011_2$

для целой части:

999	2								
1	499	2							
	1	249	2						
		1	124	2					
			0	62	2				
				0	31	2			
					1	15	2		
						1	7	2	
							1	3	2
								1	1

для дробной части:

0,	35 2
0,	70 2
1,	40 2
0,	80 2
1,	60 2
1,	20

1.2.4 Лабораторная работа Системы исчисления

Постановка задачи: написать приложение, которое преобразовывает число:

а) из двоичного представления в десятичное;

б) из десятичного в двоичное.

Порядок выполнения: пример оформления экранного интерфейса приведен на рисунке 1.2. Программа состоит из двух независимых частей. В качестве примера рассмотрим последовательность разработки левой части интерфейса и его программного кода.

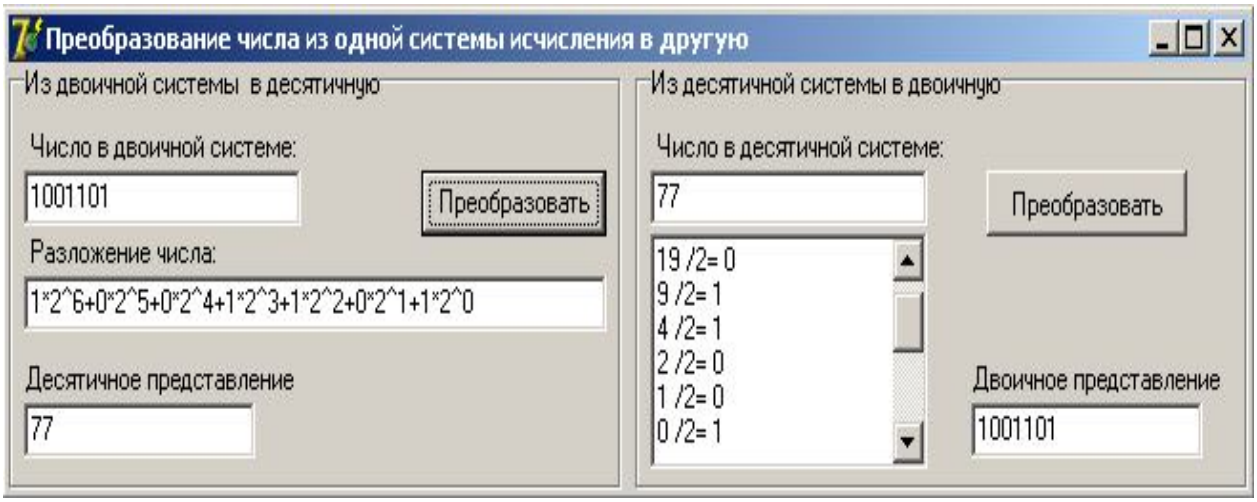


Рисунок 1.2 - Экранный интерфейс приложения

Последовательность разработки разделим на следующие шаги.

Шаг 1. Размещаем на форме компоненты в соответствии с рисунком 1.3.

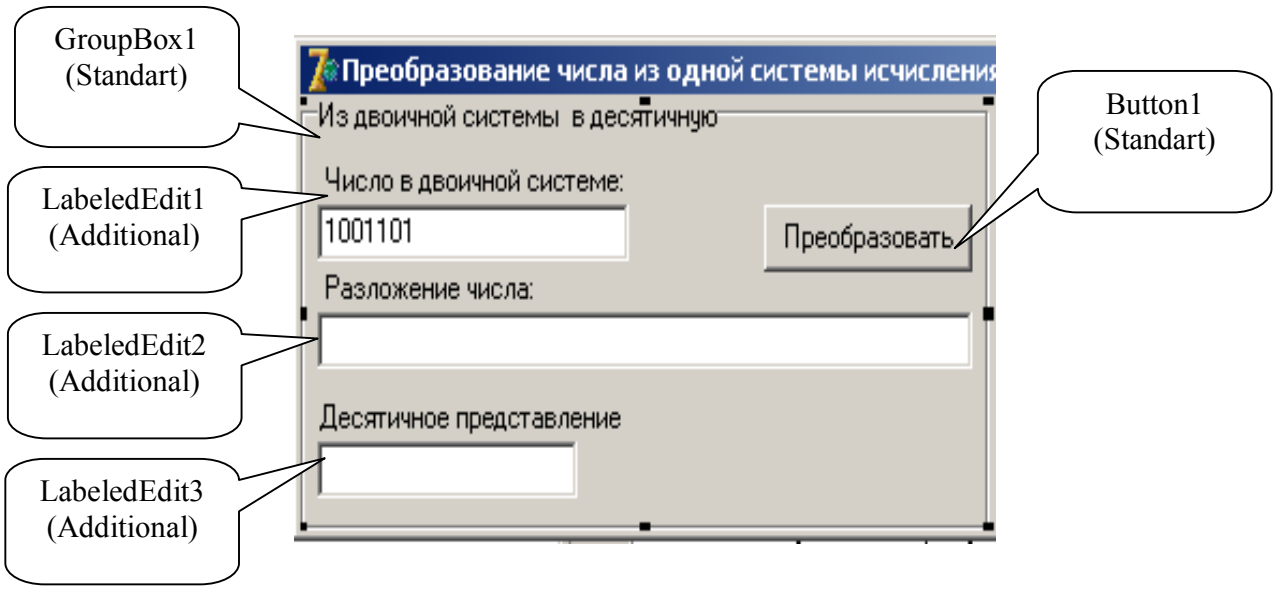


Рисунок 1.3 - Используемые компоненты на экранной форме

Шаг 2. В инспекторе объектов вписываем нужный текст в свойства используемых компонентов

Шаг 3. В процедуру кнопки Button1.Click вписываем следующий код:

```
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
Var
 E,      // будем использовать для указания степени
 Cislo, // переменная для получения числа в десятичном виде
 Dlina, // переменная для количества символов в двоичном числе
 i: integer;  // переменная цикла
// массивы строковых переменных для вывода разложения числа в Edit2
 a,b: array[1..50] of string;
 Dwoi,Stroka: string;       // строковые переменные
 Y: extended;              // Переменная для записи результата
                          // возведения в степень
begin
```

```
                          // присваиваем переменной Dwoi значение
                          // введенного двоичного числа:
  Dwoi:=LabeledEdit1.Text;
 // переменную Dlina записываем число символов в переменной Dwoi:
  Dlina:= Length(Dwoi);
  // последовательно из переменной Dwoi выделяем
  // каждый отдельный символ и сохраняем его
  // в отдельном элементе массива a[i]
  // определяем разряд (позицию) данного символа в двоичном числе
For i:=1 to Dlina do
begin
  a[i]:=Copy(Dwoi,i,1);
  b[i]:=IntToStr(Dlina-i);
end;
Stroka:='';           // очищаем строковую переменную Stroka
for i:=1 to Dlina do  // последовательно записываем в
                      //переменную Stroka разложение числа Dwoi
  If i<Dlina then Stroka:=Stroka+a[i]+'*2^'+b[i]+'+'
             else Stroka:=Stroka+a[i]+'*2^'+b[i];
 LabeledEdit2.Text:=Stroka;
                    // выводим разложение по степеням в окно
                    // полученное разложение числа преобразуем в
                   // переменную Cislo
 Cislo:=0;
 For i:=1 to Dlina do
 begin
   E:=StrToInt(b[i]); // разряд текущего символа преобразуем из
                      // строкового представления в целочисленное
   Y:= IntPower(2,E); // возводим число 2 в целую степень E
                      // накапливаем результат в переменной Cislo
 Cislo:=Cislo+StrToInt(a[i])*Round(Y);
end;
// выводим полученное значение в окно Edit3
 LabeledEdit3.Text:=IntToStr(Cislo);
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
```

```
Var
     i: integer;
 Cislo: integer;
   Sim: array[1..50] of integer;
Stroka: string;
begin
i:=0;
Cislo:=StrToInt(LabeledEdit4.Text);
While Cislo>0 do
begin
 i:=i+1;
 Sim[i]:=Cislo mod 2;     {находим остаток}
 Cislo:=Cislo div 2;      {находим целую часть}
Memo1.Lines.Strings[i-1]:=IntToStr(Cislo)+'/2= +IntToStr(Sim[i]);
end;
Stroka:='';
For i:=i downto 1 do Stroka:=Stroka+FloatToStr(Sim[i]);
LabeledEdit6.Text:=Stroka;
end;
end.
```

После запуска приложения можно использовать рисунок 1.2 в качестве тестового примера для проверки правильности расчетов.

1.2.4.1 Дополнительное задание

Написать приложение для перевода числа из десятичного представления в шестнадцатеричное и из шестнадцатеричного в десятичное.

1.2.4.2 Контрольные вопросы

1) Что такое система исчисления?
2) Какие типы систем исчисления вы знаете?

3) Что такое основа позиционной системы исчисления?
4) Как называется система исчисления, в которой значение каждой цифры не зависит от места в последовательности цифр в записи числа?
5) Какая система исчисления используется для представления чисел в памяти компьютера? Почему?
6) Каким образом осуществляется перевод чисел, если основа новой системы исчисления равняется некоторой степени старой системы исчисления?
7) Какие правила (алгоритмы) перевода чисел из одной системы исчисления в другую вы знаете?
8) По какому правилу переводятся числа из любой системы исчисления в десятичную систему исчисления?
9) Чему равно число 0xC, если его перевести в десятичную систему исчисления?
10) Чему равно число 1101b, если его перевести в десятичную систему исчисления ?
11) Что означает запись 0xDEADBEEF?
12) Что означает запись CCCXXI?
13) Что означает следующая запись100100100100b?
14) Справедлива ли запись: 0xB = 1100b?

1.3 Представление результатов шифрования в бинарном исчислении

Пусть, например, в результате шифрования в системе RSA порядковых номеров символов получено следующее шифрованное сообщение в десятичном исчислении:

4 141 45 216.

В двоичном исчислении (таблица 1.1) сообщение примет вид:

100 10001101 101101 11011000

Если оставить сообщение в таком виде, то это даст возможность криптоаналитику по разной длине кодов вычислить зашифрованные символы.

Если сделать коды одинаковой длины, то между ними можно не оставлять пробелы, тем самым затрудняя дешифрование.

Возникает вопрос выбора минимальной длины шифра символа в двоичном исчислении.

Количество бит в каждом символе шифртекста можно определить по максимальному возможному коду символа. Известно, что одним битом информации можно передать единицу, двумя битами – 3, четырьмя – 7:

x - 1

xx - 3

xxx - 7

xxxx - 15

xxxxx - 31

xxxxxx - 63

xxxxxxx - 127

xxxxxxxx - 255

xxxxxxxxx - 511

xxxxxxxxxx – 1023

В нашем примере максимальный код символа равен 216. Это число занимает 8 бит информации. Следовательно, для шифрования может быть выбран двоичный восьмибитный код:

00000100 10001101 00101101 11011000,

или

00000100100011010010110111011000.

1.3.1 Лабораторная работа Разложение и конвертирование текстового сообщения

При шифровании текстовых сообщений часто возникает задача, перевода сообщения из символьного в числовое представление. Рассмотрим одну из таких задач.

Постановка задачи: для заданной фразы нужно определить десятичный код каждого символа, а затем вывести всю фразу наоборот.

Порядок выполнения: рассмотрим возможный код приложения для этой цели, экранная форма которого представлена на рисунке 1.4.

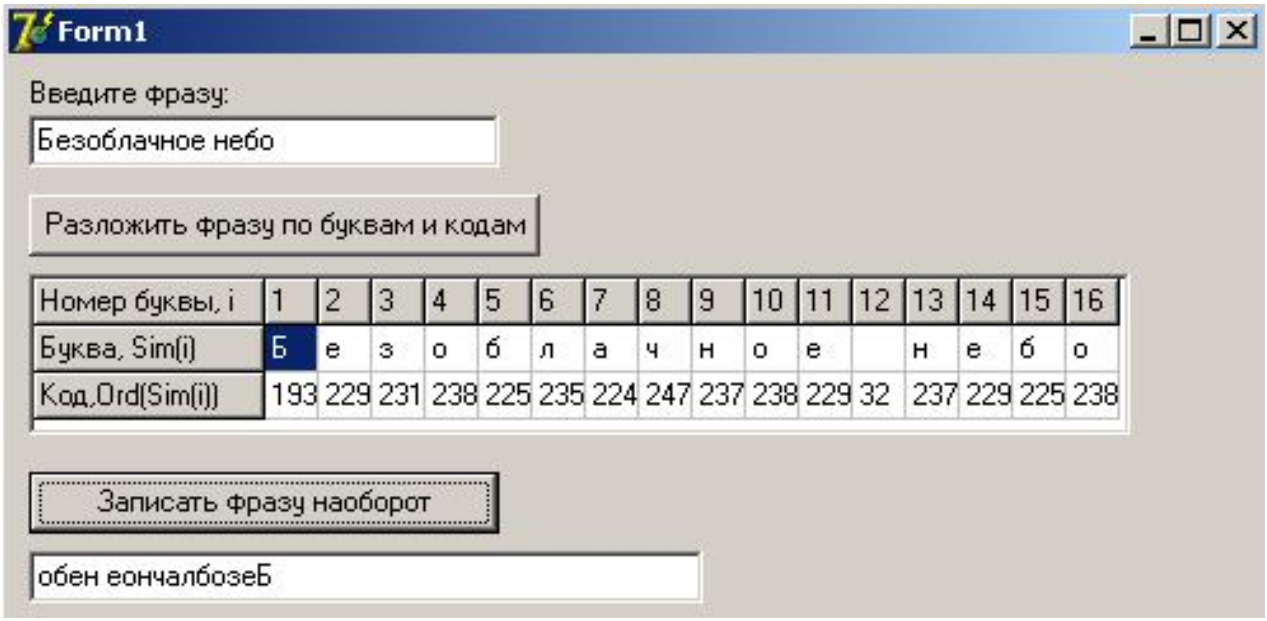


Рисунок 1.4 - Экранный интерфейс приложения

Листинг кода приложения представлен ниже

```
implementation
{$R *.dfm}
Var Dlina: integer;
procedure TForm1.Button1Click(Sender: TObject);
Var
 Stroka, Sim: string;
           i: integer;
          ch: char;
begin
 Stroka:= Edit1.Text;
 Dlina:=Length(Stroka);
 StringGrid1.ColCount:=Dlina+1;
 For i:=1 to Dlina do
  begin
```

```
    StringGrid1.Cells[i,0]:=IntToStr(i);
    Sim:=Copy(Stroka,i,1);
    StringGrid1.Cells[i,1]:=Sim;
    Ch:=Sim[1];
    StringGrid1.Cells[i,2]:=IntToStr(Ord(Ch));
   end;
   StringGrid1.Visible:=True;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
With StringGrid1 do
 begin
  ColWidths[0]:=90;
  Cells[0,0]:='Номер буквы, i';
  Cells[0,1]:='Буква, Sim(i)';
  Cells[0,2]:='Код,Ord(Sim(i))';
 end;
end;
procedure TForm1.Button2Click(Sender: TObject);
Var Slowo: string;
        i: integer;
begin
 Slowo:='';
 For i:=Dlina downto 1 do
    Slowo:=Slowo+StringGrid1.Cells[i,1];
  Edit2.Text:=Slowo;
end;
end.
```

1.3.1.1 Дополнительное задание

Вывести коды символов исходного и инвертированного выражений в двоичной системе исчисления.

1.3.1.2 Контрольные вопросы

1) В какую переменную кода запишется фраза "Безоблачное небо"?
2) Для чего служит выражение **`Length(Stroka)`**?
3) Как в коде задано число столбцов таблицы для вывода символов и чисел?
4) Что означает выражение **`Sim:=Copy(Stroka,i,1)`**? Какой тип имеют переменные **`Sim, Stroka, i`** ?
5) Что означает выражение **`Ch:= Sim[1]`** ?
6) Какой тип имеет результат выражения **`Ord(Ch)`** ? Какой смысл имеет данное выражение?
7) Что означает оператор **`StringGrid1.Visible:= true`**?
8) В чем назначение оператора **`With`**?
9) В чем назначение выражения **`Slowo:=''`** ?
10) Объясните следующее выражение: **`for i:= Dlina downto 1 do`**
11) Какой смысл имеет переменная Dlina в следующем выражении **`Dlina:=Length(Stroka)`** ? Какого типа эта переменная?
12) Расшифруйте строку кода

```
StringGrid1.Cells[i,2]:= IntToStr(Ord(Ch)).
```

1.4 Частотный анализ символов

1.4.1 Общие сведения

Криптоанализ любого шифра предполагает учет различных особенностей текстов сообщений. Одной из характеристик текстов служит повторяемость, или частота встречаемости букв текста. Данная характеристика изучается на основе эмпирических (опытных) наблюдений текстов достаточно большой длины. Наглядное представление о частотах букв дает диаграмма встречаемости (рисунок 1.5).

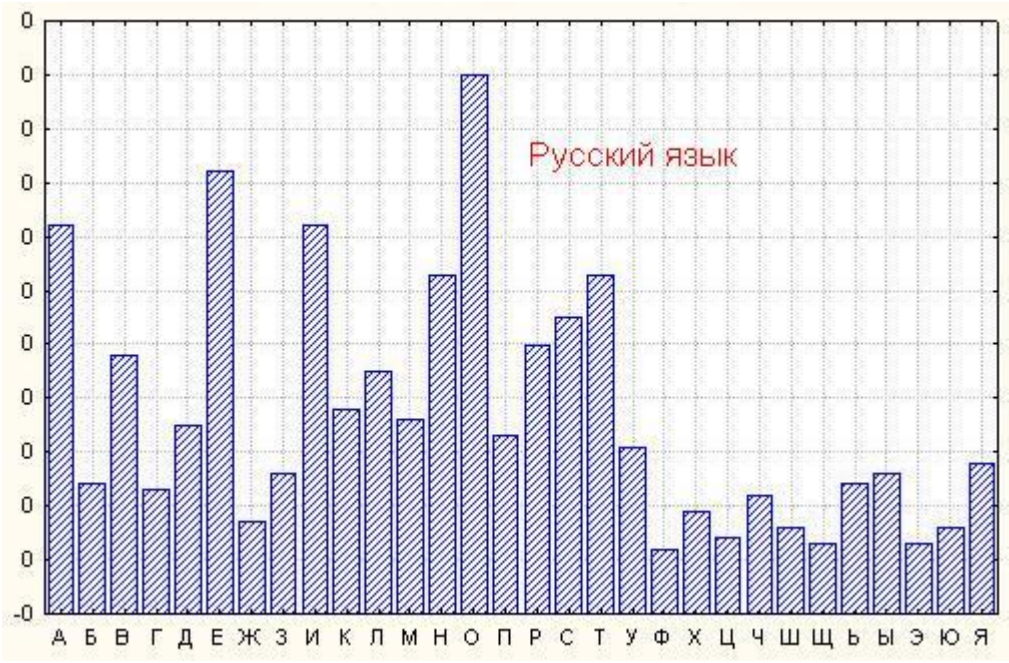


Рисунок 1.5 - Диаграмма частот букв русского языка

Имеется мнемоническое правило - запоминания десяти наиболее частых букв русского алфавита. Эти буквы составляют нелепое слово **СЕНОВАЛИТР**.

Помимо криптографии частотные характеристики открытых сообщений используются и в других сферах. Например, клавиатура компьютера, пишущей машинки - это замечательное воплощение идеи ускорения набора текста, связанное с оптимизацией расположения букв алфавита относительно друг друга в зависимости от частоты их применения.

Рассмотрим, как можно получить диаграмму, представленную на рисунке 1.5.

1.4.2 Лабораторная работа Частотный анализ символов

Постановка задачи: разработать приложение для частотного анализа символов текста на русском языке

Порядок выполнения: вариант экранной формы приложения представлен на рисунке 1.6. В верхнем левом окне формы высвечивается анализируемый текст, считанный из файла.

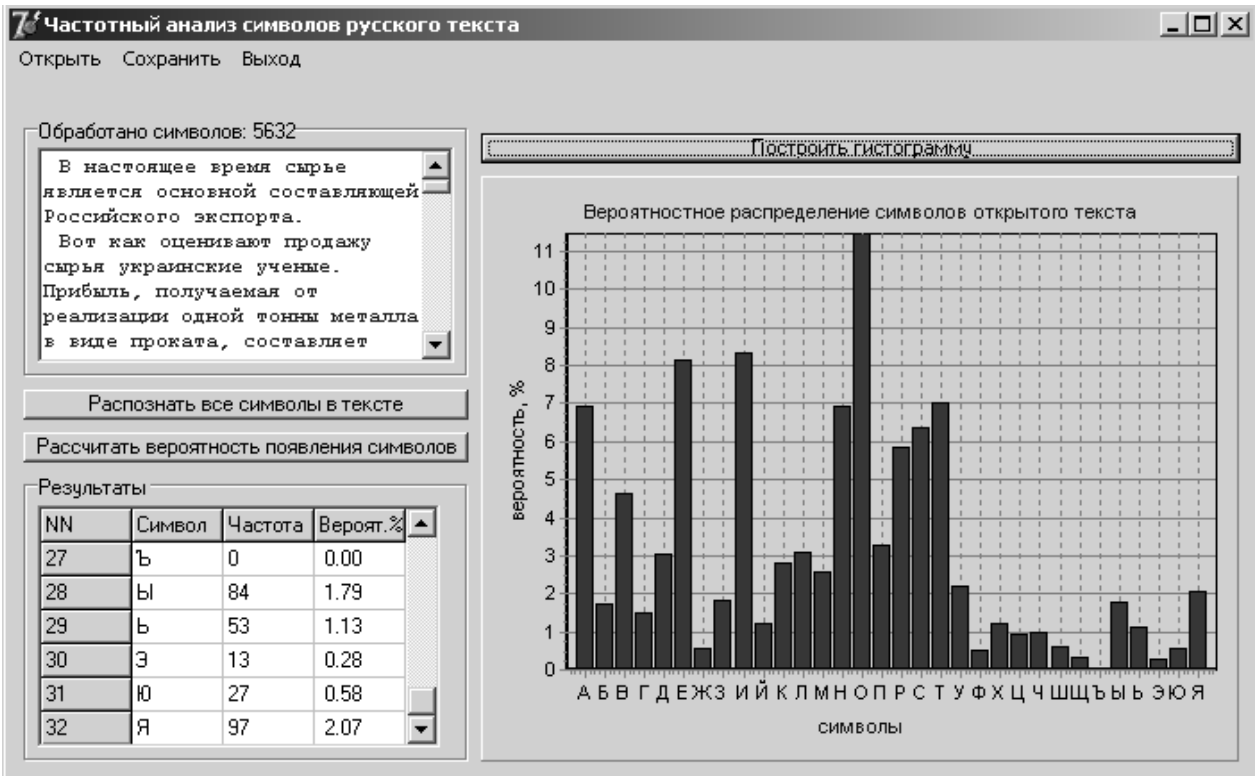


Рисунок 1.6 – Экранная форма приложения после выполнения анализа

Внизу слева представлены результаты статистической обработки символов по частоте и вероятности их встречаемости в тексте. Справа на форме представлена гистограмма вероятности встречаемости символов в тексте.

Рассмотрим последовательность написания исходного кода приложения.

Шаг 1. Формируем из компонентов **VCL** экранную форму приложения, как показано на рисунке 1.7. Используем следующие невидимые компоненты:

MainMenu – для создания меню с разделами ввода данных, сохранения результатов и выхода из программы;

OpenDialog и **SaveDialog** – для организации диалогов при считывании текста из файла и сохранения результатов, соответственно.

Видимые компоненты:

RichEdit – окно для вывода текста на экран в обогащенном формате (можно с разными шрифтами и цветом символов);

StringGrid – таблица для вывода статистической информации по частоте встречаемости символов в тексте;

Chart – для вывода графической иллюстрации результатов таблицы **StringGrid**.

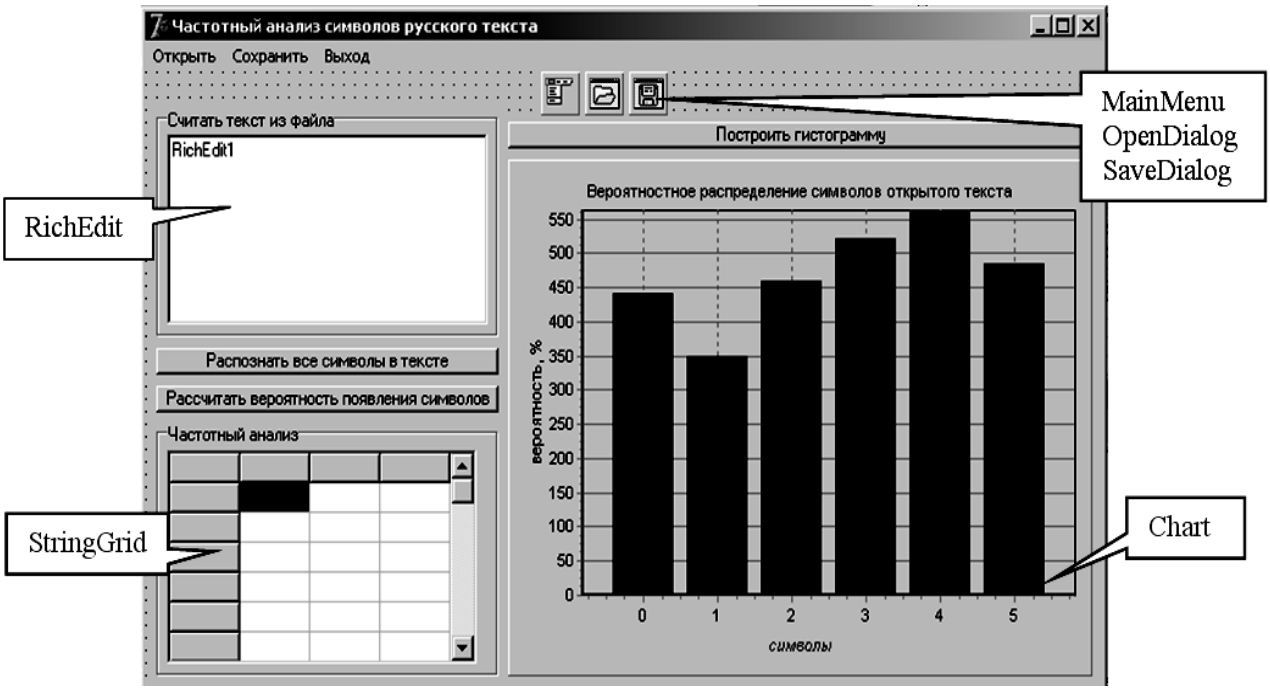


Рисунок 1.7 – Компоненты на экранной форме приложения

Использованы также три кнопки **Button** и, в качестве элементов дизайна, два компонента **GroupBox**.

Шаг 2. В разделе **Implementation**, «видимом» во всех процедурах приложения, описываем необходимые массивы **Sim, M, P** и переменную **Wsego:**

```
implementation
{$R *.dfm}
Var
 Sim: array[1..32] of string; {массив хранения русских символов}
   M: array[1..32] of integer; {массив для частоты символов}
   P: array[1..32] of real;  {массив для вероятности символов}
Wsego: word;
```

Шаг 3. В кнопку «Открыть» вписываем код процедуры поиска и считывания текстового файла в формате ***.rtf**:

```
procedure TForm1.Open1Click(Sender: TObject);
begin
 OpenDialog1.Execute;
```

```
 RichEdit1.Lines.LoadFromFile(OpenDialog1.FileName);
end;
```

Шаг 4. Дважды кликнув мышью на свободном поле формы, создаем процедуру начального заполнения заголовков таблицы **StringGrid**:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
With StringGrid1 do
 begin
  Cells[0,0]:='NN';
  Cells[1,0]:='Символ';
  Cells[2,0]:='Частота';
  Cells[3,0]:='Вероят.%';
 end;
end;
```

Шаг 5. В кнопку «Распознать» вводим код процедуры расчета частоты встречаемости символов считанного текста:

```
procedure TForm1.Button1Click(Sender: TObject);
Var
      i,j,k,poz: integer;
         Stroka: string;
         Simwol: string[1];
begin
 With RichEdit1 do
 begin
  Wsego:=0;
  {в каждой строке текста}
  For i:=0 to Lines.Count do
  begin
   {перебираем каждый символ строки В ВЕРХНЕМ РЕГИСТРЕ..}
   Stroka:=AnsiUpperCase(Lines.Strings[i]);
   For j:=1 to Length(Stroka) do
   begin
    Wsego:=Wsego+1;
    Simwol:=Copy(Stroka,j,1);
    {если это символ русского алфавита}
```

```
      poz:=0; {номер позиции символа в алфавите}
      For k:=192 to 223 do  {коды русских букв верхнего регистра}
       begin
        poz:=poz+1;
        If Chr(k)= Simwol then M[poz]:=M[poz]+1;
       end;
    end;{j}
   end; {i}
  end;  {With}
GroupBox1.Caption:=Concat
                    ('Обработано символов:', IntToStr (Wsego));
end;     {Button1Click}
```

Шаг 6. В кнопку «Рассчитать вероятность появления символов» вводим код процедуры расчета:

```
procedure TForm1.Button2Click(Sender: TObject);
Var i,n: integer;
    ss: string;
begin
 GroupBox2.Caption:='Результаты ';
 n:=0;
 For i:=1 to 32 do n:=n+M[i];
 For i:=1 to 32 do
 begin
  StringGrid1.Cells[0,i]:=IntToStr(i);
  Sim[i]:=Chr(i+191);
  StringGrid1.Cells[1,i]:=Sim[i];
  StringGrid1.Cells[2,i]:=IntToStr(M[i]);
  P[i]:=M[i]/n*100; Str(P[i]:5:2,ss);
  StringGrid1.Cells[3,i]:=ss;
 end;
 end;
```

Шаг 7. В кнопку «Построить гистограмму» вводим код процедуры:

```
procedure TForm1.Button3Click(Sender: TObject);
Var i: integer;
begin
```

```
For i:=1 to 32 do
   Chart1.Series[0].Add(P[i],Sim[i]);
end;
end.
```

Приложение готово, можно запускать на выполнение. Для его работы необходимо создать на диске файл в формате ***.rtf** с достаточно большим (от 5000 знаков) текстом.

1.4.2.1 Дополнительное задание

Дополните разработанное приложение процедурой, которая бы

а) упорядочивала символы алфавита по частоте их использования;

б) выводила на экран частотную гистограмму для упорядоченной последовательности символов;

в) выводила в отдельное окно 10 наиболее часто встречающихся символов.

1.4.2.2 Контрольные вопросы

1) Найдите на клавиатуре буквы слова СЕНОВАЛИТР. Чем объяснить расположение букв на клавиатуре компьютера?
2) Как соотносятся понятия «частота» и «вероятность» встречаемости букв текста?
3) Что означает выражение **RichEdit1.Lines.LoadFromFile(Open Dialog1.FileName?**
4) Как сделать, чтобы «шапка» таблицы **StringGrid** заполнялась при запуске приложения?
5) Каким образом все символы текста перевести в один регистр, например, верхний?
6) Что означает выражение **RichEdid1.Lines.Count**?
7) Какое значение имеет переменная **Simwol** в выражении

`k:=192; Simwol:=Chr(k)` ?

8) Что означает идентификатор **`P[i]`** в процедуре **`Button2.Click`**?
9) Сколько битов требуется, чтобы передать число 7?
10) Какое максимальное число можно зашифровать восемью битами (в бинарном представлении) ?
11) Сколько позиций потребуется для бинарного представления числа 17?
12) Для чего используется компонент **`Chart`**?
13) Что означает строка кода **`Chart1.Series[0].Add(P[i],Sim[i])`** ?
14) Согласно мнемоническому правилу запоминания какое слово образуют десять наиболее часто используемых букв русского алфавита?
15) Что иллюстрирует следующая схема?

x - 1
xx - 3
xxx - 7
xxxx - 15
xxxxx - 31
xxxxxx - 63
xxxxxxx - 127
xxxxxxxx - 255
xxxxxxxxx - 511
xxxxxxxxxx – 1023

16) Какое наименьшее число символов двоичного представления для данного десятичного шифртекста 4 141 45 216?

2 Классические шифры

2.1 Шифр Цезаря

По современной классификации шифр Цезаря относится к одноалфавитным шифрам простой замены. В таком шифре ключом является таблица подстановки, однозначно определяющая, какой символ шифрограммы будет заменять определенный символ исходного текста (таблица 2.1).

Таблица 2.1 - Таблица подстановки шифра Цезаря

Исходный алфавит																										
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	…
Алфавит подстановки																										
Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	…

Ключ в шифре Цезаря зафиксирован и определяется числом — количеством символов, на которые необходимо сдвинуть исходный алфавит, чтобы получить алфавит подстановки.

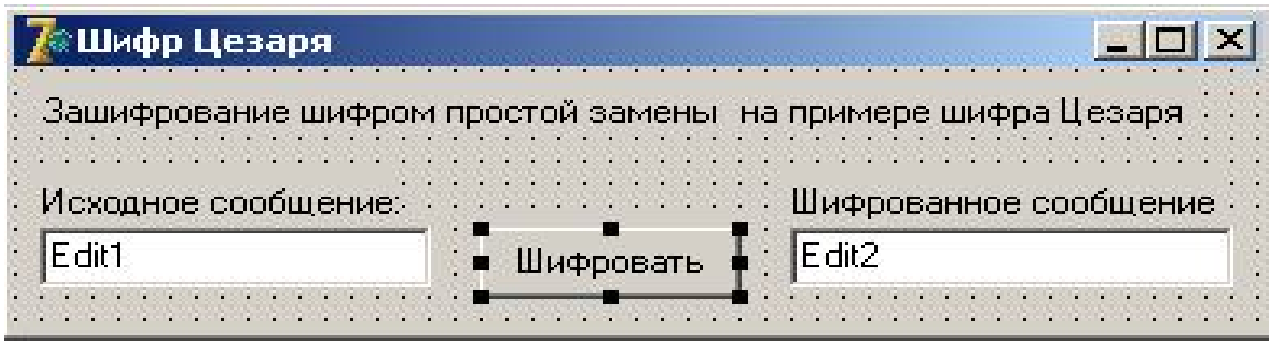
В классическом варианте алфавит просто сдвигался циклически (удаленные символы дописывались в конец) влево на три буквы. Это означает, что вместо буквы '**А**' будет записана буква '**Г**', вместо '**Б**' — буква '**Д**' и т. д. А вместо буквы "**Э**" в шифротексте появится буква '**А**'.

Например, слово «УНИВЕРСИТЕТ» после подстановки примет вид: «ЦРЛЕИУФЛХИХ».

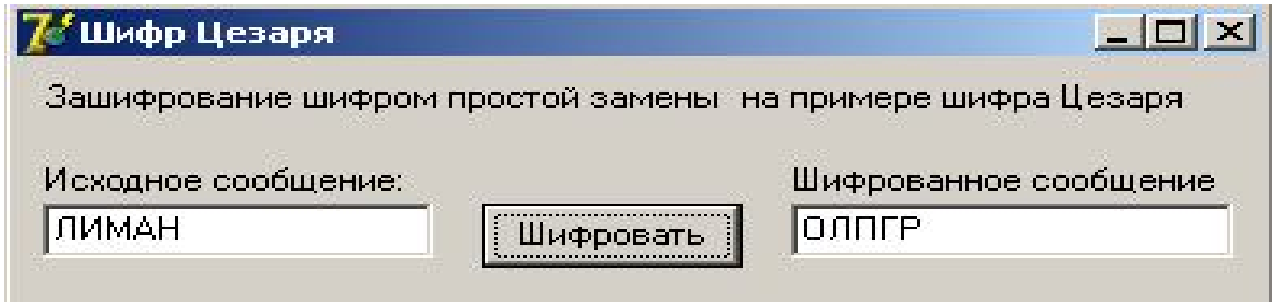
2.1.1 Лабораторная работа Шифр Цезаря

Постановка задачи: разработать приложение для зашифрования шифром Цезаря.

Порядок выполнения: создадим экранную форму с использованием компонентов **`Label, Edit`** и кнопки **`Button`** (рисунок 2.1, а).



а)



б)

Рисунок 2.1 – Экранная форма приложения для шифра Цезаря до (а) и после (б) старта

В программный код клавиши **Button** запишем следующий код приложения:

```
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
Const
  Alfavit: array [1..32] of char=
('А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н',
'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы',
'Ь', 'Э', 'Ю', 'Я'); {русский алфавит из 32 букв – без Ё}
Var
   i, j, k: integer;
   Stroka, New_Stroka: string;
   A: string[1];
begin
 Stroka:=Edit1.Text;   {ИСХОДНОЕ СООБЩЕНИЕ}
 New_Stroka:='';
{последовательно перебираем позиции символов в константе Stroka}
For i:=1 to Length(Stroka) do
```

```
 begin
A:=Copy(Stroka,i,1)//выделяем символ в позиции i константы Stroka
                   // и присваиваем его значение переменной A
 For j:=1 to 32 do // определяем порядковый номер j символа A
                   // в алфавите Alfavit
 If A=Alfavit[j] then
  begin
 k:=j+3; If k>32 then k:=k-32; // меняем порядковый номер j на k
                               // сдвигаем на три позиции
{Записываем сдвинутый символ алфавита в переменную}
New_Stroka:=Concat(New_Stroka, Alfavit[k]);
End;
End;
{Выводим в окно Edit2 шифрованное сообщение}
 Edit2.Text:=New_Stroka;
 Edit2.Visible:=True;
End;
End.
```

После старта программы и нажатия кнопки **Button** в переменную **Stroka** считывается текст исходного сообщения из окна **Memo1**. Затем зашифрованный текст, полученный сдвигом алфавита влево на 3 буквы, выводится в окно **Memo2** (рисунок 2.1, б).

Замечание: описанное приложение работает только с заглавными буквами русского алфавита.

2.1.2 Дополнительное задание

Добавьте на экранную форму еще одну кнопку **Button «Расшифровать»** и впишите в нее программный код, который

а) считывает шифрованное сообщение в окне **Edit2**;

б) расшифровывает его шифром Цезаря;

в) выводит расшифрованное сообщение в окно **Edit1**.

* Дополните приложение для шифрования и расшифрования текста содержащего все буквы русского алфавита (прописные и строчные) и знаки препинания.

2.1.3 Контрольные вопросы

1) К каким шифрам относится шифр Цезаря?
2) Как с использованием шифра Цезаря будет зашифровано слово «УНИВЕРСИТЕТ»?
3) Чему равно значение переменной **A** в выражении:

```
s:= 'УНИВЕРСИТЕТ';
A:= Length(s);
```

4) Какой тип переменных **s** и **A** в предыдущем выражении?
5) Чему равно значение переменной **A** в выражении:

```
s:= 'УНИВЕРСИТЕТ';
A:= Copy(s,3,2);
```

6) Какой тип переменной **A** в предыдущем выражении?
7) Чему равно значение **s** после выполнения операторов:

```
s:= 'УНИВЕРСИТЕТ';
s:= concat('Государственный ',s)
```

?

8) Что означает выражение: `Var A: string[1]`?
9) Чем отличаются объявления идентификаторов после ключевых слов **const** и **var**?
10) Объясните назначение выражения `k:=j+3; If k>32 then k:=k-32`?

2.2 Одноалфавитный шифр простой замены

Один из недостатков шифра Цезаря – фиксированное число символов, на которое сдвигается алфавит шифрования относительно алфавита открытого текста. По современной классификации шифр Цезаря относится к одноалфавитным шифрам простой замены. В таком шифре алфавит шифрования может быть сдвинут на любое

количество символов. Таким образом, ключ в шифре простой замены определяется числом — количеством символов, на которые необходимо сдвинуть исходный алфавит, чтобы получить алфавит шифрования. Повышенная устойчивость шифра к взлому обеспечивается тем, что ключ может меняться после каждого сообщения.

2.3 Процедура криптоанализа шифра простой замены

Задача криптоанализа состоит в несанкционированном раскрытии содержания шифрованного сообщения, применяемого шифра и используемого ключа.

Криптоаналитику приходится применять к перехваченному сообщению всевозможные шифры и рассматривать различные варианты ключей шифрования.

В случае анализа шифра простой замены задача заключается в последовательном переборе величины сдвига алфавитов открытого и зашифрованного сообщений $\Delta \in (1, 2, \ldots, N)$, где N – количество символов в алфавите.

В качестве примера напишем приложение для поиска ключа шифра постой замены. При этом будем полагать, что криптоаналитику известно, что для зашифрования использован алфавит русского языка.

2.3.1 Лабораторная работа Шифр простой замены. Процедура криптоанализа шифра простой замены.

Постановка задачи: разработать приложение для:

а) зашифрования шифром простой замены

б) криптоанализа шифра простой замены

Порядок выполнения: разработку приложения выполним в следующей последовательности:

Шаг 1. Разместим на экранной форме приложения компоненты в соответствии с рисунком 2.2.

Шаг 2. Настроим свойства компонента **`StringGrid`** в инспекторе объектов:

`ColCount` = 33 – количество столбцов;

RowCount = 3 – количество строк;

DefaultColWidth = 20 - ширина колонки, в пикселях;

DefaultRowHeight = 18 – высота строки.

Шаг 3. Для каждого из двух компонентов **Memo** через свойство **Lines** войдем в окно редактора **String List Editor** (рисунок 2.3) и клавишей **Enter** расширим область ввода текста до 20 строк.

Шаг 4. Настроим связь компонентов **LabeledEdit** и **UpDown**. С этой целью для компонента **UpDown** в инспекторе объектов:

а) в свойстве **Associate** укажем имя компонента **LabeledEdit**;

б) в свойствах **min** и **max** укажем предельные значения сдвига (ключа) шифров 0 и 32, соответственно;

в) в свойстве **Position** укажем текущее значение сдвига, равное 0.

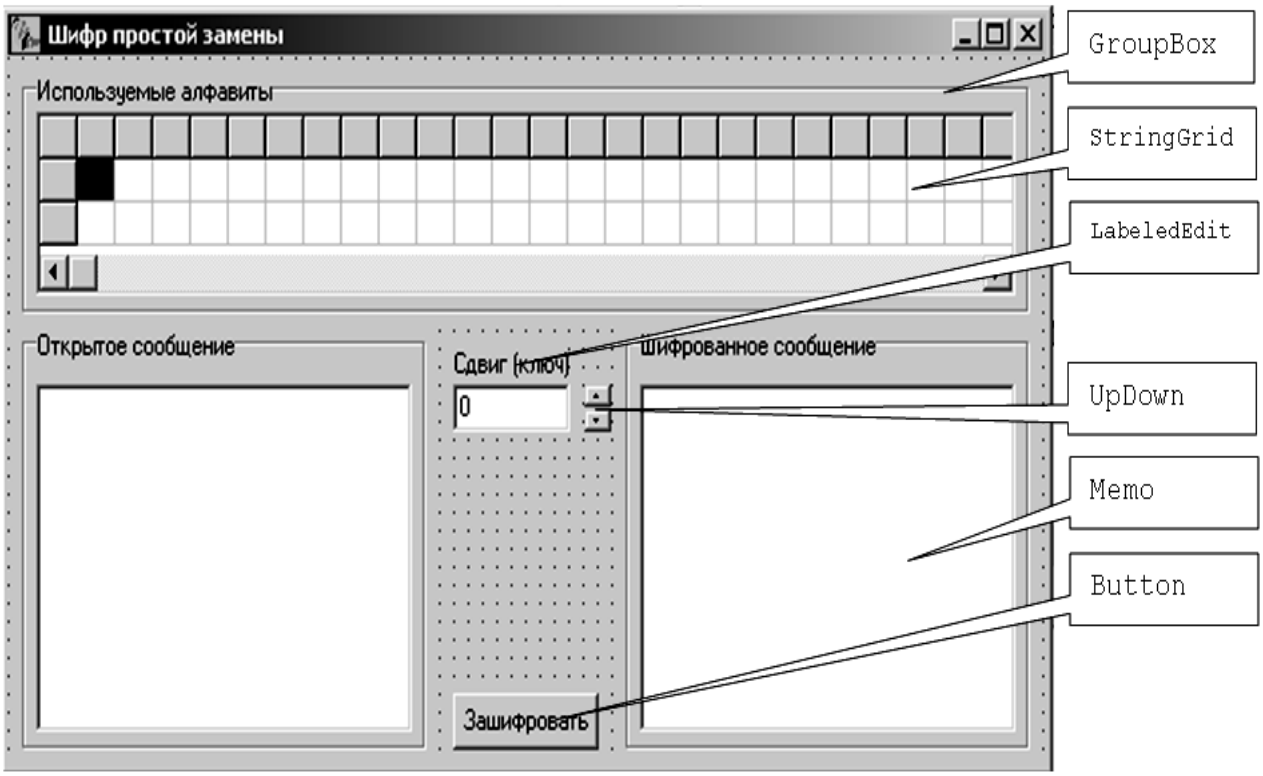


Рисунок 2.2 – Внешний вид экранной формы приложения

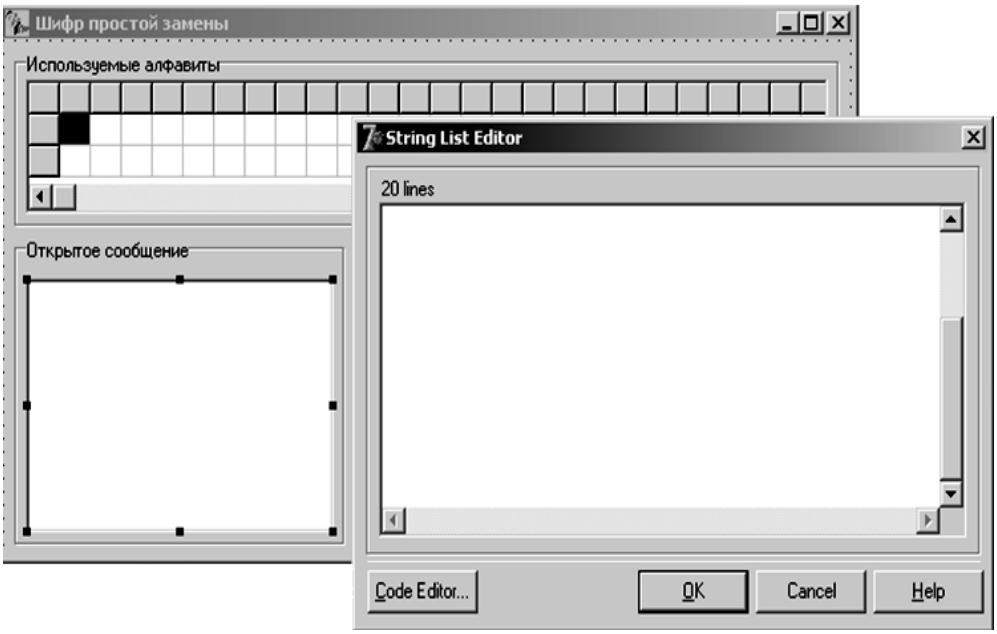


Рисунок 2.3 – Окно редактора строк компонента **Memo String List Editor**

Шаг 5. Введем соответствующие названия в заголовки компонентов **Form1, GroupBox, LabeleEdit** и **Button**.

После этого приступаем к написанию программного кода приложения.

Шаг 6. Перейдем в окно редактора кода и перед всеми будущими процедурами опишем общие для них константы и переменные:

```
implementation

{$R *.dfm}
Const
Alf_1: array[1..32] of char =
('А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н',
'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы',
'Ь', 'Э', 'Ю', 'Я');
Alf_1: array[1..32] of char =
('а', 'б', 'в', 'г', 'д', 'е', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н',
'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы',
'ь', 'э', 'ю', 'я');
Var KK: integer; {сдвиг-шифр алфавита}
```

Шаг 7. Дважды кликнув мышью по экранной форме, оказываемся в теле авто-

матически созданной процедуры **TForm1.FormCreate(Sender: TObject)**. Код, записанный в данную процедуру, автоматически выполняется при запуске формы.

Запишем в данную процедуру следующий код:

```
procedure TForm1.FormCreate(Sender: TObject);
Var i: integer;
begin
KK:=0;
with StrinGrid1 do
begin
 ColWidths[0]:=80;          Cells[0,0]:='№№';
 Cells[0,1]:='Открытый';    Cells[0,2]:='Шифр';
 For i:=1 to 32 do
 begin
Cells[i,0]:=IntToStr(i);
Cells[i,1]:=Alf_1[i];
Cells[i,2]:=Alf_1[i];
end;
end; {Width}
end; {FormCreate}
```

Теперь при каждом старте приложения на экране будем наблюдать заполненную таблицу в соответствии с рисунком 2.4.

Рисунок 2.4 – Внешний вид компонента **StringGrid** после запуска приложения

Шаг 8. Сделаем так, чтобы при изменении ключа (нажатии **UpDown** после за-

пуска приложения) автоматически обновлялся алфавит шифра в нижней строке таблицы. Для этого дважды кликнем на компоненте **UpDown**, и в тело автоматически сгенерированной процедуры **UpDownClick** впишем код:

```
procedure TForm1.UpDown1Click(Sender: TObject; Button: TUDBtnType);
Var i: integer;
begin
 KK:=StrToInt(LabelEdit1.Text);
 For i:=1 to 32 do
   begin
     if i+KK<=32 then StringGrid1.Cells[i,2]:=Alf_1[i+KK]
                 else StringGrid1.Cells[i,2]:=Alf_1[i+KK-32];
end;
  end;   {UpDown}
```

Теперь при каждом клике на компоненте **UpDown** после старта приложения автоматически будет изменяться и шрифт шифра.

Шаг 9. Остается записать код процедуры, которая считывает открытое сообщение из окна **Memo1**, зашифровывает его на текущем ключе и выводит шифрованное сообщение в окно **Memo2**. Код процедуры запишем в кнопку **Button**:

```
procedure TForm1.Button1Click(Sender: TObject);
VAR i, j, k, x: INTEGER;
 Stroka, New_Stroka: string;
   Sim: string[1];
begin
   For i:=0 to Memo1.Lines.Count-1 do
   begin
    Stroka:=Memo1.Lines.Strings[i];
    New_Stroka:='';
{обработка строки}
For j:=1 to Length(Stroka) do
begin
```

```
 Sim:=Copy(Stroka,j,1);
  For k:=1 to 32 do
   If (Sim=Alf_1[k]) or (Sim=Alf_2[k]) then
    begin
       x:=k+KK;
       if x>32 then x:=x-32;
       New_Stroka:=Concat(New_Stroka,Alf_1[x]);
       end;
     Memo2.Lines.Strings[i]:= New_Stroka;
   end;
  end;  {end i}
end; {Button}
```

Приложение готово. Теперь после его запуска в левом окне можно вводить исходный текст открытого сообщения, выбирать ключ и в правом окне получать текст зашифрованного сообщения.

Замечание: в процессе зашифрования открытый текст переводится в верхний регистр, пробелы и знаки препинания уничтожаются. Для примера на рисунке 2.5 представлен вид экрана с текстом, зашифрованном со сдвигом, равным нулю.

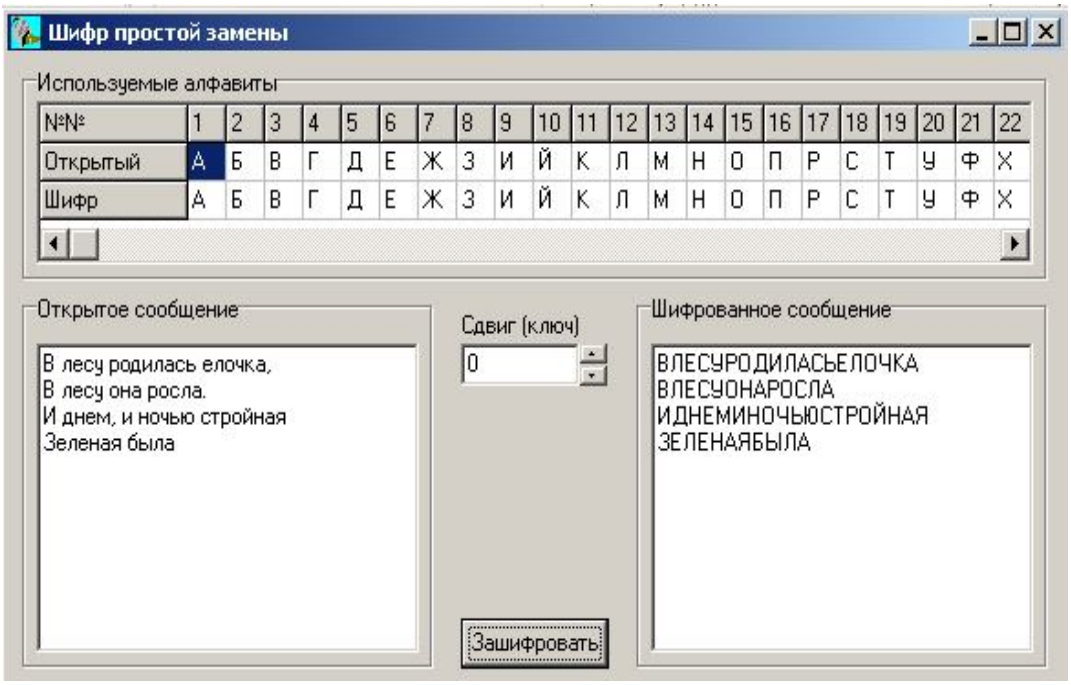


Рисунок 2.5 – Пример работы приложения

Для решения задачи криптоанализа за основу возьмем разработанное приложение, в одном окне которого выводится зашифрованное сообщение. Это сообщение будем использовать для криптоанализа.

Последовательность написания приложения:

Шаг 1. Изменим экранную форму приложения в соответствии с рисунком 2.6

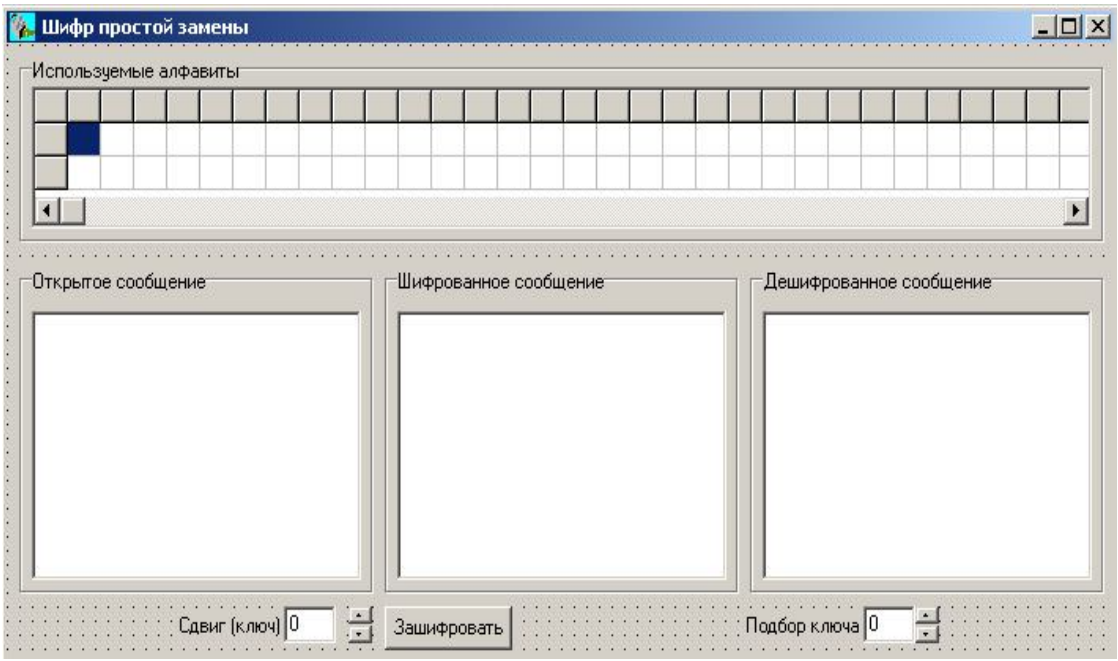


Рисунок 2.6 – Внешний вид приложения с процедурой криптоанализа

Шаг 2. В тело автоматически сгенерированной процедуры **UpDown2Click** впишем программный код:

```
procedure TForm1.UpDown2Click(Sender: TObject; Button: TUDBtnType);
var i, j, k, x: integer;
 Stroka, New_Stroka: string;
   Sim: string[1];
begin
KK:=StrToInt(LabelEdit2.Text);
For i:=0 to Memo2.Lines.Count-1 do
   begin
    Stroka:=Memo2.Lines.Strings[i];
```

```
    New_Stroka:='';
{обработка строки}
For j:=1 to Length(Stroka) do
begin
 Sim:=Copy(Stroka,j,1);
  For k:=1 to 32 do
   If (Sim=Alf_1[k]) or (Sim=Alf_2[k]) then
    begin
       x:=k-KK;
       if x>32 then x:=x-32;
       New_Stroka:=Concat(New_Stroka,Alf_1[x]);
       end;
     // Memo3.Clear
     Memo3.Lines.Strings[i]:= New_Stroka;
   end;
  end;  {end i}
end;
```

Процедура готова. После компиляции приложения и запуска на выполнение пользователь в третьем окне будет получать дешифрованный текст, соответствующий подобранному ключу. При этом можно уничтожить открытое сообщение в первом окне и замаскировать открытый ключ.

2.3.1.1 Контрольные вопросы

1) В чем состоит недостаток шифра Цезаря?
2) Чем определяется ключ в шифре простой замены?
3) Чем обеспечивается устойчивость к взлому одноалфавитного шифра простой замены?
4) Как настраивается связь компонентов **LabelEdit** и **Updown** в инспекторе объектов?
5) В чем заключается задача криптоанализа?
6) Для чего предназначен фрагмент программного кода?

```
begin
KK:=StrToInt(LabelEdit1.Text);
For i:=1 to 32 do
begin
if i+KK<=32 then StringGrid1.Cells[i,2]:=Alf_1[i+KK]
else StringGrid1.Cells[i,2]:=Alf_1[i+KK-32];
```

7) Объясните назначение фрагмента программного кода

```
For i:=1 to 32 do
begin
Cells[i,0]:=IntToStr(i);
Cells[i,1]:=Alf_1[i];
Cells[i,2]:=Alf_1[i];
```

8) В чем состоит задача криптоанализа шифра простой замены?
9) Укажите тип переменной **Sim** в выражении **Sim:=Copy(Stroka,j,1)**
10) Укажите смысл значения переменной Sim в выражении

Sim:=Copy(Stroka,i,1)

11) Какого типа данных переменная **New_Stroka** в выражении **New_Stroka:=Concat(New_Stroka,Alf_1[x])**?
12) Для каких целей используются компоненты **UpDown** и **UpDownClick**?
13) Расскажите о свойствах компонента **StringGrid**: **ColCount; RowCount**; **DefaultColWidth**; **DefaultRowHeigh**

2.4 Многоалфавитный шифр. Матрица Вижинера

2.4.1 Общие сведения

В XVI в аббат Иоганнес Тритемий разработал сдвиговую таблицу (таблица 2.2) для многоалфавитного зашифрования самым простым из возможных способов: первая буква текста шифруется первым алфавитом, вторая буква — вторым и т. д. Алфавитом открытого текста служил алфавит первой строки.

Шифр Тритемия является также первым нетривиальным примером периодического шифра. Так называется многоалфавитный шифр, правило за-

шифрования которого состоит в использовании периодически повторяющейся последовательности простых замен.

Таблица 2.2 - Таблица Тритемия

А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж
И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

В 1553 г. Джованни Баттиста Белазо предложил использовать для многоалфавитного шифра буквенный, легко запоминаемый ключ, который он назвал паролем. Паролем могло служить слово или фраза. Пароль периодически записывался над открытым текстом. Буква пароля, расположенная над буквой текста, указывала на алфавит таблицы, который использовался для зашифрования этой буквы.

Блез де Вижинер предложил использовать в качестве ключа текст самого сообщения. Такой шифр был назван самоключом. Первая строка служит алфавитом открытого текста, а первый столбец — алфавитом ключа (таблица 2.3). Для зашифрования открытого сообщения Вижинер предлагал в качестве *ключевой последовательности* использовать само сообщение.

2.4.2 Лабораторная работа Шифры многоалфавитной замены

Постановка задачи: разработать приложение для зашифрования шифром полиалфавитной замены.

Порядок выполнения: внешний вид приложения представлен на рисунке 2.7. В сдвиговой таблице алфавитов верхняя строка отображает символы открытого сообщения. Левый столбец служит для выбора символов ключа, указывающих на соответствующую строку алфавита зашифрования.

В качестве примера использования полиалфавитного шифра разработаем приложение для шифрования сообщений с ключом по принципу шифра Белазо.

Разными цветами на рисунке показана схема зашифрования первых четырех букв фразы «БЕЗОБЛАЧНОЕ НЕБО» на ключе «ЗОНД»:

а) букву «**Б**» ищем на строке алфавита с буквой «**З**» в первой колонке, получаем букву «**И**»;

б) букву «**Е**» ищем на строке алфавита с буквой «**О**» в первой колонке, получаем букву «**У**»**;**

в) букву «**З**» зашифровываем на алфавите с буквой «**Н**» в первой колонке, получаем букву «**Ф**», и так далее.

Таблица 2.3 - Таблица Вижинера

	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
А	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Б	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
В	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
Г	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
Д	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
Е	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
Ж	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
З	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж
И	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
Й	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
К	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
Л	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
М	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
Н	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
О	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
П	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
Р	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
С	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
Т	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
У	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ъ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ы	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ь	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
Э	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
Ю	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
Я	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Код приложения разместим в двух процедурах. Первая из них, **FormCreate**, при запуске приложения производит заполнение алфавитами сдвиговой таблицы. При этом символы русского алфавита заданы в виде символьного массива в разделе объявления констант, «видимом» всеми процедурами приложения.

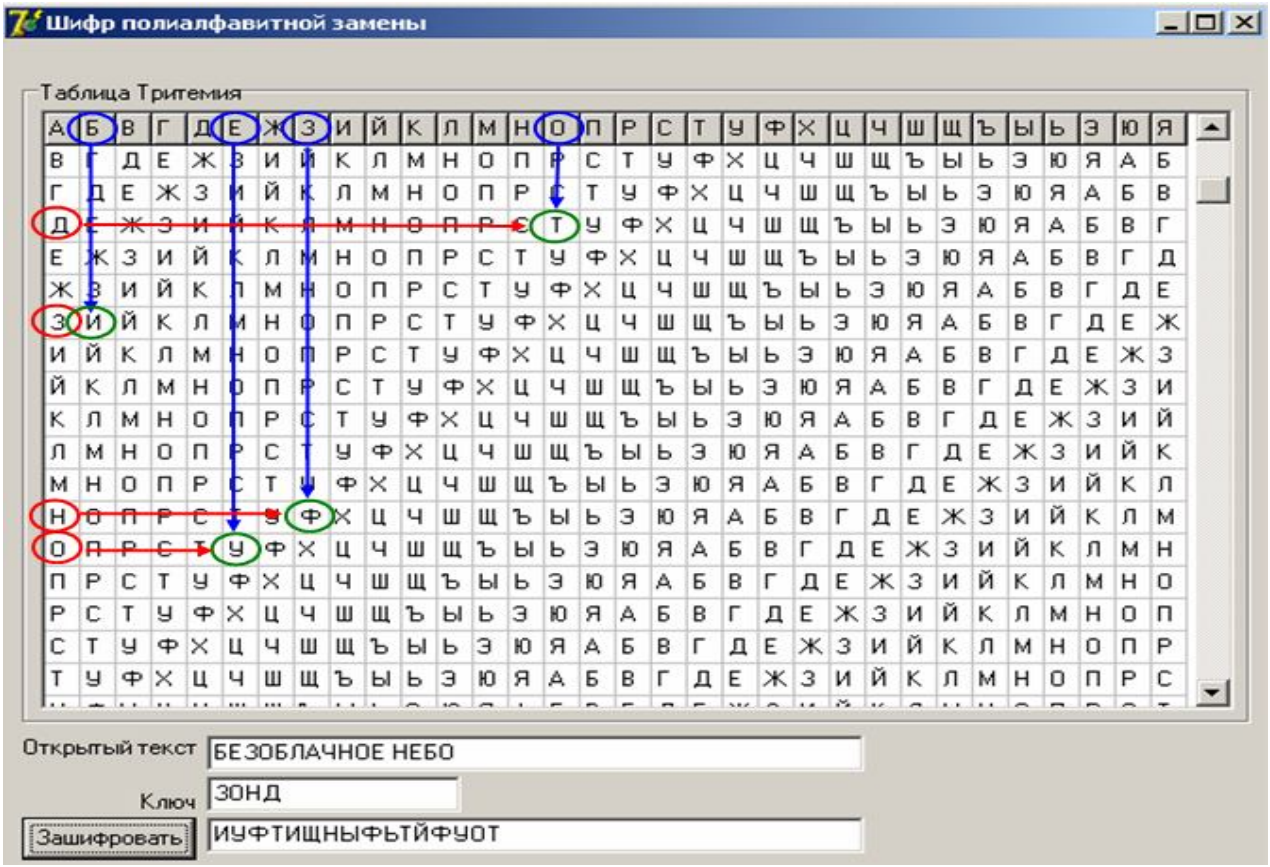


Рисунок 2.7 - Экранная форма приложения и схема шифрования открытого текста на ключе «зонд»

```
implementation
{$R *.dfm}
Const
 Alfavit: array [1..32] of char =('А','Б','В','Г','Д','Е','Ж','З',
 'И','Й','К','Л','М','Н','О','П', 'Р','С','Т','У','Ф','Х','Ц','Ч',
 'Ш','Щ','Ъ','Ы','Ь','Э','Ю','Я'); {русский алфавит из 32 букв}

procedure TForm1.FormCreate(Sender: TObject);
Var i, j, sum : integer;
begin
for i:=0 to 31 do StringGrid1.Cells[i,0]:= Alfavit[i+1];
for i:=1 to 32 do
 for j:=1 to 32 do
  begin
    sum:=i+j; If sum>32 then sum:=sum-32;
    StringGrid1.Cells[i-1,j]:= Alfavit[sum];
```

```
  end;
end;
```

Вторая процедура размещена в кнопке **Button** и вызывается по команде «Зашифровать»:

```
procedure TForm1.Button1Click(Sender: TObject);
Var Stolb, Stroka,m, i,j,k, Kluth: integer;
 New_Sim, Sim: string[1];
  Kod_Stroka: string;
begin
  Stolb:=0; k:=0;
  Kluth:=Length(Edit1.Text); j:=0;
  Kod_Stroka:='';
 For i:=1 to Length(Edit2.Text)do
  begin
   {текущий символ открытого текста}
   Sim:=Copy(Edit2.Text,i,1);
   {порядковый номер символа в открытом тексте - Nom:}
   For m:=1 to 32 do If sim=Alfavit[m-1] then Stolb:=m-1;
   {порядковый номер ключевого алфавита:}
   If j<Kluth then j:=j+1 else j:=1;
   Sim:=Copy(Edit1.Text,j,1);
   For k:=1 to 32 do
    If StringGrid1.Cells[0,k]=Sim then
      Stroka:=k-1; {номер текущего алфавита замены}
   New_Sim:= StringGrid1.Cells[Stolb,Stroka];
   Kod_stroka:=Concat(kod_Stroka,New_Sim);
  end;
  Edit3.Text:=Kod_Stroka;
end;
end.
```

Приложение готово. После компиляции и запуска на выполнение открытый текст можно зашифровать на заданном ключе и просмотреть полученное зашифро-

ванное сообщение.

2.4.2.1 Дополнительное задание

Напишите приложение для дешифрования сообщения, зашифрованного на шифре Тритемия?

2.4.2.2 Контрольные вопросы

1) Кто автор сдвиговой таблицы для многоалфавитного зашифрования?
2) В чем идея шифра Тритемия?
3) Чем шифр Белазо отличается от шифра Тритемия?
4) Кто предлагал в качестве ключевой последовательности использовать само сообщение?
5) В чем отличие шифра Вижинера от шифра Белазо?
6) Как расшифровать сообщение, зашифрованное шифром Белазо?
7) Как расшифровать сообщение, зашифрованное шифром Вижинера?
8) Что является ключом Вижинера?
9) Каково назначение процедуры **`FormCreate`**?
10) Как создать процедуру **`FormCreate`**?
11) Чем символьный тип данных отличается от строкового?
12) Что означает выражение **`Kluth:= Length(Edit1.Text)`**?
13) Для чего предназначена функция **`Concat`**?
14) Объясните смысл строки кода

```
Kod_stroka: = Concat(kod_Stroka,New_Sim)
```

15) Какого типа данных переменная **`Kod_stroka`** в предыдущем выражении?

2.5 Шифр перестановки - решетка Кардано

Выдающийся итальянский математик Джероламо Кардано состоял на службе

у папы Римского. Увлечение теорией магических квадратов привело Кардано к открытию нового класса шифров перестановок, названных решетками или трафаретами (рисунок 2.8).

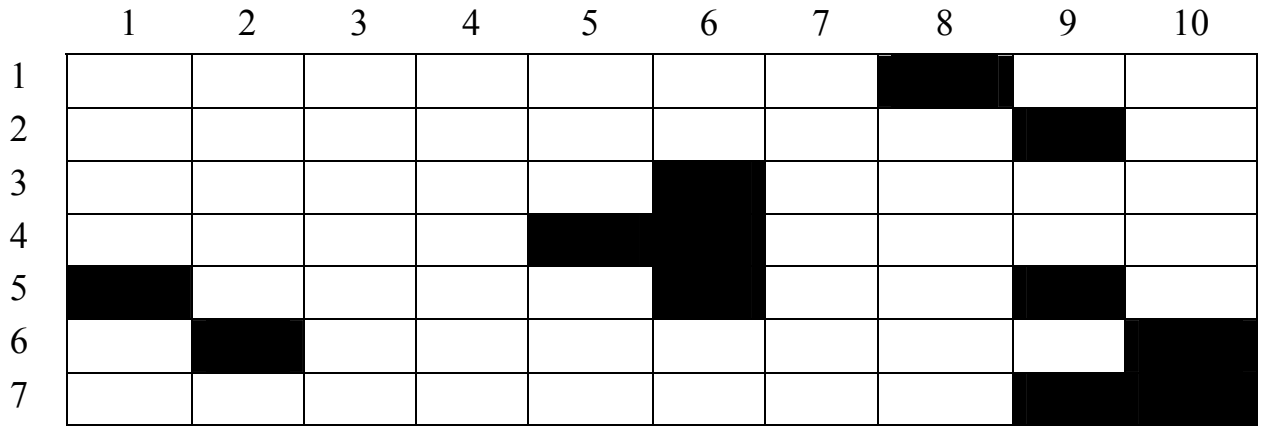


Рисунок 2.8 - Пример трафарета для шифрования

Изначально обычная решетка представляла собой лист из твердого материала, в котором через неправильные интервалы сделаны прямоугольные вырезы высотой для одной строчки и различной длины. Накладывая эту решетку на лист писчей бумаги, можно было записывать в вырезы секретное сообщение. После этого, сняв решетку, нужно было заполнить оставшиеся свободные места на листе бумаги неким текстом, маскирующим секретное сообщение.

Кардано использовал квадратную решетку, которая своими вырезами однократно покрывает всю площадь квадрата. Подобным стеганографическим методом маскировки сообщения пользовались многие известные исторические лица, например кардинал Ришелье во Франции и русский дипломат и писатель А. Грибоедов. Так, Ришелье использовал прямоугольник размера 7×10. Для длинных сообщений прямоугольник использовался несколько раз. Прорези трафарета размещались в позициях с координатами (вертикаль, горизонталь):

(1,8), (2,9), (3,6), (4,5), (4,6), (5,1), (5,6), (5,7), (5,9), (6,2), (6,10), 7,9), (7,10).

Например, следующий текст выглядит как невинное письмо (рисунок 2.9).

	1	2	3	4	5	6	7	8	9	10
1	М	О	Я	Ц	А	Р	И	Ц	А	
2	С	У	В	А	Ж	Е	Н	И	Е	М
3		П	О	С	Ы	Л	А	Ю		
4	В	Е	С	Т	Ь	В	А	М		
5	З	Д	Е	С	Ь	О	Т	К	Р	Ы
6	Л	В	Е	Р	У	В	Б	О	Г	А
7		И		Р	А	Д	О	С	Т	Ь

Рисунок 2.9 – Пример шифрованного сообщения

Однако, используя трафарет Ришелье, получим зловещую команду:

ЦЕЛЬ ВЗОРВАТЬ.

2.5.1 Лабораторная работа Шифр перестановки

Постановка задачи: разработать приложение для зашифрования сообщения шифром перестановки, в реализации Ришелье.

Порядок выполнения: внешний вид экранной формы приложения представлен на рисунке 2.10. В верхнем окне вводится текст открытого сообщения. Ключ зашифрования задается последовательностью координат ячеек таблицы, в которых будут записаны буквы сообщения.

После нажатия клавиши «Зашифровать» пустые ячейки таблицы будут заполнены буквами с помощью генератора случайных чисел.

После нажатия кнопки «Расшифровать» согласно известному ключу в ячейках таблицы остаются лишь буквы в заданных ячейках.

В примере на рисунке 2.10 слева приведена таблица с зашифрованным текстом, справа – с расшифрованным текстом. Расшифрованное сообщение выводится в нижнем окне.

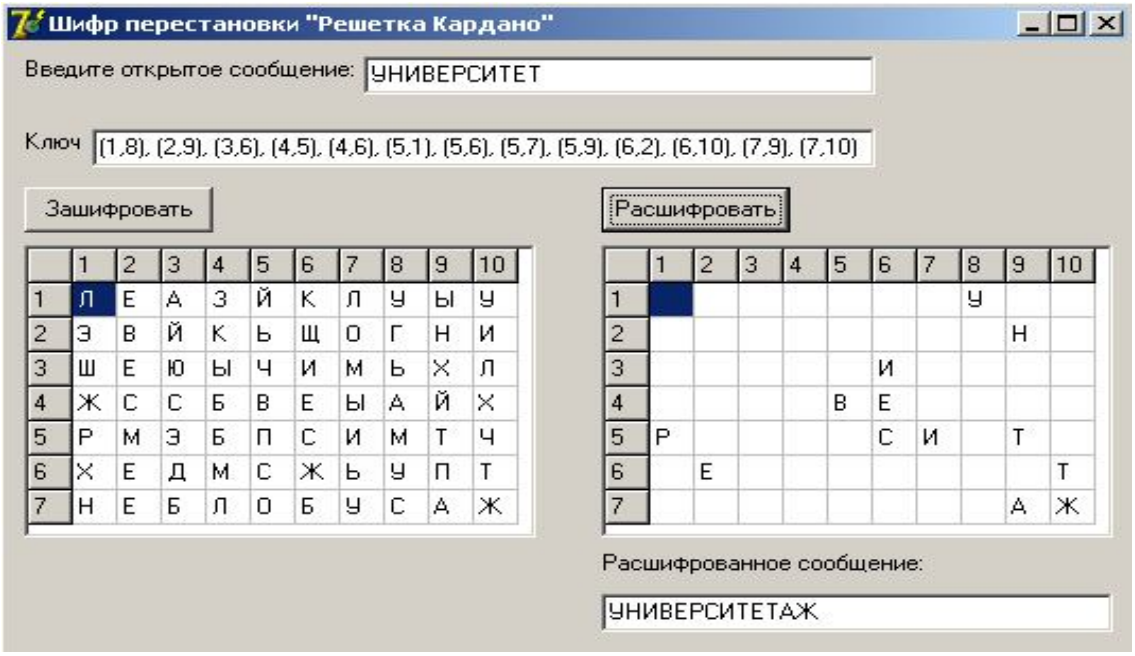


Рисунок 2.10 – Внешний вид экранной формы приложения

Код приложения представлен ниже.

В разделе **Implementation** перед всеми процедурами объявлен символьный массив букв русского алфавита. В процедуру **FormCreate**, создаваемую двумя кликами мыши на пустом поле экранной формы, вписаны координаты позиций таблиц:

```
implementation
{$R *.dfm}
Var
 Alfavit: array [1..32] of char =('А','Б','В','Г','Д','Е','Ж','З',
 'И','Й','К','Л','М','Н','О','П', 'Р','С','Т','У','Ф','Х','Ц','Ч',
 'Ш','Щ','Ъ','Ы','Ь','Э','Ю','Я');{русский алфавит из 32 букв - без Ё}
procedure TForm1.FormCreate(Sender: TObject);
Var i: integer;
begin
For i:=1 to 10 do StringGrid1.Cells[i,0]:=IntToStr(i);
For i:=1 to  7 do StringGrid1.Cells[0,i]:=IntToStr(i);
end;
```

Внешний вид экранной формы после запуска приложения показан на рисунке 2.11.

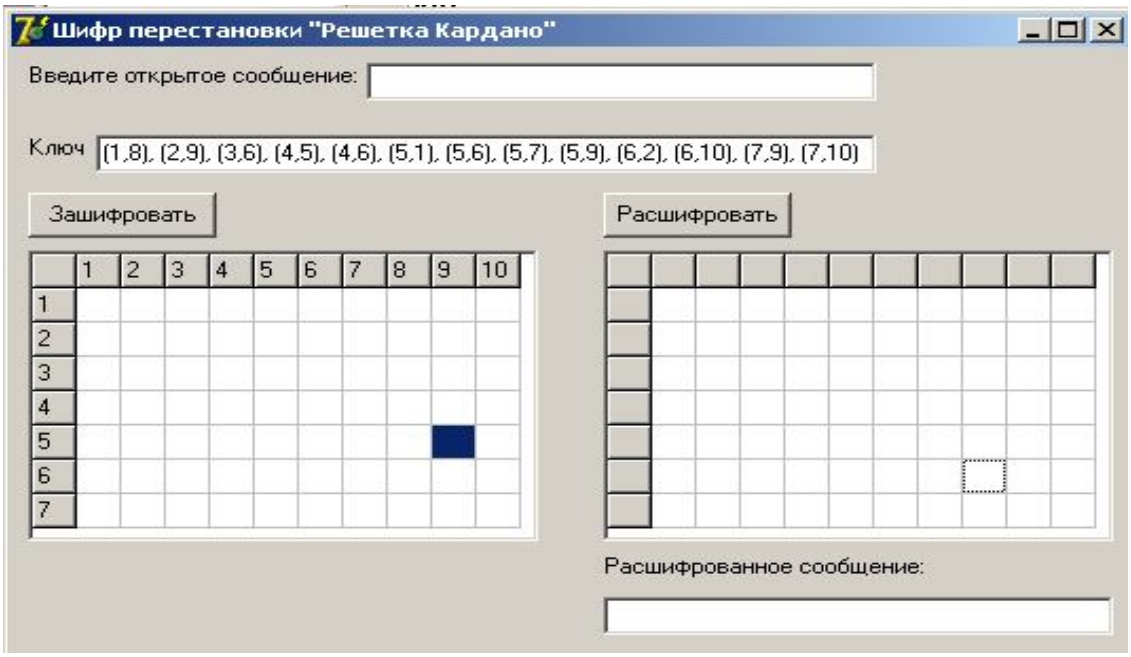


Рисунок 2.11 – Экранная форма после запуска приложения

В кнопку «Зашифровать» вводим следующий исходный текст:

```
procedure TForm1.Button1Click(Sender: TObject);
  Var
   Bukwa: string[1];
   Blok, Y, X, i, j, n, Nom, Nom1: integer;
begin
  X:=0; Y:=0; Randomize;
  For i:=1 to 10 do
    for j:=1 to 7 do
     begin
      n:=Random(32);
      StringGrid1.Cells[i,j]:=Alfavit[n];
     end;
     Blok:=Length(Edit1.Text);
     If Blok>13 then Blok:=13;
     For Nom:=1 to Blok do
     begin
       Bukwa:=Copy(Edit1.Text, Nom, 1);
       If Nom>13 then Nom1:=Nom-13 else Nom1:= Nom;
    Case Nom1 of
```

```
1:  begin Y:=1; X:=8; end;
2:  begin Y:=2; X:=9; end;
3:  begin Y:=3; X:=6; end;
4:  begin Y:=4; X:=5; end;
5:  begin Y:=4; X:=6; end;
6:  begin Y:=5; X:=1; end;
7:  begin Y:=5; X:=6; end;
8:  begin Y:=5; X:=7; end;
9:  begin Y:=5; X:=9; end;
10: begin Y:=6; X:=2; end;
11: begin Y:=6; X:=10;end;
12: begin Y:=7; X:=9; end;
13: begin Y:=7; X:=10;end;
 end; {Case}
  StringGrid1.Cells[X,Y]:= Bukwa;
 end; {do}
end;
```

В кнопку «Расшифровать» вводим код:

```
procedure TForm1.Button2Click(Sender: TObject);
Var i,Y,X: integer;
      Sim: string[1];
   Stroka: string;
begin
X:=0; Y:=0;
For i:=1 to 10 do StringGrid2.Cells[i,0]:=IntToStr(i);
For i:=1 to  7 do StringGrid2.Cells[0,i]:=IntToStr(i);
 For i:=1 to 13 do
  begin
    Case i of
1:  begin Y:=1; X:=8; end;
2:  begin Y:=2; X:=9; end;
3:  begin Y:=3; X:=6; end;
4:  begin Y:=4; X:=5; end;
```

```
5:  begin Y:=4; X:=6; end;
6:  begin Y:=5; X:=1; end;
7:  begin Y:=5; X:=6; end;
8:  begin Y:=5; X:=7; end;
9:  begin Y:=5; X:=9; end;
10: begin Y:=6; X:=2; end;
11: begin Y:=6; X:=10;end;
12: begin Y:=7; X:=9; end;
13: begin Y:=7; X:=10;end;
 end; {Case}
  StringGrid2.Cells[X,Y]:=StringGrid1.Cells[X,Y];
  Sim:=StringGrid1.Cells[X,Y];
 Stroka:=Concat(Stroka,Sim);
 Edit3.Text:=Stroka;
end;
end;
end.
```

Теперь при запуске приложения на экран будет выведена форма согласно рисунку 2.10.

2.5.1.1 Дополнительное задание

Напишите свой программный код для случая использования различных ключей зашифрования.

2.5.1.2 Контрольные вопросы

1) В чем смысл шифрования с помощью решетки Кардано?
2) К какой группе шифров относится решетка Кардано?
3) Почему на рисунке 2.10 в нижней строке с расшифрованным сообщением присутствуют лишние символы?
4) Чему равна длина ключа зашифрования в разработанном приложении?

5) Можно ли использовать для заданной таблицы (решетки) ключи зашифрования разной длины? Если да, то чему равна максимальная длина ключа?
6) Как использовать решетку, если число символов открытого сообщения значительно превышает длину ключа зашифрования?
7) Где в программном коде считывается ключ зашифрования, заданный в соответствующей строке экранной формы?
8) Что означает выражение вида **`Case Nom of`** … **`end`** в коде приложения?
9) Что означает выражение вида **`Sim:= StringGrid1. Cells [X,Y]`** в коде приложения?
10) Как в программном коде организовано случайное заполнение таблицы буквами?
11) Как в программном коде обеспечить возможность зашифрования и расшифрования на разных ключах?

3 Основы шифрования с открытым ключом

3.1 Алгоритм Эратосфена формирования простых чисел

Определение: Если целое число, большее единицы не имеет делителей, не равных единицы и самого этого числа, то такое число называется простым.

Например, 1, 3, 5, 17.

Наименьший делитель любого числа – простое число.

Теорема: Наименьший делитель числа *N* не превосходит $\sqrt{N}$.

Алгоритм Эратосфена:

1. Дана последовательность целых чисел от 2 до N

2. Первое число последовательности – простое – q. Запоминаем его. Вычеркиваем из последовательности все числа кратные q, т.е q, q^2, q^3 …

3. В полученной последовательности первое число – простое. Запоминаем его. Вычеркиваем все числа, кратные выбранному и т.д., до тех пор пока выбранное число $q \le \sqrt{N}$. Все оставшиеся в последовательности числа – простые.

3.1.1 Лабораторная работа Простые числа

Постановка задачи: разработать приложение для поиска всех простых чисел от 1 до 1000 пользуясь определением простого числа.

Порядок выполнения: экранная форма приложения представлена на рисунке 3.1.

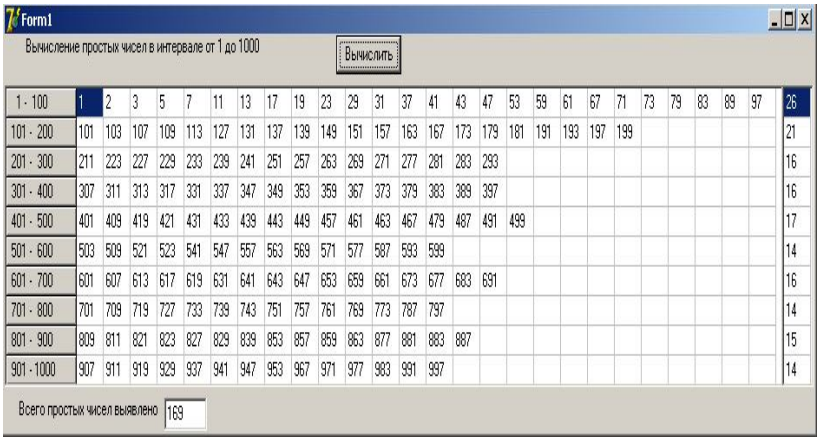


Рисунок 3.1 – Экранная форма приложения для вычисления простых чисел

Код приложения для вычисления простых чисел представлен ниже

```
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
Var
sum,     // переменная для записи общего количества простых чисел
max,     // переменная для вычисления числа столбцов в таблице
x,i,j,k: integer;          { обозначим:
       M[0] - число простых чисел в диапазоне от 1 до 100
       M[1] - число простых чисел в диапазоне от 101 до 200
                ...
       M[9] - число простых чисел в диапазоне от 901 до 1000}
M: array[0..9] of integer;
begin
  For i:=0 to 9 do M[i]:=0; // обнуляем
  for i:=1 to 1000 do       // цикл перебора
  begin
   k:=0;
   for j:=1 to i do       // перебор чисел от 1 до текущего i
{если остаток от деления равен нулю, то на единицу увеличиваем k}
     If Frac(i/j)=0 then k:=k+1;
     If k<=2 then // если k<=2, т.е. число делится само на себя
      // и на 1,то определяем клетку таблицы и выводим это число
     begin
      x:=Trunc(i/100);      //  усекаем, берем целую часть
      M[x]:=M[x]+1;
      StringGrid1.Cells[M[x],x]:=IntToStr(i);
     end;
   end;
  Max:=0; Sum:=0;
  For i:=0 to 9 do
  begin
  Sum:=Sum+M[i];
```

```
  If Max<M[i] then Max:=M[i]; //требуемое текущее число столбцов
  end;
  StringGrid1.ColCount:=Max+1;
  Edit1.Text:=IntToStr(Sum); // вывод количества простых чисел
  For i:=0 to 9 do //вывод количества простых чисел по диапазонам
    StringGrid2.Cells[0,i]:=IntToStr(M[i]);
end;
procedure TForm1.FormCreate(Sender: TObject);
Var i: integer;
    ss, ss1: string;
begin
   For i:=1 to 10 do     // формируем таблицу и ее заголовки
   begin
     Str((100*(i-1)+1):4,ss);
     Str((100*i):4,ss1);
   StringGrid1.Cells[0,i-1]:= ss+' - '+ss1;
   end;
   StringGrid1.ColWidths[0]:=80;
end;
end.
```

3.1.1.1 Дополнительные задания

1) Измените программу, чтобы она выводила все простые числа в интервале от 1000 до 2000.

2) Напишите программу, которая в заданном диапазоне чисел находит все пары взаимно простых чисел (взаимно простыми называются числа, не имеющие ни одного общего делителя кроме единицы, например 20 и 3).

3) Напишите программу, которая выводила бы все числа в интервале от 1 до 1000, которые делятся без остатка на 17.

4) Напишите программу, реализующую алгоритм Эратосфена.

3.1.1.2 Контрольные вопросы

1) Можно ли процедуру **Frac** заменить другой стандартной процедурой? Если да, то какой?
2) Расскажите о процедурах **Franc** и **Trunc**
3) В чем отличие в использовании процедур **Frac** и **Trunc**?
4) В чем отличие выражений **M[x]:=M[x]+1** и **M[x]:=M[x+1]**?
5) Какой тип имеет переменная **M[i]** в выражении **IntToStr(M[i])**?
6) Что означает срока кода **StringGrid1.Cells[M[x],x]:= IntToStr(i)?**
7) Что означает строка кода **StringGrid1.ColWidths[0]:=80**?
8) Что означает строка кода **StringGrid1.ColCount**?

3.2 Основы теории чисел

Алгоритм разложения числа на простые множители

Всякое целое число может быть представлено в виде произведения простых чисел и такое представлено единственно.

Пусть в разложении числа a на простые множители, множители P_i встречаются α_i раз, тогда $a = P_1^{\alpha_1} * P_2^{\alpha_2} .....P_m^{\alpha_m}$ - каноническое представление числа a.

Следует отметить, что в настоящее время нет достаточно эффективных алгоритмов разложения произвольного целого числа на простые множители даже в случае, когда известно, что оно разлагается в произведение двух простых чисел.

Отсутствие эффективных алгоритмов с доказуемыми оценками сложности позволяет использовать задачу разложения натуральных чисел на простые множители при обосновании стойкости некоторых криптографических алгоритмов.

Алгоритм Евклида нахождения наибольшего общего делителя

Общим делителем целых чисел $a_1, a_2, ...a_n$ будем называть всякое b_j, которое делит любой элемент из этой последовательности.

Наибольшим общим делителем чисел $a_1, a_2, ...a_n$ будем называть максимальное из их общих делителей $\max\{b_j\}$ и обозначать $НОД(a_1, a_2, ...a_n)$.

Числа $a_1, a_2, ...a_n$ называют взаимно простыми ,если $НОД(a_1, a_2, ...a_n) = 1$.

Числа $a_1, a_2, ...a_n$ называют попарно простыми, если $НОД(a_i, a_j) = 1,\ \forall i \neq j$

Свойство: Если $НОД(a_i, a_j) = 1, \forall i \neq j$ то $НОД(a_1, a_2, ...a_n) = 1$.

Теорема о представлении целых чисел: Всякое целое число a можно единственным образом представить в виде: $a = b * q + r$, причем $0 \le r < b$, где q- неполное частное, r - остаток от деления a на b.

Теорема : Пусть $a = b * q + c,$тогда $НОД(a,b) = НОД(b,c) = НОД(a,c)$.

Алгоритм Евклида нахождения НОД двух целых чисел *a* и *b*

В силу теоремы о представлении чисел

$a = b * q_1 + r_1,\ 0 \le r_1 < b$, причем $НОД(a,b) = НОД(b,r_1)$.

Рассмотрим пару чисел b и r_1: в силу теоремы о представлении чисел

$b = r_1 * q_2 + r_2,\ 0 \le r_2 < r_1$, причем $НОД(b,r_1) = НОД(r_1,r_2)$. Рассмотрим пару чисел r_1 и r_2:

$r_1 = r_2 * q_3 + r_3,\ 0 \le r_3 < r_2$, причем $НОД(r_1,r_2) = НОД(r_2,r_3)$ и т.д.

…

$r_{k-2} = r_{k-1} * q_k + r_k,\ 0 \le r_k < r_{k-1}$, причем $НОД(r_{k-2}, r_{k-1}) = НОД(r_{k-1}, r_k)$.

Корректное завершение алгоритма гарантируется тем, что остатки от деления образуют строго убывающую последовательность натуральных чисел, т.е. последовательность $r_1, r_2, r_3, ...r_k...$ убывающая. Следовательно на конечном шаге n r_n станет равным 0:

$r_{n-2} = r_{n-1} * q_n + 0$.

$НОД(b,r_1) = НОД(r_1,r_2) = НОД(r_2,r_3) = ... = НОД(r_{n-2}, r_{n-1}) = НОД(r_{n-1}, 0) = r_{n-1}$.

Рассмотрим 2 примера:

А. Найти НОД(117,45).

117 : 45 = 45 × 2 + 27,

45 : 27 = 27 × 1 + 18,

27: 18 = 18 × 1+ 9,

18 : 9 = 9 × 2+ 0.

Корректное завершение алгоритма гарантируется тем, что остатки от деления образуют строго убывающую последовательность натуральных чисел: 27, 18, 9, 0, т.е. (117, 45) = (45, 27) = (27, 18) = (18,9) =9.

Следовательно, НОД(117, 45) = 9.

Б. Найти НОД(247, 103).

247 : 103 = 103 × 2 + 41;

103 : 41 = 41 × 2 + 21;

41 : 21 = 21 × 1 + 20;

21 : 20 = 20 × 1 + 1;

20 : 1 = 1 × 20 + 0.

Следовательно, НОД (247, 103) = 1 и это есть два взаимно простых числа.

Как следствие из алгоритма Евклида, можно получить утверждение, что наибольший делитель целых чисел ***a*** и ***b*** может быть представлен в виде линейной комбинации этих чисел, т. е. существуют целые числа ***u*** и ***v*** такие, что справедливо равенство: $a \times u + b \times v = r_n$.

Пример: для чисел 107 и 45 парой таких чисел служат ***u*** =27 и ***v*** =-64:

107 × 27 – 45 × 64 = 9.

Функция Эйлера

Определение: Функцией Эйлера называют функцию $\varphi(a)$, которая определена для каждого целого положительного числа a и равна числу чисел из ряда *1,2,…* $a-1$, которые взаимно простые с a.

Теорема: Пусть $a = P_1^{\alpha_1} * P_2^{\alpha_2} \ldots . P_K^{\alpha_K}$, тогда

$$\varphi(a) = \left(p_1^{\alpha_1} - p_1^{\alpha_1 - 1}\right)\left(P_2^{\alpha_2} - p_2^{\alpha_2 - 1}\right) .. \left(pk_K^{\alpha_K} - p_k^{\alpha_K - 1}\right)$$

Свойство: Если p – простое число, то $\varphi(p) = p - 1$.

3.2.1 Лабораторная работа Каноническое разложение числа

Постановка задачи: разработать приложение для разложения числа на простые множители.

Порядок выполнения: Основное окно программы представлено на рисунке 3.2.

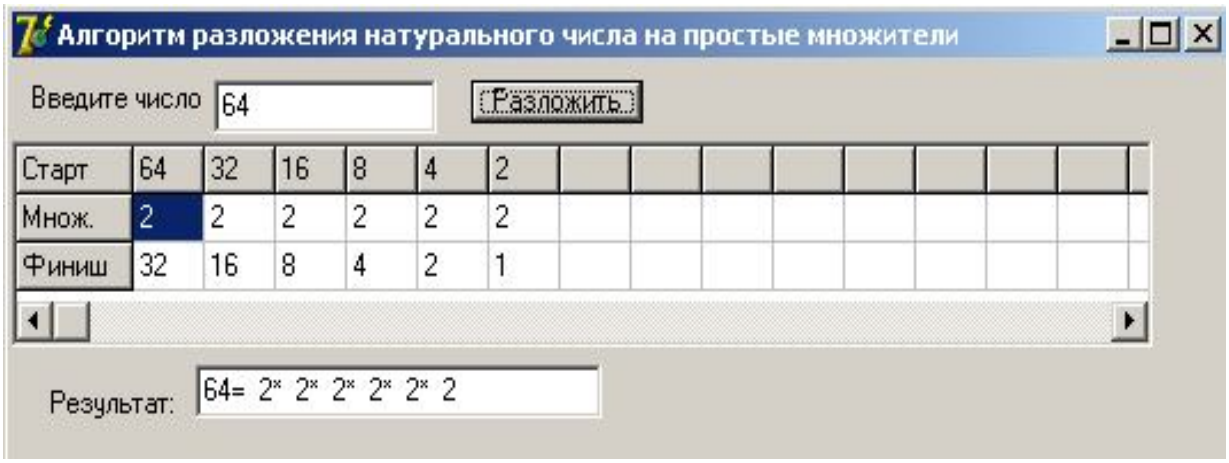


Рисунок 3.2 Окно программы «Алгоритм разложения натурального числа на простые множители»

Листинг программы представлен ниже:

```
implementation
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
Label 5;
Var
  A, i, Nomer: integer;
          Zna: array[1..100] of integer;
      ss, ss1: string;
begin
    A:=StrToInt(Edit1.Text);
    StringGrid1.Cells[0,0]:='Старт';
    StringGrid1.Cells[0,1]:='Множ.';
    StringGrid1.Cells[0,2]:='Финиш';
```

```
    StringGrid1.ColWidths[0]:=50;
For i:=1 to 100 do Zna[i]:=0;
i:=1; Nomer:=0; ss1:='';
Repeat
begin
 i:=i+1;
5: If Frac(A/i)=0 then
                    begin
                      nomer:=nomer+1;
                      Zna[nomer]:=i;
                      StringGrid1.Cells[nomer,0]:=IntToStr(A);
                      A:=Round(A/i);
                      StringGrid1.Cells[nomer,1]:=IntToStr(i);
                      StringGrid1.Cells[nomer,2]:=IntToStr(A);
                      Goto 5;
                    end; {END IF}
end;
Until i=5;
 Edit2.Text:=Edit1.Text+'=';
 For i:=1 to Nomer do
 begin
   Str(Zna[i]:3,ss);
   If i<Nomer then ss1:=ss1+ss+'*'
              else ss1:=ss1+ss;
 end;
 Edit2.Text:=Edit2.Text+ss1;
end;
end.
```

3.2.1.1 Дополнительные задания

1) Разработать программный модуль, который для произвольного числа вычисляет значение функции Эйлера.

2) Разработать функцию для вычисления НОД двух чисел по алгоритму Евклида.

3.2.1.2 Контрольные вопросы

1) Какое число называется простым?
2) В каком виде можно представить любое натуральное число, отличное от 1?
3) Как называется наибольшее целое число, делящее одновременно целые числа ***a*** и ***b***? Как оно обозначается?
4) В чем заключается алгоритм Евклида?
5) В каком случае справедливо равенство $a \times u + b \times v = r_n$?
6) Что такое НОД(a,b)?
7) Что означает строка кода **`StringGrid1.ColWidths[0]:=50`**?
8) В чем суть оператора **`Goto`**?
9) Что означает выражение **`A:=Round(A/i)`**?
10) Что означает эта запись

$117 : 45 = 45 \times 2 + 27,$

$45 : 27 = 27 \times 1 + 18,$

$27 : 18 = 18 \times 1 + 9,$

$18 : 9 = 9 \times 2 + 0.$

3.3 Модульная арифметика (арифметика вычетов)

3.3.1 Сравнения

Пусть *a* и *n* — натуральные числа. "Разделить число *a* на число *n* с остатком" — это значит найти целые числа *q* и *r*, удовлетворяющие условию

$$a = q \times n + r \text{, где } 0 \le r < n.$$

При этом число *q* называют неполным частным, а *r* - остатком от деления числа *a* на число *n*.

Например. Число 51 разделить на 2:

$$51 = 25 \cdot 2 + 1.$$

Если остаток *r* равен нулю, то говорят, что число *n* делит число *a*, или, по-другому, *n* является делителем числа *a*.

Определение: Числа a и b, которые при делении на одно и то же число m дают один и тот же остаток называют сравнительными по модулю m .

Выражение вида $a=b \ mod \ m$ называют сравнением.

Теорема: Для того, что бы a и b были сравнительными по модулю m необходимо и достаточно, что бы имело место представление $a=b+mt$, где t- целое или чтобы (a-b) делилось на m без остатка.

Например, числа 51 и 27 сравнимы по модулю 2, так как остаток у обоих равен 1:

$$51 = 25 \cdot 2 + 1,$$

$$27 = 13 \cdot 2 + 1.$$

Т.е. $51 \equiv 27 \bmod 2$.

Отсюда, в частности, следует, что число 2 делит разность чисел 51 и 27:

$$(51\text{-}27) / 2 = 24 /2 = 12 .$$

Для обозначения остатка используют так же запись вида $b = a \ (mod \ m)$.

Если $a = b \bmod m$, где a,b,m целые , то существует класс целых чисел, которые по модулю m также имеют один и тот же остаток, т.е. равноостаточные. Очевидно, что по модулю m может существовать ровно m классов, т.е. классов чисел с остатком $0, 1, 2 \ldots m\text{-}1$ и эти классы охватывают все целые числа.

Любой элемент каждого такого класса принято называть вычетом, принадлежащему к данному классу. Если от каждого класса взять по одному вычету, то получим полную систему вычетов.

Определение. Числа $a_0,...,a_k$ образуют полную систему вычетов по модулю m, если любое целое число сравнимо по модулю m с одним и только одним из этих чисел.

Любая полная система вычетов по модулю m состоит из m чисел, которые попарно не сравнимы по модулю m.

Пример: полная система вычетов m=7: 0, 1, 2, 3, 4, 5, 6

Теорема. Пусть $a_0, a_1 ..., a_k$ — полная система вычетов по модулю *m*. Пусть λ — целое число, взаимно простое с m. Тогда $\lambda a_0, \lambda a_1 ..., \lambda a_k$ — тоже полная система вычетов по модулю *m*.

Теорема. Пусть $a_0, a_1 ..., a_k$ — полная система вычетов по модулю *m*. Пусть λ — целое число. Тогда $\lambda + a_0, \lambda + a_1 ..., \lambda + a_k$ — тоже полная система вычетов по модулю *m*.

Определение. Числа $a_0, a_1 ..., a_k$ образуют приведенную систему вычетов по модулю *m*, если они взаимно просты с *m* и любое целое число, взаимно простое с *m*, сравнимо с одним и только одним из этих чисел по модулю *m*.

Пример. Приведенная система вычетов по модулю 10: 1,3,7,9.

Лемма. Все приведенные системы вычетов по модулю *m* состоят из одного и того же количества чисел, равного значению функции Эйлера от этого модуля - $\varphi(m)$.

Теорема. Если $a_0, a_1 ..., a_k$ — приведенная система вычетов по модулю *m* и λ — число, взаимно простое *m*, тогда $\lambda a_0, \lambda a_1 ..., \lambda a_k$ — приведенная система вычетов по модулю *m*.

Правила арифметики вычетов

Правила арифметики вычетов похожи на обычную арифметику:

(a + b) mod n = ((a mod n) + (b mod n)) mod n,

(a - b) mod n = ((a mod n) - (b mod n)) mod n,

(a * b) mod n = ((a mod n) * (b mod n)) mod n,

(a *(b+c)) mod n = (((a*b) mod n) + ((a*c) mod n)) mod n.

В арифметике вычетов возведение в степень выполняется без огромных промежуточных результатов. Вычисление степени некоторого числа по модулю другого числа вида $a^X \bmod n$ просто представляет собой последовательность умножений и делений. При этом существуют приемы, либо минимизирующие количество умножений по модулю, либо оптимизирующие отдельные операции по модулю.

Один из таких приемов называется цепочкой сложения, либо методом дво-

ичных квадратов и умножения.

Рассмотрим примеры.

Пример 1. Пример преобразования выражения $a^2 \bmod n$:

$a^2 \bmod n = (a*a) \bmod n = ((a \bmod n) * (a \bmod n)) \bmod n = (a \bmod n)^2 \bmod n$.

Рассмотрим достоинство такого преобразования на числовом примере. Пусть a =50, n=2. Тогда

$$a^2 \bmod n = 50^2 \bmod 2 = 2500 \bmod 2 = 0,$$

т.е. промежуточный результат вычисления (возведение в степень) давало число 2500. То же самое путем преобразования:

$$a^2 \bmod n = ((a \bmod n)^2 \bmod n = ((0)^2 \bmod 2 = 0,$$

т.е. промежуточный результат вычисления (возведение в степень) дает число 0. Таким образом, в результате преобразования удается сократить огромные результаты промежуточных вычислений, которые могут привести к переполнению формата числа в оперативной памяти компьютера и к невозможности получения окончательного результата.

Пример 2.

$a^4 \bmod n = (a^2*a^2) \bmod n = ((a^2 \bmod n)*(a^2 \bmod n)) \bmod n = ((a^2 \bmod n)^2 \bmod n$.

$a^8 \bmod n = (a^4*a^4) \bmod n = ((a^4 \bmod n)*(a^4 \bmod n)) \bmod n =$

$((((a^2 \bmod n)^2 \bmod n)* (((a^2 \bmod n)^2 \bmod n)) \bmod n = ((a^2 \bmod n)^2 \bmod n)^2 \bmod n$.

Таким образом, метод двоичных квадратов и умножения использует простую и очевидную цепочку сложений, в основе которой лежит двоичное представление числа.

Линейные сравнения

Уравнение вида $ax = b \bmod m$ называется линейным сравнением. Линейное сравнение имеет единственное решение, если $НОД(a,m) = 1$.

Пусть $НОД(a,m) = 1$, тогда на основании теоремы Эйлера мы можем записать, что $a^{\varphi(m)} = 1 \bmod m$. Умножим левую и правую часть сравнения на b: $b * a^{\varphi(m)} = b * 1 \bmod m$, тогда $a * (b * a^{\varphi(m)-1}) = b \bmod m$, следовательно решением линейного сравнения будет $x = b * a^{\varphi(m)-1} \bmod m$.

Системы линейных сравнений. Китайская теорема об остатках.

Система вида

$$\begin{cases} x = a_1 \bmod m_1 \\ x = a_2 \bmod m_2 \\ \dots \\ x = a_n \bmod m_n \end{cases},$$

где $m_1, m_2, \dots m_n \in Z$

$a_1, a_2, \dots a_n \in Z$

называется системой линейных сравнений.

Китайская теорема об остатках: Если $(m_i, m_j) = 1 \forall i \neq j$, то система линейных сравнений имеет единственное решение $x = \sum_{i-1}^{n} a_i M_i y_i \bmod M$ по модулю $M = m_1, m_2 \dots m_n$, где $M_i = \frac{M}{m_2}; y_i = M_j^{-1} \bmod m_i$.

3.3.2 Лабораторная работа Модульная арифметика

Постановка задачи: разработать программу для разложения степени числа по правилам арифметики вычетов

Порядок выполнения: в Delphi имеется встроенная процедура возведения числа ***a*** в степень ***x*** **`y:=IntPower(a,x)`**, содержащаяся в библиотеке **`Math`**. Недостаток данной процедуры, как ранее отмечено, состоит в переполнении формата результата процедуры ***y*** в оперативной памяти компьютера при относительно небольших значениях ***a*** и ***x***.

Разработаем собственную процедуру разложения степени числа по правилам арифметики вычетов, позволяющую исключить огромные результаты промежуточных вычислений.

Для упорядочивания разложения выражения **`b=a`**$^{\mathbf{x}}$ **`mod n`** по степеням числа ***x*** результат вычисления будем записывать в переменную ***c***, значения которой на каждой итерации разложения ***i*** определяется следующей логикой вычислений:

i=1: b= a^1 mod n $\rightarrow$ **c = b** ($c_1 = (1 \times b) \bmod n$);

i=2: b= a^2 mod n = (a*a) mod n = ((a mod n) * (a mod n)) mod n =(b*b) mod n

→ **c = (c*b) mod n** ($c_2 = (c_1 \times b) \bmod n$);

i=3: b= a^3 mod n = (a^2*a) mod n = ((a^2 mod n) * (a mod n)) mod n =(c*b) mod n

→ **c = (c*b) mod n** ($c_3 = (c_2 \times b) \bmod n$);

…

i=x: ($c_x = (c_{x-1} \times b) \bmod n$).

В виде алгоритма представленная последовательность итераций приведена на рисунке 3.3. Согласно алгоритму, исходный код функции разложения степени числа по правилам арифметики вычетов выглядит следующим образом:

```
Function aXmodN(a,x,n: integer): integer;
Var i,b,c: integer;
begin
i:=0; b:= a mod n; c:=0;
 While i<x do
 begin
  i:=i+1;
  If i=1 then c:=b else c:= (c*b) mod n;
 end;
 aXmodN:=c;
end;{aXmodN}
```

Таким образом, программный код вида **y:=IntPower(a,x)mod n** можно заменить выражением **y:= aXmodN(a,x,n)**. Для исследования возможностей и преимуществ разработанной функции по сравнению с использованием встроенной процедурой **Delphi y:=IntPower(a,x)** напишем специальное приложение.

Основное окно приложения для исследования вариантов разложения степени числа представлено на рисунке 3.4.

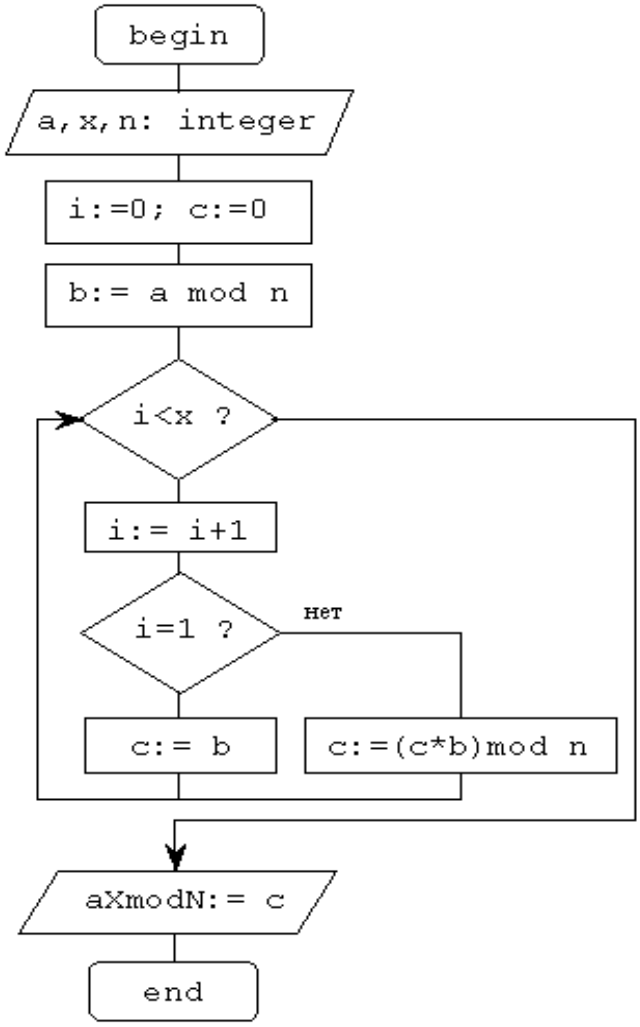


Рисунок 3.3 – Алгоритм разложения степени числа

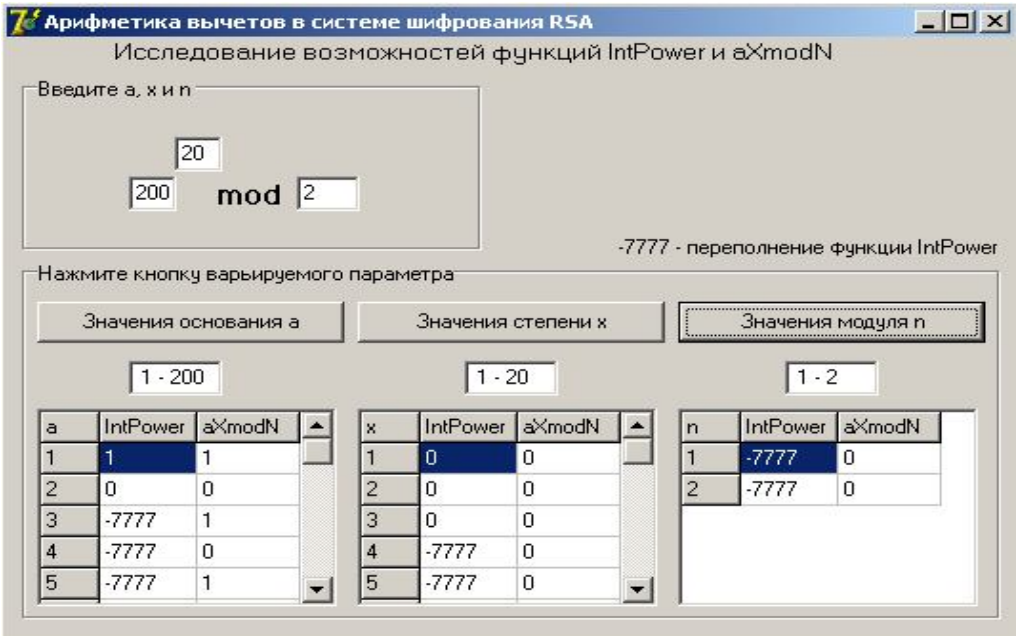


Рисунок 3.4 – Экранная форма приложения для исследования функций y:=IntPower(a,x) и y:= aXmodN(a,x,n)

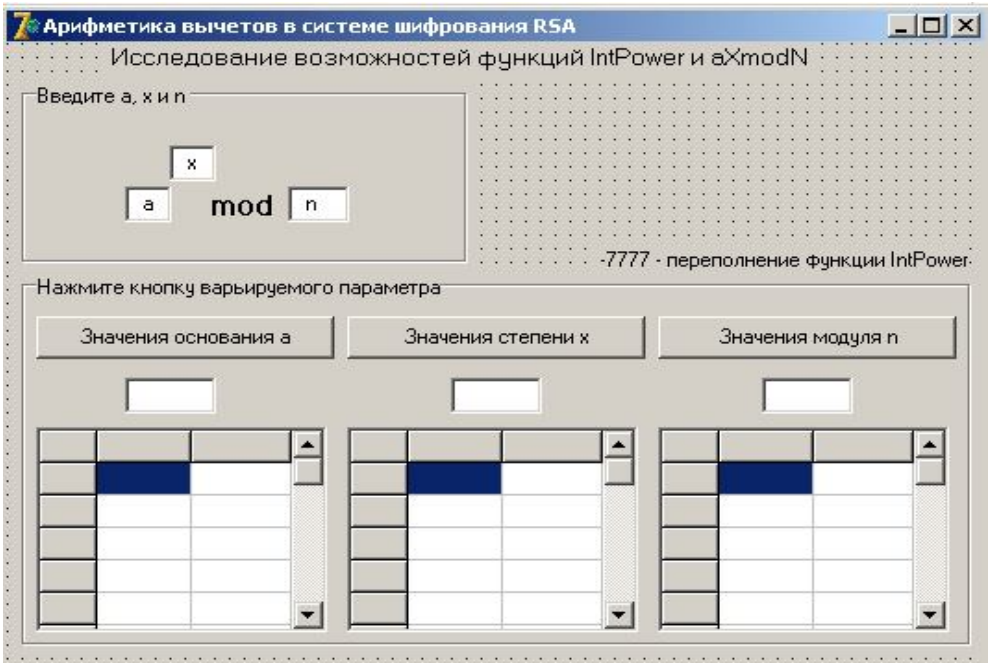


Рисунок 3.5 – Компоненты экранной формы приложения

Листинг кода программы представлен ниже:

```
implementation
{$R *.dfm}
Const Cmax=700000000; {max число типа integer}
 Var a,x,n,y: integer;
Function aXmodN(a,x,n: integer): integer;
Var i,b,c: integer;
begin
i:=0; b:= a mod n; c:=0;
 While i<x do
 begin
  i:=i+1;
  If i=1 then c:=b
         else c:= (c*b) mod n;
 end;
 aXmodN:=c;
end;{aXmodN}

procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
With StringGrid1 do
begin
 Cells[0,0]:='a';
 Cells[1,0]:='IntPower';
 Cells[2,0]:='aXmodN';
end;
 With StringGrid2 do
begin
 Cells[0,0]:='x';
 Cells[1,0]:='IntPower';
 Cells[2,0]:='aXmodN';
end;
 With StringGrid3 do
begin
 Cells[0,0]:='n';
 Cells[1,0]:='IntPower';
 Cells[2,0]:='aXmodN';
end;
end;
procedure TForm1.Button1Click(Sender: TObject);
Var i: integer;
begin
  a:=StrToInt(Edit1.Text);
  x:=StrToInt(Edit2.Text);
  n:=StrToInt(Edit3.Text);
  {подставляем a от 1 до заданного в Edit1.Text при постоянных x и n}
  Edit4.Text:= ' 1 - '+IntToStr(a);
  For i:=1 to StrToInt(Edit1.Text) do
  begin
    a:=i;
    StringGrid1.Cells[0,i]:=IntToStr(i);
    IF(IntPower(a,x)>Cmax) then y:=-7777
                           else y:=Round(IntPower(a,x)) mod n;
    StringGrid1.Cells[1,i]:=IntToStr(Y);
    Y:=aXmodN(a,x,n);
```

```
    StringGrid1.Cells[2,i]:=IntToStr(Y);
end;
  StringGrid1.RowCount:=i;
end;
procedure TForm1.Button2Click(Sender: TObject);
Var i: integer;
begin
  a:=StrToInt(Edit1.Text);
  x:=StrToInt(Edit2.Text);
  n:=StrToInt(Edit3.Text);
  {подставляем x от 1 до заданного в Edit2.Text при постоянных a и n}
  Edit5.Text:= ' 1 - '+IntToStr(x);
  For i:=1 to StrToInt(Edit2.Text) do
  begin
    x:=i;
    StringGrid2.Cells[0,i]:=IntToStr(i);
    IF(IntPower(a,x)>Cmax) then y:=-7777
                           else y:=Round(IntPower(a,x)) mod n;
    StringGrid2.Cells[1,i]:=IntToStr(Y);
    Y:=aXmodN(a,x,n);
    StringGrid2.Cells[2,i]:=IntToStr(Y);
end;
  StringGrid2.RowCount:=i;
end;
procedure TForm1.Button3Click(Sender: TObject);
Var i: integer;
begin
  a:=StrToInt(Edit1.Text);
  x:=StrToInt(Edit2.Text);
  n:=StrToInt(Edit3.Text);
  {подставляем x от 1 до заданного в Edit3 при постоянных a и n}
  Edit6.Text:= ' 1 - '+IntToStr(n);
  For i:=1 to StrToInt(Edit3.Text) do
  begin
    n:=i;
    StringGrid3.Cells[0,i]:=IntToStr(i);
```

```
    IF(IntPower(a,x)>Cmax) then y:=-7777
                           else y:=Round(IntPower(a,x)) mod n;
    StringGrid3.Cells[1,i]:=IntToStr(Y);
    Y:=aXmodN(a,x,n);
    StringGrid3.Cells[2,i]:=IntToStr(Y);
end;
  StringGrid3.RowCount:=i;
end;
end.
```

3.3.2.1 Дополнительное задание

1) Разработать программный модуль для решения линейного сравнения.

2) Разработать программный модуль для решения системы линейных сравнений.

3.3.2.2 Контрольные вопросы

1) Что значит разделить число *a* на число *n* с остатком?
2) Для чего предназначена функция **y:=IntPower(a,x)**?
3) В чем главный недостаток функции **y:= IntPower(a,x) mod n**?
4) Как называется операция нахождения числа b=a mod n?
5) В каком случае целые числа *a* и *b* называются сравнимыми по модулю *n*?
6) Какое множество целых чисел называется полной системой вычетов по модулю *n*?
7) Какая полная система вычетов обычно используется?
8) Как вычисляется степень некоторого числа по модулю другого числа вида $a^X \bmod n$? Как называется данный метод вычисления?
9) Чему равно $a^2 \bmod n$?
10) Верно ли данное выражение - $a^3 \bmod n = (a^2 * a) \bmod n$?
11) Чему равно $a^2 \bmod n$, если a=60, n=2?

12) Какому уравнению соответствует функция y:= IntPower(a,x) mod n?

13) В чем назначение данного фрагмента кода?

```
i:=0;
b:= a mod n;
c:=0;
While i<x do
begin
 i:=i+1;
 If i=1 then c:=b else c:=(c*b) mod n;
end;
```

14) В чем достоинство функции **Function aXmodN(a,x,n: integer)** при шифровании?

15) Опишите метод решения линейного сравнения.

16) Сформулируйте китайскую теорему об остатках.

3.4 Шифрсистема RSA

В семидесятых годах после работы американских математиков У.Диффи и М.Хеллмана появилась "новая криптография" - *криптография с открытым ключом.* Идея применения ассиметричной криптографии возникла из попыток найти решение двух наиболее сложных проблем, возникающих при использовании традиционного шифрования: проблема распределения ключей и проблема установления подлинности отправляющей и принимающей стороны.

При ассиметричном шифровании для расшифрования и шифрования используют разные ключи. И знание одного из них не даёт практической возможности определить второй. Поэтому ключ для шифрования может быть сделан общедоступным без потери стойкости шифра, если ключ для расшифрования сохранён в секрете.

У. Диффи и М. Хэллман сформулировали требования, выполнение которых обеспечивает безопасность открытой криптосистемы:

1) вычисление пары ключей открытия и закрытия должны быть прочными;

2) отправитель значения открытый ключ K_0 легко вычисляет криптограмму

$$C = E_{K_0}(M),$$

где M-исходное сообщение;

$E_{K_0}(\bullet)$ - преобразование защифрованных ключей K_0

C- шифрованное сообщение;

3) получатель, использует закрытый ключ K_c и криптограмму «С» легко восстанавливает исходное сообщение $M = D_{K_c}(C)$, где $D_{K_c}(\bullet)$- преобразование расшифрования на ключе K_c;

4) противник зная открытый ключ при попытке вычислить секретный ключ, наталкивается на непреодолимую проблему;

5) противник, зная открытый ключ и криптограмму , при попытке восстановить исходное сообщение также наталкивается на непреодолимую проблему.

Концепция симметричных криптосистем основана на применении однонаправленных функций. Однонаправленными называются функции F: $F: X \to Y$, обладающая следующими свойствами:

1) существует полиномный алгоритм вычисления значения $F(x)$;

2) не существует полиномного алгоритма инвертирования функции, т.е. решения уравнения $F(x) = y$ относительно x.

В криптографии широко применяются следующие односторонние функции:

1) разложение большого числа на простые сомножители $N=P*Q$;

2) задача дискретного логарифмирования $A^x \ mod \ N = y$..

В 1978 г. появилась первая шифрсистема RSA с открытым ключом, разработанная Райвестом (R), Шамиром (S) и Адлеманом (A). В настоящее время RSA является наиболее распространенной системой шифрования с открытым ключом.

Процедуру шифрования данных в системе RSA можно разбить на два этапа :

1) определение ключей;

2) шифрование данных.

Открытый текст шифруется блоками, каждый из которых меньше некоторого малого числа n. На практике длина блока равна 2^k. Таким образом $2^k < n < 2^{k+1}$.

Этап 1. Определение ключей

1) получатель выбирает два очень больших простых числа P и Q и находит их произведение $N = P * Q$

2) получатель вычисляет функцию Эйлера по формуле $\varphi(N) = (P-1)(Q-1)$

3) получатель выбирает случайное целое число K_0, такое что

$$1 < K_0 < \varphi(N),\ НОД(K_0, \varphi(N)) = 1$$

K_0 и N объявляются открытыми ключами и пересылаются отправителю.

4) Вычисляется закрытый ключ K_c как решение линейного сравнения

$$K_0 * K_c = 1 \bmod \varphi(N)$$

Этап 2. Шифрование /расшифрование данных:

Пусть M- сообщение, представленное в виде целого числа от 1 до N.

1) Отправитель вычисляет криптограмму по формуле:

$$C = M^{K_0} \bmod N$$

2) получатель расшифровывает его по формуле:

$$M = C^{K_C} \bmod N$$

Стойкость этой системы основана на проблеме разложения составного числа множители и на проблеме дискретного логарифмирования.

Рассмотрим пример работы с RSA.

Зашифруем аббревиатуру RSA. Для этого

1. Выберем два взаимно простых числа $P = 17$, Q = 31.

2. Вычислим $N = PQ = 527$ и $\varphi(N) = (P - 1) \times (Q - 1) = 480$.

3. Выберем в качестве K_0 число, взаимно простое с $\varphi(N)$, например $K_0 = 7$.

4. Найдем K_c как решение линейного сравнения $K_0 * K_c = 1 \bmod \varphi(N)$.

$$7K_c = 1 \bmod 480$$

$$K_c = 7^{\varphi(480)-1} \bmod 480$$

Каноническое разложение числа 480: $480 = 2^5 * 3 * 5$

Следовательно $\varphi(480) = (2^5 - 2^4) * (5^1 - 5^0) * (3^1 - 3^0) = (32 - 16) * 4 * 2 = 128$

$K_c = 7^{127} \bmod 480 = ((7^4 \bmod 480)^{31} * (7^3 \bmod 480)) \bmod 480 =$

$((2401 \bmod 480)^{31} * (343 \bmod 480)) \bmod 480 = 1 * 343 \bmod 480 = 343$

Проверка: 7 × 343 = 2401 = 1mod480.

Теперь представим данное сообщение в виде последовательности чисел, содержащихся в интервале $\{0,526\}$. Для этого буквы ***R***, ***S*** и ***A*** закодируем пятимерными двоичными векторами, воспользовавшись двоичной записью их порядковых номеров в английском алфавите:

R = 18 = (10010), 5 = 19 = (10011), A = 1 = (00001).

Тогда RSA = (100101001100001). Укладываясь в заданный интервал $\{0,526\}$, получаем следующее представление:

RSA = (100101001), (100001) = (M_1 = 297, M_2 = 33).

Далее последовательно шифруем M_1, и M_2:

$$C_1 = E_k(M_1) = M_1^e \equiv 297^7 (\bmod 527) = 474 .$$

При этом мы воспользовались тем, что

$297^7 = ((297^2)^3 297)(\bmod 527) = ((200^3 (\bmod 527) 297)(\bmod 527)$,

$C_2 = E_k(M_2) = M_2^e \equiv 33^7 (\bmod 527) = 407$.

В итоге получаем шифртекст: $y_1 = 474$, $y_2 = 407$.

<u>При расшифровании</u> нужно выполнить следующую последовательность действий.

Во-первых, вычислить $D_k(C_1) = (C_1)^{343} (\bmod 527)$.

Отметим, что при возведении в степень удобно воспользоваться тем, что 343 = 256 + 64 + 16 + 4 + 2 + 1. На основании этого представления получаем:

$474^2 \equiv 174(\bmod 527)$, $474^4 (\bmod 527) \equiv 237$, $474^8 (\bmod 527) \equiv 307$, $474^{16} (\bmod 527) \equiv 443$,

$474^{32} (\bmod 527) \equiv 205$, $474^{64} (\bmod 527) \equiv 392$, $474^{128} (\bmod 527) \equiv 307$, $474^{256} (\bmod 527) \equiv 443$,

в силу чего

$$474^{343} (\bmod 527) \equiv (443 \times 392 \times 443 \times 237 \times 174 \times 474)(\bmod 527) \equiv 297 .$$

Аналогично $407^{343} (\bmod 527) \equiv 33$.

Возвращаясь к буквенной записи, получаем после расшифрования: RSA.

Алгоритм RSA можно представить в виде следующей последовательности шагов.

Шаг 1. Выбираются два больших простых числа *P* и *Q*. Величина этих чисел должна быть больше 200.

Шаг 2. Вычисляется открытая компонента ключа *N:*

$$N=P*Q.$$

Шаг 3. Вычисляется функция Эйлера по формуле:

$$\varphi(N) = (P-1)\times(Q-1).$$

Шаг 4. Выбирается большое простое число $К_о$, которое является взаимно простым со значением $\varphi(N)$.

Шаг 5. Определяется число $К_с$, удовлетворяющее условию:

$$K_0 \times K_c = 1 \bmod \varphi(N).$$

Число $К_о$ принимается в качестве второй компоненты открытого ключа. В качестве секретного ключа используются числа K_0, *P, Q.*

Шаг 6. Исходная информация, независимо от ее физической природы, представляется в числовом двоичном виде. Последовательность бит разделяется на блоки длиной *L* бит, где *L* - наименьшее целое число, удовлетворяющее условию: $L \geq \log_2(n+1)$. Каждый блок рассматривается как целое положительное число *X(i)*, принадлежащее интервалу *[0, n-1]***.** Таким образом, исходная информация представляется последовательностью чисел *X(i), i =1,I.* Значение *I* определяется длиной шифруемой последовательности.

Шаг 7. Зашифрованная информация получается в виде последовательности чисел *Y(i)***,** вычисляемых по формуле:

$$Y(i) = (X(i)^{K_c} (\bmod n).$$

Шаг 8. Для расшифрования информации используется следующая зависимость:

$$X(i) = (Y(i)^{K_o} (\bmod n).$$

3.4.1 Лабораторная работа Шифр-система RSA

Постановка задачи: разработать приложение для шифрования/расшифрования в системе RSA

Порядок выполнения: экранная форма запущенного приложения представлена в соответствии с рисунком 3.6.

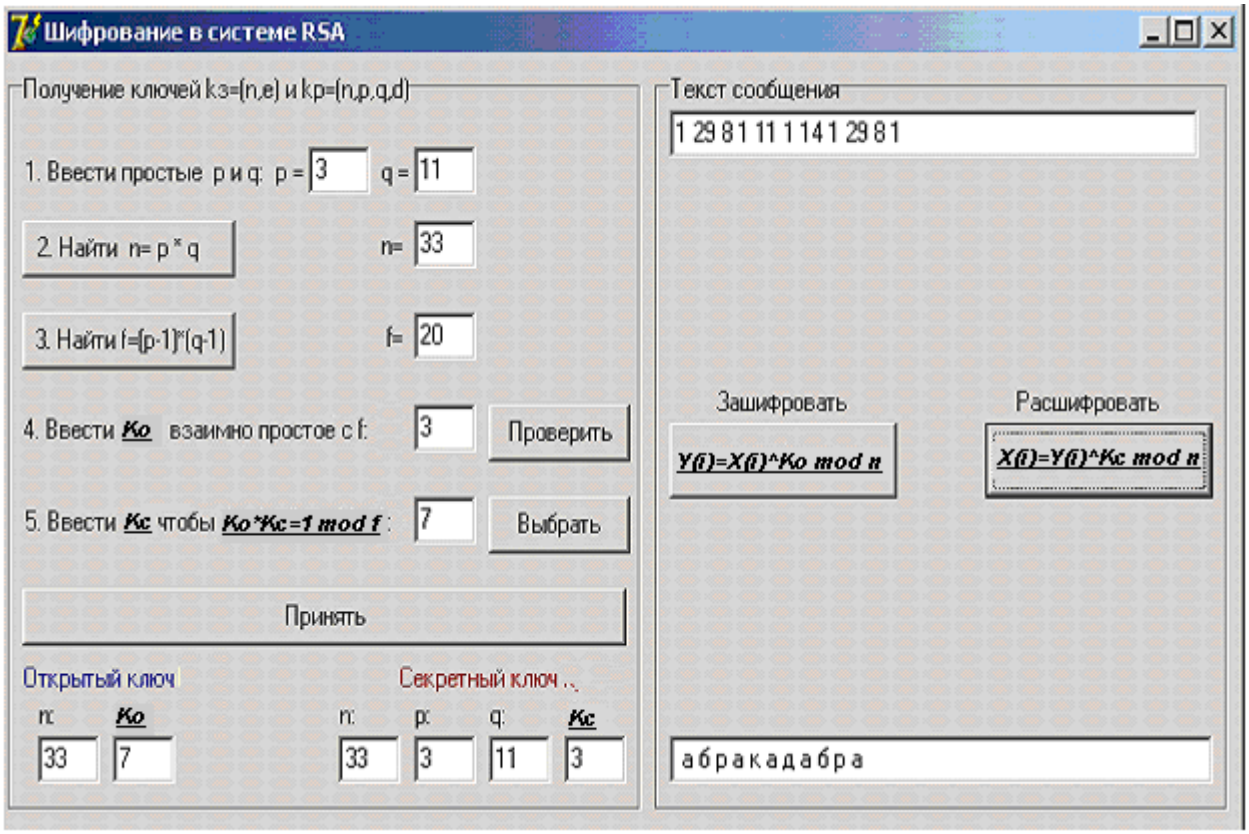


Рисунок 3.6 – Экранная форма приложения для шифрования в RSA

Листинг кода программы представлен ниже.

```
implementation
{$R *.dfm}
Var  p,q,n,f,
d {секретная экспонента - Кс},e {открытая экспонента Ко}: integer;
Procedure Shifr;
begin
 With Form1 do
 begin
```

```
 LabeledEdit5.Text:=IntToStr(n);
 LabeledEdit6.Text:=IntToStr(e);
 LabeledEdit7.Text:=IntToStr(n);
 LabeledEdit8.Text:=IntToStr(p);
 LabeledEdit9.Text:=IntToStr(q);
LabeledEdit10.Text:=IntToStr(d);
 end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
p:=StrToInt(pp.Text);
q:=StrToInt(qq.Text);
n:=p*q;
nn.Text:=IntToStr(n);
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
f:=(p-1)*(q-1);
ff.Text:=IntToStr(f);
end;
procedure TForm1.Button3Click(Sender: TObject);
Label 5;
Var
    a,b,r,NOD: integer;
   ss,ss1,ss2: string;
begin
  d:=StrToInt(dd.Text);
If f>d then begin a:=f; b:=d; end
       else begin a:=d; b:=f; end;
5: r:= a mod b;
   Str(a:2,ss); Str(b:2,ss1); Str(r:2,ss2); ss:=ss+' '+ss1+' '+ss2;
   ShowMessage(ss);
If r=0 then
         begin
          NOD:= b;
          If NOD>1 then
```

```
            begin
             Str(nod:3,ss); ss:=' НОД равен '+ss+' !';
             ShowMessage(ss); Exit;
            end
                    else ShowMessage('Правильно');
          end
        else
          begin
           a:=b; b:=r; goto 5;
          end;
end;
procedure TForm1.Button4Click(Sender: TObject);
Var
j,i: integer;
begin
StringGrid1.Visible:=true;
  j:=0;
  for i:=1 to 100 do
  begin
    if ((i*d) mod f)=1 then
      begin
        j:= j+1;
        StringGrid1.Cells[0,j-1]:=IntToStr(i);
      end;
  end;
  StringGrid1.RowCount:=j-1;
 end;
procedure TForm1.Button5Click(Sender: TObject);
Var   vr: int64;
       i: integer;
      cr: extended;
begin
 Edit2.Clear;
 For i:=1 to Length(Edit1.Text) do
 begin
  vr:=Ord(Edit1.Text[i]);
```

```
  vr:=vr-223;
  vr:=Round(intPower(vr,e));
  vr:= vr mod n;
  Edit2.Text:=Edit2.Text+IntToStr(vr)+' ';
 end;
end;
procedure TForm1.Button6Click(Sender: TObject);
Var
i, sum: integer;
vr: int64;
ss, s: string;
cr: extended;
 Kod: array[1..200] of integer;
 Sim: array[1..200] of char;
begin
Edit2.Clear;
 sum:=0; ss:='';
 For i:=1 to Length(Edit1.Text) do
 begin
  s:= Edit1.Text[i];
   if s<>' ' then ss:=ss+s
             else
    begin
               sum:=sum+1;
               Kod[sum]:=StrToInt(ss);
               ss:='';
              end;
 end;
  For i:=1 to sum do
  begin
   cr:=Kod[i];
   vr:= Round(IntPower(cr,d)) mod n;
   vr:=vr+223;
   Sim[i]:=Chr(vr);
  end;
  For i:=1 to sum+1 do Edit2.Text:=Edit2.Text+' '+Sim[i];
```

```
end;
procedure TForm1.eeChange(Sender: TObject);
begin
 StringGrid1.Visible:= false;
end;
procedure TForm1.Button7Click(Sender: TObject);
begin
  e:=StrToInt(ee.Text); Shifr;
end;
end.
```

3.4.2.1 Дополнительное задание

Доработать программу так, что бы она автоматически расчитывала секретный ключ.

3.4.2.2 Контрольные вопросы

1) Кто авторы концепции криптографии с открытым ключом?
2) Когда появилась первая шифрсистема с открытым ключом? Кем она разработана?
3) Что такое RSA?
4) Опишите процедуру зашифрования RSA
5) Опишите процедуру расшифрования RSA
6) Какому условию должен удовлетворять открытый ключ K_o?
7) Какому условию должен удовлетворять секретный ключ K_c?
8) Как вычислить значение функции Эйлера

3.5 Стойкость системы RSA. Многопользовательский вариант RSA

Сложность нахождения секретного ключа системы RSA определяется сложностью разложения числа *N* на простые множители. В связи с этим нужно выбирать

числа P и Q таким образом, чтобы задача разложения числа N была достаточно сложна в вычислительном плане. Для этого рекомендуются следующие требования:

1) числа P и Q должны быть достаточно большими, не слишком сильно отличаться друг от друга и в то же время быть не слишком близкими друг другу;

2) числа P и Q должны быть такими, чтобы наибольший общий делитель чисел $(P - 1)$ и $(Q - 1)$ был небольшим; желательно, чтобы $НОД(P-1, Q-1) = 2$;

В настоящее время самые большие простые числа, вида $N = P \times Q$, которые удается разложить на множители известными методами, содержат в своей записи 140 десятичных знаков. Поэтому, согласно указанным рекомендациям, числа P и Q в системе RSA должны содержать не менее 100 десятичных знаков.

Следует подчеркнуть необходимость соблюдения осторожности в выборе модуля RSA (числа N) для каждого из корреспондентов сети. В связи с этим можно сказать следующее.

Читатель может самостоятельно убедиться в том, что, зная одну из трех величин: P, Q или $\varphi(N)$, можно легко найти секретный ключ RSA.

Известно также, что, зная секретную экспоненту расшифрования, можно легко разложить модуль N на множители. В этом случае удается построить вероятностный алгоритм разложения N. Отсюда следует, что каждый корреспондент сети, в которой для шифрования используется система RSA, должен иметь свой уникальный модуль.

В самом деле, если в сети используется единый для всех модуль N, то такая организация связи не обеспечивает конфиденциальности, несмотря на то что базовая система RSA может быть стойкой. Выражаясь другими словами, говорят о несостоятельности протокола с общим модулем. Несостоятельность следует из того, что знание произвольной пары экспонент (K_{oi}, K_{ci}) позволяет, как было отмечено, разложить **N** на множители. Поэтому любой корреспондент данной сети имеет возможность найти секретный ключ любого другого корреспондента. Более того, это можно сделать даже без разложения $\boldsymbol{N}$ на множители.

Как отмечалось ранее, системы шифрования с открытыми ключами работают сравнительно медленно. Для повышения скорости шифрования RSA на практике

используют малую экспоненту зашифрования.

Если выбрать число K_o небольшим или таким, чтобы в его двоичной записи было мало единиц, то процедуру шифрования можно значительно ускорить. Например, выбрав $K_o = 3$ (при этом ни *P-1*, ни *Q-l* не должны делиться на 3), мы сможем реализовать шифрование с помощью одного возведения в квадрат по модулю **N** и одного перемножения.

Выбрав $K_o = 2^{16}+1 = 65537$ — число, двоичная запись которого содержит только две 1, мы сможем реализовать шифрование с помощью 16 возведений в квадрат по модулю N и одного перемножения. Если экспонента K_o выбирается случайно, то реализация шифрования по алгоритму RSA потребует s возведений в квадрат по модулю N и в среднем s/2 умножений по тому же модулю, где s - длина двоичной записи числа N .

Вместе с тем выбор небольшой экспоненты *Ko* может принести к негативным последствиям. Дело в том, что у нескольких корреспондентов могут оказаться одинаковые экспоненты.

Пусть, например, три корреспондента имеют попарно взаимно простые модули N_1, N_2, N_3 и общую экспоненту $K_o=3$. Если еще один пользователь посылает им некое циркулярное сообщение M, то криптоаналитик противника может получить в свое распоряжение три шифрованных текста $y_i = M^3 \bmod N_i$, $i = 1,2,3$. Далее он может найти решение системы сравнений

$$\begin{cases} y \equiv y_1 \bmod N_1, \\ y \equiv y_2 \bmod N_2, \\ y \equiv y_3 \bmod N_3, \end{cases}$$

лежащее в интервале $0 < y < N_1 * N_2 * N_3$. По китайской теореме об остатках (см. Приложение 3) такое решение единственно, а так как $M^3 < N_1 * N_2 * N_3$, *то* $y = M^3$. Само M можно найти, вычисляя кубический корень: $M = \sqrt[3]{y}$.

Отметим, что выбор малой экспоненты расшифрования *Kc* также нежелателен в связи с возможностью определения K_c простым перебором. Известно также, что если $K_c < \sqrt[4]{n}$, то экспоненту K_c легко найти, используя непрерывные дроби.

3.5.1 Лабораторная работа Многопользовательский вариант RSA

Постановка задачи: разработать приложение для шифрования в многопользовательском варианте RSA.

Порядок выполнения: Основное окно программы представлено на рисунке 3.7., где d - секретная экспонента Кс, e*i* - открытая экспонента Ко *i*–го пользователя.

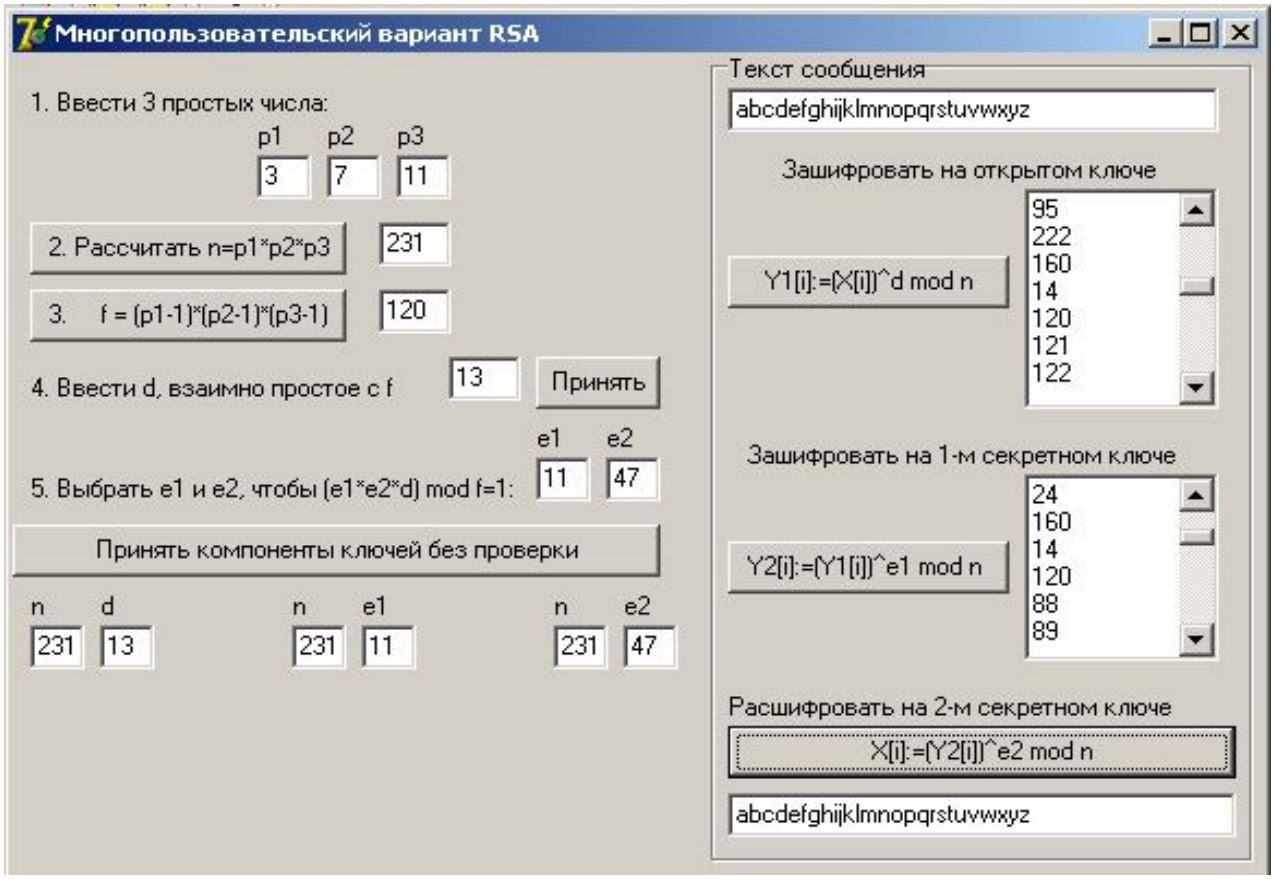


Рисунок 3.7 – Окно программы «Многопользовательский вариант RSA»

Листинг кода программы приведен ниже.

```
implementation
{$R *.dfm}
 Var
 Dlina,p1,p2,p3,n,f,d,e1,e2: integer;
Function aXmodN(a,x,n: integer): integer;
Var i,b,c: integer;
begin
```

```
i:=0; b:= a mod n; c:=0;
 While i<x do
 begin
  i:=i+1;
  If i=1 then c:=b
         else c:= (c*b) mod n;
 end;
 aXmodN:=c;
end;{aXmodN}
procedure TForm1.Button1Click(Sender: TObject);
begin
p1:=StrToInt(LabeledEdit1.Text);
p2:=StrToInt(LabeledEdit2.Text);
p3:=StrToInt(LabeledEdit3.Text);
n:=p1*p2*p3;
Edit3.Text:=IntToStr(n);
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
f:=(p1-1)*(p2-1)*(p3-1);
Edit4.Text:=IntToStr(f);
end;
procedure TForm1.Button3Click(Sender: TObject);
Label 5;
Var
    a,b,r,NOD: integer;
   ss,ss1,ss2: string;
begin
  d:=StrToInt(Edit5.Text);
If f>d then begin a:=f; b:=d; end
       else begin a:=d; b:=f; end;
5: r:= a mod b;
If r=0 then
         begin
          NOD:= b;
          If NOD>1 then
```

```
            begin
             Str(nod:3,ss); ss:=' НОД равен '+ss+' !';
             ShowMessage(ss); Exit;
            end
                    else ShowMessage('Правильно');
          end
        else
          begin
           a:=b; b:=r; goto 5;
          end;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
   e1:= StrToInt(LabeledEdit4.Text);
   e2:= StrToInt(LabeledEdit5.Text);
   LabeledEdit6.Text:=IntToStr(n);
   LabeledEdit7.Text:=IntToStr(d);
   LabeledEdit8.Text:=IntToStr(n);
   LabeledEdit9.Text:=IntToStr(e1);
   LabeledEdit10.Text:=IntToStr(n);
   LabeledEdit11.Text:=IntToStr(e2);
end;
procedure TForm1.Button5Click(Sender: TObject);
Var  i, kod: integer;
begin
 Dlina:=Length(Edit1.Text); // определяем число символов в тексте
 For i:=1 to Dlina do
 begin
  kod:=Ord(Edit1.Text[i]); //для каждого символа определяем его код
  kod:=kod{-223};
  kod:=aXmodN(kod,d,n);       // шифруем код символа на ключе d
  Memo1.Lines.Strings[i-1]:=IntToStr(kod); // заносим код в Memo1
 end;
end;
procedure TForm1.Button6Click(Sender: TObject);
Var
```

```
 i,kod: integer;
begin
 For i:=1 to Dlina do
 begin
  kod:= StrToInt(Memo1.Lines.Strings[i-1]);//считываем код символа
  kod:= aXmodN(kod,e1,n);                  //шифруем код на ключе e1
  Memo2.Lines.Strings[i-1]:= IntToStr(kod); // заносим в Memo2
 end;
end;
procedure TForm1.Button7Click(Sender: TObject);
Var
i, kod: integer;
    ss: string;
begin
 Edit2.Clear;
 For i:=1 to Dlina do
  begin
   kod:= StrToInt(Memo2.Lines.Strings[i-1]);//считываем код символа
   kod:= aXmodN(kod,e2,n);              // расшифровываем на ключе e2
   ss:=chr(kod{+223});                  // определяем символ
   Edit2.Text:=Edit2.Text+ss;           // записываем открытый текст
  end;
end;
end.
```

3.5.1.1 Контрольные вопросы

1) Для чего используют экспоненту зашифрования K_o?
2) К чему может привести выбор малой комоненты K_o и почему?
3) Почему выбор малой экспоненты расшифрования K_c нежелателен?
4) Почему каждый корреспондент должен иметь в сетисвой уникальный номер?
5) В чем сложность нахождения секретного ключа системы RSA?
6) Какие требования нужно соблюдать при выборе P и Q?

7) Не менее скольки знаков должны содержать числа P и Q в системе RSA для обеспечения высокой криптостойкости?
8) Какие величины нужно знать, чтобы найти секретный ключ RSA?
9) Из чего следует несостоятельность протокола с общим множителем?

3.6 Цифровая подпись

3.6.1 Общие сведения

Цифровая подпись сообщения позволяет зафиксировать подмену или модификацию данных сообщения. При положительных результатах проверки цифровой подписи получатель может быть уверен, что полученное сообщение пришло от субъекта, владеющего секретным ключом, и содержательная часть сообщения не подвергалась изменениям.

Если цифровая подпись получается в соответствии с официальным государственным стандартом, то она имеет юридическую силу обычной подписи под документом.

Впервые идею цифровой подписи предложили в 1976 году американские специалисты У. Диффи и М. Хеллман..

Первым по времени изобретения алгоритмом цифровой подписи был алгоритм RSA.

Предложенный в 1984 году алгоритм Эль-Гамаля позволял повысить стойкость подписи при ключе в 64 байта примерно в 1000 раз, но длина самой цифровой подписи увеличивалась в два раза и составляла 128 байт.

Алгоритм Эль-Гамаля послужил основой для разработки национального стандарта США DSA, введенного в 1991 году, и государственного стандарта РФ ГОСТ Р 34.10-94, введенного в действие с 1995 года.

3.6.2 Алгоритм цифровой подписи Эль-Гамаля

В основе алгоритма лежит задача дискретного логарифмирования.

Шаг 1. Для генерации ключей выбирается большое простое число *p* (порядка 1024 бит), такое что p-1 делится на другое простое число *q* (приблизительно 160 бит), и число *g*, такое что *g<p,* причем $g^{\frac{p-1}{q}} \bmod P \neq 1$.

Шаг 2. Отправитель выбирает случайное число *x*, такое что 1< *x* < *p и* вычисляет $y = g^X \bmod p$**.**

В результате получаются открытые ключи *p*, *g*, *y*, которые можно сделать общими для группы пользователей, и закрытый ключ *x***.**

Шаг 3. Для подписи сообщения M генерируется случайное число *k*, такое что

$$1<k<(p-1),\ НОД(k,p-1)=1.$$

Шаг 4. Далее отправитель вычисляется первый элемент подписи $a = g^K \bmod p$**.**

Шаг 5. Отправитель вычисляет второй элемент подписи *b*, как решение сравнения:

$$M = (x*a + k*b) \bmod (p-1).$$

Пара чисел *a* и *b* является цифровой подписью**.** (*M*, *a, b*) – передаются получателю. Случайное число *k* хранится в секрете.

Шаг 6. Для проверки подписи необходимо убедиться, что выполняется следующее условие:

$$y^a a^b \bmod p = g^M \bmod p\ .$$

Если условие выполнено, то получатель считает, что полученное сообщение подписано отправителем, от которого был получен ключ *y*. Кроме того, получатель считает, что в процессе передачи целостность сообщения не нарушена. В противном случае подпись считается недействительной и сообщение отвергается.

Каждая подпись требует генерации нового значения *k*, и это значение должно быть выбрано случайным образом.

Замечание: Для обеспечения стойкости подписи обычно алгоритмы цифровой подписи используют в сочетании с криптографической Хэш-функцией. В этом случае сообщение сначала хэшируют, а затем подписывают Хэш-значение сообщения:

Открытый ключ:

p – простое число (может быть общим для группы пользователей), такое что p-

1 делится на другое простое число q (приблизительно 160 бит),

$g < p$ (может быть общим для группы пользователей), такое что $g^{\frac{p-1}{q}} \bmod P \neq 1$.

$y = g^X \bmod p$.

Закрытый ключ:

$1 < x < p$.

Подпись сообщения M:

k – выбирается случайным образом, $1<k<(p-1)$, $НОД(k,p-1)=1$

a (шифротекст) = $g^k \bmod p$;

$m=H(M)$, где H – криптографическая Хэш-функция

b: $m = (x*a + k*b) \bmod (p-1)$.

(M,a,b**)** –текст и подпись передаваемые получателю

Проверка подписи:

$m=H(M)$

Проверка подписи: Если $y^a a^b \bmod p = g^m \bmod p$ то подпись верна.

3.6.3 Лабораторная работа Цифровая подпись Эль - Гамаля

Постановка задачи: разработать приложение для получения и проверки цифровой подписи на основе алгоритма Эль- Гамаля.

Порядок выполнения: экранная форма приложения представлена на рисунке 3.8.

Листинг программного кода представлен ниже.

```
implementation
{$R *.dfm}
Var p,g,x,y,k,a,b,M: integer;
 Function aXmodN(a,x,n: integer): integer;
Var i,b,c: integer;
begin
```

```
i:=0; b:= a mod n; c:=0;
 While i<x do
 begin
  i:=i+1;
  If i=1 then c:=b
         else c:= (c*b) mod n;
 end;
 aXmodN:=c;
end;{aXmodN}
```

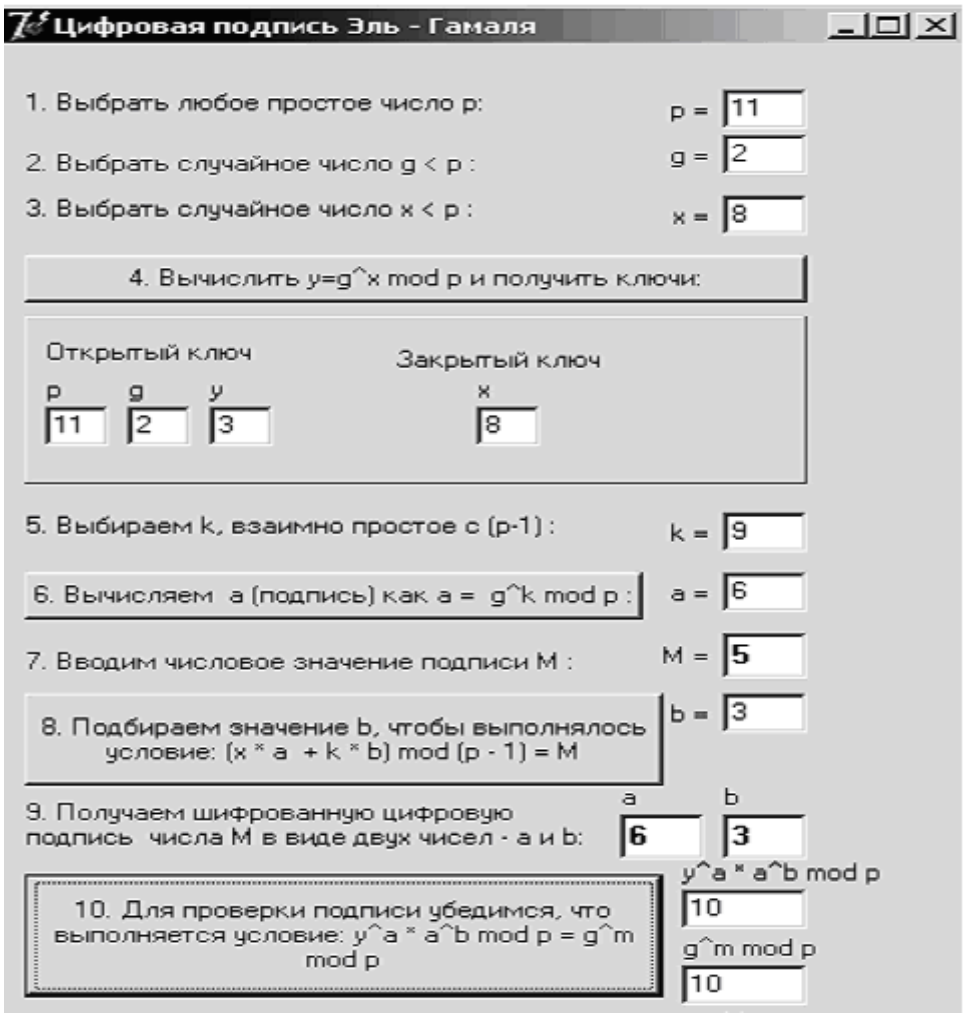


Рисунок 3.8 – Окно программы «Цифровая подпись Эль-Гамаля»

```
procedure TForm1.Button1Click(Sender: TObject);
begin
p:=StrToInt(LabeledEdit1.Text);
g:=StrToInt(LabeledEdit2.Text);
```

```
x:=StrToInt(LabeledEdit3.Text);
y:= aXmodN(g,x,p);
LabeledEdit4.Text:=IntToStr(p);
LabeledEdit5.Text:=IntToStr(g);
LabeledEdit6.Text:=IntToStr(y);
LabeledEdit7.Text:=IntToStr(x);
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
 k:= StrToInt(LabeledEdit8.Text);
 a:= aXmodN(g,k,p);
 LabeledEdit9.Text:=IntToStr(a);
end;
procedure TForm1.Button3Click(Sender: TObject);
Var i: integer;
begin
  M:=StrToInt(LabeledEdit10.Text);
 ListBox1.Visible:=true;
 i:=0;
 for b:=1 to 200 do
 begin
   If ((x*a+k*b) mod (p-1)) = M then
     begin
       ListBox1.Items.Strings[i]:=IntToStr(b);
       i:=i+1;
     end;
 end;
end;
procedure TForm1.ListBox1Click(Sender: TObject);
Var i: integer;
begin
i:=ListBox1.ItemIndex;
b:=StrToInt(ListBox1.Items.Strings[i]);
LabeledEdit11.Text:=IntToStr(b);
ListBox1.Visible:= false;
LabeledEdit12.Text:=IntToStr(a);
```

```
LabeledEdit13.Text:=IntToStr(b);
end;
procedure TForm1.Button4Click(Sender: TObject);
Var a1, a2, a3: integer;
begin
   {y^a*a^b mod p}
a1:= aXmodN(y,a,p) ;
a2:= aXmodN(a,b,p);
a3:= (a1*a2) mod p;
LabeledEdit14.Text:=IntToStr(a3);
    {g^m mod p}
a3:=aXmodN(g,m,p);
LabeledEdit15.Text:=IntToStr(a3);
end;
end.
```

3.6.3.1 Контрольные вопросы

1) Что такое цифровая подпись?
2) Кто впервые предложил идею цифровой подписи?
3) Какой алгоритм цифровой подписи был первым по времени изобретения?
4) Суть алгоритма Эль-Гамаля
5) В основу чего лег алгоритм Эль-Гамаля?
6) В чем отличие алгоритма Эль-Гамаля от алгоритма RSA?
7) Во сколько раз увеличилась длина цифровой подписи в алгоритме Эль-Гамаля?
8) Шаг 1. Выбирается любое простое число p.
   Шаг 2. Выбирается случайное число g которое должно быть меньше p.
   Укажите следующий шаг преобразований
9) Что следует из того, что условие $y^a a^b \bmod p = g^m \bmod p$ выполнено?
10) С помощью какого выражения вычисляется первый элемент подписи ***a***?
11) С помощью какого выражения вычисляется второй элемент подписи ***b***?

Список использованных источников

1 Agranovsky, A. V., Hady R. A. Crypto miracles with random oracle / The Proceedings of IEEE SIBCOM'2001, The Tomsk Chapter of the Institute of Electrical and Electronics Engineers, 2001.

2 Аграновский, А. В. Классические шифры и методы их криптоанализа / А. В. Аграновский, А. В. Балакин, Р. А. Хади, М: Машиностроение, Информационные технологии, № 10, -2001.

3 Аграновский, А. В. Криптография и открытые системы / А. В. Аграновский, Р.А. Хади, Я.М. Ерусалимский, Телекоммуникации, - 2000. - № 1.

4 Аграновский, А.В. Практическая криптография: алгоритмы и их программирование / А. В. Аграновский, Р. А. Хади. - Москва : Солон-Пресс, 2002. - 256 с.

5 Анин, Б.Ю. Защита компьютерной информации / Б.Ю. Анин, СПб.: БХВ - Петербург, - 2000. – 384с.

6 Милославская, Н. Г. Интрасети: доступ в Интернет, защита = Intranets: Internet Access, Security: учеб. пособие / Н. Г. Милославская, А. И. Толстой. - М. : Юнити, 2000. - 527 с.

7 Молдовян, А. А. Криптография: учебник / А. А. Молдовян, Н. А. Молдовян, Б. Я. Советов. - СПб. : Лань, 2001. - 219 с.

8 Основы криптографии: учеб. пособие для вузов / А. П. Алферов [и др.].- 3-е изд., испр. и доп. - М. : Гелиос АРВ, 2005. - 480 с.

9 Проскурин, В. Г. Защита в операционных системах / В. Г.Проскурин, С. В. Крутов, И. В. Мацкевич, М: Радио и связь, - 2000. - 168с.

10 Романец, Ю. В. Защита информации в компьютерных системах и сетях / Ю. В. Романец, П. А. Тимофеев, В. Ф. Шаньгин; под ред. В. Ф. Шаньгина.- 2-е изд., перераб. и доп. - М. : Радио и связь, 2001. - 376 с.

11 Ростовцев, А.Г. Алгебраические основы криптографии / А.Г. Ростовцев. - СПб.: Мир и Семья: Интерлайн, 2000. - 354 с.

12 Столлингс, В. Криптография и защита сетей / В.Столлингс, М: Вильямс, - 2001. – 672 с.

13 Устинов, Г. Н. Основы информационной безопасности / Г. Н. Устинов, М: Синтег, -2000. -248 с.

14 Чмора, А. Л. Современная прикладная криптография: учебное пособие / А. Л. Чмора, М.: Гелиос АРВ, - 2001. – 256 с.