МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

О.А. ГУЛЯЄВА

Методичні вказівки до виконання лабораторних робіт з дисципліни «Захист інформації» для студентів напряму 6.020100-документознавство та інформаційна діяльність

Дніпропетровськ
2015

Содержание

Лабораторная работа №1
Создание личных алфавитов

Пример 1. Создать собственный алфавит, как массив символов, непосредственным описанием последовательности символов.

```
procedure TForm1.Button1Click(Sender: TObject);
const
A : array[0..15] of char = ('0','1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'); // Описываем алфавит
var
i : integer;                                                    // объявляем переменную к-ва символов
begin
        {Вывод алфавита}
 Form1.Caption := '';                                           // очищаем переменную
 for i := 0 to 15 do                                            // в цикле от 0 до 15
 Form1.Caption := Form1.Caption + A[i] + '   ';                 // соединяем символы через пробел
end;
```

Пример 2. Создать алфавит, как массив символов, используя функцию Succ, указав в качестве начального параметра первый символ алфавита.

```
procedure TForm1.Button2Click(Sender: TObject);
var
A : array[1..32] of char;                                       // объявляем алфавит, как массив символов
i: integer;                                                     // объявляем переменную к-ва символов
begin
                                {Построение алфавита}
  i := 1;                                                       // определяем начальное значение переменной
  A[i] := 'A';                                                  // указываем первый символ, как русская буква (А)
  Repeat                                                        // повторяем цикл
  inc(i,1);                                                     // увеличиваем на единицу (можно i := i+1).
  A[i] := Succ(A[i-1]); //текущему элементу массива присваиваем следующий за предыдущим символом
  until i >= 32;                                                //завершаем цикл по условию ≥ 32 символа

                                {Вывод алфавита}
  Form1.Caption := '';                                          // очищаем переменную
  for i := 1 to 32 do                                           // в цикле от 1 до 32
  Form1.Caption := Form1.Caption + A[i] + ' ';                  // соединяем символы через пробел
end;
```

Пример 3. Создать обратный алфавит, как массив символов, используя функцию Pred, указав в качестве начального параметра последний символ алфавита.

```
procedure TForm1.Button3Click(Sender: TObject);
var
A : array[1..32] of char;                                       // объявляем алфавит, как массив символов
i: integer;                                                     // объявляем переменную к-ва символов
begin
                                {Построение алфавита}
  i := 1;                                                       // определяем начальное значение переменной
  A[i] := 'Я';                                                  // определяем первый символ как русская буква (Я)
  Repeat                                                        // повторяем цикл
  inc(i,1);                                                     // увеличиваем на единицу (можно i := i+1)
  A[i] := Pred(A[i-1]); //текущему элементу массива присваиваем предыдущий за предыдущим символом
  until i >= 32;                                                //завещаем цикл по условию ≥ 32 символа

                                {Вывод алфавита}
  Form1.Caption := '';                                          // очищаем переменную
  for i := 1 to 32 do  // в цикле от 1 до 32
  Form1.Caption := Form1.Caption + A[i] + '   ';                // соединяем символы через пробел
end;
```

Лабораторная работа № 2

Шифр Виженера – модифицированный шифр Цезаря, шифр XOR

Установить в форму элементы управления (страница Standard): Edit1 для ввода пароля, Memo1, Memo 2, Memo 3, Label 1, Label 2, Label 3 (надписи), GroupBox 1, GroupBox 2, GroupBox 3, в которые установить по 2 кнопки Button (рис. 1). Дать заголовки объектам.

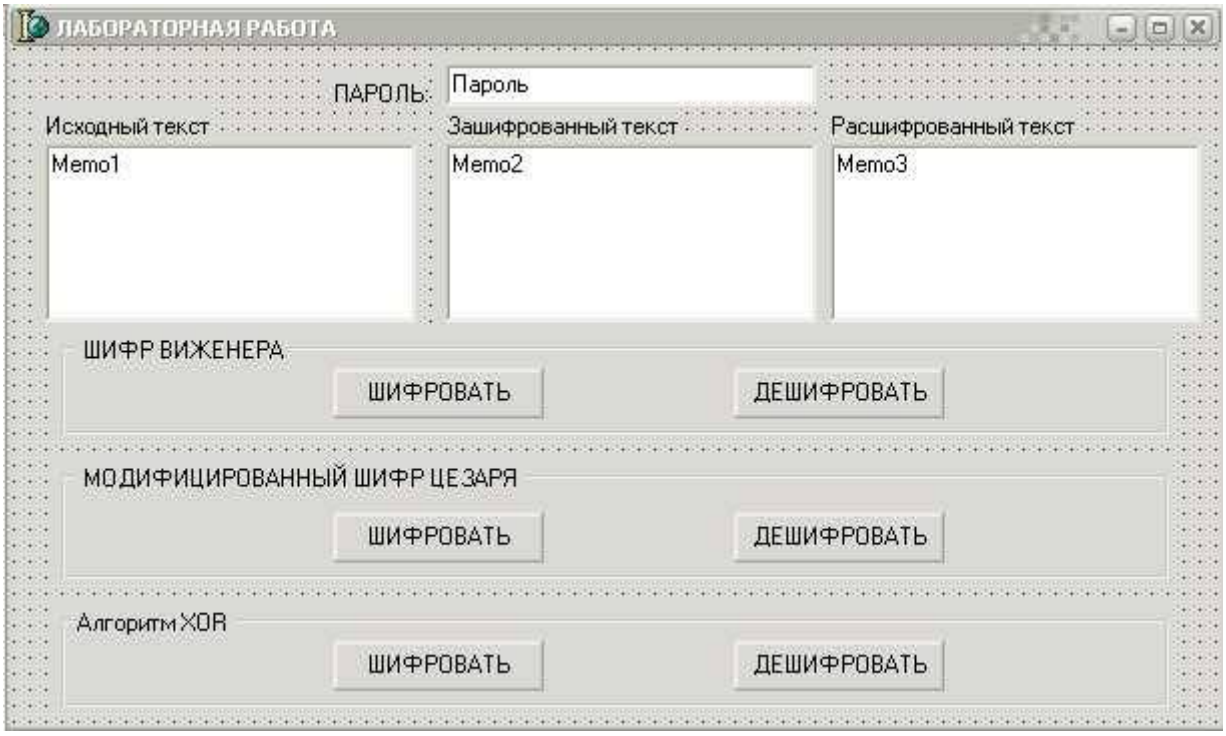

Рис. 1 Расположение компонентов в форме

В качестве алфавита будем использовать встроенную таблицу ASCII (Z_{256})

Шифр Виженера (модифицированный шифр Цезаря)
Вариант 1

Декларирование функции

```
{ Private declarations }
function VCR(PSW,TXT:string; CRT: boolean):string;                    //Виженер
```

Описание функции

```
{Шифрование, Дешифрование по Виженеру}
function TForm1.VCR(PSW,TXT:string; CRT: boolean):string;
var
i, NS:integer;                                                        // NS - номер символа пароля
TMP:string;                                                           // tmp - результирующий текст
begin
 tmp:='';                                                            // инициализация строки - "пусто"
 NS:=1;                                                             // номер первого символа пароля = 1
 for i:=1 to length(TXT) do                                        // с первого символа до конца строки:
  begin
   {---- шифрование и дешифрование ----}
   if CRT = true then                                               // если шифровать, то:
   TMP := TMP + chr(ord(TXT[i])+ord(PSW[NS]))
   else                                                             // в противном случае - дешифруем
   TMP := TMP + chr(ord(TXT[i])-ord(PSW[NS]));
   {---- вычисление следующего символа пароля ---}
```

```
    {----  наложение символов на всю строку  ---}
    NS := NS + 1;                          //вычисляем номер следующего символа пароля
    if NS > length(PSW) then  NS:=1;       // если номер символа пароля > длины пароля
                                           // номер  символа пароля = 1
   end;
  Result:=TMP;                             //Результат работы функции
end;
```

Шифрование

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 Memo2.Text :=  VCR(Edit1.Text,Memo1.Text, true);
end;
```

Расшифрование

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Memo3.Text :=  VCR(Edit1.Text,Memo2.Text,false);
end;
```

Задание. Проверьте работу функции (зашифруйте и расшифруйте произвольный текст, изменяя пароль).

Модифицированный шифр Цезаря (шифр Виженера)
Вариант 2

Декларирование функции

```
{ Private declarations }
  function VCR(PSW,TXT:string; CRT: boolean):string;      //Виженер
  function CZR(PSW,TXT:string; CRT: boolean):string;      //Цезарь
```

Описание функции

```
                {Шифрование по модифицированному Цезарю}
function TForm1.CZR(PSW,TXT:string; CRT: boolean):string;
var
i, NS:integer;                                  // NS - номер символа пароля
TMP:string;                                     // tmp - результирующий текст
begin
  tmp:='';                                      // инициализация строки - "пусто"
  NS:=1;                                        // номер первого символа пароля = 1
  for i:=1 to length(TXT) do                    // с первого символа до конца строки:
   begin
    {---- шифрование и дешифрование ----}
    if CRT = true then // если шифровать, то:
    TMP := TMP + Chr(  (Ord(TXT[i]) + Ord(PSW[NS])) mod 256  )
    else                                        // в противном случае - дешифруем
    TMP := TMP + Chr(  (Ord(TXT[i]) - Ord(PSW[NS])) mod 256  );
    {---- вычисление следующего символа пароля ---}
    {----  наложение символов на всю строку  ---}
    NS := NS + 1; //вычисляем номер следующего символ пароля
    if NS > length(PSW) then  NS:=1;            // если номер символа пароля > длины пароля
                                                // устанавливаем номер символа пароля =1
   end;
  Result:=TMP;                                  //Результат работы функции
end;
```

Шифрование

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Memo2.Text :=  CZR(Edit1.Text,Memo1.Text, true);
end;
```

Расшифрование

```
procedure TForm1.Button4Click(Sender: TObject);
begin
   Memo3.Text :=  CZR(Edit1.Text,Memo2.Text,false);
end;
```

Задание. Проверьте работу функции (зашифруйте и расшифруйте произвольный текст, изменяя пароль).

Убедитесь, что шифры Виженера и модифицированный шифр Цезаря дают один и тот же результат, т.е. реализуют одинаковый алгоритм с различной программной реализацией.

Шифрование методом XOR

Декларирование функции

```
{ Private declarations }
  function VCR(PSW,TXT:string; CRT: boolean):string;                    //Виженер
  function CZR(PSW,TXT:string; CRT: boolean):string;                    //Цезарь
  function TXT_XOR(PSW,TXT:string):string;                              // Метод
XOR
```

Описание функции

```
{Шифрование и дешифрация методом XOR}
function TForm1.TXT_XOR(PSW,TXT:string):string;
var
i, NS:integer;                                                  // NS - номер символа
пароля
TMP:string;                                                     // tmp - результирующий текст
begin
  tmp:='';                                                      // инициализация строки -
"пусто"
  NS:=1;                                                        // номер первого символа пароля = 1
  for i:=1 to length(TXT) do                                    // с первого символа до конца строки:
    begin
      {---- шифрование и дешифрование ----}
      TMP := TMP + Chr(Ord(TXT[i]) xor Ord(PSW[NS]));
      {---- вычисление следующего символа пароля ---}
      {----   наложение символов на всю строку   ---}
      NS := NS + 1;                                             //вычисляем номер следующего символ пароля
      if NS > length(PSW) then  NS:=1;                          // если номер символа пароля > длины пароля
                                                                // номер  символа пароляь=1
    end;
  Result:=TMP;                                                  //Результат работы функции
end;
```

Шифрование

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  Memo2.Text :=  TXT_XOR(Edit1.Text,Memo1.Text);
end;
```

Расшифрование

```
procedure TForm1.Button6Click(Sender: TObject);
begin
  Memo3.Text :=  TXT_XOR(Edit1.Text,Memo2.Text);
end;
```

Задание. Проверьте работу функции, изменяя пароль.

Лабораторная работа №3

Модель шифровальной машины ”ЭНИГМА”

Особенностью алгоритма ЭНИГМА является сдвиг текущего символа в строке текста на значение некоторой функции, зависимой от номера позиции символа в строке. В нашей работе будем применять функцию сдвига f(kx,i) = (kx * i)^2, где

i – порядковый номер символа в строке,

kx – множитель (*нечетное число* = 0, 1, 3, 5, 7, . .., n).

В качестве алгоритма шифрования может выступать алгоритм Цезаря с переменным шагом замены.

Шифрование по таблице ASCII $C_i = T_i + (kx * i)^2 (\mathrm{mod}\, 256)$, (1)

Расшифрование по таблице ASCII $T_i = C_i - (kx * i)^2 (\mathrm{mod}\, 256)$. (2)

При kx = 0 шифрование и расшифрование происходить не будут, так как Ci = Ti + 0 (mod 256) и Ti = Ci – 0 (mod 256).

Для выполнения работы будем использовать компоненты :

- SpinEdit (множитель), страница Samples;
- три компонента Memo (открытый текст, криптограмма, расшифрованный текст);
- две кнопки Button (1 - шифровать, 2 - расшифровать).

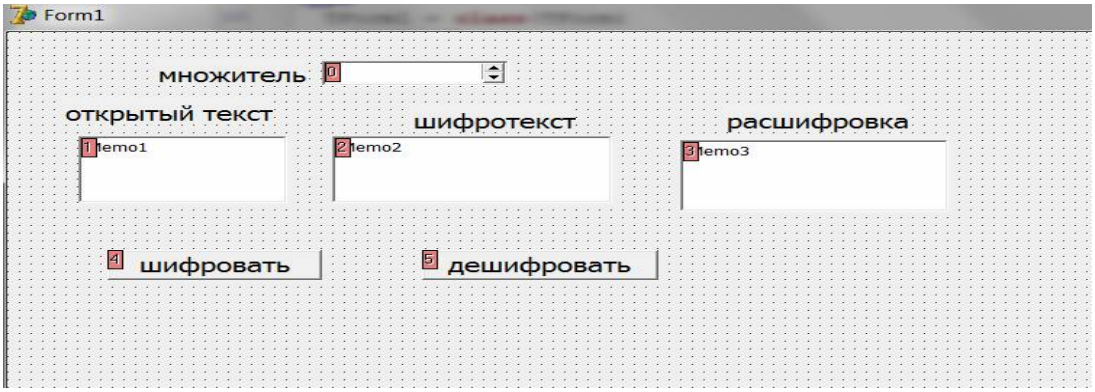

Рис. 1 Расположение компонентов в форме

Ниже приводится универсальная функция шифрования и расшифрования текста по описанному алгоритму. Входными параметрами функции являются:

- Tx – текст или криптограмма;
- kx – множитель сдвига;
- Encrupt – вид операции (true – шифрование, false – расшифрование).

Результатом работы функции является текст либо криптограмма.

Декларирование функции

```
{ Private declarations }
  function EnDeCrupt(Tx:String; Kx:Integer; Encrupt: boolean):String;
```

Описание функции

```
function TForm1.EnDeCrupt(Tx:String; Kx:Integer; Encrupt: boolean):String;
var
i: integer;
X: String;
begin
```

```
X:='';                                                    //Определение начальных значений
 for i :=1 to Length(Tx) do                               // С первого символа, до конца текста
 if Encrupt = true then          //если Encrupt = true, то: шифровать: C = T + (i*kx)^2 (mod 256)
 X := X + Chr(   (Ord(Tx[i]) + SQR(i * kx))  mod 256  )
 Else                                  // в противном случа дешифровать: T = C - (i*kx)^2 (mod 256)
 X := X + Chr(   (Ord(Tx[i]) - SQR(i * kx))  mod 256  );
 Result := X;                                                          // вывод результата
end;
```

Шифрование

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Memo2.Text := EnDeCrupt(Memo1.Text, SpinEdit1.Value, true);
end;
```

Расшифрование

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 Memo3.Text := EnDeCrupt(Memo2.Text, SpinEdit1.Value, false);
end;
```

Задание 1. Выполните шифрование и расшифрование текста при различных значениях множителя.

Задание 2. Измените функцию сдвига, например на $f(kx,i)=\frac{i}{2\pi}*kx$ и посмотрите на результат.

Шифрование

```
X := X + Chr(   (Ord(Tx[i]) + TRUNC(i/(2*pi) * kx) ) mod 256  )
```

Расшифрование

```
X := X + Chr(   (Ord(Tx[i]) - TRUNC(i/(2*pi) * kx) ) mod 256  );
```

***Примечание**: множитель kx может принимать любые значения.*

Лабораторная работа №4

Простые числа

Натуральное число *p*, большее единицы, называется **простым**, если оно имеет только 2 положительных делителя 1 и само число *p*. Например числа 3, 5, 7.

Натуральное число называется **составным**, если оно имеет более 2-х положительных делителей. Например числа 6, 8, 9.

Теорема арифметики. Любое натуральное число *n*, большее единицы, может быть разложено в произведение простых чисел (множителей), причем это разложение единственное.

Разложение натурального числа *n* имеет вид:

$n = p1^{a_1} \cdot p2^{\alpha_2} \cdot \ldots \cdot pk^{a_k}$, где $p1$, $p2$,... pk — различные простые числа, $a_1,\ldots,a_k \in N$ (множество натуральных чисел).

Задача проверки является ли число простым, а так же задача получения больших простых чисел имеют важные приложения в криптографии.

Решается задача: определить, является ли натуральное число *n* простым.

Метод пробных делений

Пусть число $n \in$ N. Если *n* – составное число, то *n* = *ab*, где $1 < a \le b$, причем $a \le \sqrt{n}$ (если *b=a, то* $n=a^2 \rightarrow a = \sqrt{n}$).

Будем искать делители числа *n*. Для этого достаточно проверить, делится ли число *n* на 2, 3, 4, . . ., $\sqrt{n}$.

Если делитель найден, то число *n* — составное (не является простым). Если, делитель не будет найден, то *n* – простое число.

Пример. Сгенерировать таблицу простых чисел в выбранном диапазоне значений.

В примере использованы:

- два компонента SpinEdit — для выбора значений из диапазона чисел;
- компонент Memo — для отображения результатов;
- компоненты Label — для заголовков;
- командная кнопка Button.

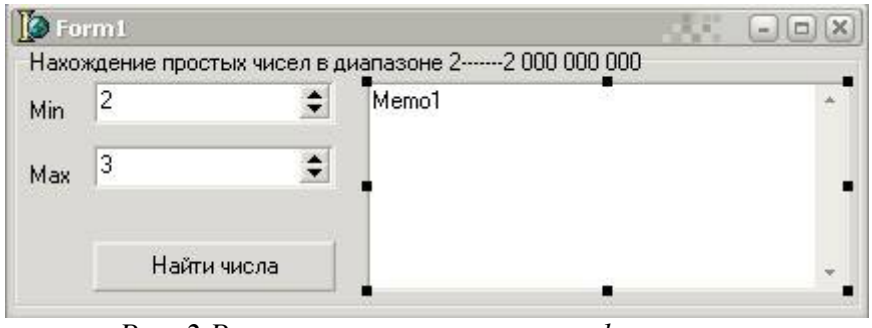

Рис. 2 Расположение элементов в форме

Функция проверки числа на простоту

Так как функция не объявляется в разделах декларирования, то описание функции необходимо располагать в Unit'e выше обработчиков событий.

```
{--- функция проверки чисел на их простоту ---}
{----- возвращает TRUE, если число простое ---}
function Simple(n:integer):Boolean;
var k:Boolean; i:integer;
begin
k:=true;
 if n<>2 then
   for i:=2 to trunc(sqrt(n))+1 do
   if n mod i = 0 then
   begin
   k := false;
   break;
 end;
Result:=k;
end;
```

Тестированиеа функции

```
// вывод простых чисел в диапазоне от N до M
procedure TForm1.Button1Click(Sender: TObject);
var
i:integer;
begin
  Memo1.Clear;
  for i := SpinEdit1.Value to SpinEdit2.Value do
  if Simple(i) = true then Memo1.Lines.Add(IntToStr(i));
end;
```

Задание. Вывести таблицу простых чисел в диапазоне от 2 до 30, от 50 до 100.

Решето Эратосфена

Алгоритм построения таблицы простых чисел в диапазоне от 2 до N.

Из исходной таблицы чисел

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ……

нужно вначале вычеркнуть все числа, делящиеся на 2, кроме 2.

Затем взять число 3 и вычеркнуть все последующие числа, делящиеся на 3.

Затем взять следующее, не вычеркнутое число (т. е. 5) и вычеркнуть все последующие делящиеся на него числа, и так далее.

В итоге останутся лишь простые числа.

Задание.. Построить таблицу простых чисел в диапазоне от 2 до 70 пользуясь алгоритмом «Решето Эратосфена».

Лабораторная работа №5

Генератор псевдослучайной последовательности

Генератор псевдослучайной последовательности чисел может быть описан рекуррентной формулой:

$$g_i = a\, g_{i-1} + b \pmod m, \qquad \text{где } i=1, 2,\ldots n \quad (1)$$

где: g_i — i-й член последовательности псевдослучайных чисел; a, b, m и g_0 — ключевые параметры.

Данная последовательность состоит из целых чисел от 0 до m–1, и если элементы g_i и g_j совпадут, то последующие участки последовательности также совпадут: $g_{i+1} = g_{j+1}$, $g_{i+2} = g_{j+2}$, и т.д.

Поэтому последовательность $\{g_i\}$ является периодической, и ее период не превышает m.

Пример. Пусть M — длина сообщения (например 500 символов),
L — количество символов в алфавите (Z256).
Необходимо создать последовательность случайных чисел (последовательность символов ключа), равную длине сообщения (M), значения которой принадлежат алфавиту длины L.

Для этого должны быть выполнены условия:

- параметр **a** взаимно прстой с M, то есть a=M+1;
- $g_0 = b \pmod L$.

Тогда формула (1) примет вид $g_i = a\, g_{i-1} + b \pmod L$, (2)

где b — ключевой параметр.

Для того чтобы период последовательности псевдослучайных чисел, сгенерированной по формуле (2), был максимальным (равным L), параметры формулы должны удовлетворять следующим условиям:

b и L —взаимно простые числа, b > L;
a–1 делится на любой простой делитель числа L;
a–1 кратно 4, если L кратно 4.

Для реализации поставленной задачи используются компоненты:

SpinEdit – для ввода секретного числа B;
Memo – для отображения результатов;

Label — для заголовка;

Button — для запуска обработчика событий (см. рис.1).

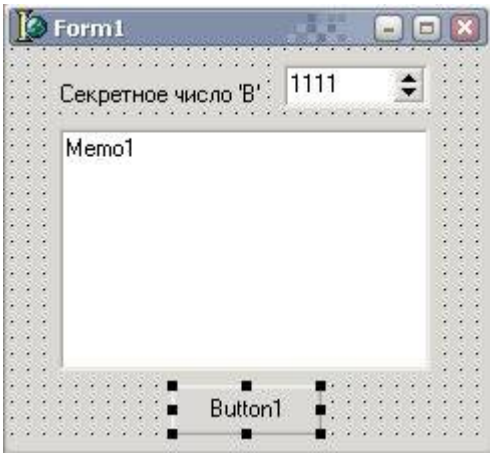

Рис.1

Процедура, реализующая алгоритм $g_i = a\, g_{i-1} + x \pmod L$, i=1, 2, …

```
 //============================================================
{Процедура формирования случайной последовательности, равной размеру передаваемого
сообщения}
                {Входные параметры: X - секретный ключ, M - размер сообщения }
                    {Выходной параметр: RCOD - массив случайных чисел}
 //============================================================
procedure RND_CODE(X, M: integer; var RCOD: array of integer);
var
i,A: integer;
begin
A:= M + 1;                                    // Число A взаимно простое с числом M
RCOD[0] := X mod 256;                         // Первый символ последовательности
 for i := 1 to M - 1 do                       // Цикл последовательности
  RCOD[i] := (A*RCOD[i-1] + X) mod 256;       //текущий символ последовательности
end;
```

Создание последовательности псевдослучайных чисел

```
procedure TForm1.Button1Click(Sender: TObject);
var
A : array[1..500] of integer;                 // Массив последовательности, равный длине сообщения
i, X: integer;
begin
   X := SpinEdit1.Value;                      // Секретный ключ
   RND_CODE(X,High(A),A);                     // Обращение к процедуре создания последовательности
   Memo1.Clear;                               // Очистка окна
   for i:= 1 to High(A) do                    // Вывод значений последовательности
   Memo1.Lines.Add('№ ' + IntToStr(i) + ' Значение числа = ' + IntToStr(A[i]));
end;
end.
```

Задание. Создать последовательность символов ключа, равную длине сообщения (300 символов), значения которой принадлежат алфавиту Z32.

Деловая игра

Пусть задан алфавит Z31.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
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

Необходимо вычислить последовательность символов ключа для текста размером M = 5, в алфавите размером L = 31 (Z31). При этом

$g_i = a\, g_{i-1} + b \pmod L$; a=M+1; $g_0 = b \pmod L$.

Порядок действий.

1. Выбираем секретный ключ ***b*** = 35, больший размера алфавита ***L***.
2. Вычисляем число a по формуле $a = M + 1 = 5 + 1 = 6$.
3. Вычисляем символ g_0 по формуле: $g_0 = b(\mathrm{mod}\, L)$ = 35 (mod 31) = 4.
4. Вычисляем первый символ g_1 по формуле $g_1 = a * g_0 + b(\mathrm{mod}\, L)$ = 6 * 4 + 35(mod 31) = 59 (mod 31) = 28.

5. Вычисляем второй символ g_2 по формуле $g_2 = a * g_1 + b (\mathrm{mod}\, L)$ = 6 * 28+ 35(mod 31) = 203 (mod 31) = 17.
6. Вычисляем третий символ g_3 по формуле $g_3 = a * g_2 + b (\mathrm{mod}\, L)$ = 6 * 17+ 35(mod 31) = 137 (mod 31) = 13.
7. Вычисляем четвёртый символ g_4 по формуле $g_4 = a * g_3 + b (\mathrm{mod}\, L)$ = 6 * 13+ 35(mod 31) = 113 (mod 31) = 20.

Таким образом, мы получили последовательность, равную:

№ 1 Значение числа = 4 символ Д
№ 2 Значение числа = 28 символ Э
№ 3 Значение числа = 17 символ С
№ 4 Значение числа = 13 символ Н
№ 5 Значение числа = 20 символ Ф

Если изменить секретный ключ ***b*** (например, пусть ***b*** = 54) то мы получим последовательность:

№ 1 Значение числа = 23 символ Ч
№ 2 Значение числа = 6 символ Ж
№ 3 Значение числа = 28 символ Ю
№ 4 Значение числа = 5 символ Е
№ 5 Значение числа = 22 символ Ц

Задание: Попарно, обменяйтесь секретным ключом ***b*** и вычислите псевдослучайные последовательности для текста заданной длины в алфавите Z31. Результаты у Вас должны совпасть.

*Таким образом, в дальнейшем мы можем рассчитывать псевдослучайную последовательность, равную длине сообщения, на основании секретного ключа **b** и любого алфавита **L**. Рассчитанную последовательность чисел целесообразно применять в алгоритме "Одноразовый блокнот" или "XOR".*

ЛАБОРАТОРНАЯ РАБОТА №6

Вычисление значения ХЭШ функции сообщения

Вычисление значения ХЭШ функции выполняется методом посимвольной свертки сообщения в соответствии с алгоритмом:

$$H = (h(T_{i-1}) \oplus T_i)\ shl\ 1, \quad \text{где} \quad i=1, 2,\ldots$$

$h(T_0) = 0$ — начальное значение функции,

T_i — текущие символы сообщения (таблица ASCII).

Порядок действий.

1. Установить в форму компоненты (см. рис. 1):
   - два компонента Edit — для ввода сообщения и получения ХЭШ функции соответственно;
   - Button — для запуска обработчика события;
   - два компонента Label — для надписей.

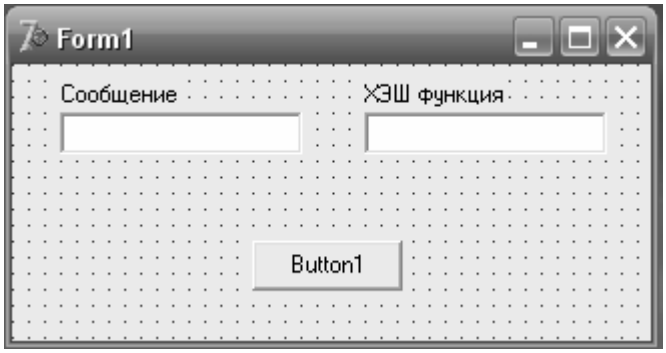

Рис. 1 Дизайн приложения

2. Обработчик события Click по кнопке имеет вид:

```
procedure TForm1.Button1Click(Sender: TObject);
var
i, HI : Integer;
S,HS : string;
begin
S := Edit1.Text; // присваиваем строке значение поля ввода
HI := 0;   // определяет начальное значение ХЭШ функции = 0
 for i := 1 to length(S) do // в цикле от 1 до конца строки
 begin
 HI := (HI xor ORD(S[i])) shl 1; // вычисляем текущее значение функции
 end;
 HS := IntToHex(HI,8); // преобразуем значение функции в 16-ти ричный формат
 Edit2.Text := HS;     // выводим значение функции
end;
```

Задание. Ввести сообщение, вычислить значение ХЭШ функции сообщения.

Лабораторная работа № 7
Алгоритм рекурсивного вычисления наибольшего общего делителя

Наибольшим общим делителем натуральных чисел a и b (НОД (a, b)) называется наибольшее число, на которое делятся a и b без остатка.

Например, НОД (12, 9) = 3.

Натуральное число *p*, большее единицы, называется ***простым***, если оно делится нацело только на 1 и на себя.

Два числа называются ***взаимно простыми*** (relatively prime), если их наибольший общий делитель равен **1**. Например, НОД (3, 5) = 1. Числа 3 и 5 являются взаимно простыми.

Если b нацело делится на a , то НОД*(a, b)* = a

Математик *Эйлер*, живший в 18 веке, обнаружил интересный факт:

НОД(a, b) = НОД(b Mod a, a)

Этот факт можно использовать для быстрого вычисления наибольшего общего делителя (greatest common divisor, **GCD**).

Например: GCD(9, 12) = GCD(12 Mod 9, 9)
= GCD(3, 9)
= 3

На каждом шаге числа становятся все меньше, так как 1<= b Mod *a* < *a*. В какой-то момент b разделится на *a* нацело, и работа процедуры завершится.

Открытие *Эйлера* закономерным образом приводит к рекурсивному алгоритму вычисления наибольшего общего делителя.

Рекурсивными процедурами (recursive procedure) называются процедуры, вызывающие сами себя.

Для выполнения работы используются два компонента SpinEdit (страница Samples) и командная кнопка Button (рис.1).

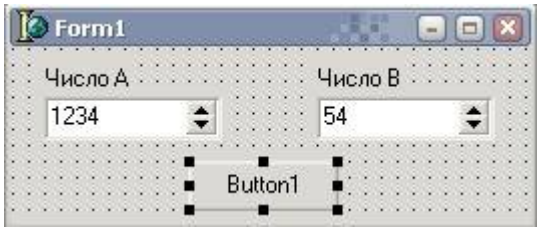

Рис.1

Декларирование функции

```
private
   { Private declarations }
  Function GCD(A, B:integer): Integer;
```

Описание функции

```
                              {Для функции безразлично (A,B) или (B,A)}
Function TForm1.GCD(A, B:integer): Integer;
begin
   If B Mod A = 0 Then                                  // Делится ли B на A нацело?
   Result := A                                          // Да. Функция выполнена.
   else
   Result := GCD(B mod A, A);                           //' Нет. Рекурсия (вызов самой себя).
end;
```

Вызов функции

```
procedure TForm1.Button1Click(Sender: TObject);
var
A, B, X : integer;
begin
 A := 12; B := 9;                    //числа A и B
 X := GCD(A,B);                      // результат работы функции
          {вывод информации}
 if X = 1 then  Form1.Caption := 'Числа вз. простые!' else  Form1.Caption := 'НОД = ' + IntToStr(X);
end;
```

Задание. Внести изменения в программу, что бы числа вводились из компонента SpinEdit.

Лабораторная работа №8

Шифрование мультипликативным ключом

Приводится пример программы шифрования и расшифрования текста. При этом используется некоторая функция, изменяющая стартовый (начальный) ключ шифрования: ключ шифрования изменяется в зависимости от текущего номера символа в тексте и некоторого множителя.

Для реализации программы использованы следующие компоненты (рис.1):

- два компонента SpinEdit – для ввода стартового (начального) значения ключа и множителя (страница Samples);
- три компонента Memo – для ввода текста, представления шифротекста и расшифрованного текста (страница Standard);
- две кнопки Button – для запуска обработчиков событий(страница Standard) ;
- два компонента Label — для задания заголовков (страница Standard) .

Задание. Исследовать приведенную программу, описать в формализованном виде алгоритм, реализованный функцией CCR.

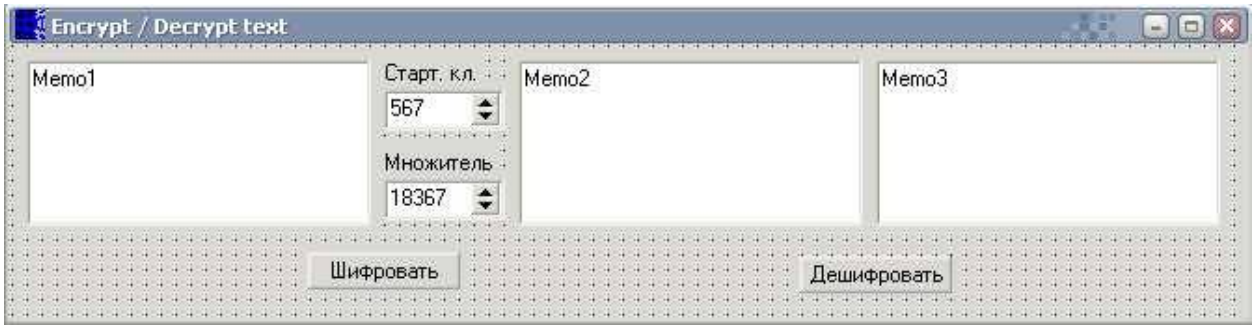

Рис. 1 Компоненты, используемые в программе

Глобальные переменные уровня модуля формы

```
var
  Form1: TForm1;
  StartKey, MultKey: integer;                    // Переменные - ключи
implementation
```

Функция шифрования и расшифрования

```
function CCR(const TXT:string; StartKey,MultKey:Integer; CRT:Boolean): string;
var
 i: Integer;
begin
 Result:='';
 for i:=1 to Length(TXT) do
  begin
   Result := Result + Chr(Ord(TXT[i]) xor (StartKey shr 8));
   if CRT = true then
   StartKey := (Ord(Result[i]) + StartKey) * MultKey
   else
   StartKey:=(Ord(TXT[i]) + StartKey) * MultKey;
  end;
end;
```

Обработчик события «шифрование»

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  StartKey := SpinEdit1.Value;                    //(567) Значение стартового ключа
  MultKey := SpinEdit2.Value;                     //(18367) Значение множителя
  Memo2.Text := CCR(Memo1.Text, StartKey, MultKey, true);
end;
```

Обработчик события «расшифрование»

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  StartKey := SpinEdit1.Value;                    // (567) Значение стартового
ключа
  MultKey := SpinEdit2.Value;                     // (18367) Значение множителя
  Memo3.Text:= CCR(Memo2.Text, StartKey, MultKey, false);
end;
```