Міністерство освіти і науки України
Національний університет водного господарства
та природокористування

Кафедра обчислювальної техніки

04-04-191

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабораторних робіт
з дисципліни
"Захист інформації в комп'ютерних системах"
студентами напряму підготовки
6.050102 "Комп'ютерна інженерія"

Частина ІІ

Рекомендовано
методичною комісією
напряму підготовки
"Комп'ютерна інженерія"
Протокол № 7
від 03 березня 2017 р.

Рівне 2017

Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни "Захист інформації в комп'ютерних системах" студентами напряму підготовки 6.050102 "Комп'ютерна інженерія". Частина ІІ. / П. В. Ольшанський, – Рівне: НУВГП, 2017, – 32 с.

Упорядник П. В. Ольшанський, старший викладач кафедри обчислювальної техніки.

Відповідальний за випуск
Б.Б. Круліковський, кандидат технічних наук, доцент, завідувач кафедри обчислювальної техніки.

Зміст

Лабораторна робота №5.

Тема: Програмна реалізація блочного алгоритму шифрування DES

Мета роботи. Створити програму для шифрування файлів довільного типу найпоширенішим в світі алгоритмом DES (Data Encryption Standard), відлагодити проект Delphi з декількома вкладеними модулями, вивчити криптографічні прийоми для надійного перемішування і збивання блоків інформації в процесі шифрування, інтегрувати програму в спільний пакет з раніше розробленими алгоритмами.

Теоретичні відомості

При використанні блочного шифру інформація поступає на вхід шифруючого пристрою (мікросхеми) порціями однакової довжини - блоками, кожен з яких шифрується окремо.

Американський стандарт шифрування DES розроблений в 1973-1976 роках групою математиків фірми IBM, рекомендований для використання в комерційних системах захисту інформації, детально вивчений і апробований на практиці. Були розроблені спеціальні теоретичні методи криптоаналізу - диференційний та лінійний, які можуть бути використані для аналізу окремих етапів чи неповних реалізацій, однак до цього часу не існує практичних методів зламування алгоритму DES, крім повного перебору ключів.

Довжина блоку шифрування - 64 біти, довжина ключа - 56 бітів. Швидкодія сучасних комп’ютерів дозволяє виконувати з допомогою паралельних мережевих обчислень перебрати всі можливі

ключі 2^{56} за декілька днів, а з допомогою спеціально побудованих суперкомп'ютерів навіть за декілька годин.

Шифр складається з послідовності бітових перестановок і підстановок, орієнтований на апаратну реалізацію. Архітектура сучасних процесорів не пристосована до бітових операцій DES, що ускладнює ефективну програмну реалізацію.

Кроки алгоритму DES.

1. Початкова перестановка бітів (Initial Permutation) згідно з таблицею ***transpos_number***

```
procedure InitialTransposition64(a:block64bits;
                                 var b:block64bits);
var i:byte;
begin
   for i:=1 to 64 do
                  b[i]:=a[transpos_number[i]]
end;
```

2. Розбивання блока на ліву і праву половини

```
procedure  Break_Block64_to_Left32_and_Right32;
var i:byte;
begin
  for i:=1 to 32 do
  begin
    left32[i]:=block64[i];
    right32[i]:=block64[i+32]
  end;
end;
```

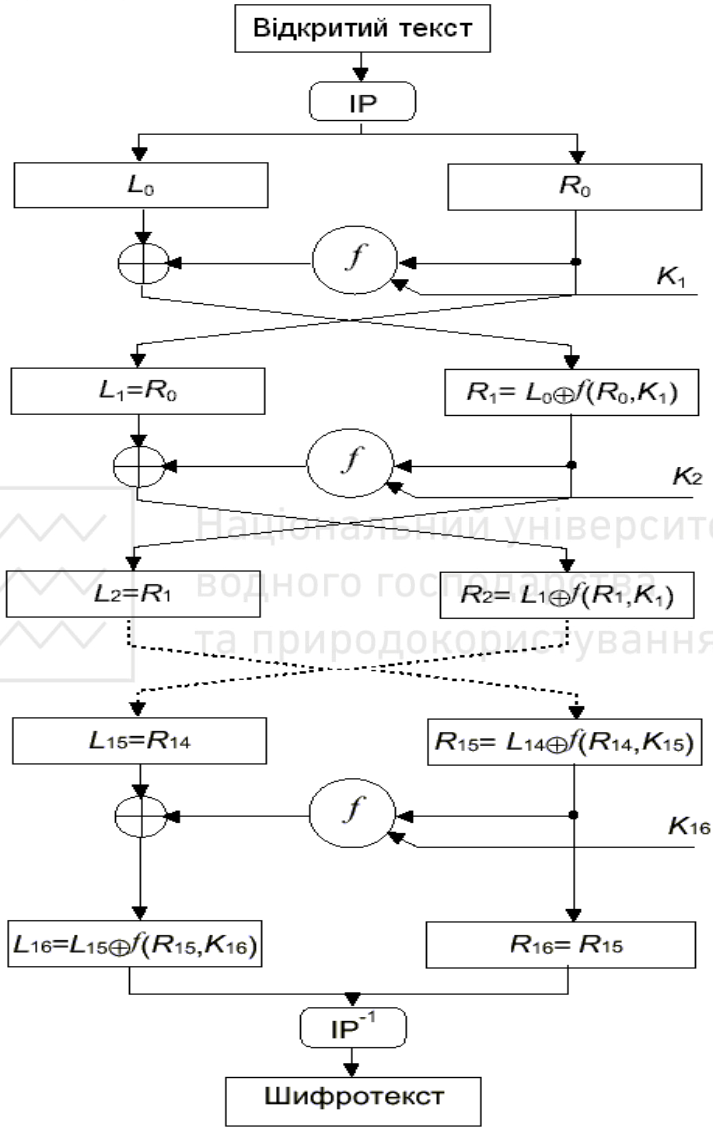

Рис 6. Схема перетворень блоку даних на етапах алгоритму DES

3. Повторення 16 однакових етапів

```
for i:=1 to 16 do
    begin
       Generate_Key48_for_Step(i);
       StepDES(i);
       if i<16 then
       begin
         left32 :=new_left32;
         right32:=new_right32;
       end
    end;
```

4. Об'єднання половин блоку

```
procedure  Merge_Left32_and_Right32_to_b64;
var i:byte;
begin
  for i:=1 to 32 do
  begin
    b64[i]:=new_left32[i];
    b64[i+32]:=new_right32[i]
  end;
end;
```

5. Кінцева перестановка (обернена до початкової)

```
procedure FinalTransposition64(a:block64bits;
                               var b:block64bits);
var i:byte;
begin
   for i:=1 to 64 do  b[transpos_number[i]]:=a[i]
end;
```

Обернений алгоритм DES (розшифровування) співпадає з прямим, тому використовується та сама процедура. Відрізняється тільки порядок підключів на окремих етапах - використовується обернена послідовність підключів.

Напрям шифрування визначається параметром типу

```
Direction=(Direct,Back);
```

Опис одного етапу DES.

1. Права половина блока розширяється з допомогою дублювання окремих бітів до 48 бітів з використанням таблиці (масиву) ***permut***.

```
procedure  Right32_to_48;
var i:integer;
begin
  for i:=1 to 48 do
                b48[i]:=right32[permut[i]];
end;
```

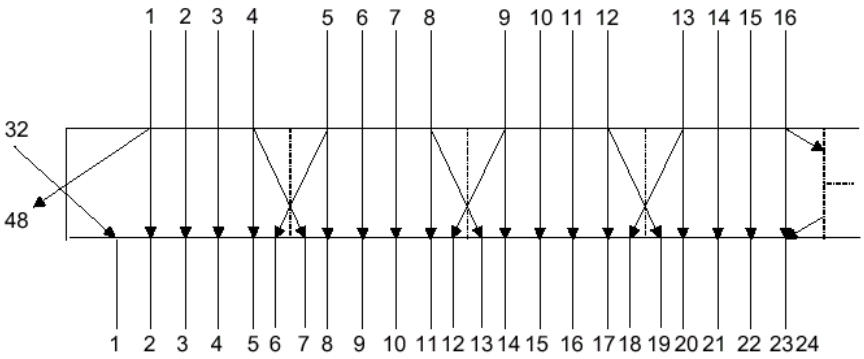

Рис 7. Перестановка з розширенням

2. Виконується побітова операція XOR з 48-бітовим підключем, який спеціально готується для кожного етапу (опис див. нижче)

```
procedure  b48_XOR_key48;
var j:byte;
begin
   for j:=1 to 48 do
       if b48[j]=key48[j] then block48[j]:=0
                          else block48[j]:=1
end;
```

3. **Центральна частина етапу і всього алгоритму - підстановка з допомогою S-блоків**

48-бітний блок розбивається на 8 шестибітових блоків. Для кожного блоку використовується своя таблиця S-підстановки, яка має 4 рядки і 16 стовпчиків. Перший і шостий біт шестибітового блока визначають номер рядка, біти з 2 по 5 визначають номер стовпчика. В результаті зчитується число в діапазоні від 0 до 15, яке утворює 4-бітовий результуючий блок. 8 чотирибітових блоків дають 32-бітовий результат шифрування S-блоками.

```
procedure  S_Block48_to_b32;
type block6bits=array[1..6]of bit;
     block4bits=array[1..4]of bit;
var i,j,n,n1,n2:byte;
     b6:array[1..8]of block6bits;
     b4:array[1..8]of block4bits;
begin
```

```
  for i:=1 to 8 do
      for j:=1 to 6 do b6[i][j]:=b48[(i-1)*8+j];
  for i:=1 to 8 do  // 8 S_blocks:  6bits_to_4bits
  begin
    n1:=b6[i][1]*2+b6[i][6];
    n2:=b6[i][2]*8+b6[i][3]*4+b6[i][4]*2+b6[i][5];
    n:=s[i,n1,n2];
    b4[i][1]:=n div 8;
    b4[i][2]:=(n-8*b4[i][1]) div 4;
    b4[i][3]:=(n-8*b4[i][1]-4*b4[i][2]) div 2;
    b4[i][4]:=n mod 2;
  end;
  for i:=1 to 32 do b32[i]:=b4[i div 8+1][i mod 4 +1];
end;
```

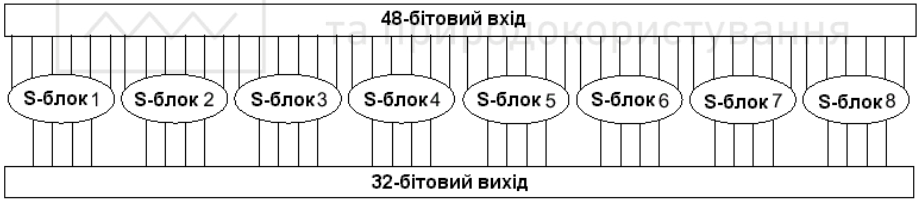

Рис 8. Схема роботи S-блоків

4. Біти одержаного 32-розрядного блока переставляються (P-блок)

```
      procedure  P_Block32;
      var i:byte;
      begin
        for i:=1 to 32 do r32[i]:=b32[p[i]];
      end;
```

5. Виконується побітова операція XOR лівої половини , яка від початку етапу не зазнала змін, із перетвореною правою половиною

```
procedure  Left32_XOR_r32;
var j:byte;
begin
  if k=16 then for j:=1 to 32 do
              if left32[j]=r32[j] then new_left32[j]:=0
                                  else new_left32[j]:=1
  else for j:=1 to 32 do
        if left32[j]=r32[j] then new_right32[j]:=0
                            else new_right32[j]:=1
end;
```

6. На кожному етапі змінюється тільки половина 64-бітового блоку, інша половина переставляється місцями із зашифрованою. На останньому 16 етапі перестановка не виконується.

```
if k=16 then new_right32:=right32
        else new_left32 :=right32
```

Основна частина процедури для одного етапу DES має вигляд

```
procedure StepDES(k:byte);
. . . (опис процедур етапу - наведені вище)
begin
   Right32_to_48;
   b48_XOR_key48;
   S_Block48_to_b32;
   P_Block32;
   left32_XOR_r32;
   if k=16 then new_right32:=right32
           else new_left32 :=right32
end;
```

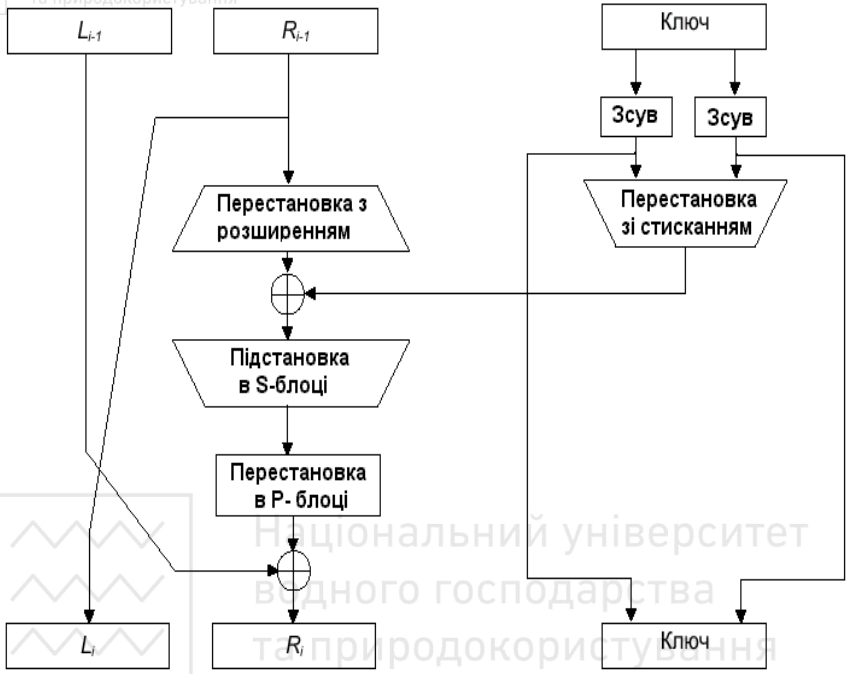

Рис 9. Схема одного етапу алгоритму DES

Перетворення ключа

1. Із заданого 64-бітового ключа формується 56-бітний ключ відкиданням кожного 8 біта

```
procedure KeyTransposition64_to_56;
var
    i:byte;
begin
      for i:=1 to 56 do
            key56[i]:=key64[key_transpos[i]]
end;
```

2. 56-бітний ключ розбивається на дві 28-бітні половини

```
procedure   Break_key56_to_left28_and_right28;
var i:byte;
begin
   for i:=1 to 28 do
   begin
     left_key28[i] :=key56[i];
     right_key28[i]:=key56[i+28]
   end
end;
```

3. (Повторюється на кожному з 16 етапів). До кожної з половинок застосовується операція циклічного зсуву вліво (для прямого ходу алгоритму) або вправо (для розшифровування). Кількість бітів зсуву на кожному етапі відрізняється і задається таблицями ***Shift_L*** та ***Shift_R***.

```
const
   shift_L:array[1..16]of byte=
                  (1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1);
   shift_R:array[1..16]of byte=
                  (0,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1);
procedure Cyclic_SHL_left_key28(k:byte);
var i,it:byte;btemp:bit;
begin
  for it:=1 to shift_L[k] do
  begin
    btemp:=left_key28[1];
    for i:=1 to 27 do left_key28[i]:=left_key28[i+1];
    left_key28[28]:=btemp
  end
```

```
end;
procedure Cyclic_SHL_right_key28(k:byte);
var i,it:byte;btemp:bit;
begin
  for it:=1 to shift_L[k] do
  begin
    btemp:=right_key28[1];
    for i:=1 to 27 do right_key28[i]:=right_key28[i+1];
    right_key28[28]:=btemp
  end
end;
procedure Cyclic_SHR_left_key28(k:byte);
var i,it:byte;btemp:bit;
begin
  for it:=1 to shift_R[k] do
  begin
    btemp:=left_key28[28];
    for i:=28 downto 2 do
                    left_key28[i]:=left_key28[i-1];
    left_key28[1]:=btemp
  end
end;
procedure Cyclic_SHR_right_key28(k:byte);
var i,it:byte;btemp:bit;
begin
  for it:=1 to shift_R[k] do
  begin
    btemp:=right_key28[28];
    for i:=28 downto 2 do
```

```
    right_key28[i]:=right_key28[i-1];
  right_key28[1]:=btemp
 end
end;
```

3. З одержаних після зсувів 28-бітових частин формується 48-бітовий ключ, який приймає участь в операції XOR з правою частиною блоку даних на даному етапі алгоритму (див. вище)

```
procedure Step_Key_48;
var i:byte;
begin
  for i:=1 to 28 do
  begin
    k56[i]:=left_key28[i];
    k56[i+28]:=right_key28[i]
  end;
  for i:=1 to 48 do
                key48[i]:=k56[p56key48[i]];
end;
```


Загальний вигляд процедури формування ключа для етапу DES

```
procedure Generate_Key48_for_Step(k:byte);
. . .    (опис констант і процедур - наведено вище)
begin
 if  Dir_or_back=Direct then
    (прямий хід алгоритму - шифрування)
 begin
  Cyclic_SHL_left_key28(k);
  Cyclic_SHL_right_key28(k);
```

```
 end
 else
      (зворотній хід алгоритму - розшифровування)
 begin
  Cyclic_SHR_left_key28(k);
  Cyclic_SHR_right_key28(k);
 end;
 Step_Key_48;
end;
```

Загальний вигляд процедури шифрування DES

(Оформлена у вигляді модуля DES.PAS. Програма для перевірки, відлагодження та аналізу роботи модуля DES_PROG.PAS, див. Додаток 2)


```
unit DES;  //  Copyright by P. Olszanski, 2004
interface
uses Tables; (див. Додаток 1.)
type
  bit=0..1;
  block64bits=array[1..64] of bit;
  Direction=(Direct,Back);
procedure DES_alg(
           Dir_or_back:Direction; //напрям шифрування
           start_block64,         //вхідний блок
           key64      :block64bits;// ключ
     var end_block64 :block64bits);// результуючий блок
implementation
type   // типи даних для проміжних результатів
  block56bits=array[1..56] of bit;
```

```
  block48bits=array[1..48] of bit;
  block32bits=array[1..32] of bit;
  block28bits=array[1..28] of bit;
var
// внутрішні масиви модуля - відповідають регістрам DES
   key56,k56      :block56bits;
   left_key28,
   right_key28    :block28bits;
   key48          :block48bits;
   block64, b64   :block64bits;
   b48,block48    :block48bits;
   left32, right32,
   b32,r32,new_left32,
   new_right32    :block32bits;
   i:byte;// лічильник
procedure DES_alg(
         Dir_or_back:Direction; //напрям шифрування
         start_block64,         //вхідний блок
         key64      :block64bits;// ключ
    var end_block64 :block64bits);// результуючий блок
. . . (опис внутрішніх процедур - наведено вище)
begin
    // початкова перестановка
    InitialTransposition64(start_block64,block64);
    // розбиття на ліву і праву половину
    Break_Block64_to_Left32_and_Right32;
    // формування 56-бітного ключа
    KeyTransposition64_to_56;
    // розбиття ключа на ліву і праву половину
```

```
    Break_key56_to_left28_and_right28;
    // повторення 16 етапів
    for i:=1 to 16 do
    begin
     // генерування 48-бітового ключа для даного етапу
      Generate_Key48_for_Step(i);
     // виконання процедури для i-ого етапу DES
      StepDES(i);
      if i<16 then //для всіх етапів, крім останнього
      begin
     // підготовка вхідних даних для наступного етапу
        left32 :=new_left32;
        right32:=new_right32;
      end
    end;
    // об'єднання половинок
    Merge_Left32_and_Right32_to_b64;
    // кінцева перестановка - обернена до початкової
    FinalTransposition64(b64,end_block64);
end;
begin
end.
```

Задавати вхідні дані і використовувати результати у вигляді бітових масивів не завжди зручно, тому пропонується процедура, яка працює з блоками у вигляді восьмисимвольних рядків. (Оформлена у вигляді модуля DES8char.PAS. Програма для перевірки, відлагодження та аналізу роботи модуля Test_DES.PAS, див. Додаток 3)

```
Unit DES8char;
interface
uses DES;
type  str8=string[8];
procedure DES_string8(dir_or_back:Direction;
                      string8,string_key:str8;
                      var end_string8:str8);
implementation
procedure DES_string8(dir_or_back:Direction;
                      string8,string_key:str8;
                      var end_string8:str8);
var
  start_block64,key64,end_block64:block64bits;
procedure Start(s:str8 ;  var bl:block64bits);
var i,j:byte; c:char; b,filter:byte;
begin
 for i:=1 to 8 do
 begin
   c:=s[i];
   b:=ord(c);
   filter:=128;
   for j:=1 to 8 do
   begin
     if b AND filter =0 then bl[(i-1)*8+j]:=0
                        else bl[(i-1)*8+j]:=1;
     filter:= filter SHR 1
   end
 end
end;
```

```
procedure  Finish(be:block64bits;var s8:str8);
var
   i,j,k:byte;
begin
  s8:='12345678';
  for i:=1 to 8 do
  begin
    k:=0;
    for j:=1 to 8 do
                  k:=2*k+be[(i-1)*8+j];
    s8[i]:=chr(k);
  end;
end;
begin
    Start(string8,start_block64);
    Start(string_key,key64);
    DES_alg(dir_or_back,start_block64,key64,
                        end_block64);
    Finish(end_block64,end_string8)
end;

begin
end.
```


Графічне зображення перетворень блоку даних в DES алгоритмі на рис. 6 та блок-схема DES-алгоритму з вказанням розрядності операцій на рис.10.

Паскаль-програма для шифрування довільних файлів блоками по 8 символів (64 біти) DES-алгоритмом - FILE_DES.PAS див. Додаток 4.

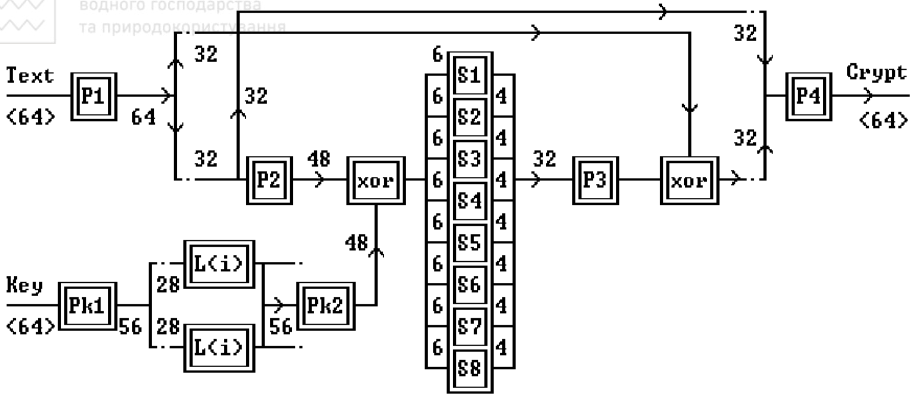

Рис.10 Блок-схема DES-алгоритму

Пояснення до блок-схеми

Text - початковий текст (блок 64 бітів);

Crypt - зашифрований блок;

Key - 64-х розрядний ключ;

числа - розрядність на даній вітці алгоритму;

P, Pk - перестановки;

S - підстановка 6 бітів => 4 біти ;

L(i) - зсув (i -- номер ітерації) ;

XOR - додавання за модулем 2 (виключне АБО);


- конкатенація (об'єднання) бітових рядків;


- розбиття блоків на два;


- обмежений точками відрізок повторюється 16 разів ;

Перестановки виконуються за формулою D[i] = A[P [i]], де

A - вхідний масив бітів;

D - результат перестановки (масив бітів) ;

P - таблиця перестановок (масив індексів для масиву A) ;

S - підстановка 6=>4. У відповідність шести бітам ставиться чотири. Підстановка проводиться за наступним правилом: нехай шість бітів на вході - (abcdef), тоді (af) визначають номер рядка, а (bcde) - номер стовпчика. Номери рядка і стовпчика визначають результат в діапазоні від 0 до 15, який зчитується з S-таблиці. Наприклад, при використанні таблиці S6, число 58 (111010) переводиться в 13 (1101).

Рис 11. Вигляд головної форми Delphi - проекту

Хід роботи

1. Відкрийте проект з попередньої лабораторної роботи в Delphi, доповніть кнопками Button8 і Button9 згідно рис.11.

2. Для кнопок Button8 і Button9 створіть спільний обробник події натискання за зразком попередніх лабораторних робіт, в якому послідовно читається і шифрується алгоритмом DES інформація з вхідного файлу порціями по 64 біти (8 символів) та записується послідовно в результуючий файл.

В модулі головної форми в розділі uses проекту підключіть модулі DES і DES8char.

```
procedure TForm1.Button8Click(Sender: TObject);
var Sen:Tobject;
procedure DES_algorithm;
var f,f1:file of char;  time0,time1:TDateTime;
      FileLen :integer; millisec :comp;
      c:char;    s,ss:string;
      k, k100, i, j, L, i1, num : integer;
      key_number : array[1..255] of integer;
      smemo1,smemo2 : string;
      dir_or_back:Direction;
      string8,string_key,end_string8:str8;
begin
    Sen:=Sender;
    string_key:='12345678';
    if psw1<>'' then
    begin
    while Length(psw1)<8 do psw1:=psw1+psw1;
    for  i:=1 to 8 do string_key[i]:=psw1[i];
    end;
    Form3.Caption:='Алгоритм шифрування DES';
    Form3.Label1.Caption:='Вхідний файл:'+name1;
    AssignFile(f,name1);
```

```
Reset(f); FileLen:= FileSize(f);
L:=FileLen div 8;     k100:= L div 100;
Form3.Label3.Caption:='Довжина файла '+
                        IntToStr(FileLen)+'байтів';
Form3.Label5.Caption:=' ';
Form3.Label2.Caption:=
               'Зашифрований файл:'+name1+'.cip';
AssignFile(f1,name1+'.cip' );
Rewrite(f1);    time0:=Time;
if Sen=Button8 then
     dir_or_back:=Direct // шифрування
else if Sen=Button9 then
          dir_or_back:=Back; // розшифровування
Form3.Gauge1.Progress:=0; // покажчик індикатора
smemo1:=''; smemo2:='';
for k:=1 to L do  //k- лічильник кількості блоків
begin
  s:=''; // початок формування блока із 8 символів
  for i:=1 to 8 do
  begin
     read(f,c);    // читання символа з файла
     s:=s+c       // формування блока із 8 символів
  end;
  string8:=s;   // шифрування блока із 8 символів
  DES_string8(dir_or_back,string8,string_key,
                   end_string8);
  // запис зашифрованого блока у файл
  for j:=1 to 8 do  write(f1,end_string8[j]);
    // формування рядків для виведення на екран
```

```
     smemo1:=smemo1+s; smemo2:=smemo2+end_string8;
    if length(smemo1)>63 then
    begin   //виведення на екран рядків по 64 символів
      Form3.Memo1.Lines.Add(smemo1);
      Form3.Memo2.Lines.Add(smemo2);
      smemo1:='';smemo2:=''//очищення рядкових буферів
    end ;
    if k mod k100=0 then
    begin // відлік відсотків на індикаторі
       Form3.Gauge1.Progress:=Form3.Gauge1.Progress+1;
       // виведення системного часу
       Form3.Label4.Caption:='Час '+TimeToStr(Now);
       Form3.Refresh; // оновлення форми
    end;
 // очищення полів Memo для запобігання переповнення
    if k mod 2000=0 then
    begin Form3.Memo1.Clear;  Form3.Memo2.Clear end;
   end;
// остання неповна порція записується у вихідний файл
   for i:=1 to FileLen mod 8 do
   if not eof(f) then
   begin
     Read(f,c);  smemo1:=smemo1+c;
     Write(f1,c); smemo2:=smemo2+c
   end;
     //виведення на екран останніх рядків
   Form3.Memo1.Lines.Add(smemo1);
   Form3.Memo2.Lines.Add(smemo2);
   time1:=Time-time0; // обчислення часового проміжку
```

```
    Form3.Label4.Caption:='Час шифрування '+
          FormatDateTime(' n хв. ss,zz сек.',time1);
millisec:=TimeStampToMsecs(DateTimeToTimeStamp(time))-
          TimeStampToMsecs(DateTimeToTimeStamp(time0));
    Form3.Label5.Caption:='Швидкість    '+
      IntToStr(Trunc(FileLen*1000/millisec))+' б/сек';
    // закінчення роботи з файлами
    closefile(f);closefile(f1);
    ShowMessage('Збережено у файлі '+name1+'.cip');
end;
begin
   OpenDialog1.Title:=
      'Виберіть файл для шифрування/розшифровування';
   OpenDialog1.Execute;
   if OpenDialog1.FileName='' then exit;
   name1:=OpenDialog1.FileName;
   Form1.Hide; Form2.ShowModal;
   if Form2.ModalResult=6 then
   begin
     ShowMessage('Ваш пароль '''+psw1+''''');
     Form2.Close;
   end;
   Form3.Show; DES_algorithm; Form3.Close;Form1.Show
end;
```

Контрольні запитання.

- *Як довести зворотність алгоритму DES?*
- *Які етапи алгоритму DES не впливають на надійність шифрування? Чи можна їх пропустити?*

- *Які методи криптоаналізу можна застосувати (теоретично) для зламування текстів, зашифрованих алгоритмом DES?*
- *Чи змінює алгоритм DES частотні характеристики тексту?*
- *Запропонуйте свій алгоритм шифрування останньої неповної порції тексту.*
- *В комбінації з якими іншими відомими Вам методами можна використовувати розглянутий в роботі алгоритм шифрування?*
- *Які недоліки в реалізації можуть ослабити надійність розглянутого алгоритму шифрування?*
- *Який тип файлів використовується в алгоритмі? Чому запропонований алгоритм шифрує файли будь-якого типу?*
- *Як можна оптимізувати реалізований в роботі алгоритм для прискорення шифрування файлів?*

Додаток 1.

Таблиці перестановок та S-блоків алгоритму DES (у вигляді модуля)

```
unit Tables;
interface
const transpos_number:array[1..64] of byte=
(58,50,42,34,26,18,10, 2,60,52,44,36,28,20,12, 4,
 62,54,46,38,30,22,14, 6,64,56,48,40,32,24,16, 8,
 57,49,41,33,25,17, 9, 1,59,51,43,35,27,19,11, 3,
 61,53,45,37,29,21,13, 5,63,55,47,39,31,23,15, 7);
key_transpos   : array[1..56] of byte=
(57,49,41,33,25,17, 9, 1,58,50,42,34,26,18,
 10, 2,59,51,43,35,27,19,11, 3,60,52,44,36,
 63,55,47,39,31,23,15, 7,62,54,46,38,30,22,
 14, 6,61,53,45,37,29,21,13, 5,28,20,12, 4);
permut:array[1..48] of byte=
```

```
(32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9,
  8, 9,10,11,12,13,12,13,14,15,16,17,
 16,17,18,19,20,21,20,21,22,23,24,25,
 24,25,26,27,28,29,28,29,30,31,32, 1);
s : array[1..8,0..3,0..15] of byte=
{S1}(((14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7),
 (0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8),
 (4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0),
 (15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13)),
{S2} ((15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10),
(3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5),
(0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15),
(13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9)),
{S3} ((10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8),
(13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1),
(13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7),
(1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12)),
{S4} ((7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15),
(13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9),
(10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4),
(3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14)),
{S5} ((2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9),
(14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6),
(4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14),
(11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3)),
{S6} ((12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11),
(10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8),
```

```
(9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6),
(4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13)),
{S7} ((4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1),
(13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6),
(1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2),
(6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12)),
{S8} ((13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7),
(1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2),
(7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8),
(2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11)));
p:array[1..32] of byte=
(16,7,20,21,29,12,28,17, 1,15,23,26, 5,18,31,10,
  2,8,24,14,32,27, 3, 9,19,13,30, 6,22,11, 4,25);
p56key48:array[1..48] of byte=
(14,17,11,24, 1, 5, 3,28,15, 6,21,10,
 23,19,11, 4,26, 8,16, 7,27,20,13, 2,
 41,52,31,37,47,55,30,40,51,45,33,48,
 44,49,39,56,34,53,46,42,50,36,29,32);
implementation
begin  end.
```

Додаток 2.

Програма для перевірки та відлагодження модуля DES.

```
program DES_prog;
uses crt,DES;
type  str8=string[8];
var  string8,string_key,end_string8:str8;
  start_block64, key64, end_block64,
```

```
dec_block64:block64bits;   i:byte;
procedure Start(s:str8 ;  var bl:block64bits);
var i,j:byte; c:char; b,filter:byte;
begin
 for i:=1 to 8 do
 begin
   c:=s[i];  b:=ord(c);  filter:=128;
   for j:=1 to 8 do
   begin
     if b AND filter =0 then bl[(i-1)*8+j]:=0
                        else bl[(i-1)*8+j]:=1;
     filter:= filter SHR 1
   end
 end
end;
procedure  Finish(be:block64bits;var s8:str8);
var i,j,k:byte;
begin  s8:='12345678';
  for i:=1 to 8 do
  begin
    k:=0;
    for j:=1 to 8 do k:=2*k+be[(i-1)*8+j];
    s8[i]:=chr(k);
  end;
end;
procedure writeBlock64(bl:block64bits);
var i,j:byte;
begin
 for i:=1 to 8 do
```

```
 begin
    for j:=1 to 8 do Write(bl[(i-1)*8+j]:1); write(' ')
  end;
  writeln
end;
begin {main}
   clrscr; randomize;
   string8:='abcdefgh'; string_key:='12345678';
   for i:=1 to 8 do  string8[i]:=chr(random(256));
   for i:=1 to 8 do  string_key[i]:=chr(random(256));
   { Initialisation of binary arrays}
   writeln(' string block = ',string8);
   Start(string8,start_block64);
   for i:=1 to 8 do write(ord(string8[i]):8);writeln;
   writeBlock64(start_block64);
   writeln(' string-key  = ',string_key);
   Start(string_key,key64);
   for i:=1 to 8 do  write(ord(string_key[i]):8);
   writeln;
   writeBlock64(key64);
   DES_alg(direct,start_block64,key64,end_block64);
   writeBlock64(end_block64);
   Finish(end_block64,end_string8);
   writeln(end_string8);
   DES_alg(back,end_block64,key64,dec_block64);
   writeBlock64(dec_block64);
   Finish(dec_block64,end_string8);
   writeln(end_string8);
end.
```

Додаток 3.
Програма для перевірки та відлагодження модуля DES8char.

```
program TEST_DES;
uses crt,DES,DES8char;
var
  string8,string_key,end_string8:str8;  i:integer;
begin
  clrscr; randomize;
  string8:='abcdefgh';string_key:='12345678';
  for i:=1 to 8 do  string8[i]:=chr(random(256))  ;
  for i:=1 to 8 do  string_key[i]:=chr(random(256));
  writeln(' string block = ',string8);
  writeln(' string-key  = ',string_key);
  DES_string8(direct,string8,string_key,end_string8);
  writeln('encoded block = ',end_string8);
  DES_string8(back,end_string8,string_key,string8);
  writeln('decoded block = ',string8);
end;
```

Додаток 4. Паскаль-програма для шифрування довільних файлів DES-алгоритмом.

```
program DES_FILE;
uses crt,DES,DES8char;
var
  string8,string_key,end_string8:str8;
  i,j,FileLen:integer; f,f1,f2: file of char;
  fname,fname1,fname2:string; c:char; s:string;
  dir_or_back :Direction;
begin
```

```
  clrscr; Writeln('Enter filename for encryption');
  Readln(fname); Assign(f,fname);  Reset(f);
  FileLen:=FileSize(f);  clrscr;
  Writeln('Enter filename for result');
  Readln(fname1); Assign(f1,fname1);  Rewrite(f1);
  Writeln('Enter key (8 characters)');
  Readln(string_key);
  if string_key='' then  string_key:='12345678';
  Writeln('Press D (direct DES) or B (back)');
  repeat
    c:=Readkey;
    if UpCase(c)='D' then dir_or_back:=Direct
    else dir_or_back:=Back;
  until  UpCase(c) in ['D','B'];
  for i:=1 to FileLen div 8 do
  begin
    s:='';
    for j:=1 to 8 do
    begin  Read(f,c); s:=s+c  end;
    string8:=s;
    DES_string8(dir_or_back,string8,string_key,
                                    end_string8);
    for j:=1 to 8 do  write(f1,end_string8[j]);
  end;
  for i:=1 to FileLen mod 8 do
  if not eof(f) then
  begin Read(f,c); Write(f1,c)  end;
  Close(f);Close(f1)
end.
```