

Uniwersytet Zielonogórski	Wykonali:	Grupa:	Nr ćwiczenia: 8	Ocena:
Laboratorium				
Temat ćwiczenia: Zarys algorytmów kryptograficznych.		Prowadzący:	Data wyk. ćw.	Data odd. spr.

1. Program dokonujący szyfrowania i deszyfrowania prostym algorytmem XOR.

```
Program _XOR;
uses crt;
var tekst_zak,tekst,klucz,kl_odk:string;

function suma_mod2(A,B:Char):char;
begin
suma_mod2:=chr(ord(A) XOR ord(B));
end;

procedure Szyfruj(tekst:string; klucz:string; var tekst_zak:string);
var i:byte; pos:integer;
begin
tekst_zak:= "";
pos:=1;
for i:=1 to length(tekst) do
begin
tekst_zak:=tekst_zak+suma_mod2(tekst[i],klucz[pos]);
if pos<length(klucz) then pos:=pos+1
else pos:=1;
end;
end;

procedure Odszyfruj(tekst_zak:string; kl_odk:string; var tekst:string);
var i:byte; pos:integer;
begin
pos:=1;
for i:=1 to length(tekst_zak) do
begin
tekst[i]:=suma_mod2(tekst_zak[i],kl_odk[pos]);
if pos<length(kl_odk) then inc(pos)
else pos:=1;
end;
end;

BEGIN
Clrscr;

write('Podaj tekst do zaszyfrowania: ');
readln(tekst);

write('Podaj klucz: ');
readln(klucz);

Szyfruj(tekst,klucz,tekst_zak);

writeln('Ciag zaszyfrowany:', tekst_zak);
```

```

write('Podaj klucz odszyfrujacy: ');
readln(kl_odk);

Odszyfruj(tekst_zak,kl_odk,tekst);

writeln('Po odszyfrowaniu:', tekst);

readln
end.

```

2. Program dokonujący szyfrowania i deszyfrowania prostym algorytmem RSA.

- kod źródłowy

```

program rsa;
uses crt;
type Tab_dwoj=array[1..30] of longint;
     Tab_tekst=array[1..30] of longint;
var b,p,q,n,w,kl_deszyf,kl_szyf:longint; Tekst:string; M:Tab_tekst; dlugosc,j:byte;

procedure zamien(l:longint;var T:Tab_dwoj;var i:longint); {zamiana liczby na kod dwójkowy}
var x,y:longint;
begin
i:=0;
  repeat
    i:=i+1;
    x:=l mod 2;
    l:=l div 2;
    T[i]:=x;
  until l=0;
end;

procedure red_mod(x,a,z:longint;var w:longint); {redukcja modularna}
var h:Tab_dwoj; i:longint;
begin
zamien(a,h,i);
w:=1;
  repeat
    w:=sqr(w) mod z;
    if h[i]=1 then w:=w*x mod z;
    i:=i-1;
  until i=0;
end;

procedure odwrot(a,z:longint;var kl_deszyf:longint); {wyznaczenie klucza deszyfrującego}
var q2,ni,r,r2,s,s2,q1,s1,r1:longint;
begin
q2:=1;r2:=0;s2:=a;
q1:=0;r1:=1;s1:=z;

repeat

  z:=s2 div s1;
  kl_deszyf:=q2-z*q1;
  r:=r2-z*q1;

```

```

s:=s2 mod s1;
q2:=q1;q1:=kl_deszyf;
r2:=r1;r1:=r;
s2:=s1;s1:=s;
until s=1;
end;

begin
Clrscr;
write('Podaj p (liczba pierwsza): ');
readln(p);
write('Podaj q (liczba pierwsza): ');
readln(q);
write('Podaj klucz szyfrujący (liczba pierwsza): ');
readln(kl_szyf);

n:=p*q;
b:=(p-1)*(q-1);

odwrot(kl_szyf,b,kl_deszyf);

writeln('d = ',kl_deszyf,' n = ',n);

write('Podaj tekst do zaszyfrowania: ');
readln(Tekst);

dlugosc:=Length(Tekst);

write('ZASZYFROWANY TEKST: ');

for j:=1 to dlugosc do begin
M[j]:=Ord(Tekst[j]);           {wpis do tablicy kodu ascii kazdego znaku}
red_mod(M[j],kl_szyf,n,w);     {kodowanie kazdego znaku}
M[j]:=w;                       {zapis zakodowanego tekstu, ale w kodzie ASCII}
write(Chr(M[j])); end;        {zamiana zaszyfrowanego tekstu z kodu ASCII na znak i jego wypis na ekran}

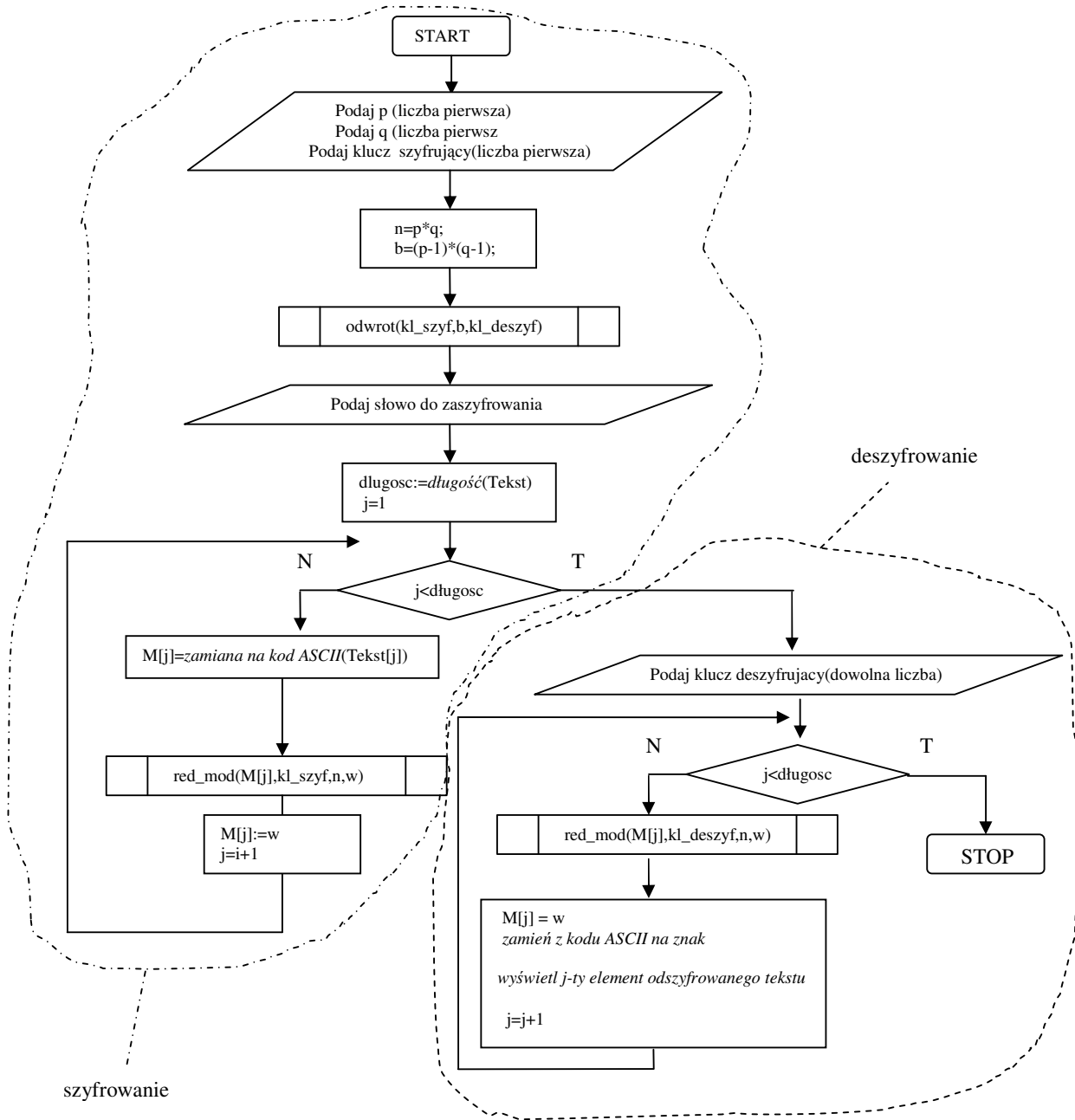
write('Podaj klucz deszyfrujący (dowolna liczba): ');
readln(kl_deszyf);
write('ROZSZYFROWANY TEKST: ');

for j:=1 to dlugosc do begin   {rozkodowanie}
red_mod(M[j],kl_deszyf,n,w);
M[j]:=w;                       {zapis rozkodowanego tekstu, ale w kodzie ASCII}
write(Chr(M[j])); end;        {zamiana z kodu ASCII na znak}

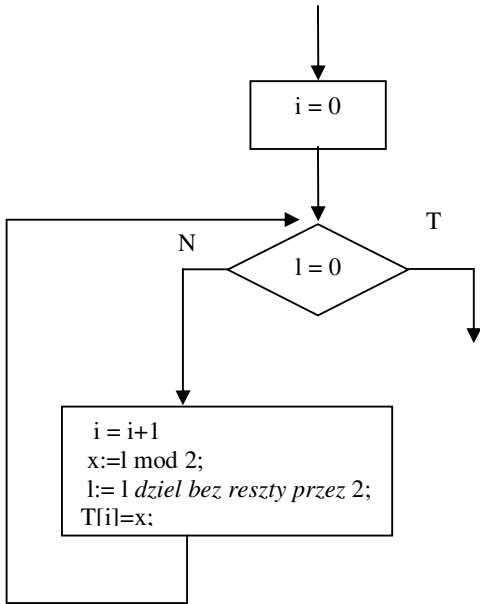
readln;
end.

```

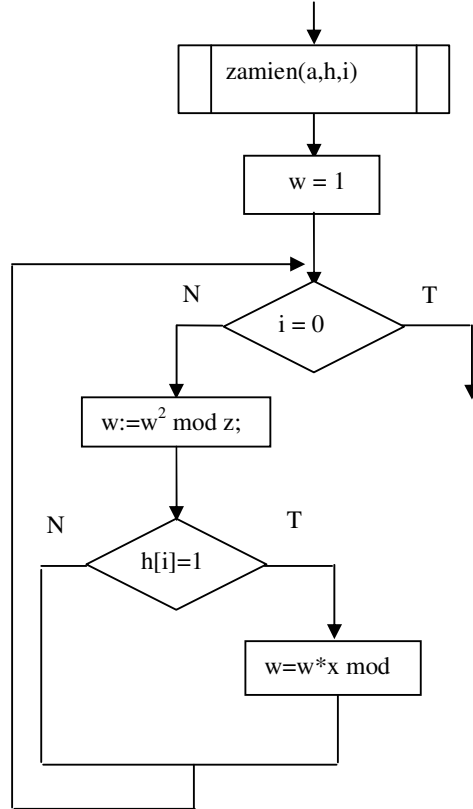
- schemat blokowy programu głównego



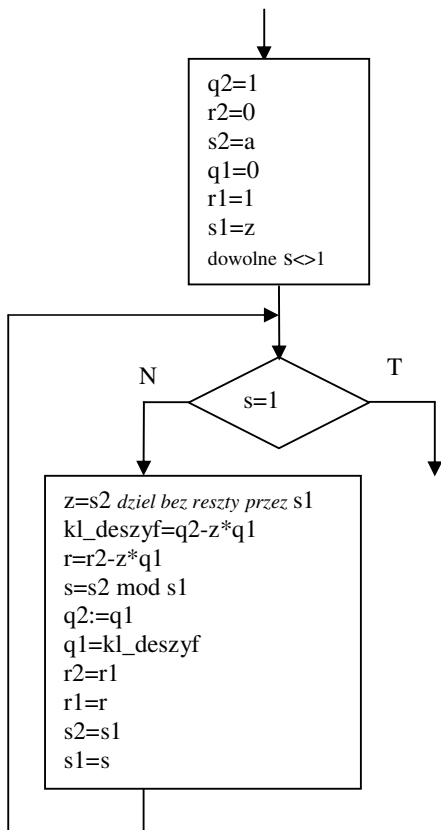
Procedura zamien (l:longint;var T:Tab_dwoj;var i:longint)



procedure red_mod(x,a,z:longint;var w:longint);



procedure odwrot(a,z:longint;var kl_deszyf:longint)



Wnioski:

W ćwiczeniu tym zajęliśmy się podstawowymi metodami szyfrowania. W celu zaszyfrowania informacji posłużyliśmy się prostym algorytmem *XOR* i *RSA*.

Zasada działania algorytmu *XOR* jest bardzo prosta. Szyfrogram (zaszyfrowany tekst) jest wynikiem binarnego sumowania *modulo 2* tekstu z kluczem. Ten sam klucz wykorzystywany jest zarówno do szyfrowania jak i deszyfrowania, ponieważ dwukrotnie sumowanie *modulo 2* tej samej wartości daje w wyniku tę wartość. Stosowanie tego algorytmu nie daje żadnego bezpieczeństwa. Nam jednak nie udało się złamać tego szyfru.

Nasz program szyfrujący i deszyfrujący algorytmem *XOR* składa się z następujących podprogramów:

- funkcji *suma_mod2(A,B:Char):char*, która binarnie sumuje *modulo 2* dwa składniki;
- procedury *Szyfruj(tekst:string; klucz:string; var tekst_zak:string)*, która szyfruje tekst przy pomocy klucza; klucz jak i tekst podaje użytkownik; w tej procedurze wykorzystywana jest funkcja *suma_mod2*; za pomocą tej funkcji dodajemy *modulo 2* *i*-ty składnik tekstu z *j*-tym składnikiem klucza; jeśli długość klucza była krótsza od długości tekstu, to po zsumowaniu kolejnego elementu tekstu z ostatnim elementem klucza, następnym krokiem była suma kolejnego elementu tekstu z pierwszym elementem klucza; przed wykonaniem operacji sumowania poszczególne elementy tekstu i klucza zamieniane były na kod ASCII
- procedury *Odszyfruj(tekst_zak:string; kl_odk:string; var tekst:string)*, która działała analogicznie jak procedura *Szyfruj*; różnica polegała na tym, że sumowane był zaszyfrowany tekst z kluczem;

W programie głównym najpierw przeprowadzana jest operacja szyfrowania, a po jej zakończeniu – operacja deszyfrowania. Oczywiście w rzeczywistości powinny być to dwie odrębne operacje.

Algorytm *RSA* jest jednym z najbezpieczniejszych i najpopularniejszych szyfrów. Jego działanie oparte jest na tzw. arytmetyce modularnej.

Nasz program szyfrujący i deszyfrujący algorytmem *RSA* składa się z następujących podprogramów:

- procedury *zamien(l:longint;var T:Tab_dwoj;var i:longint)*, która zamienia liczbę na odpowiadający jej kod dwójkowy
- procedury *red_mod(x,a,z:longint;var w:longint)*, która przeprowadzała redukcję modularną; redukcja modularna pozwala unikać dużych wyników pośrednich przy operacji potęgowania modularnego, składającego się z sekwencji mnożeń i dzieleni;
- procedury *odwrot(a,z:longint;var kl_deszyf:longint)*, która obliczała odwrotność multiplikatywną $x = a^{-1} \text{ mod } n$;

W programie głównym najpierw przeprowadzana jest operacja szyfrowania, a po jej zakończeniu – operacja deszyfrowania. Oczywiście w rzeczywistości powinny być to dwie odrębne operacje. Publikuje się tylko wartość klucza szyfrującego i wartość *n*, a klucz deszyfrujący jest tajny – zna go tylko osoba, która uprawniona jest do znajomości zaszyfrowanych danych.