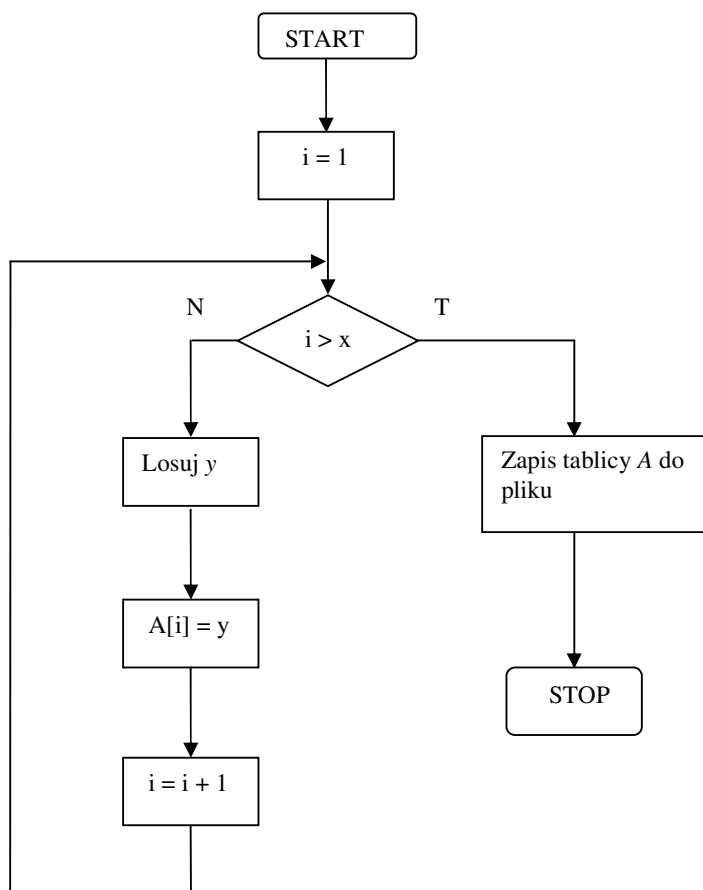


<b>Uniwersytet Zielonogórski</b>	Wykonali:	Grupa:	Nr ćwiczenia: 3	Ocena:
<b>Laboratorium</b>				
Temat ćwiczenia: Wybrane metody sortowania wewnętrznych.		Prowadzący:	Data wyk. ćw.	Data odd. spr.

1. Program losujący  $x$  elementową tablicę liczb typu integer i zapisujący wylosowaną tablicę  $A$  do pliku.

- schemat blokowy

kod źródłowy w języku Turbo Pascal



```
program wpis;  
uses crt;  
const x=30000;  
type Tab = array[1..x] of integer;  
var A:Tab; i:word; f:file of Tab; S:string;
```

BEGIN

```
Randomize;  
S:='d:\zad1';
```

```
Assign(f,S);  
Rewrite(f);
```

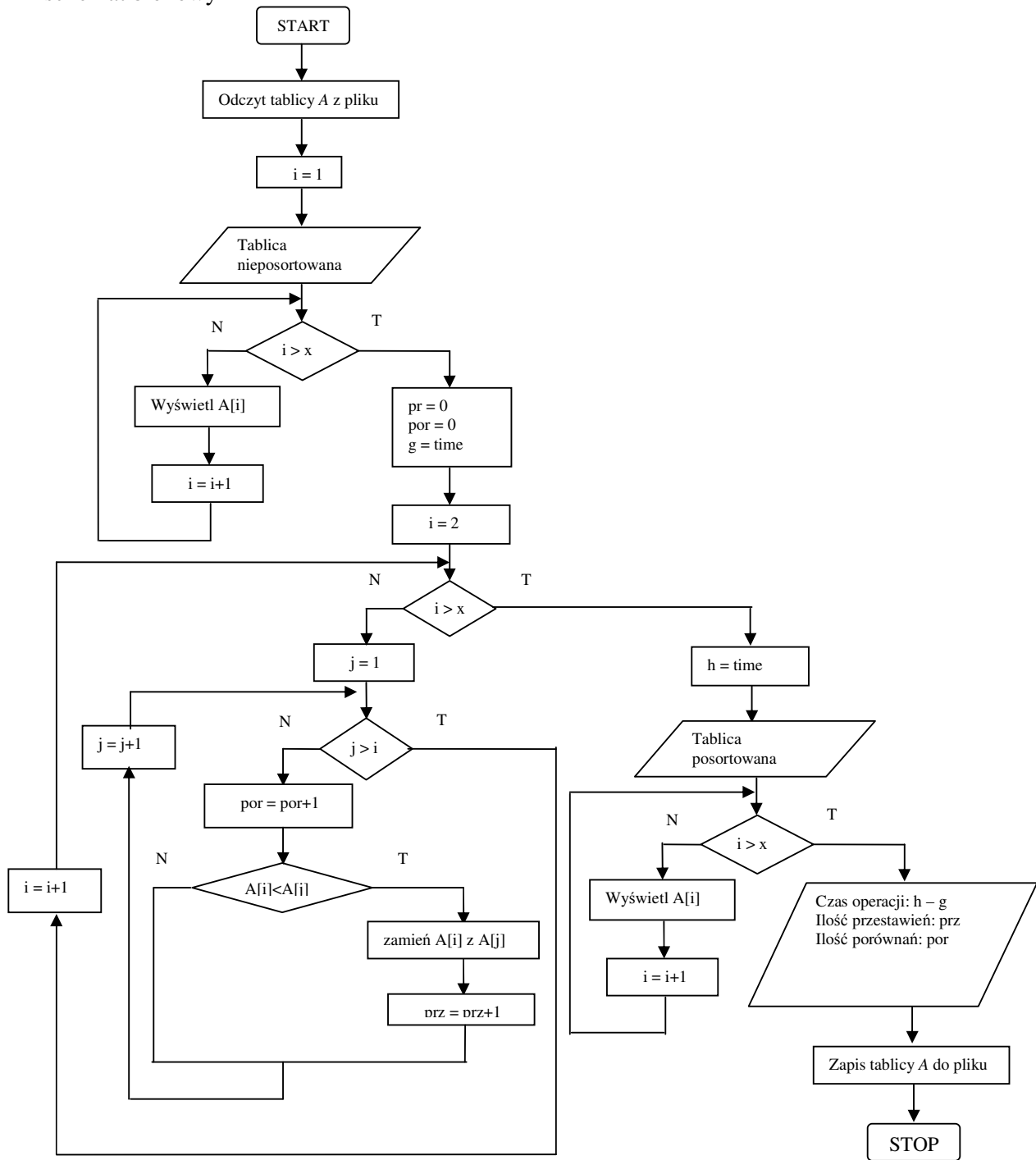
```
for i:=1 to x do  
A[i]:=Random(100)+1;
```

```
write(f,A);  
Close(f);
```

END.

## 2. Sortowanie przez proste wstawianie.

- schemat blokowy



- kod źródłowy w języku Turbo Pascal

```

program pros_wstaw;
uses crt,dos;
const x=30000;
type Tab = array[1..x] of integer;
var A:Tab; i,j,y,min,buf,g,m,s1,ss1,s2,ss2:word; prz,por:Longint; S,Sa:String; f,f1:file of Tab;

BEGIN
Clrscr;
S:='d:\zad1';
Assign(f,S);
Reset(f);
Read(f,A);
Close(f);

writeln('TABLICA nieposortowana: ');
    for i:=1 to x do write(A[i], ' ');

prz:=0;
por:=0;
GetTime(g,m,s1,ss1);

for i:=2 to x do
    for j:=1 to i do begin
        por:=por+1;
        If A[i]<A[j] then begin
            buf:=A[i];
            A[i]:=A[j];
            A[j]:=buf;
            prz:=prz+1;    end;
        end;

GetTime(g,m,s2,ss2);
writeln('TABLICA posortowana: ');
    for i:=1 to x do write(A[i], ' ');

writeln('Czas rozpoczęcia operacji: ',s1,' : ',ss1);
writeln('Czas zakończenia operacji: ',s2,' : ',ss2);
writeln('Ilość przestawień: ',prz);
writeln('Ilość porównań: ',por);

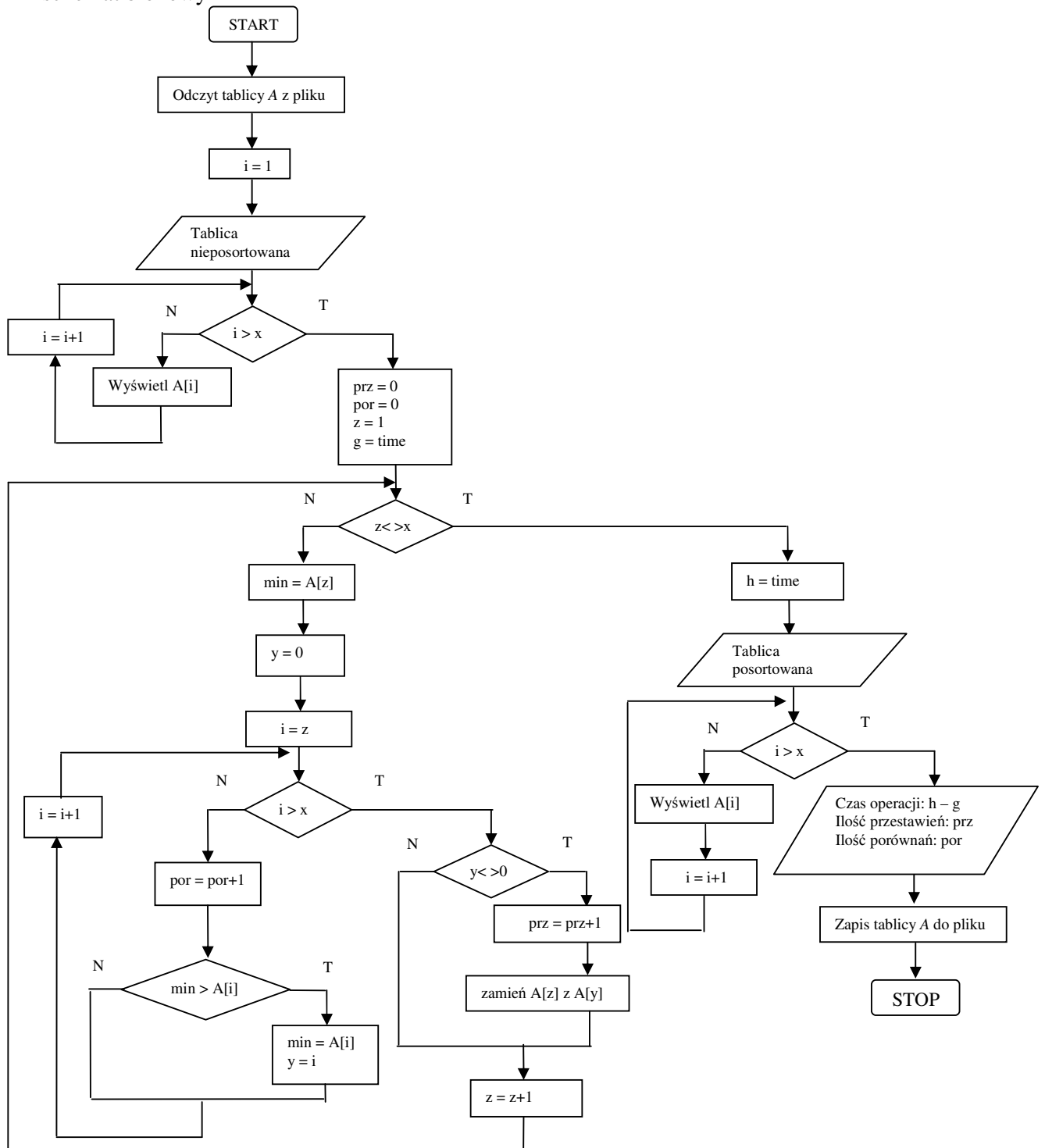
Sa:='d:\zad_1';
Assign(f1,Sa);
Rewrite(f1);
write(f1,A);
Close(f1);

readln
END.

```

### 3. Sortowanie przez proste wybieranie.

- schemat blokowy



- kod źródłowy w języku Turbo Pascal

```

program pros_wyb;
uses crt,dos;
const x=30000;
type Tab = array[1..x] of integer;
var A:Tab; i,z,y,min,buf,g,m,s1,ss1,s2,ss2:word; prz,por:Longint; S,Sa:String; f,f1:file of Tab;

BEGIN
Clrscr;
S:='d:\zad1';
Assign(f,S); Reset(f);
Read(f,A); Close(f);

writeln('TABLICA nieposortowana: ');
  for i:=1 to x do write(A[i],', ');

z:=1; prz:=0; por:=0;
GetTime(g,m,s1,ss1);

while z<>x do begin
min:=A[z];
y:=0;

for i:=z to x do begin
por:=por+1;
If min>A[i] then begin min:=A[i]; y:=i; end; end;

If y<>0 then begin
buf:=A[z];
A[z]:=A[y];
A[y]:=buf;
prz:=prz+1; end;
z:=z+1; end;
GetTime(g,m,s2,ss2);

writeln('TABLICA posortowana: ');
  for i:=1 to x do write(A[i],', ');

writeln('Czas rozpoczęcia operacji: ',s1,' : ',ss1);
writeln('Czas zakończenia operacji: ',s2,' : ',ss2);
writeln('Ilość przestawień: ',prz);
writeln('Ilość porównań: ',por);

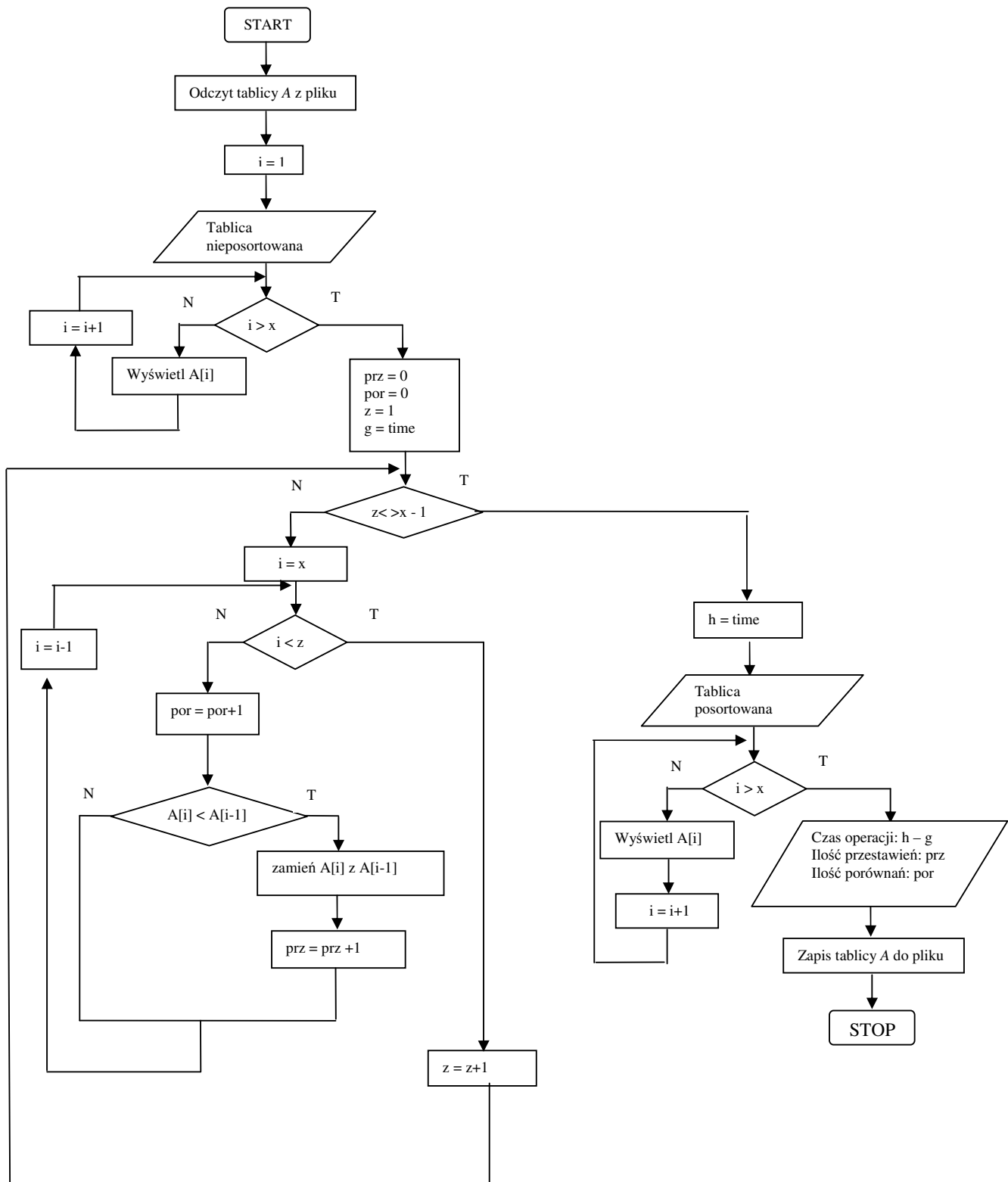
Sa:='d:\zad_2';
Assign(f1,Sa); Rewrite(f1);
write(f1,A); Close(f1);

readln
END.

```

#### 4. Sortowanie bąbelkowe.

- schemat blokowy



- kod źródłowy w języku Turbo Pascal

```

program babel;
uses crt,dos;
const x=30000;
type Tab = array[1..x] of integer;
var A:Tab; i,z,buf,g,m,s1,ss1,s2,ss2:word; prz,porz:Longint; S,Sa:String; f,f1:file of Tab;

BEGIN
Clrscr;

S:='d:\zad1';
Assign(f,S); Reset(f);
Read(f,A); Close(f);

writeln('TABLICA nieposortowana: ');
  for i:=1 to x do write(A[i],', ');

z:=1; prz:=0; por:=0;
GetTime(g,m,s1,ss1);

while z<>x-1 do begin

for i:=x downto z do begin
por:=por+1;
  If A[i]<A[i-1] then begin
    buf:=A[i];
    A[i]:=A[i-1];
    A[i-1]:=buf;
    prz:=prz+1;  end; end;

z:=z+1;      end;

GetTime(g,m,s2,ss2);

writeln('TABLICA posortowana: ');
for i:=1 to x do write(A[i],', ');

writeln('Czas rozpoczęcia operacji: ',s1,' : ',ss1);
writeln('Czas zakończenia operacji: ',s2,' : ',ss2);
writeln('Ilość przestawień: ',prz);
writeln('Ilość porównań: ',por);

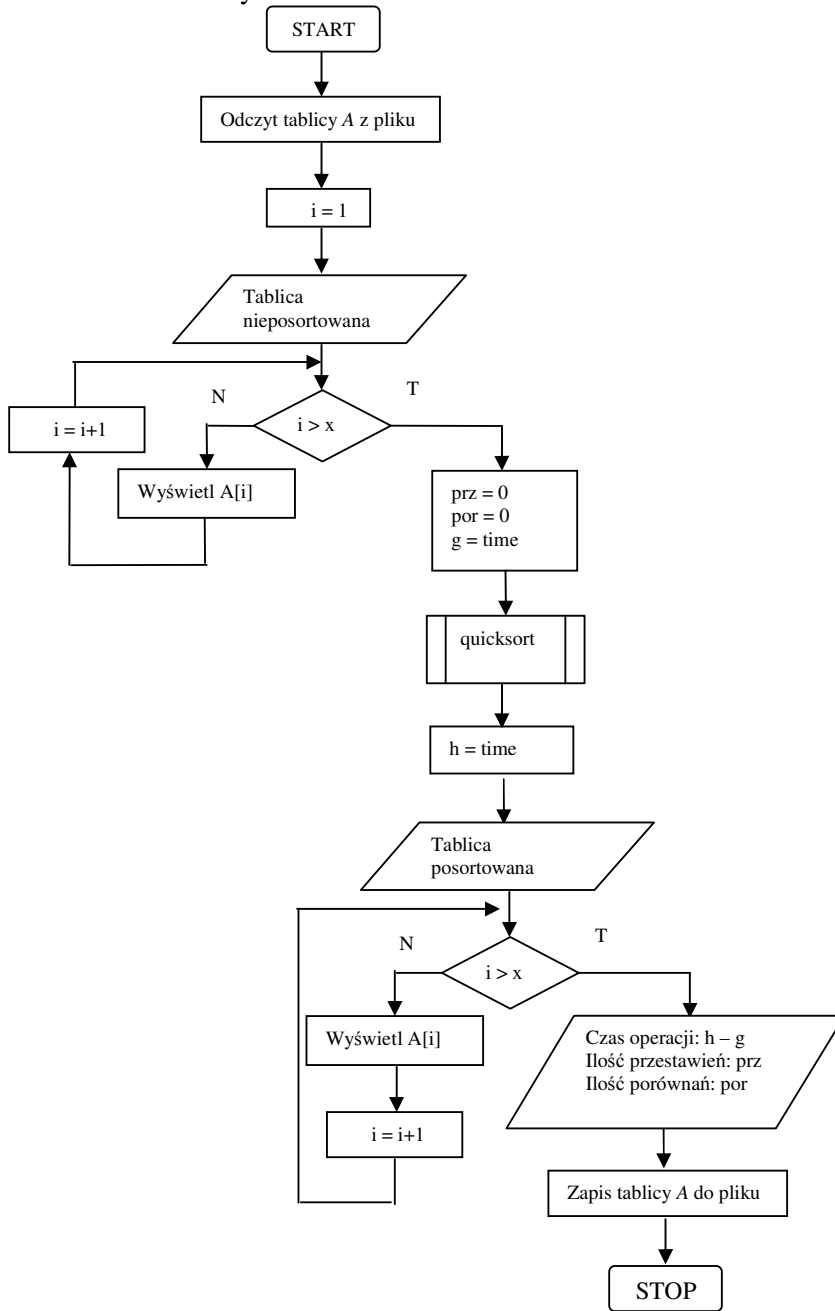
Sa:='d:\zad_3';
Assign(f1,Sa);Rewrite(f1);
write(f1,A);Close(f1);

readln
END.

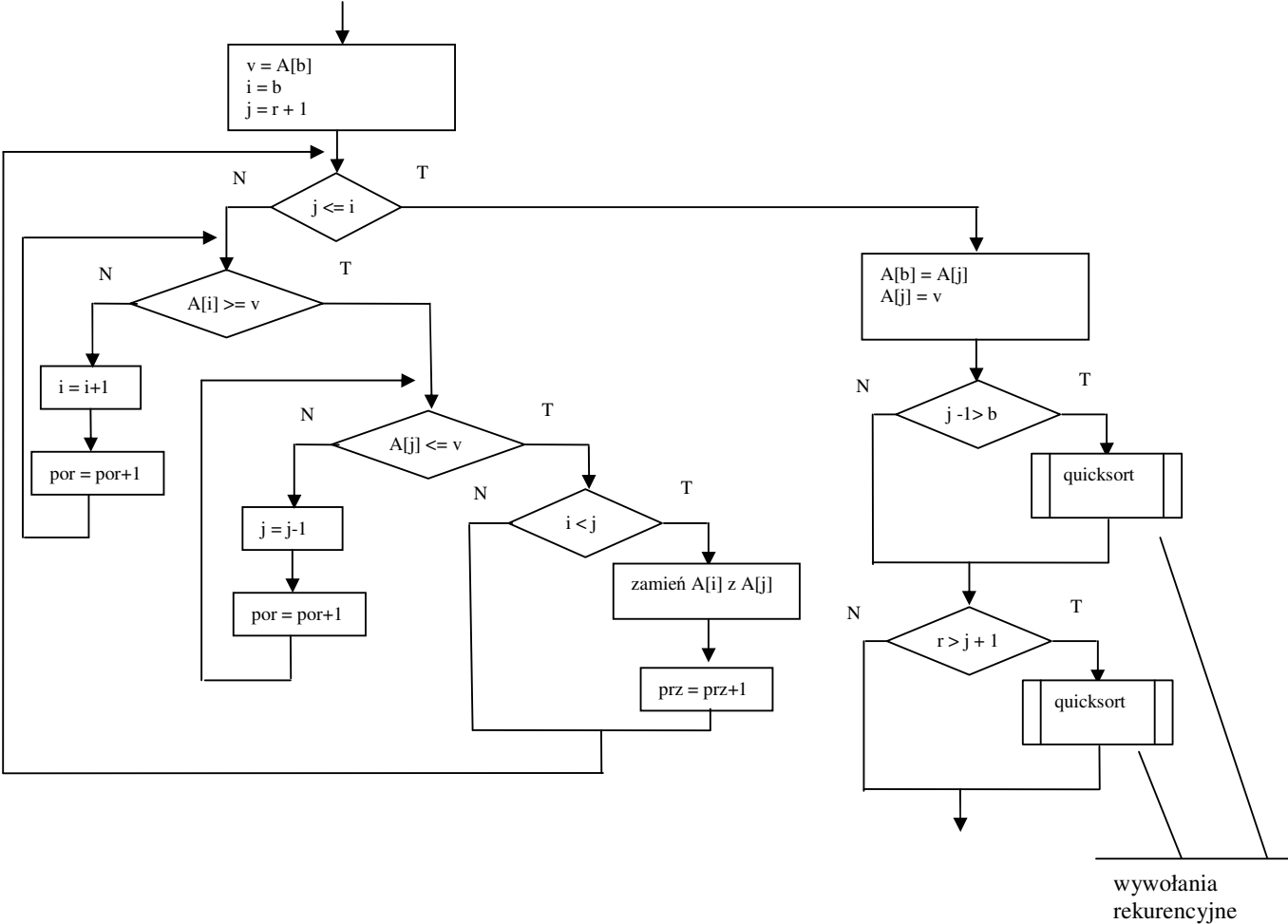
```

## 5. Sortowanie szybkie (Quick Sort).

- schemat blokowy



Schemat blokowy podprogramu quicksort.



- kod źródłowy w języku Turbo Pascal

```

program quick;
uses crt,dos;
const x=30000;
type Tab=array[1..x] of integer;
var A:Tab; l,k,g,m,s1,ss1,s2,ss2:word; prz,por:Longint; S,Sa:String; f,f1 :file of Tab;

procedure quicksort(b,r:word);
var i,v,j,k,buf:word;
begin
v:=A[b]; i:=b; j:=r+1;
repeat
  repeat
    i:=i+1;
  until A[i]>=v;

  repeat
    j:=j-1;
  until A[j]<=v;

  if i<j then begin
    buf:=A[i];
    A[i]:=A[j];
    A[j]:=buf;
    prz:=prz+1; end;
until j<=i;

  A[b]:=A[j]; A[j]:=v;

  if j-1 >b then quicksort(b,j-1);
  if r>j+1 then quicksort(j+1,r);
end;

Begin
Clrscr; Randomize;
S:='d:\zad1';
Assign(f,S); Reset(f);
Read(f,A); Close(f);

writeln('TABLICA przed posortowaniem: ');
  for i:=1 to x do write(A[i], ' ');

por:=0; prz:=0;
GetTime(g,m,s1,ss1);
quicksort(1,x);
GetTime(g,m,s2,ss2);
writeln('TABLICA po posortowaniu: ');
  for i:=1 to x do write(A[i], ' ');
writeln('Czas rozpoczęcia operacji: ',s1,' : ',ss1);
writeln('Czas zakończenia operacji: ',s2,' : ',ss2);
writeln('Ilość przestawień: ',prz);
writeln('Ilość porównań: ',por);

Sa:='d:\zad_4';
Assign(f1,Sa); Rewrite(f1);
write(f1,A); Close(f1);
readln;
End.

```

## 6. Porównanie metod sortowania.

Metoda	Rozmiar tablicy	Ilość przestawień	Ilość porównań	Czas operacji [s]
Proste wstawianie	31 995	43 416 376	511 856 009	5,87
	20 000	25 462 253	200 009 999	2,47
Proste wybieranie	31 995	31 970	511 856 009	5,32
	20 000	19 979	200 009 999	2,3
Bąbelkowe	31 995	256 137 945	511 856 007	11,59
	20 000	100 298 266	200 009 997	4,55
Quick Sort	31 995	107 153	574 506	0
	20 000	61 603	344 570	0

### Wnioski:

Nasze badania przeprowadzaliśmy na tablicach 20 000 i 31 995 elementowych. Jest to maksymalny rozmiar tablicy w Turbo Pascalu, ponieważ tablica nie może przekraczać 64 kB. Tablice były wypełniane liczbami z zakresu od 0 do 3000.

Najlepszą metodą okazała się metoda szybkiego sortowania (Quick Sort). Jej czas działania był niemożliwy do zbadania, ponieważ za mały był zakres tablicy.

Drugą co do szybkości działania, była metoda sortowania przez proste wybieranie. Była ona szybsza w działaniu niż metoda sortowania przez proste wstawianie, ponieważ wybierała ona kolejny najmniejszy element tablicy i od razu umieszczała go w odpowiednim miejscu po lewej stronie tablicy. Potrzebna była nam dodatkowa zmienna *min* (najmniejszy element) i *y* (indeks najmniejszego elementu). Natomiast metoda przez proste wstawianie przestawiała element na lewą część tablicy, która następnie była sortowana.

Najgorszą metodą okazała się metoda sortowania bąbelkowego. Jej tak duży czas operacji spowodowany dużą liczbą przestawień. Jeżeli najmniejszy element umieszczony jest na końcu tablicy, to musimy wykonać  $n-1$  przestawień, żeby umieścić go na początku tablicy.<sup>1</sup>

Zwiększenie zakresu liczb, którymi były wypełniane tablice w trzech pierwszych metodach powodowały zwiększenie czasu operacji i liczby przestawień, natomiast liczba porównań nie ulegała zmianie. Spowodowane to jest tym, że porównane (sprawdzone) musiały być wszystkie elementy, a liczba przestawień zależała od zakresu.

W Quick Sort'cie zwiększenie zakresu liczb, którymi były wypełniane tablice powodowało zmniejszenie liczby przestawień, a zwiększenie liczby porównań. Czas był niemożliwy do zbadania.