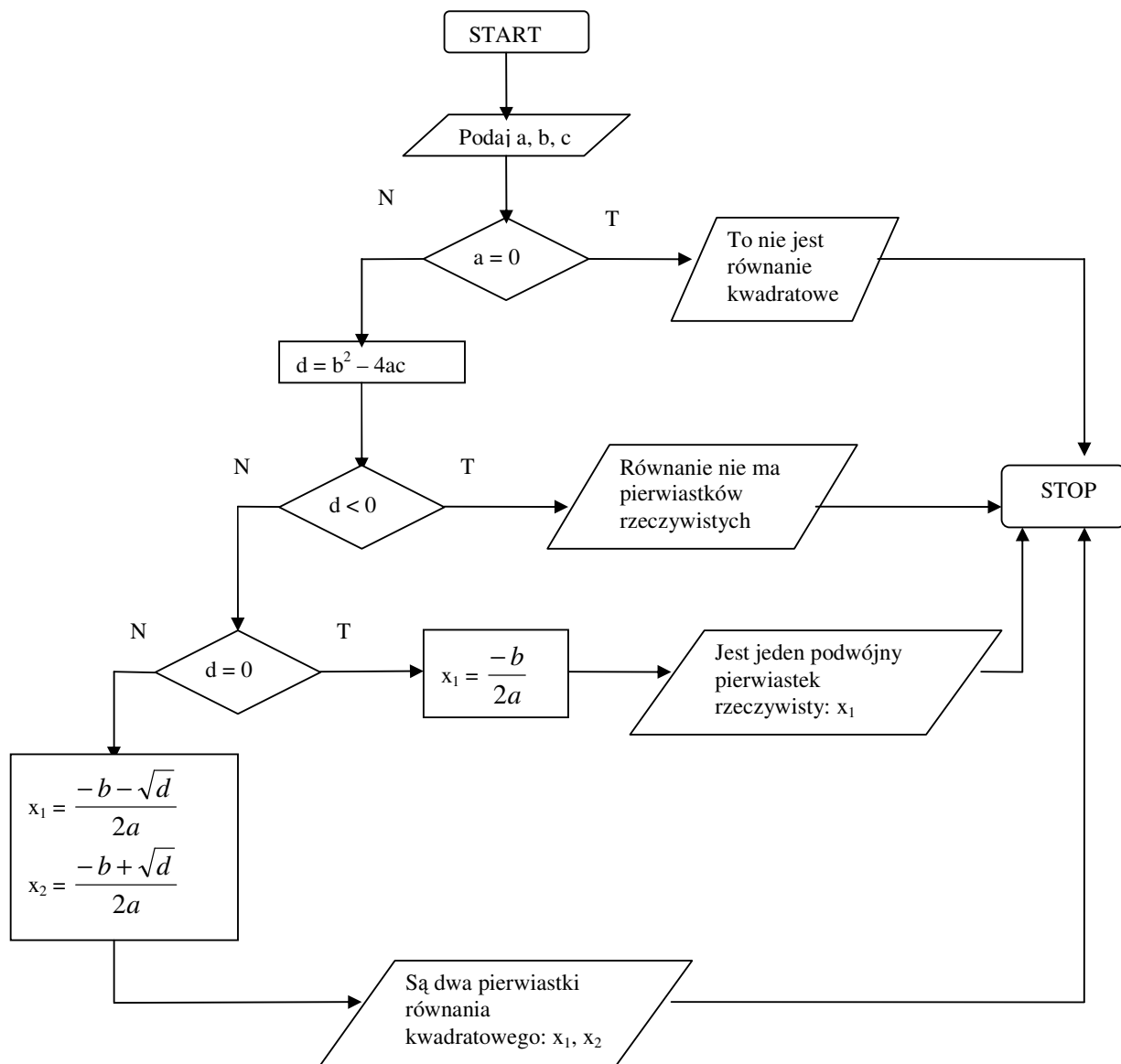


<b>Uniwersytet Zielonogórski</b>	Wykonali:	Grupa:	Nr ćwiczenia: 1	Ocena:
<b>Laboratorium</b>				
Temat ćwiczenia: Schematy blokowe.	Prowadzący:	Data wyk. ćw.	Data odd. spr.	

1. Obliczanie pierwiastków równania kwadratowego w postaci:  $ax^2 + bx + c = 0$

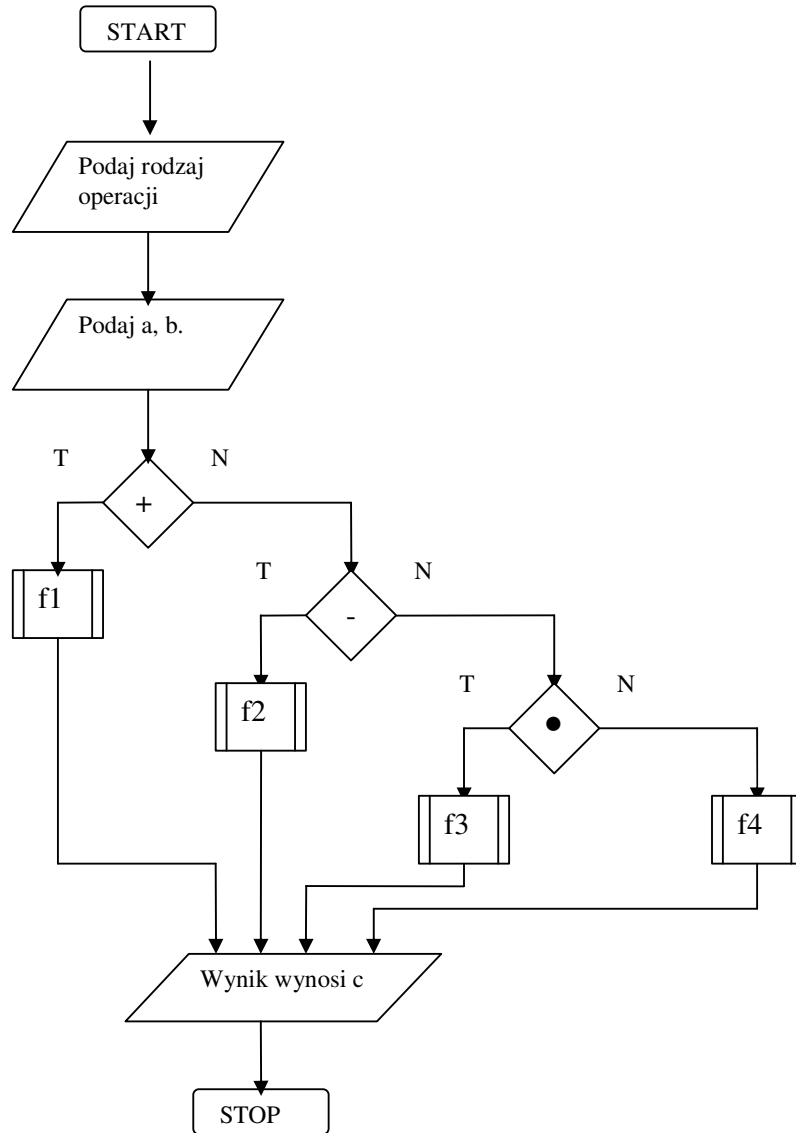
- schemat blokowy



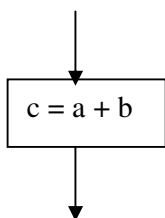


2. Wykonywanie czterech podstawowych działań matematycznych tj. +, -, • oraz /, gdzie pierwszym zapytaniem algorytmu powinno być pytanie o rodzaj operacji, a dopiero kolejnymi pytanie o argumenty tej operacji.

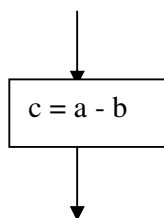
- schemat blokowy



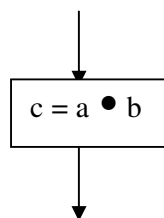
procedura f1



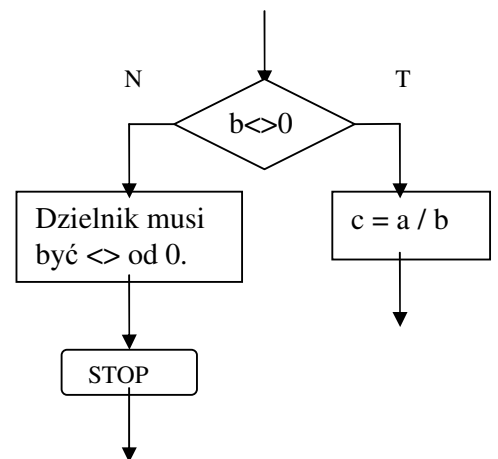
procedura f2



procedura f3



procedura f4



- kod źródłowy w języku Turbo Pascal

```
program kalkulator;
uses crt;
var a,b:integer; c:real; z:char;

procedure f1(a,b:integer;var c:real);
begin
c:=a+b;
end;

procedure f2(a,b:integer;var c:real);
begin
c:=a-b;
end;

procedure f3(a,b:integer;var c:real);
begin
c:=a*b;
end;

procedure f4(a,b:integer;var c:real);
begin
If b=0 then halt(1) else
c:=a/b;
end;

BEGIN
Clrscr;
write('Podaj rodzaj operacji: ');
readln(z);

If z in ['+', '-', '*', '/'] then begin

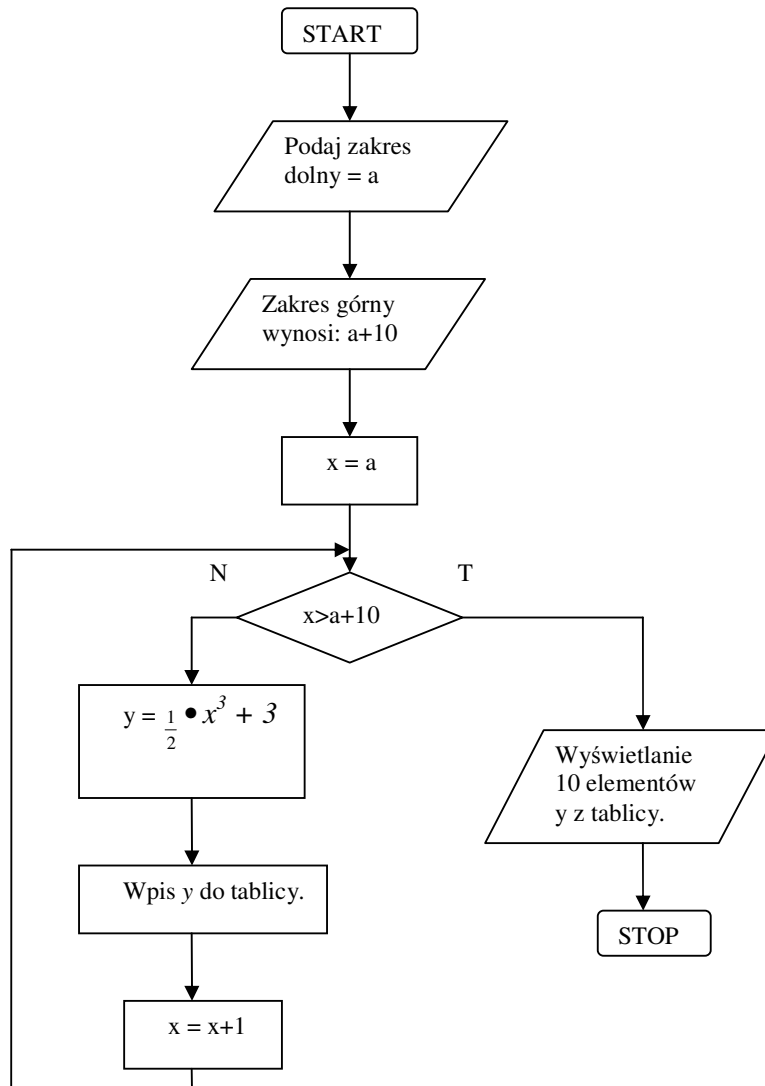
write('Podaj a: '); readln(a);
write('Podaj b: '); readln(b);

Case z of
'+':f1(a,b,c);
'-':f2(a,b,c);
'*':f3(a,b,c);
'/':f4(a,b,c); end;

write('Wynik: ',c:0:2);
readln          end;
END.
```

3. Tablicowanie funkcji wyrażonej wzorem  $f_I(x) = \frac{1}{2} \cdot x^3 + 3$

- schemat blokowy



- kod źródłowy w języku Turbo Pascal

```
program funkcja;
uses crt;
type Tab=array[1..10] of real;
var a:integer; x:integer; y:real; T:Tab; i:byte;

BEGIN
clrscr;

write('Podaj zakres dolny: '); readln(a);

write('Zakres gorny wynosi: ',a+10);

x:=a;
i:=1;

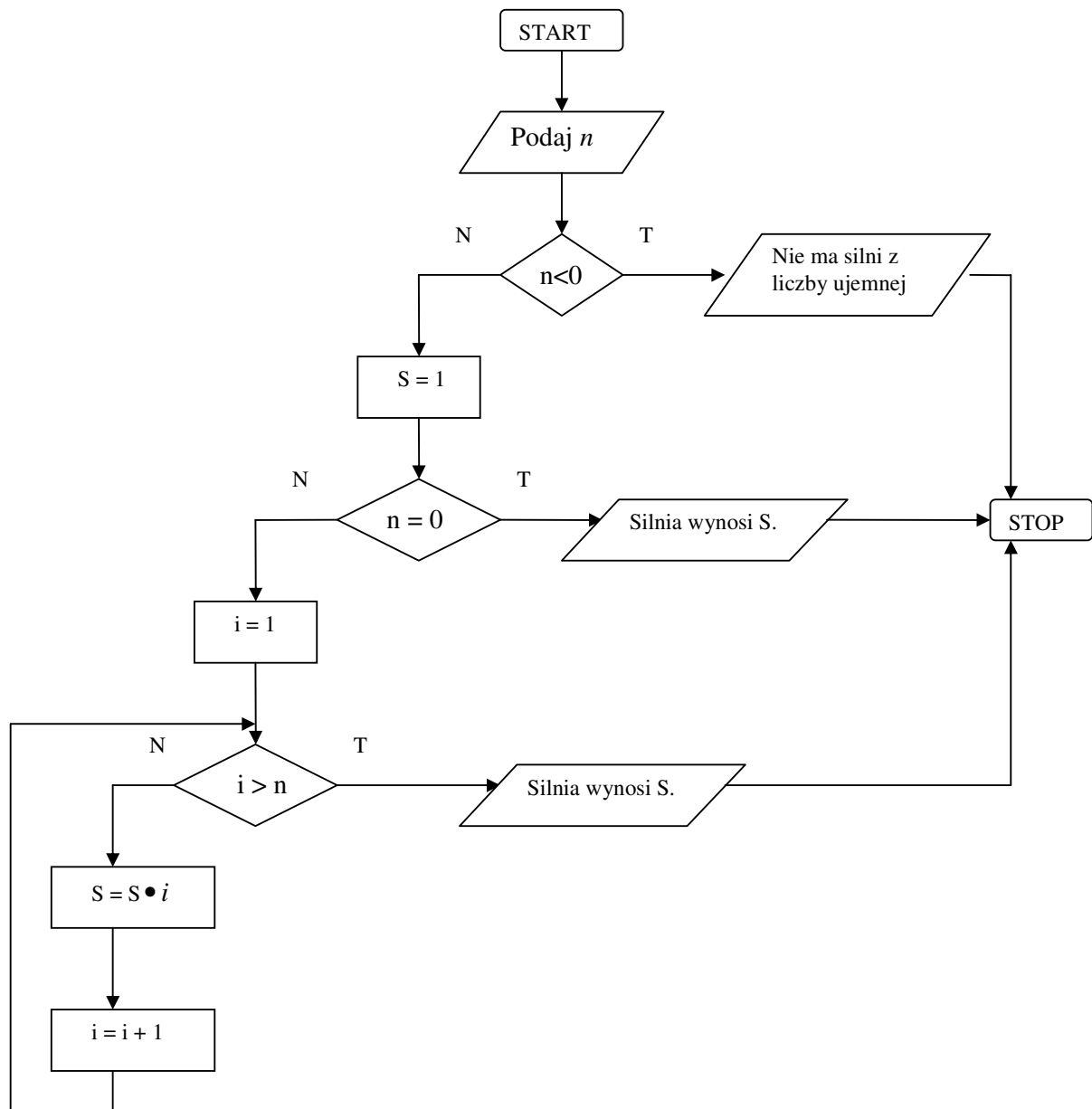
for x:=a to a+10 do begin
  y:=0.5*x*Sqr(x)-3;
  T[i]:=y;
  i:=i+1;          end;

writeln;
writeln;

for i:=1 to 10 do write(T[i]:0:1,' ');
readln
END.
```

#### 4. Iteracyjne obliczanie funkcji silnia.

- schemat blokowy



- kod źródłowy w języku Turbo Pascal

```

program silnia;
uses crt;
var N:integer; S:word; i:byte;

BEGIN
Clrscr;
write('Podaj N: '); readln(N);

If N<0 then write('Nie ma silni z liczby ujemnej') else begin
S:=1;
If N=0 then write('Silnia wynosi: ',S) else begin
i:=1;
  for i:=1 to N do S:=S*i;

  write('Silnia wynosi: ',S);          end;
end;

readln
END.

```

### **Wnioski:**

Schematy blokowe są jednym ze sposobów zapisu algorytmu działania programu. Są one bardzo obrazowe i pokazują działanie algorytmu za pomocą ilustracji. Ilustracje składają się z figur zwanych skrzynkami oraz połączeń między nimi. Skrzynki obrazują rozpoczęcie działania algorytmu, zakończenie działania algorytmu, pobranie danych i ich wyprowadzenie, wykonanie instrukcji, sprawdzenie warunku. Strzałki ilustrują sposób poruszania się po schemacie, czyli kolejność wykonywania działań.

W przykładzie pierwszym umieszczone są trzy warunki. Pierwszy zabezpiecza nas przed możliwością dzielenia przez zero, drugi i trzeci sprawdza, czy wartość delty jest ujemna, dodatnia, czy zero. Do sprawdzenia tych trzech warunków (dotyczących delty) wystarczą tylko dwie operacje warunku. Najpierw sprawdzamy, czy wartość delty jest mniejsza od zera. Jeśli nie jest, to sprawdzamy, czy równa jest zero. Jeśli nie jest, to wiadomo, że jej wartość może być tylko dodatnia. Kod źródłowy jest pełnym odzwierciedleniem schematu blokowego.

W przykładzie drugim korzystaliśmy z bloków podprogramów, które są procedurami dla kodu źródłowego i definiujemy je poza programem głównym. Do sprawdzenia jednego z czterech działań matematycznych wystarczyły tylko trzy operacje warunku. W podprogramie f4 umieściliśmy zabezpieczenie w przypadku dzielenia przez zero. W kodzie źródłowym umieściliśmy dodatkowo sprawdzenie, czy wybrano jedno z czterech możliwych działań.

Tablicowanie funkcji przeprowadzaliśmy na przedziale 10 elementowym. Działanie programu zaczynało się od zapytania o przedział dolny, a przedział górny był o 10 większy od przedziału dolnego. Wszystkie wartości funkcji wpisywane były do tablicy i wypisywane na ekran pod koniec działania programu. Wykorzystaliśmy tu tylko jedną operację warunku, która sprawdzała, czy nie przekroczyliśmy progu górnego.

Przy obliczaniu silni, w przypadku  $n < 0$  kończyliśmy działanie programu komunikatem o niemożliwości obliczenia silni z liczby ujemnej, w przypadku  $n = 0$  komunikatem „*Silnia wynosi 1*”. Gdy  $n > 0$ , to przy pomocy operacji warunku obliczaliśmy wartość funkcji. W kodzie źródłowym odpowiada jej pętla *for*.