Sposoby przedstawiania algorytmów.

1. Instrukcje sterujące w programie, tabulacja

Zadanie 1

Przedstaw w jprostej wersji algorytm wyznaczania pierwiastków równania kwadratowego.

Specyfikacja algorytmu

Dane wejściowe:

- a, b, c – parametry równania (różne od zera liczby rzeczywiste);

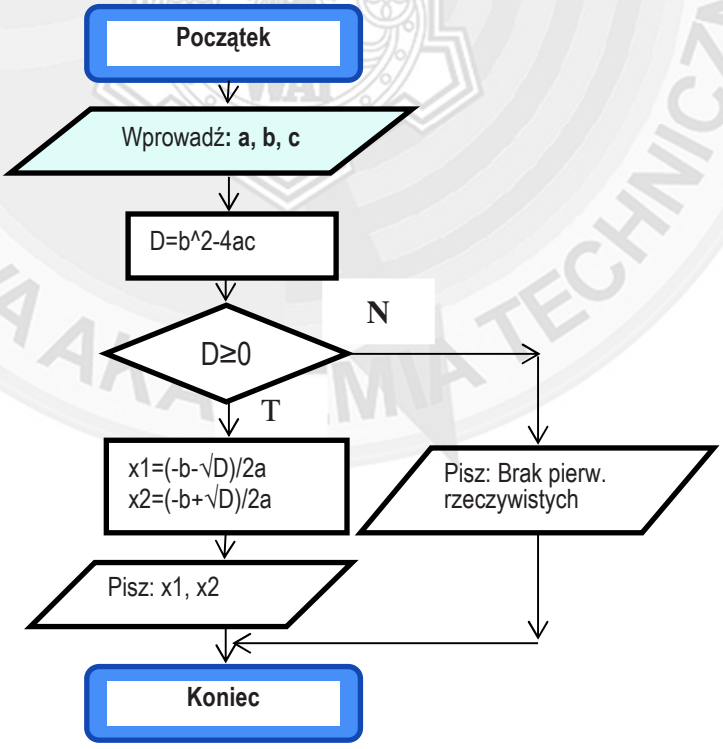
Wyniki:

- x1, x2 – wartości pierwiastków .

Zmienne pomocnicze:

D- zmienna przechowująca obliczoną wartość tzw. delty równania

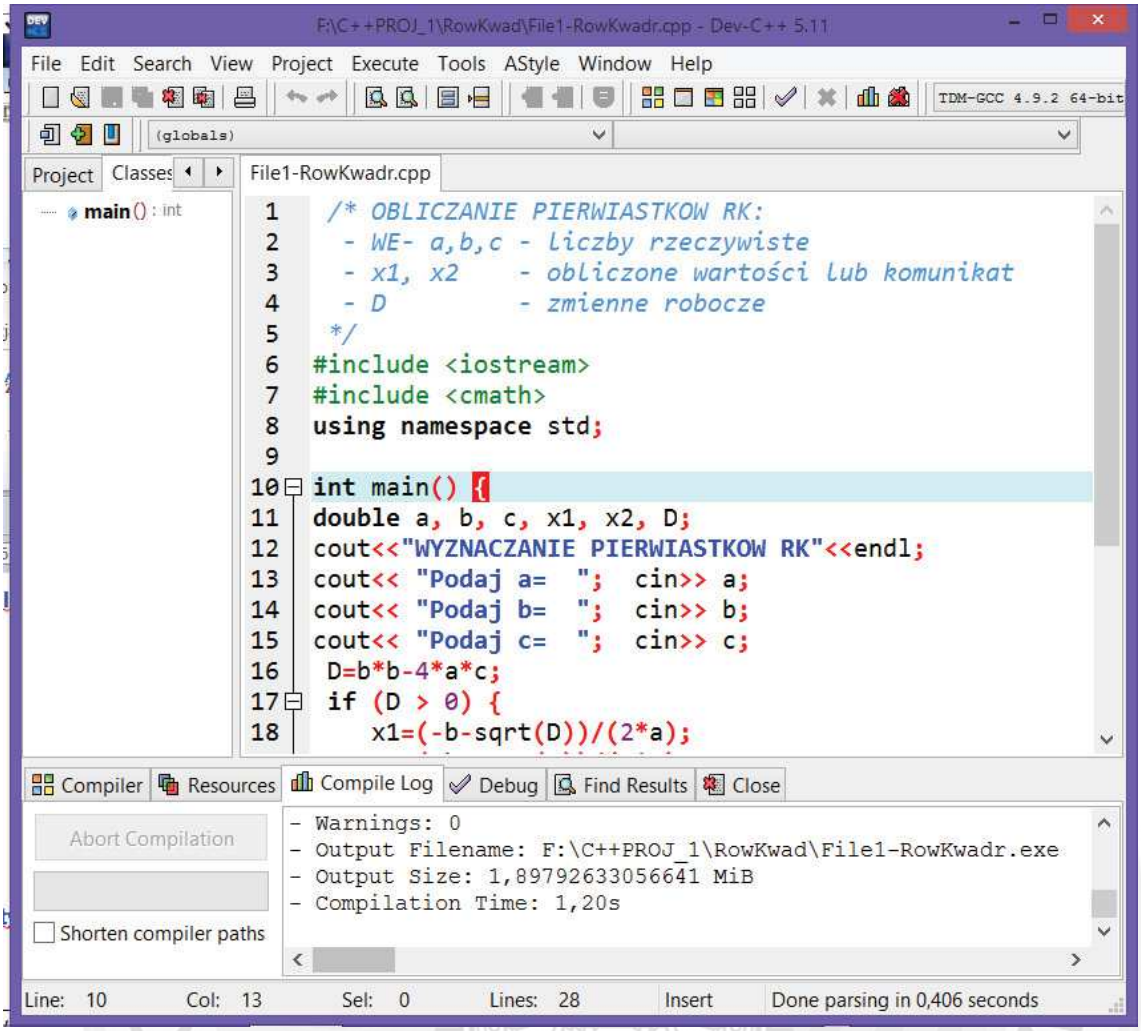
a	b	c	Del	Rozwiązanie
a≠0	dowolne	dowolne	Del>0	x1=(-b-$\sqrt{Del}$)/2a x2=(-b+$\sqrt{Del}$)/2a
			Del=0	x1=x2= -b/2a
			Del<0	Brak pierwiastków rzeczywistych
a=0	To nie jest równanie kwadratowe			



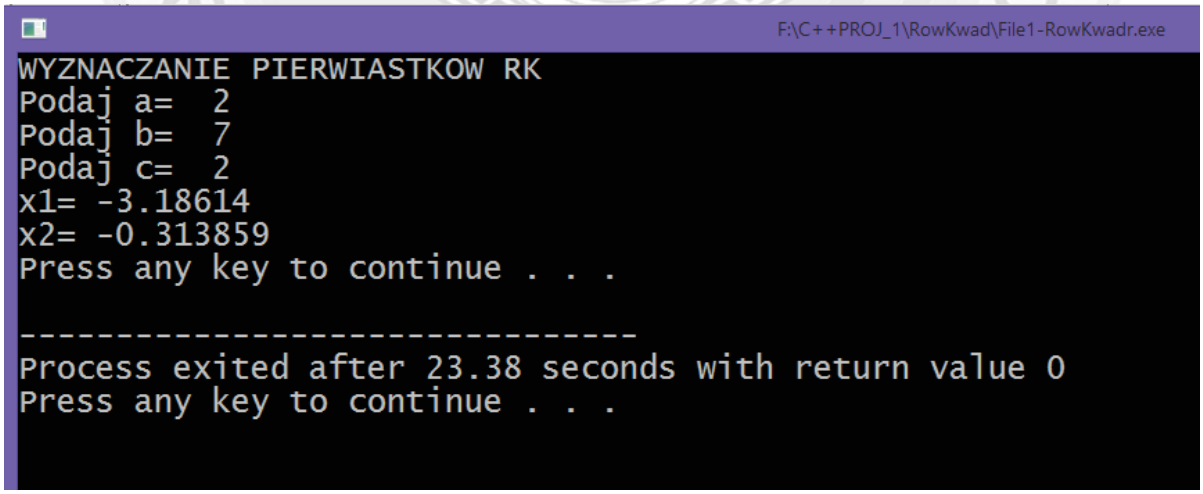
Rys.1. 1. Prosty schemat blokowy algorytmu z rozgałęzieniem

```
1: /* OBLICZANIE PIERWIASTKOW RK:
2: - WE- a,b,c - liczby rzeczywiste
3: - x1, x2 - obliczone wartości lub komunikat
4: - D - zmienne robocze                    */
5: #include <iostream>
6: #include <cmath>
7: using namespace std;
8: int main() {
9: double a, b, c, x1, x2, D;
10: cout<<"WYZNACZANIE PIERWIASTKOW RK"<<endl;
11: cout<< "Podaj a= "; cin>> a;
12: cout<< "Podaj b= "; cin>> b;
13: cout<< "Podaj c= "; cin>> c;
14: D=b*b-4*a*c;
15: if (D > 0) {
16:  x1=(-b-sqrt(D))/(2*a);
17:  x2=(-b+sqrt(D))/(2*a);
28:  cout<< "x1= " <<x1<< endl;
19:  cout<< "x2= " <<x2<< endl;
20: }
21: else
22:  cout<< "Brak pierw. rzeczywistych"<<endl;
23: system("pause");
24: return 0;
25: }
```

Rys. 1.2. Program w C++ rozwiązujący zadanie 1(wyznaczania pierwiastków RK)



Rys. 1.3. Fragment programu w C++ - widok w środowisku programistycznym DEV C++5.11



Rys. 1.4. Widok przykładowych wyników programu z rys. 1.3 na ekranie konsoli środowiska DEV C++5.11

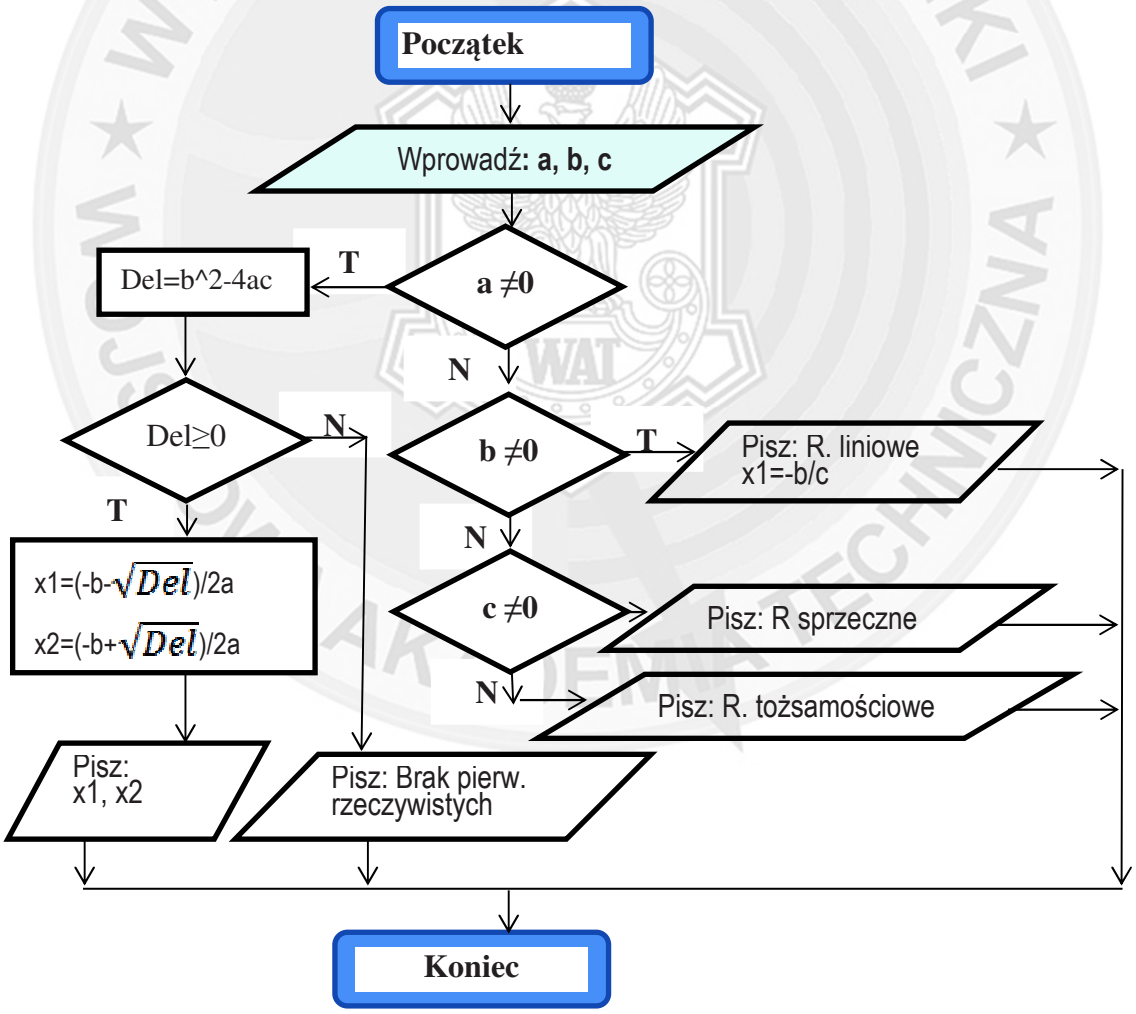
Złożone instrukcje sterujące w algorytmie i programie

Zadanie 2

Przedstaw schemat blokowy algorytmu wyznaczania pierwiastków równania kwadratowego. Uwzględnij identyfikację postaci równania (badanie parametru a). Przedstaw program w C++

Specyfikacja:
Dane wejściowe: a, b, c – zadawane współczynniki równania
Wynik: x1, x2 - wyznaczone wartości pierwiastków lub komunikat

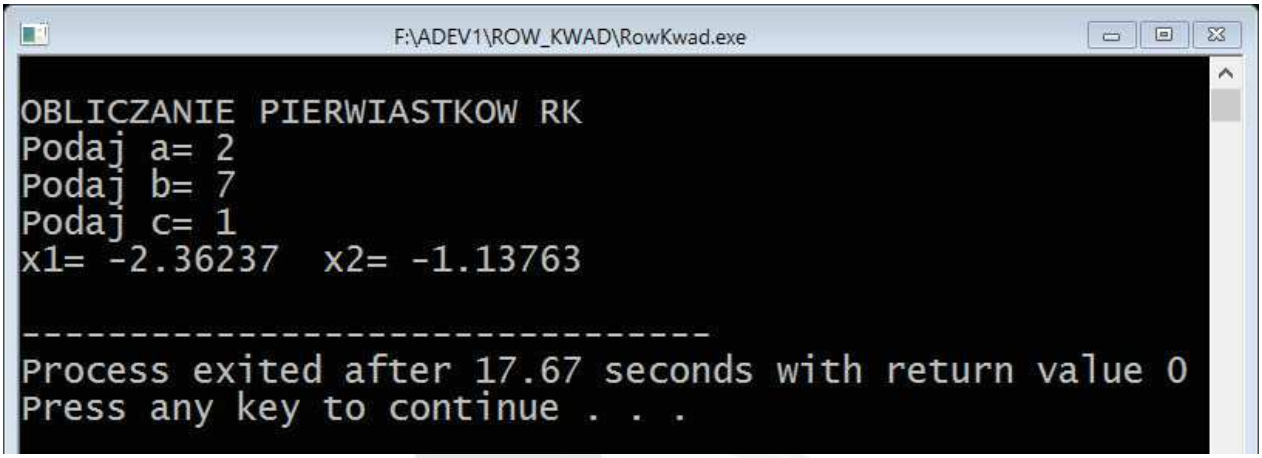
a	b	c	Del	Rozwiązanie
a≠0	dowolne	dowolne	Del>0	x1=(-b-$\sqrt{Del}$)/2a x2=(-b+$\sqrt{Del}$)/2a
			Del=0	x1=x2= -b/2a
			Del<0	Brak pierwiastków rzeczywistych
a=0	b≠0	dowolne	brak	Równanie liniowe
		c≠0		Równanie sprzeczne
		c=0		Równanie tożsamościowe



Rys. 2.1. Schemat blokowy algorytmu wyznaczania pierwiastków RK

```cpp
#include <iostream>
#include <cmath>
using namespace std;
int main(int argc, char** argv) {
     float a, b, c, Del, x1,x2;
     cout<<"\nOBLICZANIE PIERWIASTKOW RK";
    cout<<"\nPodaj a= ";
    cin>> a;
    cout<<"Podaj b= ";   cin >> b;
    cout<<"Podaj c= ";    cin>> c;
    if (a!=0){
                  //To jest rownania kwadrat.
              Del=b*2 -4*a*c;
            if(Del>0){
              x1=(-b-sqrt(Del))/(2*a);
              x2=(-b+sqrt(Del))/(2*a);
              cout<<"x1= "<<x1;
              cout<<"  x2= "<<x2<<endl;
            }
            else if (Del==0){
                x1=-b/(2*a);
                cout<<"x1=x2= "<<x1<<endl;
            }
            else{
                 cout<<"Brak pierw. rzeczywistych\n";
            }
    }
    else{
                   // To nie jest RK!
              if (b!=0){
              cout<<"To jest rownanie liniowe\n";
              }
               else if (c!=0){
              cout<<"Rownanie jest sprzeczne\n";
                }
                else{
                 cout<<"Rownanie tozsamosciowe\n";
                }
     }
        return 0;
}
```

Rys. 2.2. Program obliczania piwrwiastków RK w C++



Rys. 2.3. Przykładowe wyniki programu

W algorytmie na rys. 2.2 widoczne są bardziej złożone rozgałęzienia, co przenosi się na tabulację widoczną w programie. Tu program wyznaczania pierwiastków RK posiada zrozbudowane sterowanie pozwalające na wykluczenie rozwiązań, gdy równanie nie jest kwadratowe. Zapewniono to przez wstępne badanie parametru a. Zakresy zastosowanych tu dwóch instrukcji if zaznaczono na rys. 2.2 liniami pionowymi. Wewnętrzna instrukcja if jest bardziej rozbudowana i będzie wykorzystywana, jeśli równanie jest kwadratowe (tzn. a~0). W przedstawionej wersji zastosowano też inny sposób przedstawiania wyników niż w wersji z rys. 1.2. Tu są one wyświetlane w jednym wierszu.

W rozbudowanych programach zagnieżdżenie może obejmować setki instrukcji sterujących. Przykład zagnieżdżenia czterech instrukcji pokazano w rozwiązaniu zadania 3 polegającego na badaniu różnych własności wprowadzonej liczby (dodatnia, parzysta, itp.).

Zadanie 3

Komputer prosi o podanie liczby. W przypadku gdy liczba jest dodatnia i nie ułamkowa komputer identyfikuje czy jest to liczba parzysta czy nieparzysta. Przedstaw program w Matlabie badający liczbę ze względu na podane cechy. Narysuj schemat blokowy.

Na rys. 3.1. przedstawiono zapis programu z uwzględnieniem tabulacji (przesunięć) pokazującej zasięg działania danej instrukcji. Dodatkowo zasięg zaznaczono liniami pionowymi.

```
% WEJ: L- liczba badana wprowadzana z klawiatury
%          Skrypt bada liczby dodatnie i określa ich cechy
 %           (np. ułamek, liczba całkowita, parzysta itp)
L=input(' Wprowadz L= ');
if L >0
       disp('L - jest dodatnie');
      if L >=1
          disp('   L- jest liczbą większą od 1');
          if L== round(L)
             disp('     L- jest całkowite')
                  if L/2 == round(L/2)
                      disp('       L-jest parzyste');
                  else
                      disp('       L-jest nieparzyste');
                  end % if L/2==...
          else
                  disp('   L-jest niecałkowite');
          end  % L==...
      else
           disp('   L- jest ułamkiem');
      end %  if L>=1
elseif L==0
       disp(' L - jest równe zero!')
else
        disp(' L- jest ujemne !');
end %if L>0
```

Rys. 3.1. Zapis programu badania liczby z uwzględnieniem tabulacji

Zadanie 4

Sformułuj algorytm sprawdzający czy z boków o długościach a, b, c można zbudować trójkąt.

Specyfikacja algorytmu:

Dane wejściowe:

a, **b**, **c** – zadane długości boków,

Wynik

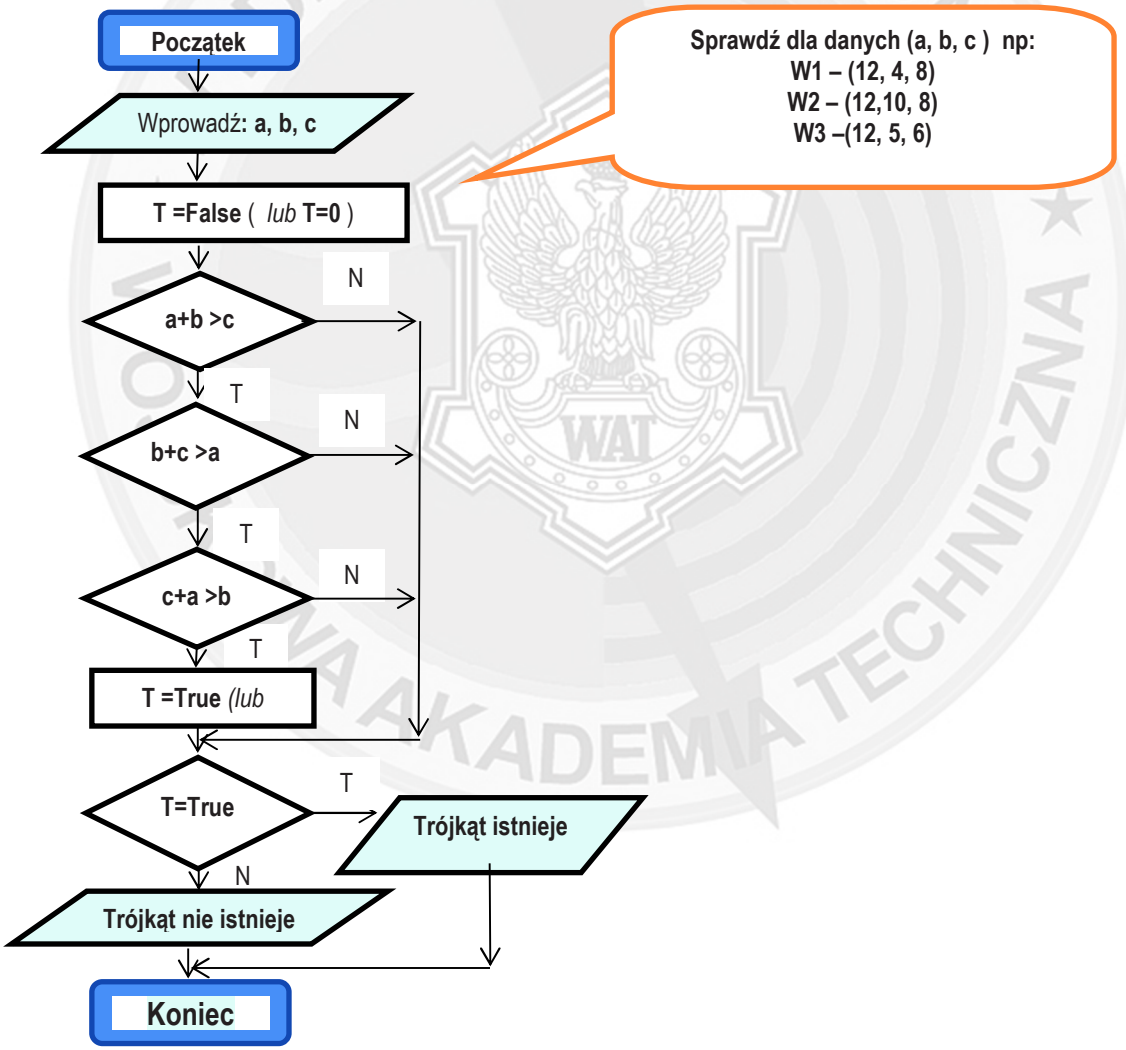
T – zmienna logiczna (np. od nazwy Trójkąt Istnieje) przyjmująca wartości:

T=Tak (True) – jeśli trójkąt istnieje lub **T=Nie (False)** – jeśli trójkąt nie istnieje

Lista kroków

1. Wprowadź: **a**, **b**, **c**
2. Zmiennej **T** przypisz wartość początkową **False**
3. Jeśli a+b >c oraz a+c >b oraz c+a >b to **T=True**
4. **Jeśli T=False** *to pisz:* ***Trójkąt nie istnieje***
5. **Jeśli T=True** *to pisz:* ***Trójkąt istnieje***
6. Koniec

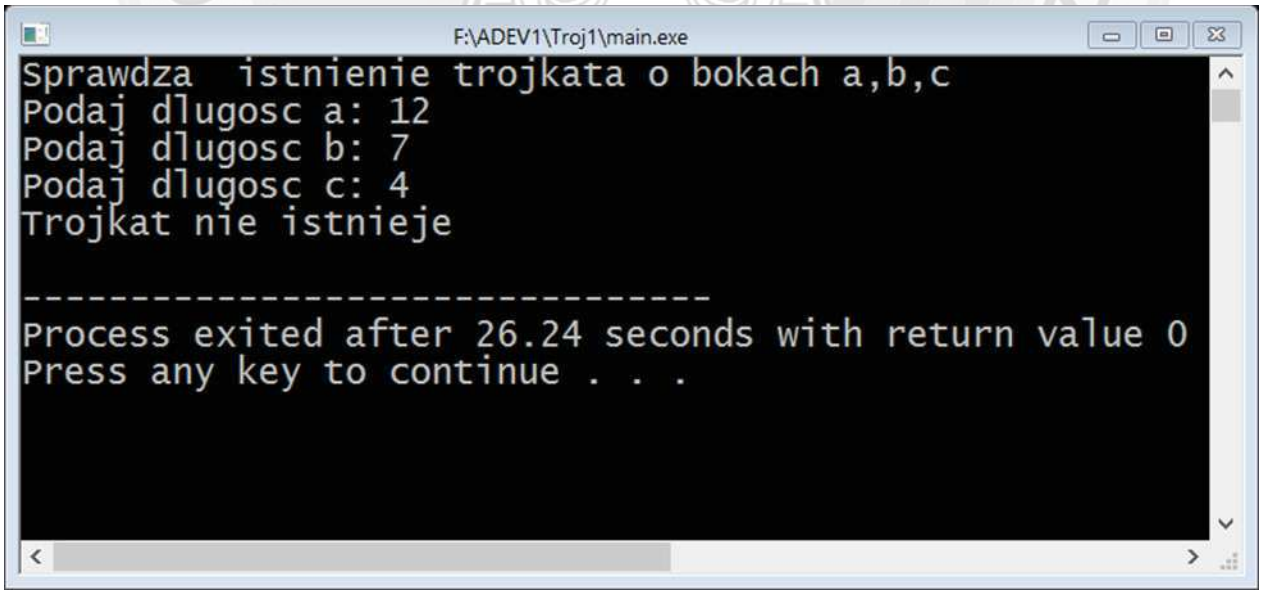
Schemat blokowy



Rys. 4.1 Schemat blokowy algorytmu zadania

```cpp
#include <iostream>
using namespace std;
int main(){
float a,b,c = 0;
bool T = false;
cout << "Sprawdza istnienie trojkata o bokach a,b,c\n";
cout << "Podaj dlugosc a: "; cin>>a;
cout << "Podaj dlugosc b: "; cin>>b;
cout << "Podaj dlugosc c: ";cin>>c;
if ((a+b) > c){
       if ((b+c) > a){
             if ((c+a) > b){
          T = true;
             }
        }
}
if (T == true){
     cout << "Trojkat istnieje \n";
 }
else {
    cout << "Trojkat nie istnieje \n";
}
return 0;
}
```

Rys. 4.2 Program w C++

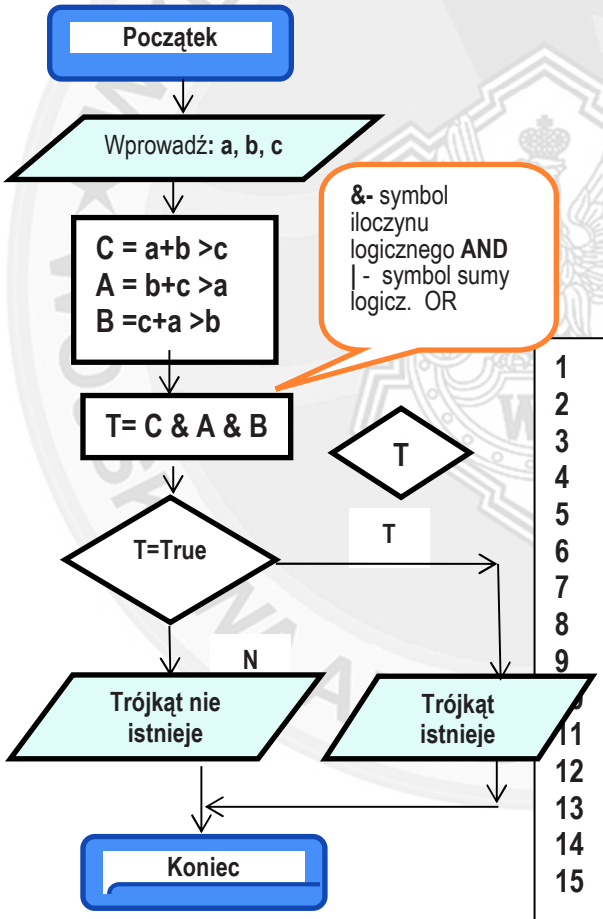


Rys. 4.3 Przykład wyników programu

Użycie zmiennych i operacji logicznych w schematach blokowych i programach

Zadanie 5
Sformułuj algorytm sprawdzający czy z boków o długościach a, b, c można zbudować trójkąt. Użyj zmiennych i operacji logicznych.

Specyfikacja algorytmu:
Dane wejściowe:
a, **b**, **c** – zadane długości boków (zmienne liczbowe, liczby dodatnie),
Zmienne pomocnicze (robocze)
A, B, C – zmienne logiczne przyjmujące wartości 0 lub 1
np. C=1 (True) jeśli jest prawdziwa relacja a+b>c a jeśli nieprawdziwa to C=0 (False).
Wynik
T – zmienna logiczna (np. od nazwy Trójkąt) przyjmująca wartości:
T=Tak (True) – jeśli trójkąt istnieje lub **T=Nie (False)** – jeśli trójkąt nie istnieje



Rys. 5.1. Schemat blokowy

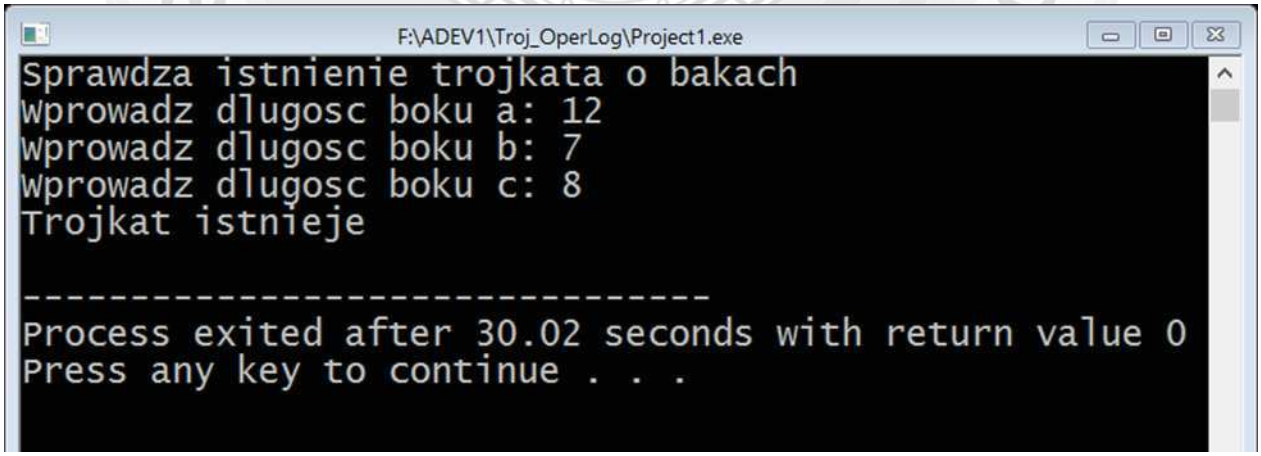
```
1   clc; clear;  close all
2   disp('Algorytm sprawdza istnienie  trójkąta );
3   disp('o bokach  a, b, c') ;
4   a = input('Podaj   bok a: ');
5   b = input('Podaj  bok b: ');
6   c = input('Podaj  bok c: ');
7     C= a+b > c % A, B,C, T beda  wyswietlone
8    A= b+c > a
9     B= c+a > b
      T = A&B&C
11  if T == true
12     disp('Trójkąt istnieje');
13  else
14     disp('Trójkąt nie istnieje');
15  end;
```

Rys. 5.2. Program w Matlabie

```cpp
#include <iostream>
using namespace std;
int main(){
  float a,b,c = 0;
  bool T = true;
  cout << "Sprawdza istnienie trojkata o bakach\n";
  cout << "Wprowadz dlugosc boku a: ";
  cin>>a;
  cout << "Wprowadz dlugosc boku b: ";
  cin>>b;
  cout << "Wprowadz dlugosc boku c: ";
  cin>>c;

  bool C, A, B;
   C=a+b>c;
   A=b+c>a;
   B=c+a>b;
  T=A && B && C;
  if (T == true){
        cout << "Trojkat istnieje \n";
  }
  else {
        cout << "Trojkat nie istnieje \n";
  }
return 0;
}
```

Rys. 5.2. Program w C++



Rys. 5.3. Przykład wyników programu

Sposoby przedstawiania algorytmów
Zastosowanie pętli w algorytmach

Prosty algorytm mieszany, iteracyjny i liniowy

Zadanie 6

Sformułuj algorytm i napisz program obliczający sumę ciągu N liczb naturalnych. Wartość N jest wprowadzana z klawiatury. Wyświetl wyniki podając ilość sumowanych liczb i wartość ich sumy.
(*Przedstaw listę kroków, pseudokod, schemat blokowy i program).*

Specyfikacja algorytmu:

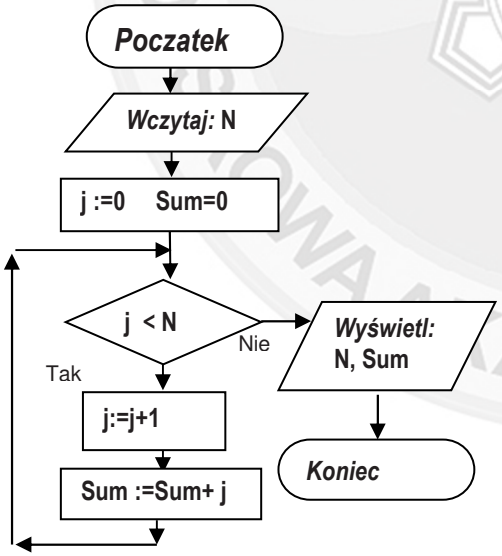
I Dane wejściowe:
- N- liczba całkowita, dodatnia określająca zakres sumowania (N>0);

Wyniki:
- Sum- liczba określająca sumę ciągu liczb Sum=1+2+…..+N .

Zmienne pomocnicze:
j - kolejna liczba całkowita

II Lista kroków:
1. Wczytaj z klawiatury ilość sumowanych liczb ***N***.
2. Zmiennej ***Sum*** przypisz wartość początkową *0*.
   Zmiennej *j* wartość ***0***;
3. Zwiększ ***j=j+1***
4. **Do dotychczasowej sumy Sum dodaj liczbę *j***
   Sum=Sum+j
5. **Jeśli *j <N* to wróć do kroku 3. Nie to p. 6.**
6. Wyświetl wartość zmiennej ***N*** i ***Sum***.
7. **Koniec**

III. Schemat blokowy algorytmu.



Rys. 6.1 Schemat blokowy algorytmu programu sumowania kolejnych liczb

IV Pseudokod

```
Program SumaLiczb1-N
zmienne:
   N, j, Sum      - całkowite
początek
   czytaj(N)
   j:=0   Sum:=0
   powtarzaj dopóki  j  < N
        j:=j+1
        Sum:=Sum +j
Wyświetl (N, Sum)
Koniec
```

Rys. 6.2. Pseudokod algorytmu

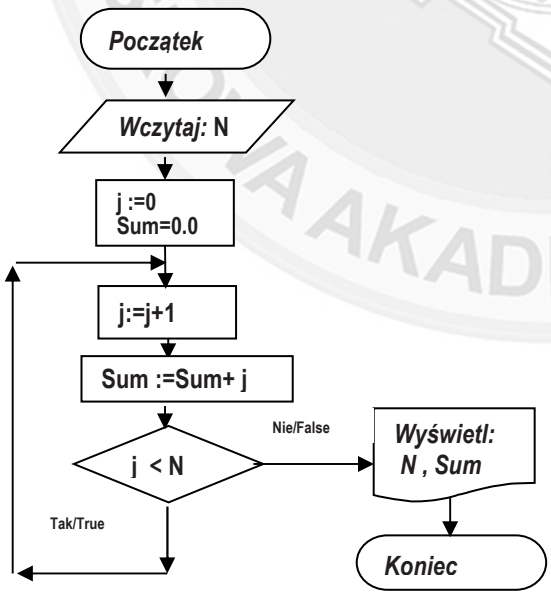
W algorytmie 6.1 wynik powstaje przez wielokrotne zwiększanie zmiennej Sum o kolejną liczbę naturalną. Taki algorytm określamy nazwą iteracyjnego a każdy krok iteracją. Algorytm 6.1 można

```
1    #include <iostream>
2    /* Program oblicza sumę kolejnych liczb 1...N */
3    using namespace std;
4    int main(int argc, char** argv) {
5    int N, Sum=0;
6    cout<<"Oblicza sume licz calk. 1...N";
7    cout<<"\nPodaj N>0 N= ";
8    cin>>N;
9    for (int j=1; j<=N; j++)//j=j+1lubj++
10   {
11     Sum=Sum+j; // lub Sum +=j;
12   }
13   cout<<"Dla N="<<N<<" Sum= "<<Sum<<endl;
14   return 0;
15   }
```

Rys. 6.3. Program w C++

określić jako iteracyjny, ponieważ główne obliczenia są wykonywane w pętli. Ale też można go uznać jako mieszany (sekwencyjnie realizowane działania z iteracyjnie realizowanym blokiem sumującym).

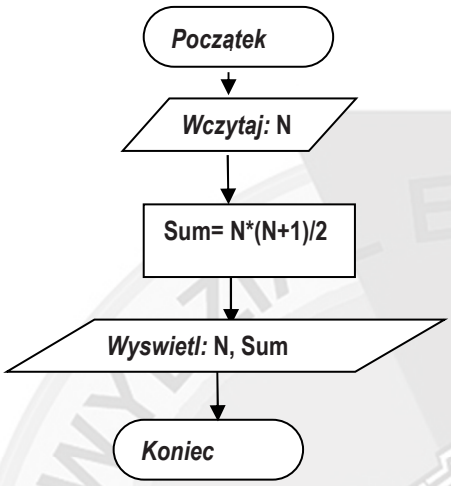
Rysunek 6.4 przedstawia inny sposób kontroli liczby powtórzeń pętli. Tu najpierw do bieżącej wartości Sum dodawana jest kolejna liczba a następnie sprawdzane jest czy podaną operacje ponownie powtórzyć.



Rys. 6.4. Schemat blokowy- wersja z badaniem powtarzania po wykonaniu iteracji

W schemacie na rys. 6.1 operację dodawania Sum=Sum+j poprzedza kontrola, co powoduje, że pętla nie wykona się, jeśli będzie zadane N=0. Wynikiem będzie Sum=0. Natomiast przy algorytmie 6.4 dla N=0 będzie wynik Sum =1 - a więc niepoprawny.
Stąd przy specyfikacji danych wejściowych dla przykładu 1 jest wymóg aby N było większe do zera. Wówczas obydwa algorytmy będą dawały poprawne wyniki.

Zadanie 6 może być rozwiązane np. z wykorzystaniem wzoru Pascala- rysunek 6.5.



Rys. 6.5. Schemat blokowy- wersja z wykorzystaniem wzoru Pascala

Przedstawiony algorytm wykorzystujący do obliczeń wzór Pascala jest prostym algorytmem sekwencyjnym (liniowym). Nie ma tu żadnych rozgałęzień ani powtarzania obliczeń. Wyszczególnione etapy są realizowane kolejno jeden po drugim.

Zadanie 7

Sformułuj algorytm obliczający sumę ciągu N liczb wprowadzanych do komputera. Ilość liczb jest pierwszą wprowadzaną wartością i poprzedza wprowadzane liczby. Przedstaw liste kroków, schemat blokowy i zapis algorytmu w pseudokodzie.

Specyfikacja algorytmu:

I Dane:

Dane wejściowe:

- **N- liczba całkowita określająca ilość wprowadzanych liczb;**
- $L_1, L_2, ..., L_j, ..., L_N$ **-ciąg N liczb rzeczywistych**

Wynik:

Sum- liczba rzeczywista określająca sumę wprowadzonych dotychczas liczb.

Zmienne pomocnicze:

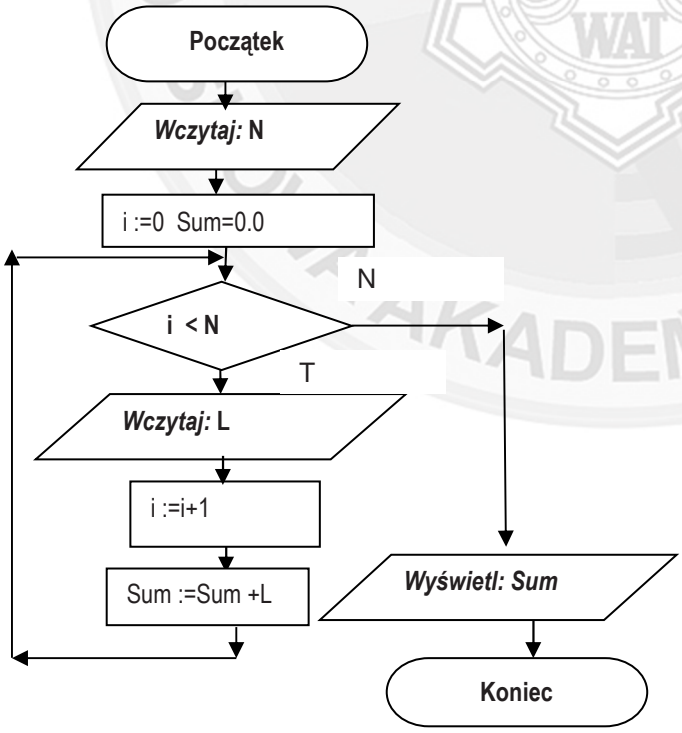
i - liczba całkowita (licznik powtórzeń);

liczba- liczba rzeczywista przyjmująca wartość kolejno wprowadzanych liczb

II Lista kroków:

1. Wczytaj z klawiatury do zmiennej ***N*** ilość wprowadzanych liczb.
2. Zmiennej ***Sum*** przypisz wartość początkową *0*.
3. Zmiennej ***i*** przypisz wartość początkową ***0***.
4. Wczytaj z klawiatury kolejną liczbę i zapamiętaj ją w zmiennej ***L.***
5. Zwiększ wartość licznika ***i*** o ***1 (jeden)***.
6. Do dotychczasowej sumy (**Sum**) dodaj wczytaną liczbę (***L***).
7. Jeśli ***i <N*** to wróć do kroku 4.
8. Wyświetl wartość zmiennej ***Sum***.
9. Koniec

III Schemat blokowy



Rys. 7.1. Schemat algorytmu rozwiązania zadania 4.

IV Pseudokod

```
Program SumaLiczb
zmienne:
        N, i      - całkowite
        L, Sum - rzeczywiste
początek
   czytaj(N)
   i:=0    Sum:=0
   powtarzaj dopóki i < N
       czytaj(L)
        i:=i+1
       Sum:=Sum +L
pisz(Sum)
koniec
```

Rys. 7.2. Algorytm w pseudokodzie .

Dla poprzedniego zadania przedstaw inne wersje zapisu algorytmu w pseudokodzie.

Zapisz w pseudokodzie przykładowe wersje algorytmu (z zadania 4) sumującego N wprowadzanych liczb do komputera. Ilość liczb jest pierwszą wprowadzaną wartością i poprzedza wprowadzane liczby.

Poniżej, przedstawiono (dla porównania w tabeli a), wersję algorytmu w pseudokodzie (oznaczoną na rys. w poprzednim zadaniu jako IV Pseudokod) oraz pokazano 3 inne formy zapisu tego algorytmu w pseudokodzie. Różnią się one głównie sposobem zapisu pętli wczytującej N liczb.

a)

IV Pseudokod:

```
Program  SumaN_Liczb
zmienne
        N, i     - całkowite
        L, Sum- rzeczywiste
początek
   czytaj(N)
   i:=0
   Sum:=0
powtarzaj  dopóki  i  < N
        czytaj(L)
        i:=i+1
        Sum:=Sum +L
pisz(Sum)
koniec
```

b)

Pseudokod I:

```
Program  SumaN_Liczb
zmienne
        N, i     - całkowite
        L, Sum- rzeczywiste
begin
   read(N)
   i:=0
   Sum:=0
while  i  < N  do
        read(liczba)
        i:=i+1
        Sum:=Sum +L
write(Sum)
end
```

c)

Pseudokod II:

```
Program  SumaN_Liczb
zmienne
        N, i     - całkowite
        L, Sum- rzeczywiste
begin
   read(N)
   Sum:=0
for i:=1 step 1 until N do
   begin
     read(L)
     Sum:=Sum +L
   end
write(Sum)
end
```

d)

Pseudokod III:

```
Program  SumaN_Liczb
zmienne
        N, i     - całkowite
        L, Sum- rzeczywiste
begin
   read(N)
   i:=0
   Sum:=0
do
     read(L)
      i:=i+1
      Sum:=Sum +L
 while  i  < N
write(Sum)
end
```

Zadanie 8

Sformułuj algorytm i napisz program obliczający sumę Sum ciągu N liczb wprowadzanych do komputera. Ilość liczb N jest pierwszą wprowadzaną wartością. Wyświetl wyniki podając w jednym wierszu ilość sumowanych liczb i wartość ich sumy.
Przedstaw schemat blokowy i zapisz algorytm w MATLABIE oraz C++.

Dane wejściowe:

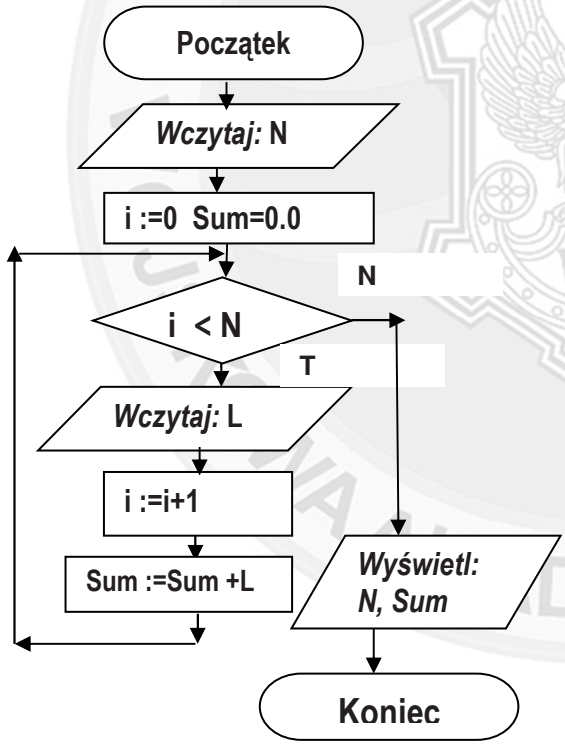
- **N- liczba całkowita, dodatnia określająca ilość wprowadzanych liczb (N>0);**
- **ciąg N liczb rzeczywistych $L_1, L_2,\ldots,L_N$**

Wyniki:

Sum- liczba rzeczywista określająca sumę wprowadzonych dotychczas liczb.

Zmienne pomocnicze:

i - liczba całkowita (licznik powtórzeń);
L- liczba rzeczywista przyjmująca
Li- wartość kolejno wprowadzanych liczb $L_1, L_2,\ldots,L_N$



Rys 8.1. Schemat blokowy (powtórzony z rys. 7.1)

```
1    /* Program oblicza sumę kolejnych liczb 1...N */
2    #include <iostream>
3    using namespace std;
4    int main(int argc, char** argv) {
5      int N;
6      float L, Sum=0;
7      cout<< "Oblicza sume wczyt. liczb rzecz.\n";
6      cout<<"\nPodaj N>0 N= ";
7      cin>>N;
8      for (int j=1; j<=N; j++)  {
9          cout<<"Podaj L= ";
10         cin>>L;
11        Sum=Sum+L; // lub Sum +=j;
12     }
13     cout<<"Dla N="<<N<<" Sum="<<Sum<<endl;
14     return 0;
15   }
```

Rys. 8.3. Program w C++

Rys. 8.2. Program w C++

Przykład 9

Sformułuj algorytm i napisz program wczytujący ciąg N liczb wprowadzanych do komputera i wyznaczający największą z wczytanych liczb. Ilość liczb N jest pierwszą wprowadzaną liczbą.
(*Przedstaw algorytm obliczeń w postaci listy kroków oraz schematu blokowego).*

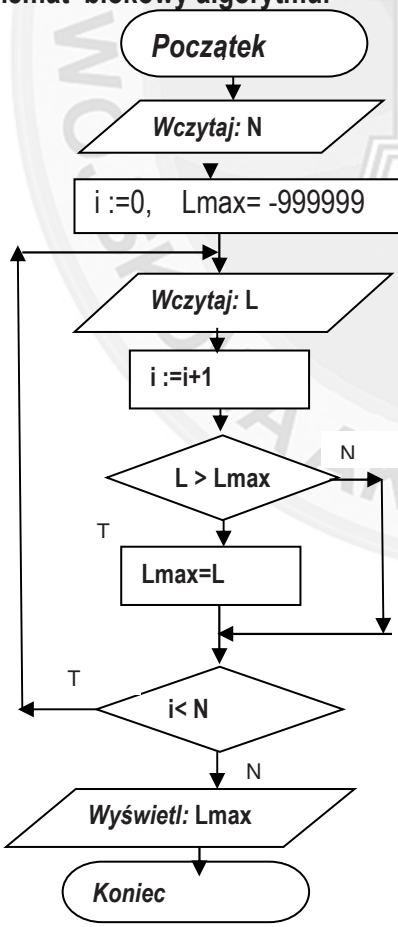
Dane wejściowe:
- **N- liczba całkowita, dodatnia określająca ilość wprowadzanych liczb;**
- **ciąg N liczb rzeczywistych $L_1, L_2,\ldots,L_N$**

Wyniki:
Lmax- najwieksza z wprowadzonych liczb.

Zmienne pomocnicze:
i - liczba całkowita (licznik powtórzeń);
LI- liczba rzeczywista przyjmująca wartość kolejno wprowadzanych liczb $L_1, L_2,\ldots,L_N$

Lista kroków:
1. Wczytaj z klawiatury liczbę określającą ilość liczb i zapisz ją w zmiennej ***N***.
2. Zmiennej ***Lmax*** przypisz wartość początkową np. **–999999.0.**
3. Zmiennej ***i*** przypisz wartość początkową ***0***.
4. **Wczytaj z klawiatury i-tą liczbe**
5. **i zapamiętaj ją w zmiennej *L*.**
   Jeśli L > Lmax to wykonaj:
   Lmax=L
6. **Zwiększ wartość licznika *i* o *jeden*.**
7. **Jeśli *i <N* to wróć do kroku 4.**
8. Wyświetl wartość zmiennej ***Lmax***. **Koniec**

. Schemat blokowy algorytmu.



Rys. 9.1. Schemat blokowy

Na rys. 9.2. przedstawiono przykładowy program rozwiązujący zadanie.

```
#include <iostream>
using namespace std;
int main(int argc, char** argv) {
int N;     float L, Lmax=-999999;
cout<<"Program wyznacza max wprowadz. liczbe";
cout<<"\nPodaj ile liczb N= ";
cin>>N;
for (int j=1; j<=N; j++) { //j++ odpowiada j=J+1
      cout<< "Podaj L= ";
      cin>>L;
      if (L>Lmax) {
          Lmax=L;
      }
}
cout<<"Dla N="<<N<<" max="<<Lmax<<endl;
return 0;
}
```

Rys. 9.2. Program w C++

Zadanie 10

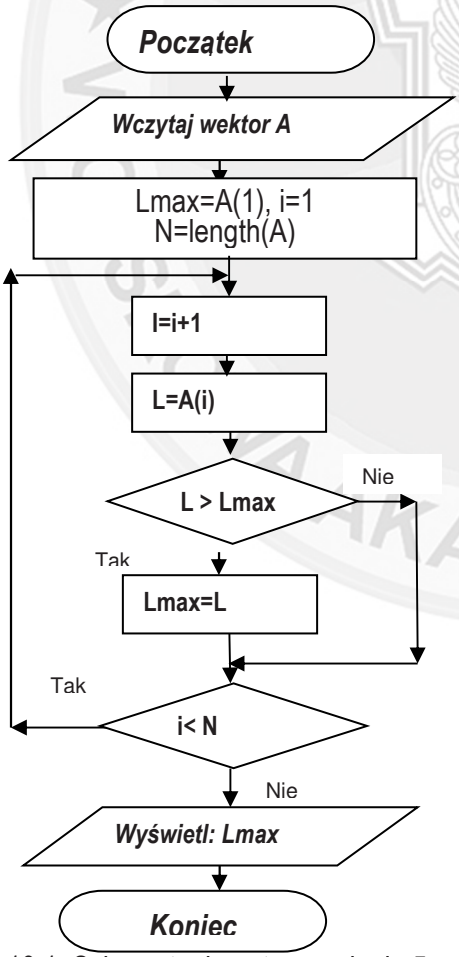
Sformułuj algorytm i napisz program wyznaczający największą z liczb zapamiętanych w wektorze A.
(*Przedstaw algorytm obliczeń w postaci listy kroków oraz schematu blokowego).*

Dane wejściowe:
- **A – wektor zawierający ciąg N liczb rzeczywistych $L_1, L_2,\ldots,L_N$**

Wyniki:
Lmax- najwieksza z wprowadzonych liczb.

Zmienne pomocnicze:
N- liczba całkowita określająca ilość liczb w wektorze A;

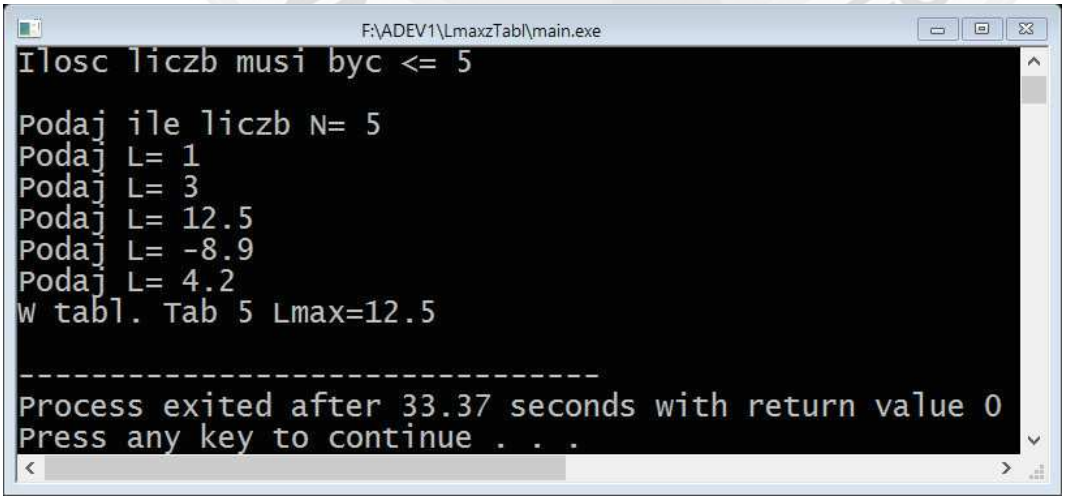
Lista kroków:
1. Wczytaj z klawiatury wektor A.
2. Zmiennej N przypisz wartość określającą ilość liczb w A
3. Zmiennej ***Lmax*** przypisz wartość początkową np. A(1)
4. Zmiennej ***i*** przypisz wartość początkową ***0***.
5. **Pobierz z wektora i-tą liczb i zapamiętaj ją w zmiennej L**
6. **Jeśli L > Lmax to wykonaj:**
   Lmax=L
7. **Zwiększ wartość licznika *i* o *jeden*.**
8. **Jeśli *i <N* to wróć do kroku 5.**
9. Wyświetl wartość zmiennej ***Lmax***.
10. **Koniec**



Rys. 10.1. Schemat algorytmu zadania 5

```
1   #include <iostream>
2   using namespace std;
3   int Rozm=5;
4   int main(int argc, char** argv) {
5   int N;
6   float L, Lmax, Tab[Rozm];
7   cout<<"Program wyznacza max liczbe z tablicy"<<endl;
8   cout<<"Ilosc liczb musi byc <= "<<Rozm;
9   cout<<"\n\nPodaj ile liczb N= ";
10  cin>>N;
11  // ---Wprowadzanie liczb do tablicy-------
12  for (int j=1; j<=N; j++)  {/// j++ odpowiada j=J+1
13        cout<<"Podaj L= ";
14         cin>>L;
15       Tab[j]=L;
16  }
17  //---------------------------------------
18  Lmax=Tab[1];
19  for (int j=1; j<=N; j++) {
20       L=Tab[j];
21       if (L>Lmax) {
22            Lmax=L;
23           }
24  }
25  cout<<"W tabl. Tab "<<N<<" Lmax="<<Lmax<<endl;
26  return 0;
27  }
```

Rys. 10.2 Zapisz program w C++



Rys. 10.3. Przykład wyników

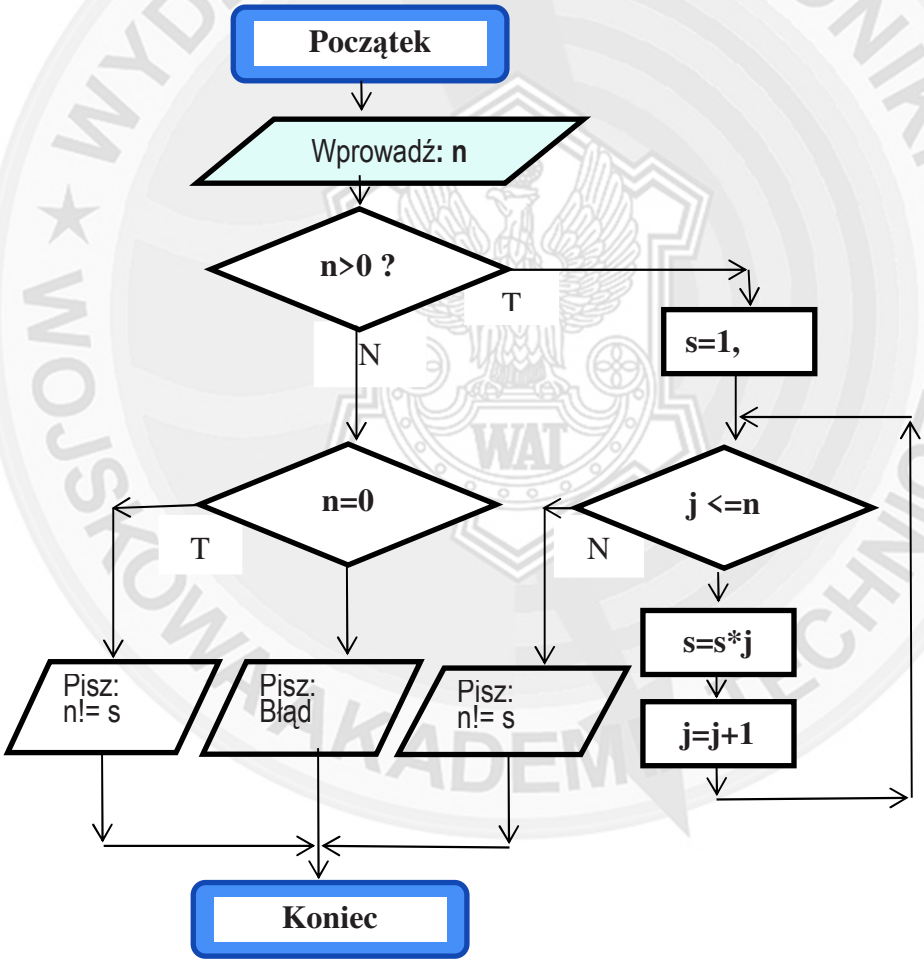
Zadanie 11

Przedstaw schemat blokowy obliczania silni dla zadanej liczby n.

Specyfikacja:
Dane wejściowe: n – zadana liczba (n- całkowite)
Wynik: s= n! - silnia liczby lub komunikat

Wartość n! jest wyznaczana ze wzoru:

$$n! = \begin{cases} 1 * 2 * \ldots * (n-1) * n & -dla\ n > 0 \\ 1 & -dla\ n = 0 \\ nie\ istnieje & -dla\ n < 0 \end{cases}$$



Rys. 11.1. Schemat blokowy algorytmu

```
1   #include <iostream>
2   using namespace std;
3   //#include <cmath>
4   int main() {
5   long n, s, j;
6   cout <<"Wprowadz n= ";
7   cin >> n;
8   if (n>0 ) {
9       s=1;
10      for (j=1; j<=n; j++)
11        s=s*j;
12     cout<<"Dla n= " <<n<< " n!= "<<s<<endl;
13  }
14  else {
15     if (n==0){
16     s=1;
17  }
18  else
19    cout<<"Dla n<0 n! nie istnieje\n";
20  return 0;
21  }
```

Rys. 11.3 Program w C++