ALGORYTMY I SPOSOBY ICH PRZEDSTAWIANIA

1. ALGORYTM - INFORMATYCZNY OPIS PLANU ROZWIĄZANIA

Pojęcie algorytmu

Wg Macieja M. Sysło ([8], s. 20).

Algorytm jest przepisem rozwiązywania postawionego zadania, będącym dokładnie określonym układem elementarnych instrukcji wraz z porządkiem ich wykonania.

Wg Władysława Turskiego ([9], s. 59)

Przez *algorytm* należy rozumieć opis obiektów łącznie z opisem czynności, które należy wykonać łącznie z tymi obiektami, aby osiągnąć określony cel.

Wg Stefana Węgrzyna ([10], s. 12)

Algorytm jest przepisem opisującym krok po kroku rozwiązanie problemu lub osiągnięcie jakiegoś celu.

Pojęcie algorytmiki

Algorytmika jest nazwą dziedziny zajmującej się algorytmami i ich własnościami, projektowaniem i analizą algorytmów.

(Dawid Harel: „Rzecz o istocie informatyki - algorytmika”.)

Pochodzenie terminu „algorytm”

Termin algorytm pochodzi od brzmienia fragmentu nazwiska: Muhammad ibn Musa al-Chorezmi[1].

Cechy algorytmu

S. Węgrzyn podaje następujące *cechy algorytmu*:

1. *Skończoność* (realizowany ciąg operacji powinien mieć swój koniec).
2. *Określoność* (zarówno operacje, jak i porządek ich wykonywania powinny być ściśle określone, nie zostawiając miejsca na dowolną interpretację użytkownika).
3. *Ogólność* (algorytm nie ogranicza się do jednego, pojedynczego, szczegółowego przypadku, ale odnosi się do pewnej klasy zadań).
4. *Efektywność* (algorytm powinien prowadzić do rozwiązania możliwie najprostszą drogą).

2. SPOSOBY PRZEDSTAWIANIA ALGORYTMÓW

- **Słowny opis algorytmu**

Przykład. Utworzyć algorytm i przedstawić jego słowny opis dla podanego zadania: obliczyć wartość funkcji

$$f(x)=\begin{cases}\dfrac{x}{|x|}, & dla\ x\neq 0,\\ 0, & dla\ x=0.\end{cases}$$

Rozwiązanie

Dane: Dowolna liczba rzeczywista x.

Wynik: Wartość funkcji f(x) określonej następującym wzorem:

$$f(x)=\begin{cases}-1, \text{dla x} < 0,\\ 0, \text{dla x} = 0,\\ 1, \text{dla x} > 0.\end{cases}$$

- **Opis algorytmu w postaci listy kroków**

Przykład. Algorytm obliczania wartości funkcji

$$f(x)=\begin{cases}-1, \text{dla x}<0,\\ 0, \text{dla x}=0,\\ 1, \text{dla x}>0.\end{cases} \qquad (*)$$

Dane: Dowolna liczba rzeczywista x.

Wynik: Wartość funkcji f(x) określonej wzorem (*).

Krok 0. Wczytaj wartość danej x.

Krok 1. Jeśli x > 0, to f(x) = 1. Zakończ algorytm.

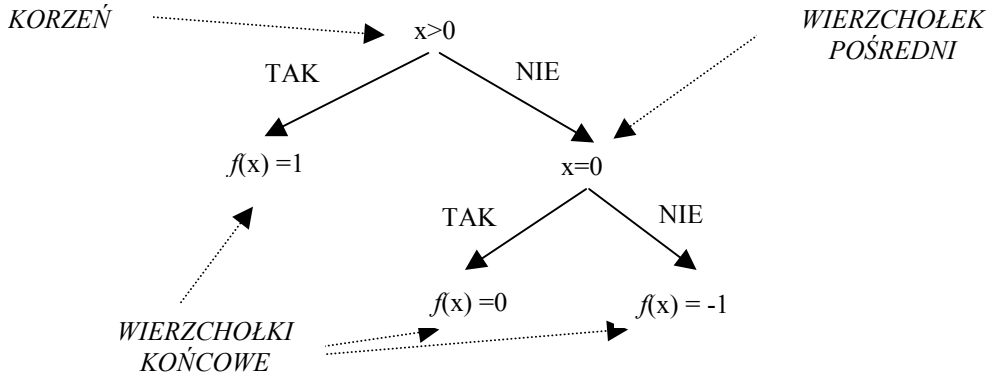
Krok 2. {W tym przypadku $x \le 0$.} Jeśli x = 0, to f(x) = 0. Zakończ algorytm.

Krok 3. {W tym przypadku x < 0.} Mamy f(x) = -1. Zakończ algorytm.

- **Drzewo algorytmu (drzewo obliczeń)**

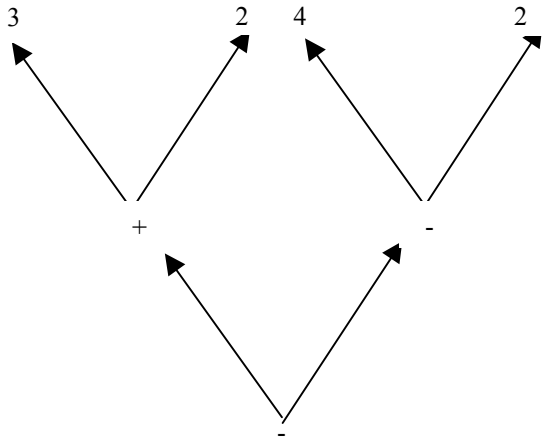
Przykład. Drzewo algorytmu obliczania wartości funkcji

$$f(x)=\begin{cases}-1, \text{dla x}<0\\ 0, \text{dla x}=0\\ 1, \text{dla x}>0\end{cases}$$



- **Drzewo wyrażenia**

Przykład. Zbudować drzewo wyrażenia: $(3+2)-(4-2)$



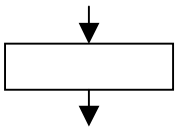
- **Schemat blokowy algorytmu**

  Zasady budowy:

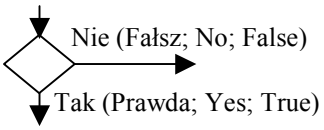
  - każda operacja, relacja lub informacja umieszczona jest w skrzynce;
  - kolejność wykonywania operacji wyznaczają połączenia między skrzynkami;
  - każde połączenie jest zaczepione początkiem do skrzynki, a końcem do innej skrzynki lub innego połączenia; żadne połączenie nie rozdziela się;
  - skrzynki przybierają kształty: prostokąta, rombu (lub sześcioką-ta), równoległoboku, okręgu lub owalu.

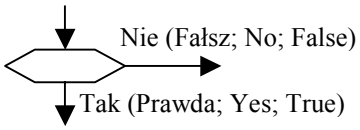
Rodzaje skrzynek:

- *Skrzynka operacyjna*

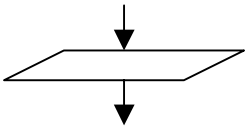


- *Skrzynka warunkowa* (*decyzyjna*)

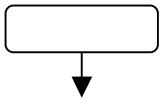


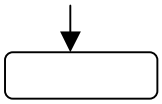


- *Skrzynka wprowadzania i wyprowadzania informacji* (skrzynka *We/Wy*)



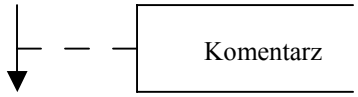
- *Skrzynki graniczne* START i STOP





- *Skrzynka łącznikowa*

- *Skrzynka komentarza*



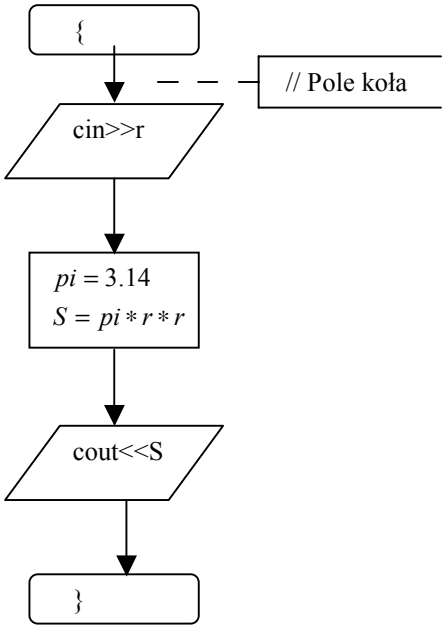
Typowe struktury schematów blokowych:

- *schematy blokowe liniowe*
- *schematy blokowe z rozgałęzieniami*
- *schematy blokowe cykliczne*

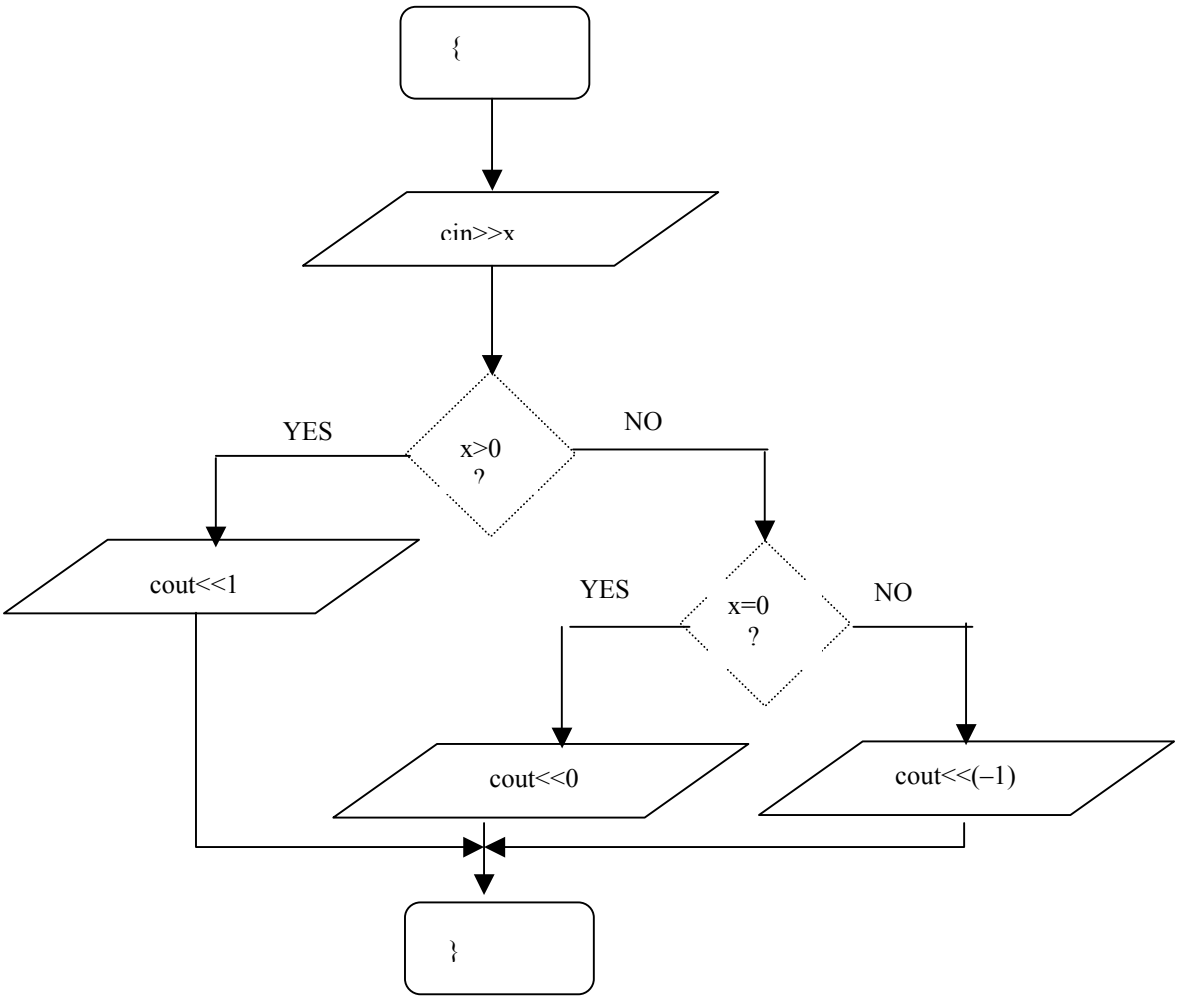
Przykład

a) Schemat blokowy liniowy obliczania pola *S* powierzchni koła o promieniu *r.*

Rozwiązanie



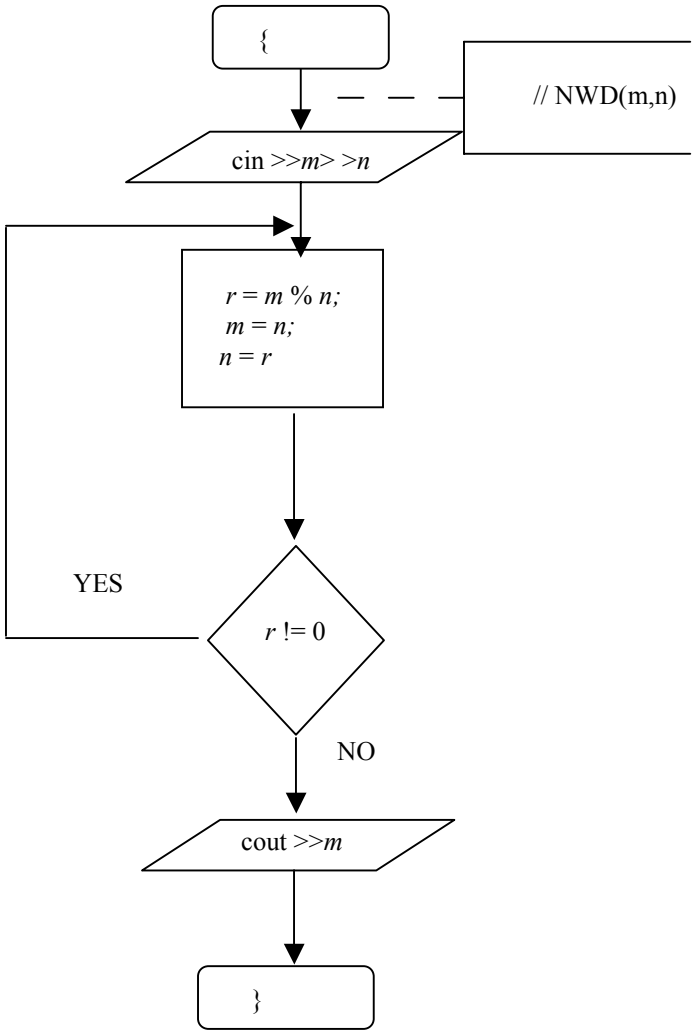
b) Schemat blokowy (z rozgałęzieniami) obliczania wartości funkcji *f*(*x*) określonej wzorem (*):



c) Schemat blokowy cykliczny (z pętlą) obliczania największego wspólnego dzielnika dwóch liczb całkowitych nieujemnych metodą Euklidesa

Dane: Dwie liczby naturalne *m* i *n*, m≤n.

Wynik: NWD(*m, n*) – największy wspólny dzielnik *m* i *n*.



- **Elementy języka programowania C++**

1. *Nagłówek programu*

void main()

// Pole_rombu

2. *Deklaracja stałych*

const <typ wartości>< nazwa> = <wartość>;

const float waga=70.5;

3. *Deklaracja zmiennych*

<typ wartości> <lista nazw>;

int i, j;

Wybrane dostępne typy:

unsigned, **int**, **long int** (całkowity),

float, **double**, **long double** (rzeczywisty),

char (znakowy),

typedef <typ wartości**><**nazwa typu**>** [górny_zakres]; (tablica jednowymiarowa)

Np. **typedef int** tab[10];

4. *Treść programu*

{ <instrukcja złożona> }

Np. {*I1;I2;...In;*}, gdzie *Ij* - instrukcja

Instrukcje:

- *Instrukcja przypisania*:

  x = w;

  gdzie: *x* - nazwa zmiennej, *w* - wyrażenie.

- *Instrukcja wejścia*:

  cin >> *x*;

  gdzie: *x* - nazwa zmiennej.

  W wersji polskiej: czytaj (*x*).

- *Instrukcja wyjścia*

  cout << *w*;

  gdzie *w* - wyrażenie.

  W wersji polskiej: pisz(*w*)

- *Instrukcje selekcji* (warunkowe):

  if (*W*) *I*; albo **if** (*W*) *I1;* **else** *I2;*

  gdzie: *W* - warunek, *I1* oraz *I2* - instrukcje.

  W wersji polskiej: **jeśli** *W* **to** *I*
  albo
  jeśli *W* **to** *I1* **w przeciwnym przypadku** *I2*

- *Instrukcje iteracji*

  while (*W*) *I*;

  gdzie: *W* - warunek, *I* - instrukcja.

  W wersji polskiej: **dopóki** *W* **wykonuj** *I;*

  do {*I* } **while** (*W*);

  gdzie: *I* - instrukcja, *W* - warunek.

  W wersji polskiej: **powtarzaj** *I* **aż do** *W*

- Szczególny przypadek instrukcji iteracji

  for (*Z=A1; Z=A2 ; Z++*) *I;*

  gdzie: *Z* - zmienna, *A1* i *A2* - wyrażenia arytmetyczne typu całkowitego, *I* - dowolna instrukcja.

  W wersji polskiej: **dla** *Z=A1* **do** *A2* **z krokiem** *1* **wykonuj** *I*

- *Instrukcja złożona*

  {*I1; I2;...;In;*}

  I1; I2;...;In - ciąg instrukcji prostych lub złożonych.

Typowe struktury programów w języku C++

- *Programy liniowe*
- *Programy z rozgałęzieniami*
- *Programy cykliczne*

Przykład 1. Program liniowy

Napisać program w języku C++ algorytmu obliczania pola *S* koła o długości promienia *r.* Za liczbę *pi* przyjąć 3.14.

```
#include<iostream.h>
#define pi 3.14
void main()
//wersja 1
{
float r,s;
cout<<"r=";cin>>r;
s=pi*r*r;
cout<<"s="<<s;
}
```

```
#include<iostream.h>
void main()
//wersja 2
{
const int pi = 3.14;
float r,s;
cout<<"r=";cin>>r;
s=pi*r*r;
cout<<"s="<<s;
}
```

Przykład 2. Program z rozgałęzieniami

A. Napisać program w języku C++ algorytmu obliczania wartości funkcji *y = f(x)* określonej wzorem:

$$f(x)=\begin{cases}-1, \text{dla } x<0,\\ 0, \text{dla } x=0,\\ 1, \text{dla } x>0.\end{cases}$$

```
#include<iostream.h>
void main()
// Obliczanie wartości funkcji y=f(x)
{
float x;
cin >> x;
        if (x>0) cout<<"1";
        else if (x==0) cout<<"0";
                else cout<<"-1";
}
```

B. Napisać program w języku C++ algorytmu obliczania wartości funkcji *y = f(x)* określonej wzorem:

$$f(x)=\begin{cases}\dfrac{x}{|x|}, & dla\ x\neq 0,\\ 0, & dla\ x=0.\end{cases}$$

```
#include<iostream.h>
#include<math.h>
void main()
// Wersja 1
{
float x,y;
cout<<"x=";cin>>x;
if (x==0) y=0;
else y=x/abs(x);
cout<<"y="<<y;
}
```

```
#include<iostream.h>
#include<conio.h>
#include<math.h>

enum decyzja {jeden, dwa} ;
decyzja p  ;
void main()
// Wersja 2
{
clrscr();
float x,y;
//int p;
cout<<"x=";cin>>x;
        if (x!=0) p=dwa; else p=jeden;
switch ( p )
{
case jeden: y=0; break;
case dwa: y=x/abs(x); break;
default: ;
}
cout<<"y="<<y;
}
```

Przykład 3. Programy cykliczne.

A. Napisać program w języku C++ obliczania największego wspólnego dzielnika dwóch liczb całkowitych dodatnich *m* i *n* z użyciem funkcji reszty z dzielenia całkowitego %.

```
#include<iostream.h>;
void main()
// Nieznana liczba powtórzeń
{
int m,n,r;
cout<<"m=";cin>>m;
cout<<"n=";cin>>n;
do
{
r=m%n;
m=n;
n=r;
}
while (r!=0);
cout<<"nwd="<< m;

}
```

B. Napisać program w języku C++ obliczania silni *s* liczby całkowitej nieujemnej *n.*

```
#include<iostream.h>
void main()
//Znana liczba powtórzeń
{
int n,i,s;
cout <<"n="; cin>>n;
i=0;s=1;
while(i<n)
{i++;
s=s*i;
}
cout<<"s="<<s<<endl;
}
```

C. Napisać program w języku C++ obliczania pierwiastka kwadratowego z dowolnej liczby rzeczywistej dodatniej a korzystając z wzoru:

$$x_0 = a,$$
$$x_{i+1} = \frac{1}{2}(x_i + \frac{a}{x_i}) \text{ dla } i = 0,1,...$$

Program kończy działanie po wykonaniu *n* iteracji.

```
#include<iostream.h>

void main()
{
const int m=10;
typedef float tab[m];
tab x;
float a;
int n,i;
cout<<"n=";cin>>n;
cout<<"a=";cin>>a;
x[0]=a;
for (i=0;i<=n;i++)
x[i+1] = 0.5*(x[i]+a/x[i]) ;

cout<<"x="<<x[n]<<endl;
}
```

ZADANIA DO SAMODZIELNEGO ROZWIĄZANIA

Zadanie 1. Utworzyć drzewo wyrażenia: $(a+b)\cdot c/(a-b)$.

Zadanie 2. Utworzyć algorytm wyznaczania rozwiązania równania $ax + b = 0$ o współczynnikach rzeczywistych *a*, *b* i zmiennej rzeczywistej *x*. Algorytm opisać graficznie przy pomocy schematu blokowego oraz napisać program rozwiązywania tego równania w języku C++.

Zadanie 3. Korzystając z algorytmu Euklidesa wyznaczyć największy wspólny dzielnik liczb: 153 i 85.

LITERATURA

[1] Aho, A. V., Hopcroft, J. E., Ullman, J. D.: *Projektowanie i analiza algorytmów komputerowych,* PWN, Warszawa 1983.

[2] Harel, D.: *Rzecz o istocie informatyki. Algorytmika,* WNT, Warszawa 1992.

[3] Suraj, Z., Rumak, T.: *Algorytmiczne rozwiązywanie zadań i problemów,* Rzeszów 1995.

[4] Sysło, M. M.: *Algorytmy,* WSiP, Warszawa 1997.

[5] Wróblewski, P.: *Algorytmy, struktury danych i techniki programowania*, Helion, 1996.