

1. WPROWADZENIE

1.1. Algorytm, instrukcja, program

Chcąc rozwiązać jakieś zadanie, musimy opracować *algorytm*, czyli metodę rozwiązania tego zadania, zakładającą dojście do wyniku w drodze realizacji sekwencji pewnych działań elementarnych. Elementarne działanie może być wykonane przez komputer w odpowiedzi na przekazane mu polecenie, zwane *instrukcją*. Algorytm można przedstawić w postaci grafu działania, pseudokodu, lub ciągu instrukcji w określonym języku programowania. Ta ostatnia postać zapisu algorytmu nosi nazwę *programu komputerowego*.

Jako prosty przykład ilustrujący te pojęcia, rozważmy obliczanie rozwiązań znanego równania kwadratowego: $ax^2 + bx + c = 0$. Algorytm postępowania można zapisać słownie (tj. w tak zwanym autokodzie), jak następuje:

1. Określ wartości parametrów a, b, c
2. Oblicz wartość wyrażenia: $d = b^2 - 4ac$
3. Jeżeli $d \geq 0$ to wykonaj, co następuje:

a/ oblicz rozwiązania jako:

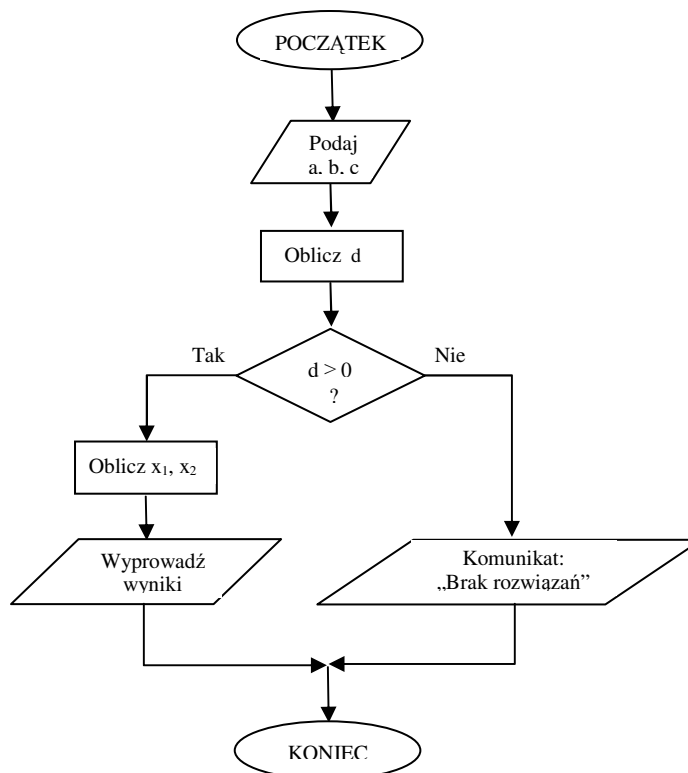
$$x_1 = (-b + \sqrt{d}) / 2a$$

$$x_2 = (-b - \sqrt{d}) / 2a$$

b/ wyprowadź wyniki

W przeciwnym przypadku wydrukuj komunikat: „Brak rozwiązań”.

Zapisany powyżej algorytm można przedstawić także w postaci grafu działania (rys. 1.1), w którym wierzchołki oznaczają realizację operacji składowych lub sprawdzanie warunków, a krawędzie łączące wierzchołki odpowiadają kolejności wykonania operacji.



Rys.1.1. Graf algorytmu obliczania pierwiastków równania kwadratowego

Na rysunku 1.2 pokazano ten sam algorytm, zapisany w postaci programów, czyli sekwencji instrukcji, w językach Turbo Pascal oraz Borland C++. Na razie nie wszystkie szczegóły tych zapisów są zrozumiałe. Staną się jasne po przeczytaniu kolejnych rozdziałów.

<pre> uses Crt; var a,b,c,d,x1,x2:Real; begin Readln(a,b,c); d:=b*b-4*a*c; if d>=0 then begin x1:=(-b+Sqrt(d))/(2*a); x2:=(-b-Sqrt(d))/(2*a); Writeln('x1=',x1); Writeln('x2=',x2); end else Writeln('Brak rozwiazan.');</pre>	<pre> #include <iostream.h> #include <math.h> void main() { float a,b,c,d,x1,x2; cin>>a>>b>>c; d=b*b-4*a*c; if (d>=0) { x1=(-b+sqrt(d))/(2*a); x2=(-b-sqrt(d))/(2*a); cout<<"x1="<<x1<<endl; cout<<"x2="<<x2<<endl; } else cout<<"Brak rozwiazan.";</pre>
---	---

Rys. 1.2. Programy znajdowania pierwiastków równania kwadratowego w TP i C++

1.2. Program źródłowy i wynikowy. Kompilacja

Języki algorytmiczne (zwane też językami wysokiego poziomu), które zazwyczaj stosuje się do zapisu programów (Basic, Pascal, C i inne), są łatwe do użycia i interpretacji przez człowieka, ponieważ stosowane w nich zapisy są podobne do języka naturalnego. Nazwy instrukcji i typów danych, słowa kluczowe, nazwy standardowych podprogramów itp. są słowami języka angielskiego lub ich skrótami. Również przy zapisie wyrażeń matematycznych lub logicznych stosuje się notację zbliżoną do powszechnie używanej.

Jednak program zapisany w języku algorytmicznym nie nadaje się do sterowania obwodami elektronicznymi komputera, które potrafią jedynie interpretować instrukcje zapisane w postaci kodów binarnych (zerojedynkowych).

Tę trudność rozwiązuje się, dzieląc proces opracowania programu na dwa etapy:

1. *Edycja*, tj. pisanie tekstu programu *źródłowego* w wybranym *języku algorytmicznym*
2. *Kompilacja*, tj. automatyczne tłumaczenie programu źródłowego na odpowiadający mu program *wynikowy* w *języku wewnętrznym* komputera, a więc w postaci sekwencji kodów zerojedynkowych

Rysunek 1.3 przedstawia postać pierwszych 16 bajtów postaci wynikowej obu programów z rysunku 1.2, uzyskanych w wyniku kompilacji. Są to niewielkie fragmenty początkowe tych programów. Pierwszy program wynikowy w całości zajmuje 7680 bajtów, a drugi ÷ 47 860 bajtów pamięci.

Teoretycznie można by było pisać program od razu w języku wewnętrznym. Rzut oka na rysunek 1.3 wystarczy jednak, by zauważyć, że takie postępowanie byłoby niezwykle nużące i bardzo mało wydajne. Prowadziłoby także do licznych błędów. Z tych względów w praktyce nigdy nie jest stosowane.

01001101	01001101
01011010	01011010
00000000	10000000
00000000	00000001
00001111	01000000
00000000	00000000
00101100	00001100
00000000	00000000
00001101	00100000
00000000	00000000
00101000	00000000
00000100	00000000
00101000	11111111
10100100	11111111
11111011	11010000
00000001	00000111
.

Rys. 1.3. Początkowe bajty programów z rysunku 1.2 w języku wewnętrznym komputera