

Podstawowe konstrukcje algorytmiczne

ALGORYTM – przepis postępowania, którego wykonanie prowadzi do rozwiązania określonego problemu w skończonym (realnym) czasie.

SPECYFIKACJA – ściśle zdefiniowanie problemu (zadania) algorytmicznego. Oprócz samego algorytmu zawiera wyszczególnienie: danych wraz z warunkami początkowymi oraz wyników wraz z warunkami końcowymi. Określa też związek między wynikami a danymi.

Algorytm można zapisać m.in. w postaci:

- opisu słownego (rzadko stosowany, nieprecyzyjny)
- listy kroków
- schematu blokowego (sieci działań)
- pseudokodu (sztucznego języka naturalnego, podobnego do języka Pascal)
- strukturogramu

Przykład

Skonstruuj algorytm obliczający pole prostokąta o bokach podanych przez użytkownika i zapisz go w postaci listy kroków, schematu blokowego i pseudokodu. Sporządź również specyfikację tego algorytmu.

Rozwiązanie

Specyfikacja algorytmu (tj. spodziewana postać danych wejściowych i wyników):

Dane:

a – długość jednego z boków (liczba rzeczywista większa od zera)

b – długość drugiego boku (liczba rzeczywista większa od zera)

Wynik:

P – pole prostokąta (liczba rzeczywista większa od zera)

1. Lista kroków

Krok 1:

Pod zmienną a wczytaj długość pierwszego boku

Krok 2:

Pod zmienną b wczytaj długość drugiego boku

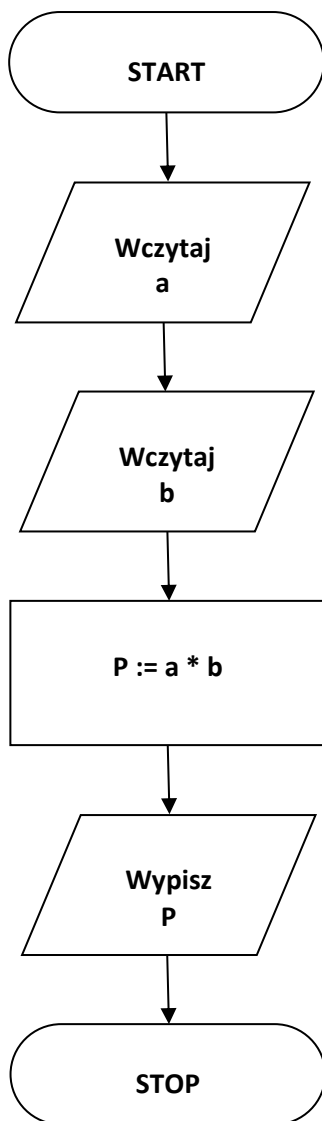
Krok 3:

Oblicz P jako iloczyn a i b

Krok 4:

Wypisz P

2. Schemat blokowy

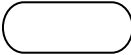
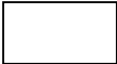





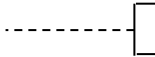




3. Pseudokod

PROGRAM Pole_prostokąta	{nagłówek programu}
ZMIENNE	
a, b, P: rzeczywiste	{deklaracje zmiennych}
POCZĄTEK	{początek właściwego programu}
Czytaj(a)	{wczytanie zmiennej a}
Czytaj(b)	{wczytanie zmiennej b}
P:=a*b	{instrukcja przypisania}
Wypisz(P)	{wypisanie wyniku na ekranie}
KONIEC	{koniec programu}

Algorytm przedstawiony jako schemat blokowy jest niezależny od języka, w jakim będzie napisany program komputerowy, co daje możliwość wymiany między programistami korzystającymi z różnych typów komputerów i języków programowania.

Za pomocą schematu blokowego łatwo sprawdzić logiczną poprawność algorytmu oraz przeprowadzić ewentualną korektę. Schematy blokowe nie są jednak dogodne do opisu bardziej rozbudowanych algorytmów. Najczęściej używane symbole schematów blokowych przedstawiono w tabeli.

Lp.	Symbol	Nazwa symbolu	Znaczenie symbolu
1.		Początek, koniec	Oznaczenie miejsca rozpoczęcia, zakończenia lub przerwania algorytmu
2.		Operator	Działanie (operacja do wykonania)
3.		Operator wejścia / wyjścia	Wprowadzanie danych do pamięci lub wyprowadzanie danych z pamięci
4.	 lub 	Element decyzyjny	Operacja określająca wybór jednej z alternatywnych dróg działania
5.		Łącznik stronicowy	Wejście lub wyjście z wyodrębnionych fragmentów schematu znajdujących się na jednej stronie
6.		Łącznik międzystronicowy	Wejście lub wyjście z wyodrębnionych fragmentów schematu znajdujących się na różnych stronach
7.		Komentarz	Oznaczenie miejsca na wyjaśnienia lub uwagi
8.		Droga przepływu danych	Więź informacyjna między poszczególnymi operacjami
9.		Droga przepływu danych o wskazanym kierunku przepływu	Więź informacyjna między poszczególnymi operacjami

Przy konstruowaniu schematów blokowych należy przestrzegać następujących zasad:

1. Rysunek musi być czytelny i staranny.
2. Ogólny przepływ informacji powinien przebiegać z góry na dół oraz z lewa na prawo.
3. Należy pamiętać o strzałkach wskazujących kierunek przepływu informacji.
4. Nie zapominać o oznaczeniach TAK/NIE (T/N, TRUE/FALSE, T/F itp.) poszczególnych wyjść elementów decyzyjnych.
5. Najczęściej w jednym operatorze powinna występować jedna instrukcja.
6. W miarę możliwości należy unikać przecięć linii łączących. Jeśli nie jest to możliwe, rysunek przecięcia powinien być jednoznaczny, aby uniknąć pomylenia linii.
7. W wypadku konstruowania pętli wielokrotnych należy unikać stosowania przecięć linii łączących. Rysunek powinien być na tyle wyraźny, aby było widać, która pętla jest bardziej wewnętrzna.
8. Należy używać komentarzy, pisanych od razu przy rysowaniu schematu blokowego. Komentarze nie mogą być jednak zbyt banalne (np. opisujące typ operatora).
9. Należy zwrócić szczególną uwagę na dobór nazw zmiennych, aby możliwie najlepiej odzwierciedlały one reprezentowane przez siebie wielkości. Dla przykładu zmienne niezależne mogą być oznaczane przez x , y , z , kąty geometryczne przez α , β , γ , zmienne dotyczące zagadnień ekonomicznych przez KOSZT, CENA, ZYSK, VAT, itp. Są to przykłady nazw zwyczajowych.
10. Nazwy zmiennych nie powinny być zbyt długie, łatwo się wówczas przy wielokrotnym przepisywaniu pomylić. Z tego samego powodu należy unikać nazw uduchowionych np. QFW04RJK.

Podstawowe struktury na schematach blokowych

1. Sekwencja – charakteryzuje się tym, że w sieci działań występują jedynie operatory, a każdy z nich posiada tylko jedno wejście. Przykładem jest algorytm obliczający pole prostokąta, długość okręgu itp.

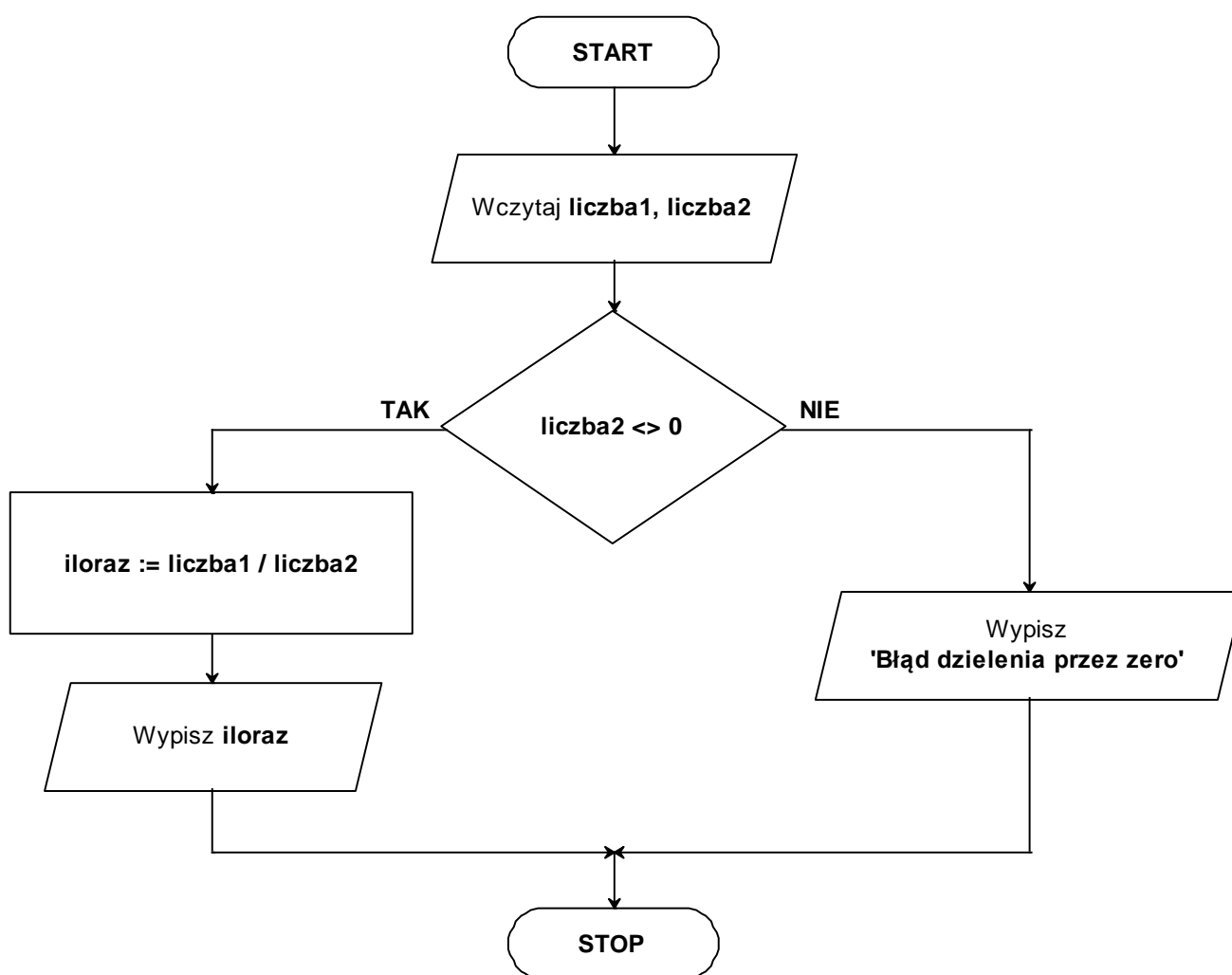
2. Rozwidlenie i decyzja

Sekwencyjne schematy blokowe dotyczą łatwych algorytmów, więc występują rzadko, ponieważ nie uwzględniają rozwiązań niejednoznacznych. Znacznie częściej mamy do czynienia z algorytmami, w których rozwiązanie jakiegoś problemu zależy od spełnienia określonych warunków.

Przykład 1.

Sporządź algorytm obliczający iloraz dwóch liczb rzeczywistych *liczba1* i *liczba2*. Uwzględnij przypadek, gdy $liczba2 = 0$. Algorytm zapisz w postaci schematu blokowego.

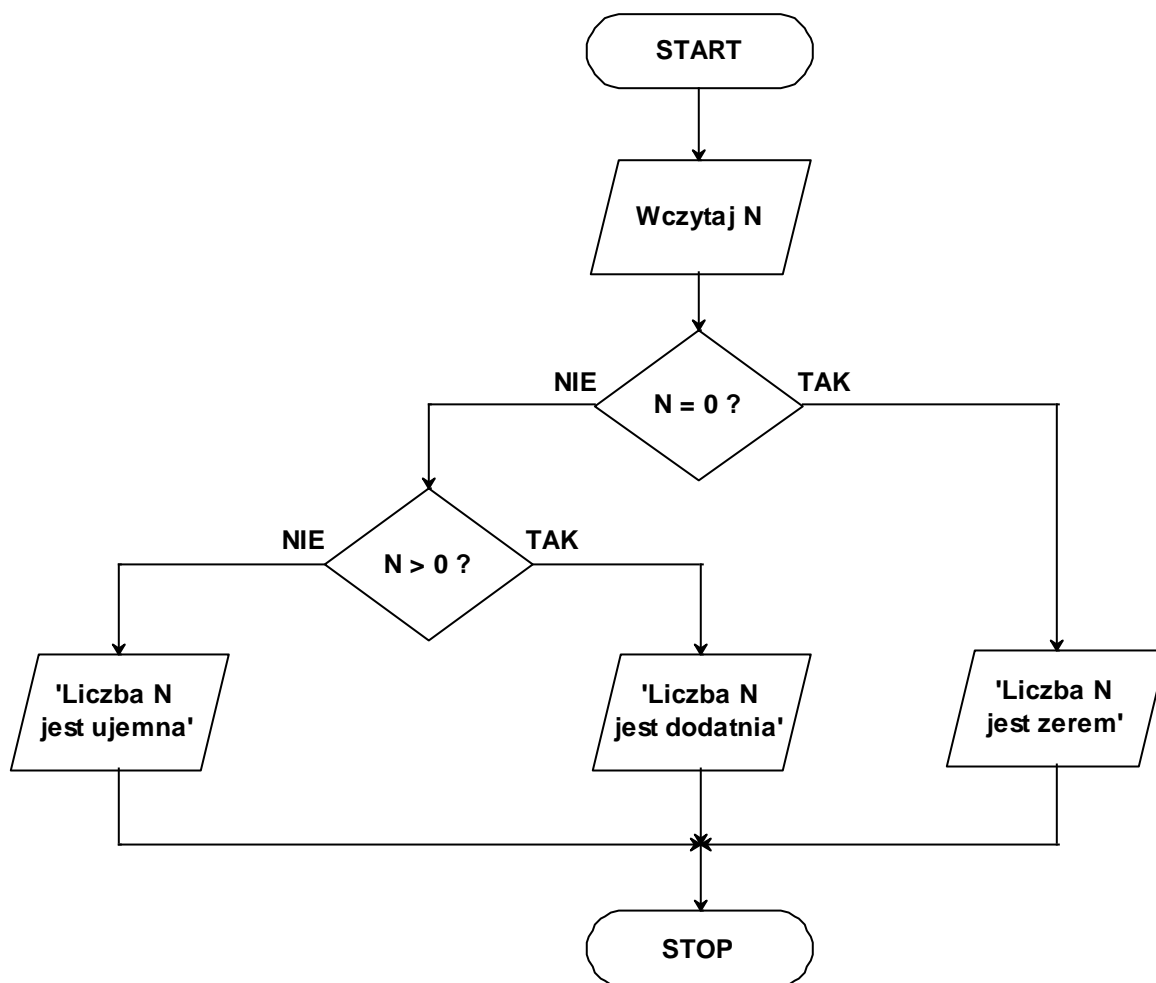
Rozwiązanie:



Przykład 2.

Opracuj algorytm, który wczytuje z klawiatury dowolną liczbę *N* i bada jej znak, tzn. określa, czy jest ona dodatnia, ujemna lub zerem.

Rozwiązanie:



Przykład 3.

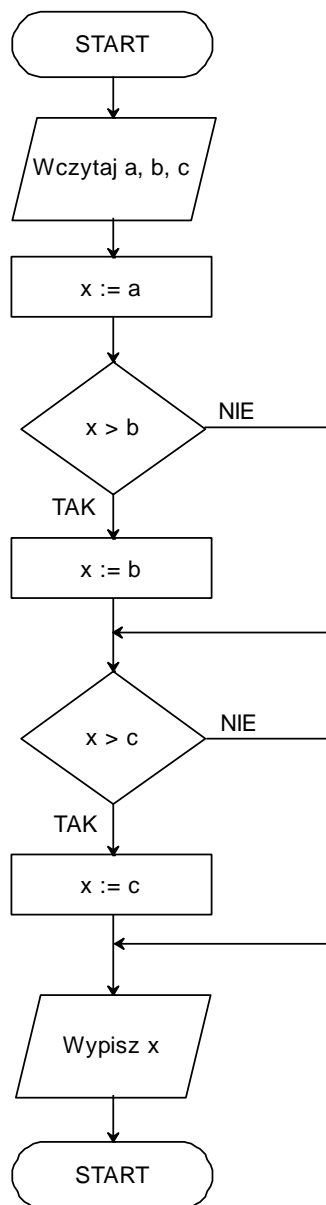
Skonstruuuj algorytm znajdujący minimalną z trzech liczb a , b , c oraz przetestuj jego działanie za pomocą przykładowego zestawu liczb. Zapisz algorytm w postaci listy kroków i schematu blokowego.

Rozwiązanie:

Na ogół proste problemy algorytmiczne mają niewiele różnych rozwiązań, trudniejsze można rozwiązać na dużo sposobów. Oto jeden z nich, wykorzystujący tzw. zmienną pomocniczą.

Lista kroków może mieć postać:

- podaj trzy dowolne liczby a , b , c ,
- przypisz zmiennej x wartość równą a ,
- porównaj x z b ,
- jeśli $x > b$, to zmiennej x przypisz wartość równą b . W przeciwnym wypadku wartość zmiennej x nie ulega zmianie,
- porównaj x z c ,
- jeśli $x > c$, to zmiennej x przypisz wartość równą c . W przeciwnym wypadku wartość zmiennej x nie ulega zmianie,
- koniec postępowania. Wartość zmiennej x oznacza rozwiązanie zadania.



Przykładowy zestaw liczb do testowania algorytmu musi zawierać wszystkie możliwe relacje, jakie zachodzą między liczbami. Tylko w ten sposób można sprawdzić, czy algorytm daje poprawne wyniki niezależnie od zestawu wprowadzonych liczb.

Przykładowe zestawy liczb a, b, c	Relacje zachodzące między liczbami a, b, c
{-1, 2, 3}	$a < b \wedge a < c$
{-1, 3, -1}	$a < b \wedge c = a$
{3, 7, 6}	$a < b \wedge c > a \wedge c < b$
{1, 5, 5}	$a < b \wedge c = b$
{1, 2, 3}	$a < b \wedge b < c$
{5, 5, -2}	$a = b \wedge c < a$
{4, 4, 4}	$a = b \wedge c = a$
{-1, -1, 0}	$a = b \wedge c > a$
{4, 2, 1}	$a > b \wedge c < b$
{4, 2, 2}	$a > b \wedge c = b$
{5, 1, 3}	$a > b \wedge c > b \wedge c < a$
{7, 3, 7}	$a > b \wedge c = a$
{4, 2, 8}	$a > b \wedge c > a$

Przykład 4.

Opracuj algorytm sortujący w kolejności rosnącej za pomocą porównań trzy liczby wprowadzone przez użytkownika (bez używania tablic).

Rozwiązanie:

