INFORMATYKA

Algorytmy

http://www.infoceram.agh.edu.pl

ALGORYTM

ALGORYTM to skończony ciąg jasno zdefiniowanych czynności, wskazujący kolejność operacji koniecznych do rozwiązania zadanego problemu. Słowo "algorytm" pochodzi od starego angielskiego słowa *algorism*, oznaczającego wykonywanie działań przy pomocy liczb arabskich (w odróżnieniu od *abacism* – przy pomocy abakusa). Wg innych źródeł nazwa algorytm pochodzi od nazwiska matematyka perskiego, nazywającego się Abu Abdullah Muhammad ibn Musa al-Chuwarizmi.

Algorytm umożliwia przejście systemu/układu ze stanu początkowego do końcowego.

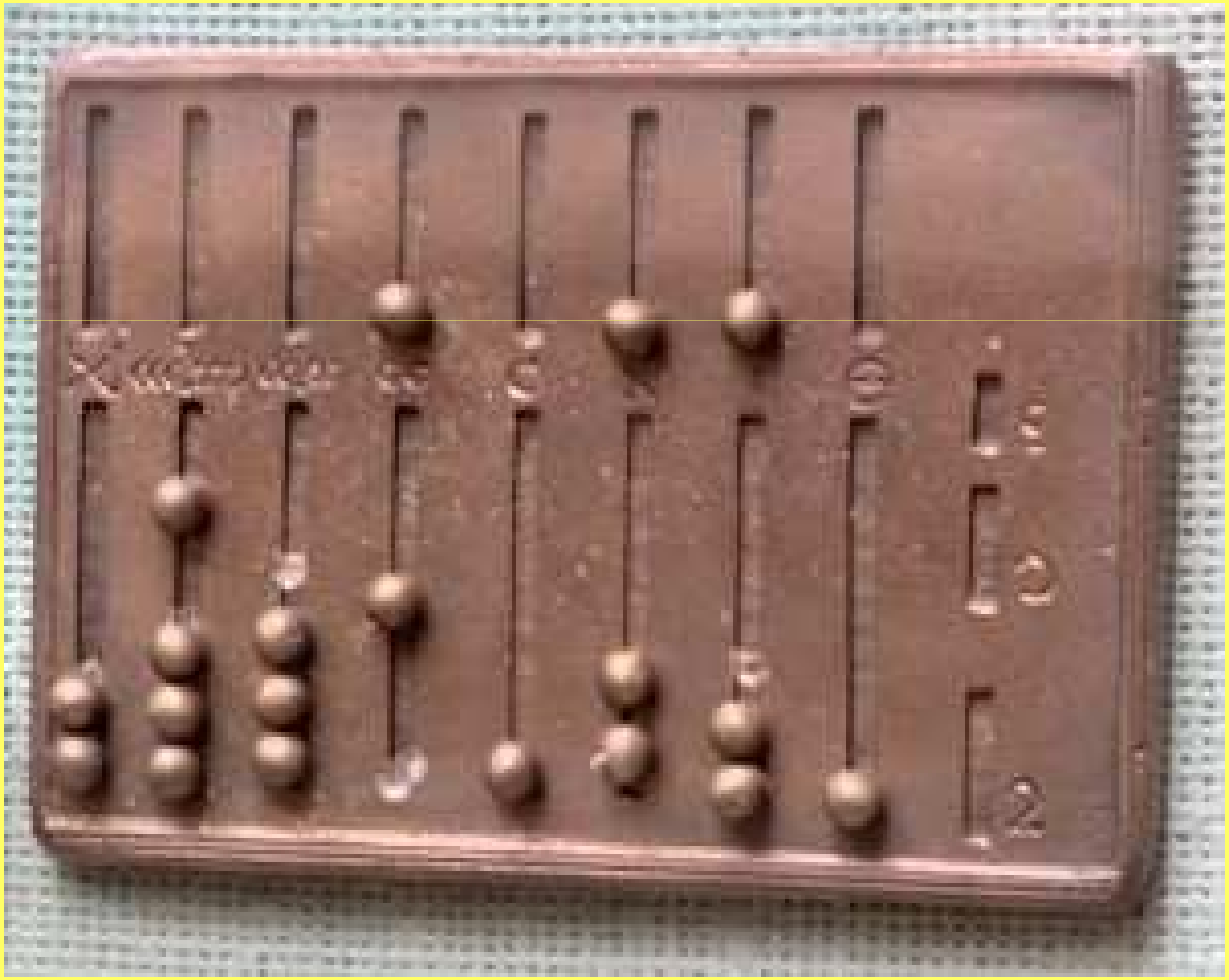
Typowym poglądowym przykładem algorytmu, stosowanym w życiu codziennym jest przepis kuchenny.

Algorytmy zwykle formułowane są w sposób ścisły w oparciu o język matematyki.

UWAGA

Należy zdawać sobie sprawę z różnicy między algorytmem, będącym niezależnym od jego implementacji przepisem, a programem, który może zostać zinterpretowany i wykonany przez komputer.

REKONSTRUKCJA RZYMSKIEGO ABAKUSA Z BRĄZU



CECHY ALGORYTMU

- **Skończoność**. Wykonanie algorytmu zawsze kończy się po skończonej liczbie kroków.
- **Poprawne zdefiniowanie**. Każdy krok algorytmu opisany jest precyzyjnie i jednoznacznie.
- **Efektywne zdefiniowanie.** Operacje algorytmu powinny być jak najprostsze, dające się wykonać w jak najkrótszym możliwym czasie.

Dane wejściowe. Są to wartości znane przed rozpoczęciem działania algorytmu lub dostarczane w czasie jego wykonywania.

Dane wyjściowe – wynik działania algorytmu. Algorytm generuje dane wyjściowe, powiązane w pewien sposób z danymi wejściowymi.

Fazy procesu rozwiązywania problemów

- Sformułowanie problemu (specyfikacja)
- Analiza problemu i znalezienie metody jego rozwiązania, tzn. algorytmu
- Zakodowanie algorytmu (napisanie programu)
- Uruchomienie i przetestowanie programu

Analizujemy problem rozbijając go na mniejszą ilość podproblemów, a następnie określamy czynności jakie należy wykonać, aby zrealizować dany fragment.

Czynności te w algorytmie nazywamy **krokiem**. Krokiem może być operacja prosta (elementarna) lub złożona.

METODY KONSTRUOWANIA ALGORYTMU

- **Metoda zstępująca** (od ogółu do szczegółu) – zadany problem dzielimy na coraz mniejsze logicznie domknięte moduły. W rezultacie uzyskujemy opis pojedynczych czynności w instrukcjach języka.

- **Metoda wstępująca** (od szczegółu do ogółu) – zaczynamy rozwiązanie problemu od skonstruowania obliczeń dla jego najistotniejszych fragmentów. Następnie takie fragmenty łączymy i uzupełniamy o instrukcje wprowadzania danych, wyprowadzania wyników i inne.

Podstawowe paradygmaty tworzenia algorytmów komputerowych

Dziel i zwyciężaj –problem dzielony jest na kilka mniejszych, a te znowu są dzielone, aż ich rozwiązania staną się oczywiste

Programowanie dynamiczne – problem dzielony jest na kilka podproblemów, ranga każdego z nich jest oceniana i wyniki analizy niektórych prostszych zagadnień wykorzystuje się do rozwiązania głównego problemu

Metoda zachłanna – podproblemy nie są analizowane dokładnie, tylko wybierana jest najbardziej obiecująca w danym momencie droga rozwiązania

Programowanie liniowe – rozwiązanie problemu oceniane jest przez pewną funkcję jakości i następnie poszukiwane jest jej minimum

Poszukiwanie i wyliczanie – zbiór danych jest przeszukiwany aż do odnalezienia rozwiązania,

Heurystyka – człowiek na podstawie swojego doświadczenia tworzy algorytm, który działa w najbardziej prawdopodobnych warunkach, rozwiązanie zawsze jest przybliżone.

Najważniejsze techniki stosowania algorytmów komputerowych

Proceduralność – algorytm dzielony jest na szereg podstawowych procedur, wiele algorytmów współdzieli wspólne biblioteki standardowych procedur, z których są one wywoływane w razie potrzeby,

Praca sekwencyjna – wykonywanie kolejnych procedur algorytmu, według kolejności ich wywołań, na raz pracuje tylko jedna procedura,

Praca wielowątkowa – procedury wykonywane są sekwencyjnie, lecz kolejność ich wykonania jest trudna do przewidzenia dla programisty,

Praca równoległa – wiele procedur wykonywanych jest w tym samym czasie, wymieniają się one danymi,

Rekurencja – procedura lub funkcja wywołuje sama siebie, aż do uzyskania wyniku lub błędu,

Obiektowość – procedury i dane łączone są w pewne klasy reprezentujące najważniejsze elementy algorytmu oraz stan wewnętrzny wykonującego je urządzenia,

Algorytm probabilistyczny – algorytm działa poprawnie z bardzo wysokim prawdopodobieństwem, ale wynik nie jest pewny.

ALGORYTM GOTOWANIA JAJKA NA MIĘKKO

Krok 1. Włóż jajko do gotującej się wody.
Krok 2. Zanotuj czas początkowy t_0.
Krok 3. Odczytaj czas aktualny t.
Krok 4. Oblicz $\Delta t = t - t_0$.
Krok 5. Jeśli $\Delta t < 3$ min., to przejdź do kroku 3.
Krok 6. Wyjmij jajko z gotującej się wody. Zakończ algorytm.

ALGORYTM EUKLIDESA

- pierwszy znany algorytm wyznaczania największego wspólnego dzielnika (NWD) dwóch liczb naturalnych

Przykład: Znaleźć NWD liczb 1677 i 744

Jeżeli od większej liczby odejmiemy liczbę mniejszą, wartość największego wspólnego dzielnika nie ulegnie zmianie.

1677 – 744 = 933
933 – 744 = 189
744 – 189 = 555
555 – 189 = 366
366 – 189 = 177
189 – 177 = 12
177 – 12 = 165
165 – 12 = 153
153 – 12 = 141
141 – 12 = 129
129 – 12 = 117
117 – 12 = 105

105 – 12 = 93
93 – 12 = 81
81 – 12 = 69
69 – 12 = 57
57 – 12 = 45
45 – 12 = 33
33 – 12 = 21
21 – 12 = 9
12 – 9 = 3
9 – 3 = 6
6 – 3 = 3
3 – 3 = 0

NWD = 3

ALGORYTM EUKLIDESA

- wersja wykorzystująca operację reszty z dzielenia

1. oblicz *c* jako resztę z dzielenia *a* przez *b*
2. zastąp pozycję *a* liczbą *b*, a pozycję *b* liczbą *c*
3. jeżeli pozycja *b* = 0, to szukane NWD = *a*, w przeciwnym wypadku przejdź do 1

Przykład: Znaleźć NWD liczb 1677 i 744

```
1677  /  744 =   2,  R = 189
 744  /  189 =   3,  R = 177
 189  /  177 =   1,  R =  12
 177  /   12 = 14,  R =   9
  12  /    9 =   1,  R =   3
   9  /    3 =   3,  R =   0
```

NWD = 3

PROBLEM KOMIWOJAŻERA

Komiwojażer wyrusza ze stolicy i ma dokładnie raz odwiedzić *n* miast. Problem polega na znalezieniu najkrótszej drogi.

Liczba tras = *n*!

Liczba miast	**Liczba tras**	**Czas operacji**
4	24	$0{,}25 \times 10^{-9}$ s
16	$2{,}09 \times 10^{13}$	1,4 doby
25	$1{,}55 \times 10^{25}$	5×10^{6} lat
49	$6{,}08 \times 10^{62}$	2×10^{44} lat

SPOSOBY PRZEDSTAWIANIA ALGORYTMÓW

- słowny - na ogół mało dokładny
- lista kroków
- schemat blokowy
- drzewo algorytmu
- pseudo-kod

SPOSOBY PRZEDSTAWIANIA ALGORYTMÓW
opis słowny

Przykład:

Obliczyć wartość funkcji: $f(x)=\frac{x}{|x|}$

Rozwiązanie:

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

1. Dla liczb ujemnych |x| = -x, a więc $f(x)=-1$

2. Dla liczb dodatnich |x| = x, a więc $f(x)=1$

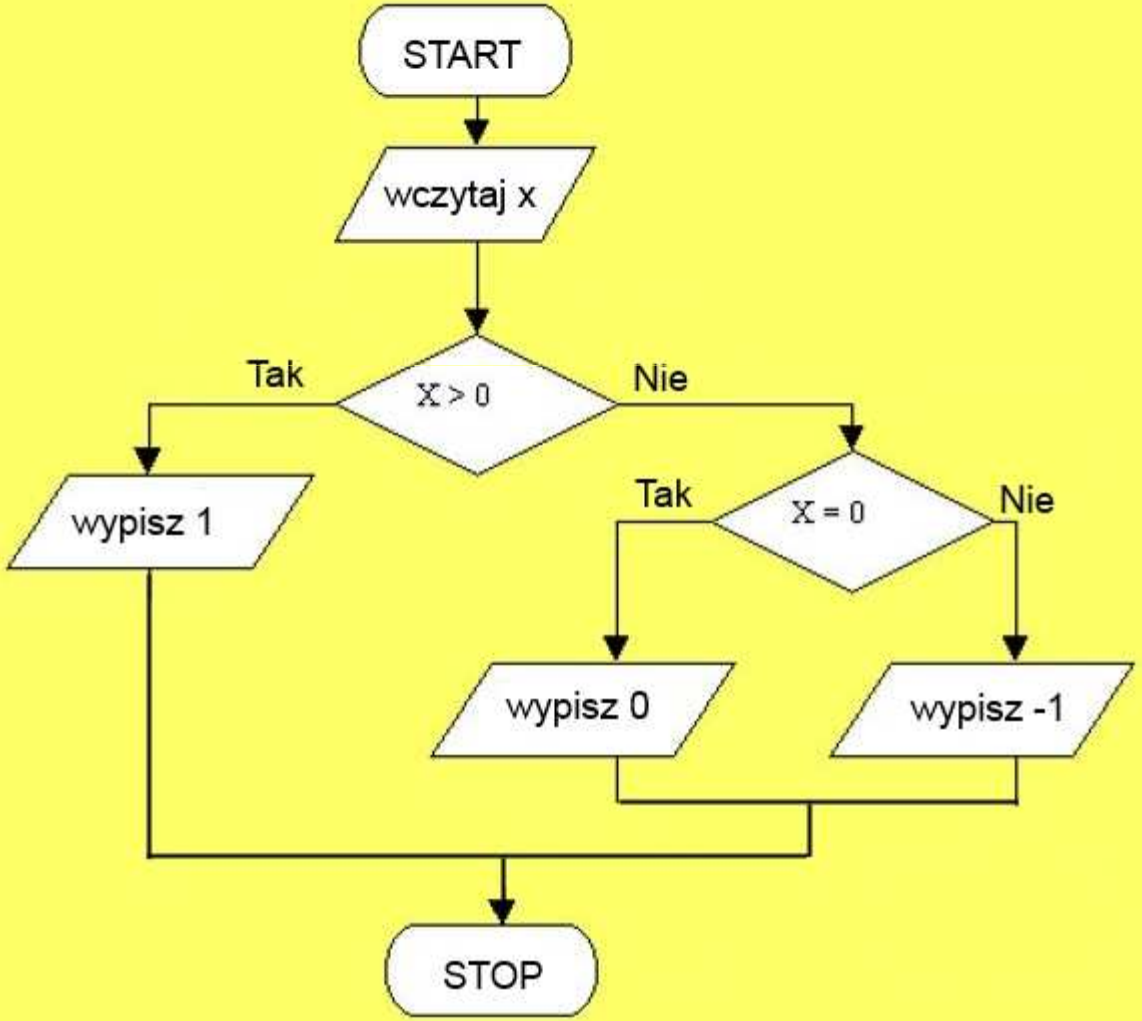
3. Jeśli x = 0, to $f(x)=0$

SPOSOBY PRZEDSTAWIANIA ALGORYTMÓW
lista kroków

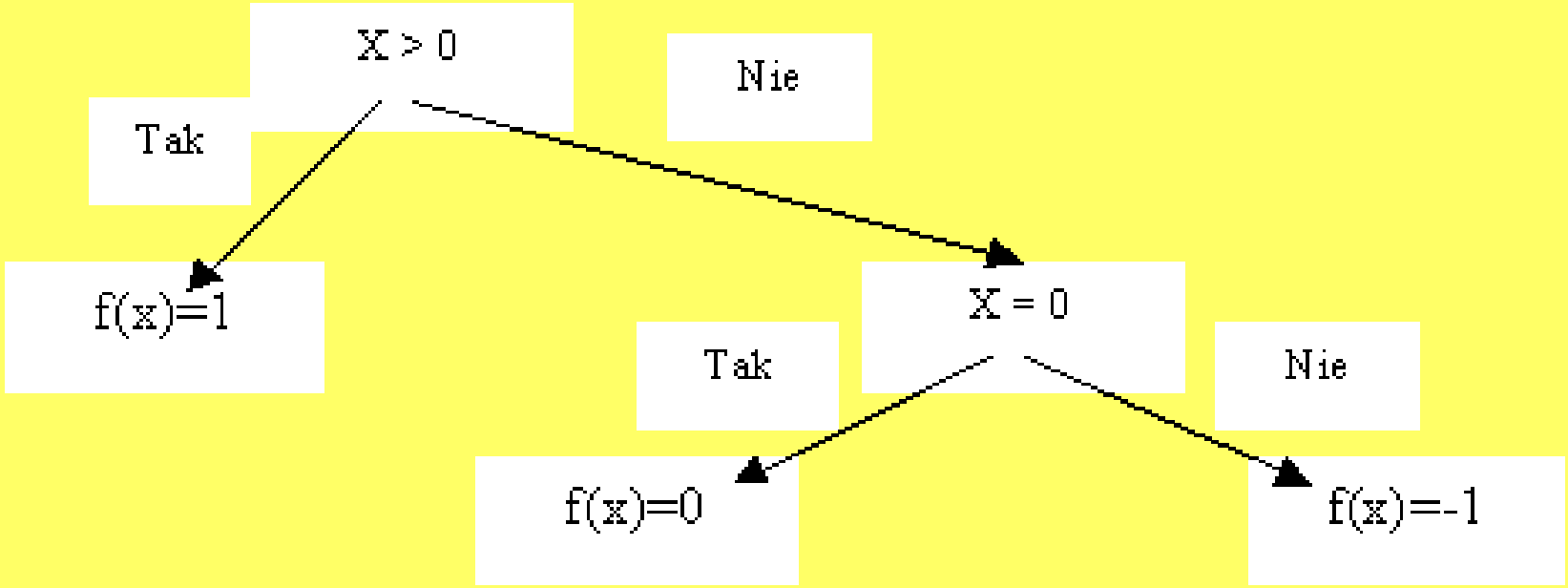
Dane: Dowolna liczba rzeczywista x.
Wynik: Wartość funkcji f(x)

Krok 1. Wczytaj wartość danej x.
Krok 2. Jeśli x > 0, to f(x)=1. Zakończ algorytm.
Krok 3. Jeśli x = 0, to f(x)=0. Zakończ algorytm.
Krok 4. Jeśli x < 0, to f(x)=-1. Zakończ algorytm.

SPOSOBY PRZEDSTAWIANIA ALGORYTMÓW
schemat blokowy



SPOSOBY PRZEDSTAWIANIA ALGORYTMÓW
drzewo algorytmu



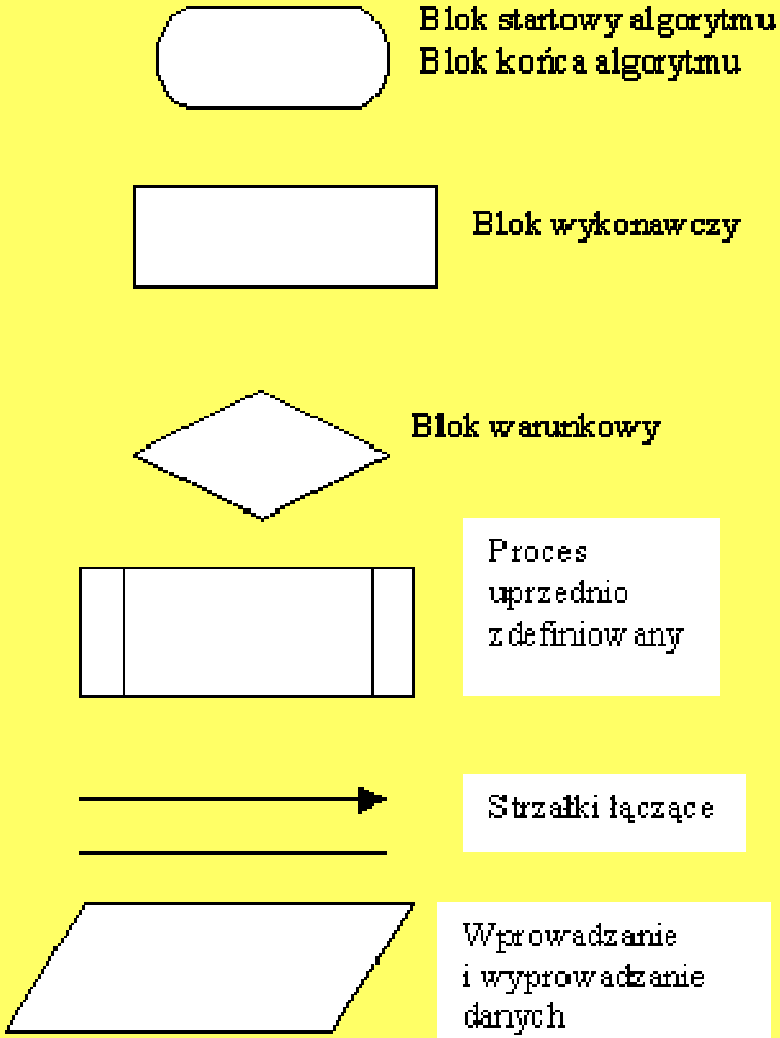
ZASADY RYSOWANIA SCHEMATÓW BLOKOWYCH

Aby przekazać dany algorytm człowiekowi lub maszynie, należy ten algorytm zapisać. Od wieków algorytmy zapisywano w językach naturalnych. Komputery muszą mieć jednoznacznie opisane czynności do wykonania. Powszechnie stosowanym obecnie językiem opisu algorytmów jest tzw. schemat blokowy.
Podstawową zasadą obowiązującą przy budowie schematów blokowych, jest łączenie strzałek przedstawiających kolejność wykonywanych czynności z elementami ilustrującymi te czynności.

Reguły rysowania schematów blokowych:

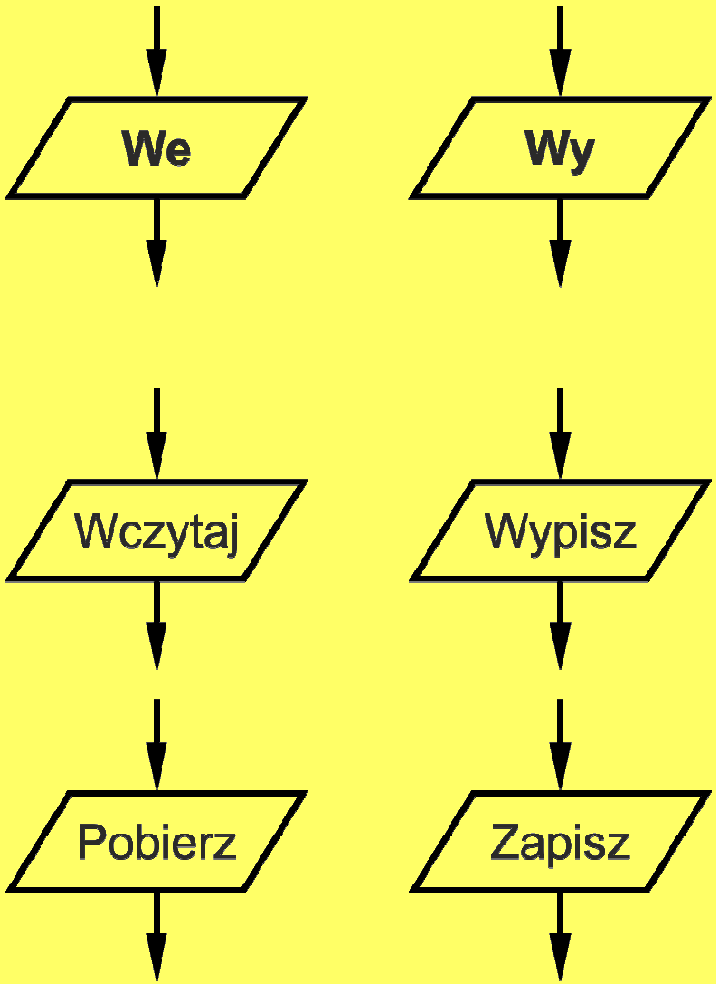
- Po zbudowaniu schematu blokowego nie powinno być takich strzałek, które znikąd wychodzą, lub donikąd nie dochodzą.
- Każdy schemat blokowy musi mieć tylko jeden element startowy oraz jeden element końca algorytmu.

ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH



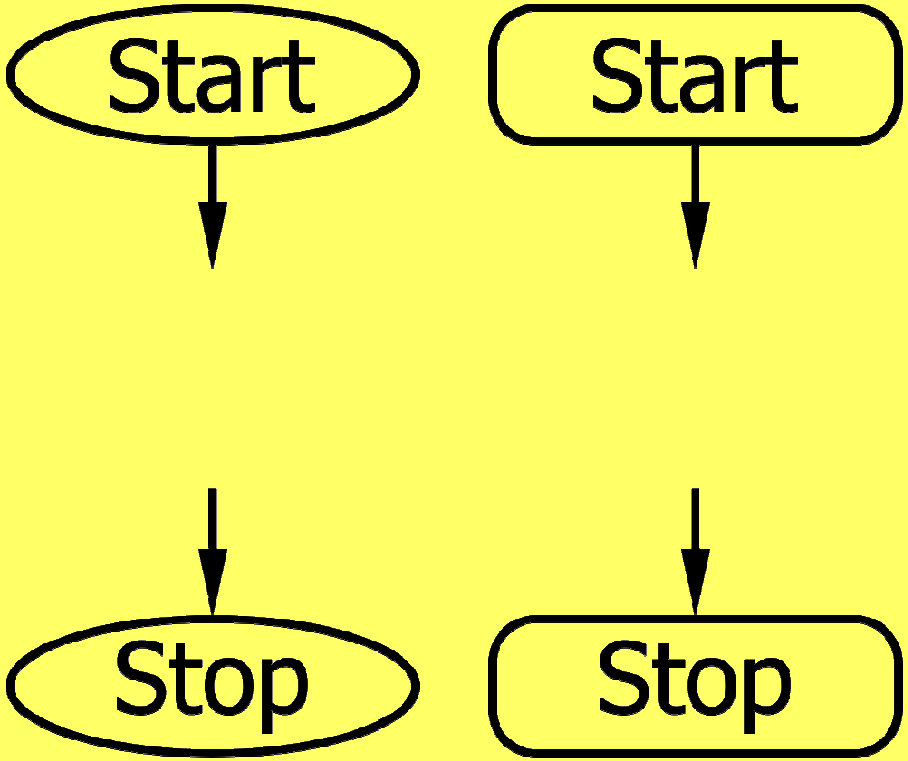
ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

Blok wprowadzania i wyprowadzania informacji (Wejścia/Wyjścia)



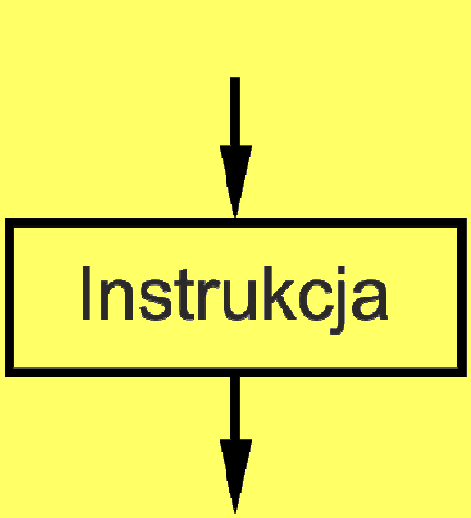
ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

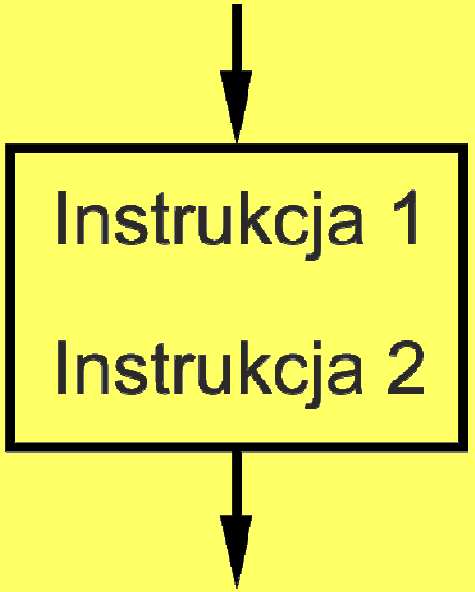
Bloki wskazujące na początek i koniec algorytmu



ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

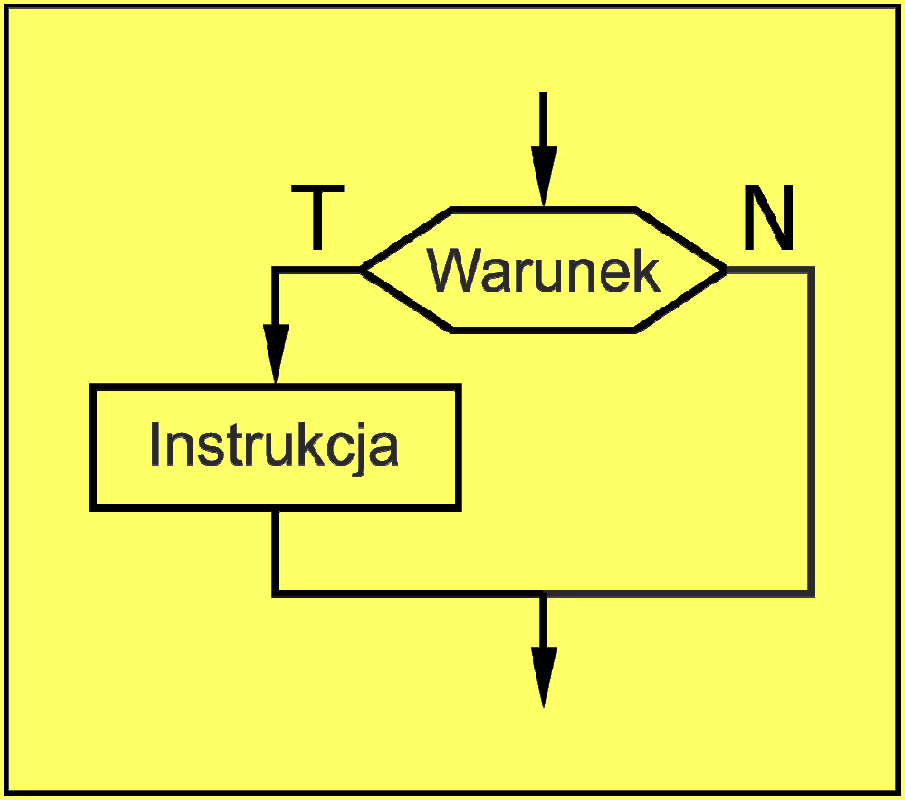
Blok instrukcji – kilka instrukcji zapisana jedna pod drugą





ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

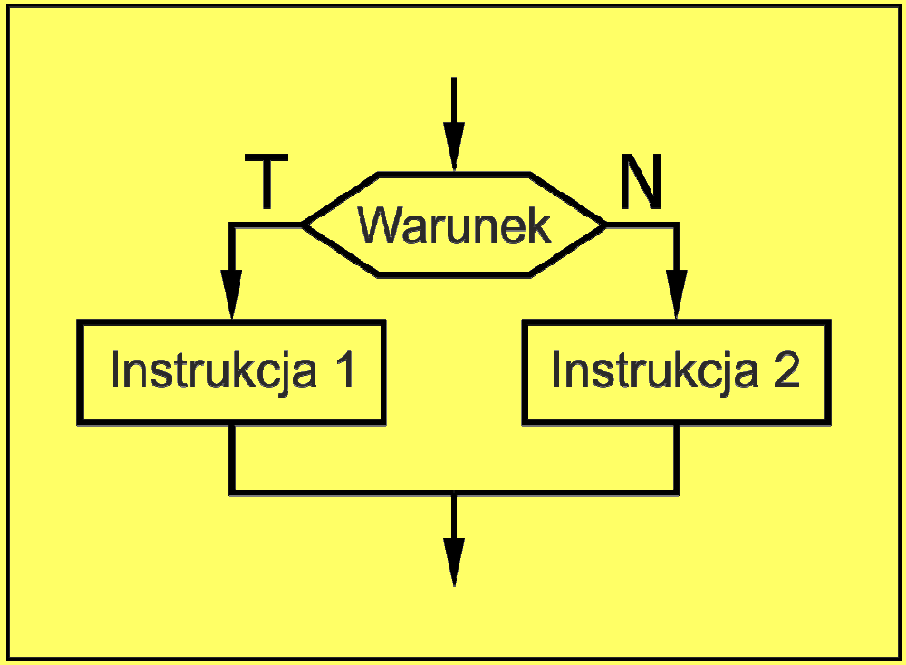
Zdanie decyzyjne – zdanie zawierające decyzję podejmowaną w algorytmie: jeśli (if) ... to



Struktura prosta

ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

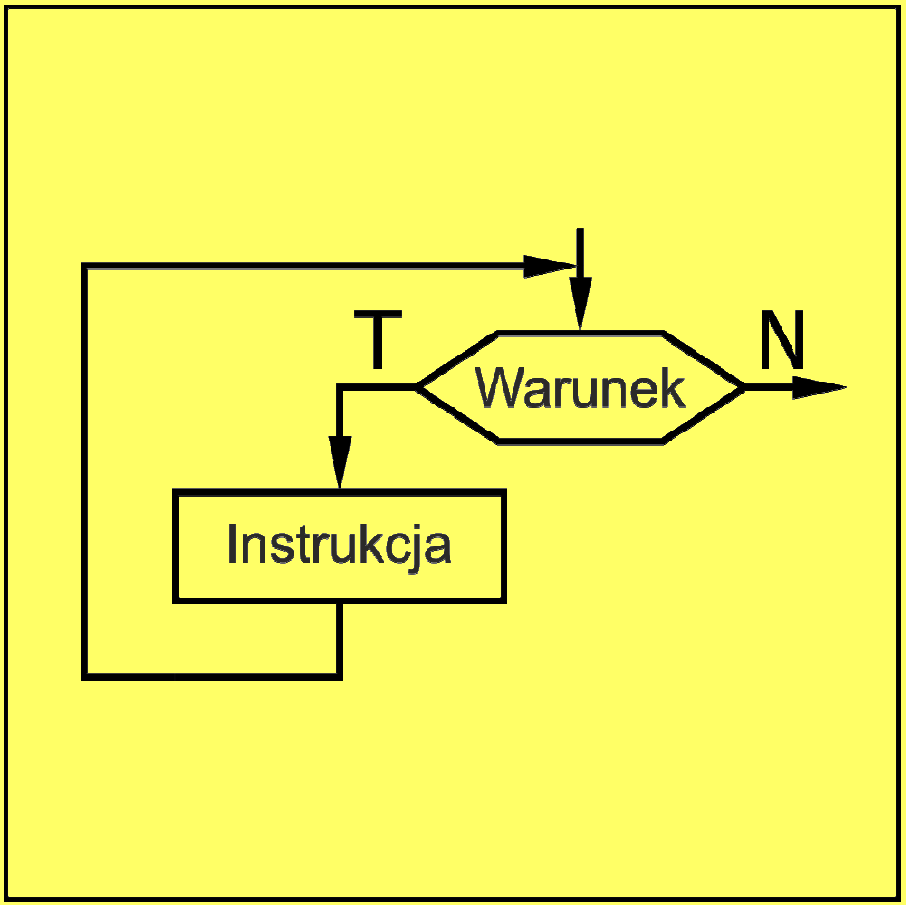
Zdanie decyzyjne – zdanie zawierające decyzję podejmowaną w algorytmie: jeśli (if) ... to



Struktura z alternatywą

ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

Zdanie iteracyjne – zdanie zawierające decyzję podejmowaną w pętli: dopóki (while) ... to

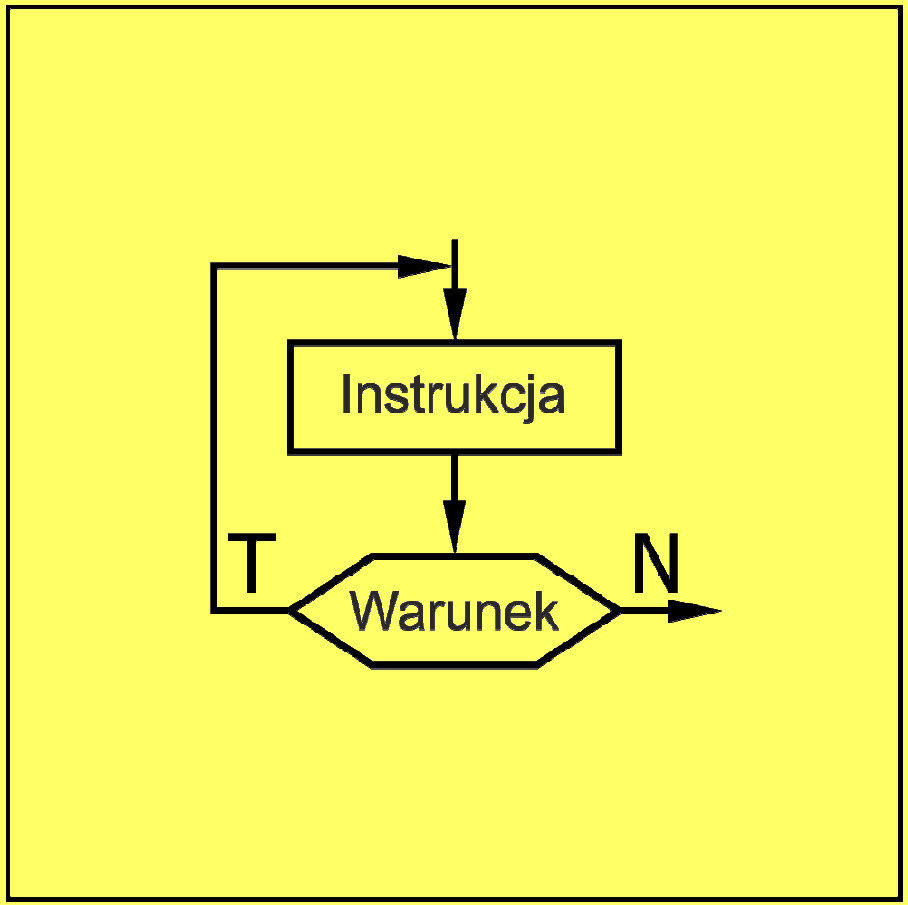


Uwaga: sprawdzenie warunku odbywa się **przed** wykonaniem instrukcji1

Instrukcja 1 może nigdy nie zostać wykonana

ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

Zdanie iteracyjne – zdanie zawierające decyzję podejmowaną w pętli: wykonuj ... dopóki (do ... while)

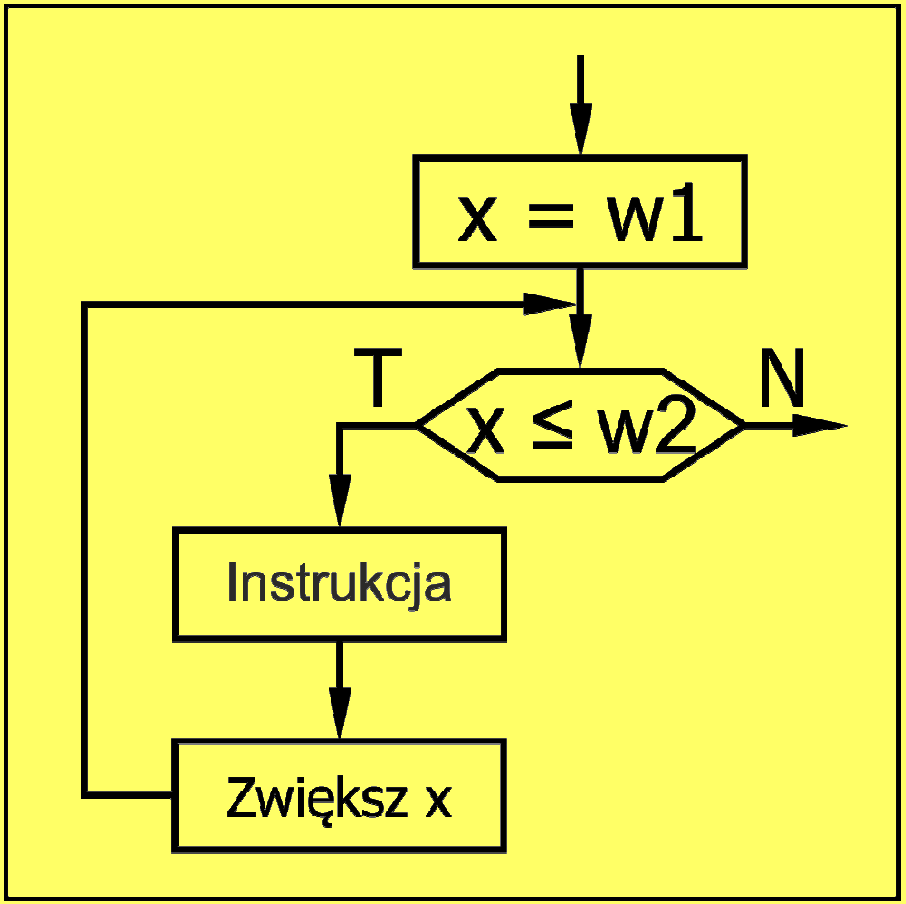


Uwaga: sprawdzenie warunku odbywa się **po** wykonaniu instrukcji1

Instrukcja 1 zawsze wykonywana jest przynajmniej raz.

ELEMENTY STOSOWANE PRZY RYSOWANIU SCHEMATÓW BLOKOWYCH

Zdanie iteracyjne – zdanie zawierające decyzję podejmowaną w pętli: dla (for)



Uwaga: sprawdzenie warunku odbywa się **przed** wykonaniem instrukcji1

Instrukcja 1 może nigdy nie zostać wykonana

OPERATORY STOSOWANE W SCHEMATACH BLOKOWYCH

Operator – w programowaniu konstrukcja językowa jedno-, bądź wieloargumentowa zwracająca wartość

- Logiczne
- Arytmetyczne
- Przypisania
- Porównania
- Inkrementacji i dekrementacji

OPERATORY LOGICZNE

Operator logiczny – operator działający na argumentach reprezentujących wartości logiczne, zwraca w wyniku wartość logiczną.

Podział operatorów logicznych:

jednoargumentowe:

- negacja

dwuargumentowe

- koniunkcja
- alternatywa
- alternatywa wykluczająca
- implikacja
- ekwiwalencja.

OPERATORY LOGICZNE

Antynomia kłamcy

Gdy ktoś mówi "Ja kłamię.", to kłamie, czy mówi prawdę? Jeśli założymy, że kłamie, to mówi prawdę, jeśli założymy, że mówi prawdę, to kłamie.

***

Antynomia cyrulika sewilskiego

Cyrulik goli wszystkich tych i tylko tych, co nie golą się sami. Czy sam siebie goli?

***

Antynomia satrapy

Pewien władca postanowił, że przy wjeździe do jego państwa straż graniczna będzie pytała każdego gościa o cel wizyty, następnie przydzielona mu będzie ochrona służb specjalnych, czuwających by nic mu się nie stało i wyjechał zadowolony po zrealizowaniu celu. Jeśli jednak okaże się, że przyjezdny swojego celu nie zrealizował, powinien zostać zgładzony. Aby uniknąć nadużyć, wprowadzono surowe kary dla agentów, którzy zabiją kogoś, kto swój cel wizyty zrealizował. Co powinni zrobić agenci z przybyszem, który oświadcza, że przyjechał do ich kraju po to, aby go uśmiercili?

Źródło: http://www.matematyka.wroc.pl/book/logika

OPERATORY LOGICZNE

Negacja (NOT)	
p	-p
0	1
1	0

Koniunkcja (AND)		
p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Alternatywa (OR)		
p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

Alternatywa wykl. (XOR)		
p	q	$p \veebar q$
0	0	0
0	1	1
1	0	1
1	1	0

Implikacja ( )		
p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

Ekwiwalencja ( )		
p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

gdzie:

1 – zdanie prawdziwe

0 – fałszywe

OPERATORY LOGICZNE

Negacja (NOT)	
p	-p
0	1
1	0

p – jest pogodnie

gdzie:

1 – zdanie prawdziwe

0 – fałszywe

OPERATORY LOGICZNE

Koniunkcja (AND)		
p	q	p∧q
0	0	0
0	1	0
1	0	0
1	1	1

„W tym roku byłem w Australii i Afryce”

p – W tym roku byłem w Australii

q – W tym roku byłem w Afryce

gdzie:

1 – zdanie prawdziwe

0 – fałszywe

$p \wedge q$

0 0 0

0 0 1

1 0 0

1 1 1

OPERATORY LOGICZNE

Alternatywa (OR)		
p	q	p∨q
0	0	0
0	1	1
1	0	1
1	1	1

„Jutro będzie padał śnieg lub deszcz”

p – Jutro będzie padał śnieg
q – Jutro będzie padał deszcz

gdzie:
1 – zdanie prawdziwe
0 – fałszywe

p ∨ q

0 0 0
0 1 1
1 1 0
1 1 1

OPERATORY LOGICZNE

Alternatywa wykl. (XOR)		
p	q	p$\veebar$q
0	0	0
0	1	1
1	0	1
1	1	0

„Albo jutro będzie padał śnieg albo deszcz”

p – Jutro będzie padał śnieg
q – Jutro będzie padał deszcz

gdzie:
1 – zdanie prawdziwe
0 – fałszywe

p $\veebar$ q

0 0 0
0 1 1
1 1 0
1 0 1

OPERATORY LOGICZNE

Implikacja ( )		
p	q	p→q
0	0	1
0	1	1
1	0	0
1	1	1

„Jeśli jutro będzie ciepło, to pójdziemy na basen”

p – Jutro będzie ciepło

q – Jutro pójdziemy na basen

gdzie:

1 – zdanie prawdziwe

0 – fałszywe

p → q

0 1 0

0 1 1

1 0 0

1 1 1

OPERATORY LOGICZNE

Ekwiwalencja ( )		
p	q	p↔q
0	0	1
0	1	0
1	0	0
1	1	1

„Zaliczenie przedmiotu przez studenta oznacza uzyskanie pozytywnej oceny końcowej”

p – Student uzyskał pozytywną ocenę końcową
q – Student zaliczył przedmiot

gdzie:
1 – zdanie prawdziwe
0 – fałszywe

p ↔ q

0 1 0
0 0 1
1 0 0
1 1 1

Sprawdzanie poprawności wnioskowań

Dane: Zdanie 1, Zdanie 2, Zdanie 3, …
Wniosek: Zdanie zawierające wniosek
Schemat postępowania:

a) sformułowanie zdań prostych: p,q,r i wniosku s
b) zapisanie Zdań 1-3 przy użyciu zdań prostych i spójników logicznych w postaci:

$$\frac{\text{Dane}}{\text{Wniosek}} \qquad \frac{\text{Zdanie 1, Zdanie 2, Zdanie 3, …}}{s}$$

c) założenie, że z prawdziwych danych wynika błędny wniosek

$$\text{Dane} \rightarrow \text{Wniosek}$$

$$\begin{array}{ccccc} & & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{array}$$

$$[Z1] \wedge [Z2] \wedge [Z3] \ldots \rightarrow s$$

$$\frac{\begin{array}{c}1\\ \text{Dane}\end{array}}{\text{Wniosek}} \qquad \frac{\begin{array}{c}1\ \ 1\ \ 1\\ \text{Z1, Z2, Z3…}\end{array}}{\text{Wniosek}}$$

$$\begin{array}{c|c} & \\ 0 & 0 \end{array}$$

p → q
0 1 0
0 1 1
1 0 0
1 1 1

Jeśli przesłanki (dane) są prawdziwe, a wniosek błędny, to wnioskowanie nie jest poprawne

Sprawdzanie poprawności wnioskowań

„Jeśli Wacek dostał wypłatę to jest w barze lub u Zdziśka. Wacka nie ma w barze. Zatem Wacek nie dostał wypłaty”

p – Wacek dostał wypłatę
q – Wacek jest w barze
r – Wacek jest u Zdziśka

$$\frac{p \rightarrow (q \vee r),\ -q}{-p}$$

Badając, czy wniosek wynika z przesłanek, sprawdzamy, czy możliwa jest sytuacja aby wszystkie przesłanki były prawdziwe, a jednocześnie wniosek fałszywy:

1	1	1	10	1 1	0 1 1	10
$p \rightarrow (q \vee r)$,	-q	$p \rightarrow (q \vee r)$,	-q	$p \rightarrow (q \vee r)$,	-q	
-p		-p		-p		
0		01		01		

Przesłanki są prawdziwe, a wniosek błędny – wnioskowanie nie jest poprawne

Sprawdzanie poprawności wnioskowań

„Jeśli Wacek dostał wypłatę to jest w barze lub u Zdziśka. Wacka nie ma w barze. Zatem Wacek nie dostał wypłaty”

p – Wacek dostał wypłatę
q – Wacek jest w barze
r – Wacek jest u Zdziśka

$$\frac{p \rightarrow (q \vee r),\ -q}{-p}$$

Badając, czy wniosek wynika z przesłanek, sprawdzamy, czy możliwa jest sytuacja aby wszystkie przesłanki były prawdziwe, a jednocześnie wniosek fałszywy:

1	1	1	10	1 1	0 1 1	10
$p \rightarrow (q \vee r)$,	-q	$p \rightarrow (q \vee r)$,	-q	$p \rightarrow (q \vee r)$,	-q	
-p		-p		-p		
0		01		01		

Przesłanki są prawdziwe, a wniosek błędny – wnioskowanie nie jest poprawne

Sprawdzanie poprawności wnioskowań

„Jeśli na imprezie był Zdzisiek i Wacek, to impreza się nie udała. Impreza się udała. Zatem na imprezie nie było Zdziśka lub Wacka ”

p – Zdzisiek był na imprezie
q – Wacek był na imprezie
r – impreza się udała

$$\frac{(p \wedge q) \rightarrow -r,\ r}{-p \vee -q}$$

1 1
$(p \wedge q) \rightarrow -r,\ r$
$-p \vee -q$
0

1 01 1
$(p \wedge q) \rightarrow -r,\ r$
$-p \vee -q$
0

0 1 01 1
$(p \wedge q) \rightarrow -r,\ r$
$-p \vee -q$
0 sprzeczność

0 1 01 1
$(p \wedge q) \rightarrow -r,\ r$
$-p \vee -q$
0 0 0

0 1 01 1
$(p \wedge q) \rightarrow -r,\ r$
$-p \vee -q$
01 0 01

1 0 1 1 01 1
$(p \wedge q) \rightarrow -r,\ r$
$-p \vee -q$
01 0 01

Wnioskowanie O.K.

Sprawdzanie poprawności wnioskowań

„Jeśli Kaśka zdąży na autobus, to przyjedzie, lub gdyby nie zdążyła, to przełożymy nasze spotkanie”. Kaśka nie zdążyła na autobus. Przełożyliśmy spotkanie”

p – Kaśka zdążyła na autobus
q – Kaśka przyjechała
r – Przekładamy spotkanie

$$\frac{(p \rightarrow q) \vee (\neg p \rightarrow r),\ \neg p}{r}$$

1 1	0 1 10 0 0 10	0 1 1 10 0 0 10
$(p \rightarrow q) \vee (\neg p \rightarrow r),\ \neg p$	$(p \rightarrow q) \vee (\neg p \rightarrow r),\ \neg p$	$(p \rightarrow q) \vee (\neg p \rightarrow r),\ \neg p$
r	r	r
0	0	0

Przesłanki są prawdziwe, a wniosek błędny – wnioskowanie nie jest poprawne. Kaśka wprawdzie nie zdążyła na autobus, ale przecież mogła przyjechać innym środkiem lokomocji, więc przekładanie spotkania było niecelowe ☺

Sprawdzanie poprawności wnioskowań

„Jeśli Lolek jest agentem, to agentem jest też Bolek, zaś nie jest nim Tola. Jeśli Bolek jest agentem, to jest nim też Lolek lub Tola. Jeśli jednak Tola nie jest agentem, to jest nim Lolek, a nie jest Bolek. Tak więc to Tola jest agentem”.

L – Lolek jest agentem
B – Bolek jest agentem
T – Tola jest agentem

$$\frac{L \overset{1}{\rightarrow} (B \wedge \text{-}T),\ B \overset{1}{\rightarrow} (L \vee T),\ \text{-}T \overset{1}{\rightarrow} (L \wedge \text{-}B)}{T}$$

0

$$\frac{L \overset{1}{\rightarrow} (B \overset{10}{\wedge} \text{-}T),\ B \overset{1}{\rightarrow} (L \overset{0}{\vee} T),\ \text{-}T \overset{10}{\rightarrow} 1 \rightarrow (L \wedge \text{-}B)}{T}$$

0

Sprawdzanie poprawności wnioskowań

$$\overset{1}{L} \rightarrow (B \wedge \overset{10}{-T}),\ \overset{1}{B} \rightarrow (L \vee \overset{0}{T}),\ \overset{10}{-T} \rightarrow (\overset{1}{L} \wedge -B)$$

T

0

$$\overset{1}{L} \rightarrow (B \wedge \overset{10}{-T}),\ \overset{1}{B} \rightarrow (L \vee \overset{0}{T}),\ \overset{10}{-T} \rightarrow (\overset{1}{L} \wedge -B)$$

T

0

sprzeczność

$$1\ 1\quad 0\ 0\ 10\quad 0\ 1\quad 1\ 1\ 0\quad 10\ 1\quad 1\ 1\ 10$$
$$L \rightarrow (B \wedge -T),\ B \rightarrow (L \vee T),\ -T \rightarrow (L \wedge -B)$$

T

0

Sprzeczność pokazuje, iż nie jest możliwa sytuacja, aby przesłanki były prawdziwe, a wniosek fałszywy. Wnioskowanie jest więc poprawne – Tola zawsze była jakaś taka podejrzana ☺

Sprawdzanie poprawności wnioskowań

Przykład zdania złożonego.
„Jeśli wybory wygra lewica to znów wzrosną podatki i spadnie tempo rozwoju gospodarczego, ale jeśli wygra prawica lub tak zwana centroprawica, to powstanie bardzo słaby rząd i albo będziemy przez cztery lata świadkami gorszących skandali, albo za rok będą nowe wybory”.
Czy zdanie jest prawdziwe ???

$$[p \rightarrow (q \wedge r)] \wedge \{(s \vee t) \rightarrow [u \wedge (w \vee z)]\}$$

Literatura:
Krzysztof Wieczorek, „Logika dla opornych”, 2002

OPERATORY ARYTMETYCZNE

Operator podstawienia	=
Operator potęgowania	^
Operator mnożenia	*
Operator dzielenia	/
Operator dzielenia całkowitego	\
Operator modulo	Mod
Operator dodawania	+
Operator odejmowania	-

OPERATORY PRZYPISANIA

Operator przypisania przypisuje wartość prawego argumentu lewemu

operator 1 = operator 2

a = 5

OPERATORY PORÓWNANIA

Operator porównania porównuje ze sobą dwa argumenty i zwraca jedynkę jeżeli warunek jest spełniony lub zero jeżeli nie jest.

Op1 > Op2	- większe
Op1 < Op2	- mniejsze
Op1 >= Op2	- większe lub równe
Op1 <= Op2	- mniejsze lub równe
Op1 = Op2	- równe
Op1 <> Op2	- różne

OPERATORY INKREMENTACJI I DEKREMENTACJI

Operatory inkrementacji zwiększa, a dekrementacji zmniejsza argument o jeden.

Op1 = Op1 + 1 - inkrementacja

Op1 = Op1 - 1 - dekrementacja

Ciekawostka – w języku C++ inkrementacji oraz dekrementacji można dokonać znacznie szybciej – pisząc:

Op1++

Op1--

Pseudokod

Jest to taki sposób zapisu algorytmu, który, zachowując strukturę charakterystyczną dla kodu zapisanego w danym języku programowania, rezygnuje ze ścisłych reguł składniowych tego języka na rzecz prostoty i czytelności. Kroki algorytmu opisywane są często za pomocą formuł matematycznych lub zdań w języku naturalnym.

W chwili obecnej brak jest ogólnie akceptowanego standardu zapisu pseudokodu. Najczęściej stosowana jest składnia oparta na składni istniejących języków programowania (Basic, Pascal, C).

Schemat blokowy można uznać za graficzny wariant pseudokodu.

Elementy składniowe pseudokodu

Elementami składniowymi są zdania. Rodzaje zdań:

- **Zdania proste (instrukcja)** – określa czynność do wykonania
  (np. przypisz średniej wartość zero)

- **Zdania decyzyjne jeśli (if)** – zdanie zawierające decyzję podejmowaną w algorytmie

  – Struktury prostej
  jeśli warunek **to**
  instrukcja
  (np. jeśli średnia jest mniejsza niż 3.0 to
  wstaw 2.0)

Elementy składniowe pseudokodu

- **Zdanie iteracyjne dopóki (while)** – określa sytuacje, w której pewne czynność należy powtarzać dopóki warunek jest spełniony (prawdziwy)

  dopóki warunek **wykonuj**
  instrukcje

  (np. **dopóki** procent opanowanej wiedzy studenta jest niższy od 50 % **wykonuj**
  stawianie oceny niedostatecznej)

Elementy składniowe pseudokodu

- **Zdania decyzyjne jeśli (if)** – zdanie zawierające decyzję podejmowaną w algorytmie

  – Struktury z alternatywą

  jeśli warunek **to**
  instrukcja1
  w przeciwnym przypadku
  instrukcja2

  (np. **jeśli** następny dzień to niedziela **to**
  wyłącz budzik
  w przeciwnym przypadku
  nastaw go na 6.00)

Elementy składniowe pseudokodu

- **Zdanie iteracyjne wykonuj … dopóki (do … while)** – określa sytuacje, w której pewne czynność należy powtarzać w zależności od spełnienia określonego warunku

wykonuj
instrukcję
dopóki warunek

(np. **wykonuj**
wczytaj wartość liczby,
wykonaj operacje na liczbie
dopóki nie wczytano liczbę równą zero)

Elementy składniowe pseudokodu

- **Zdanie iteracyjne dla (for)** – określa sytuacje, w której pewne czynność należy powtarzać określoną ilość razy

  dla warunki **wykonuj**
  instrukcje

  (np. **dla** x=1,2,…,100 **wykonuj**
  wypisz wartość x)

Przykłady algorytmów zapisanych w pseudokodzie

1. Oblicz średnią z ciągu liczb

   Dane: ciąg 100 liczb rzeczywistych

   Wynik: średnia arytmetyczna

   Wczytaj ciąg liczb

   oblicz średnią arytmetyczną

   wydrukuj wynik

Przykłady algorytmów zapisanych w pseudokodzie

1. Oblicz średnią z ciągu liczb (algorytm dokładny)

Dane: ciąg 100 kolejnych liczb naturalnych

Wynik: S - średnia arytmetyczna

```
suma = 0     {przypisz zmiennej pomocniczej suma wartość 0}
i = 1        {przypisz zmiennej pomocniczej i wartość 1}
powtarzaj
    suma = suma + i
    i = i + 1
aż i > 100   {oblicz średnią arytmetyczną. Każdą kolejną wartość ciągu
              dodaj do zmiennej suma a następnie zwiększ i o 1}
S = suma / 100 {podziel wartość zmiennej suma przez długość ciągu}
Pisz (S)     {wypisz wynik}
```

ALGORYTM HORNERA 3 STOPNIA

Aby obliczyć wartość wielomianu 3 stopnia zazwyczaj korzysta się z algorytmu, w którym wykonuje się 6 mnożeń i 3 dodawania).

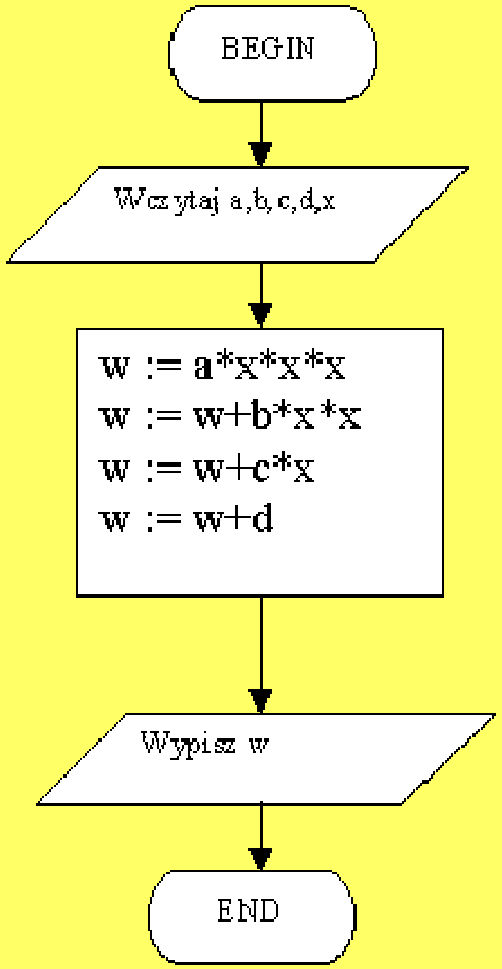
$$w(x) = ax^3 + bx^2 + cx + d = axxx + bxx + cx + d$$

Aby zmniejszyć liczbę działań, wielomian 3 stopnia można przekształcić do następującej postaci:

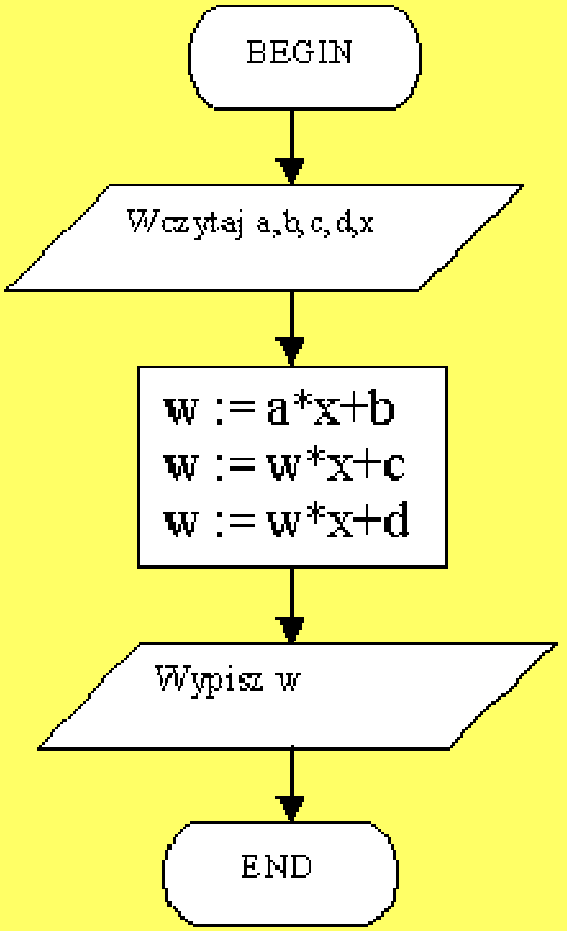
$$w(x) = ax^3 + bx^2 + cx + d = (ax^2 + bx + c)x + d = ((ax + b)x + c)x + d$$

ALGORYTM HORNERA 3 STOPNIA

$$w(x) = ax^3 + bx^2 + cx + d = axxx + bxx + cx + d$$



$$w(x) = ax^3 + bx^2 + cx + d = (ax^2 + bx + c)x + d = ((ax + b)x + c)x + d$$



ALGORYTM HORNERA 3 STOPNIA

Niezależnie od rodzaju urządzenia, na którym wykonywane są obliczenia, najistotniejszą rzeczą jest metoda, zastosowana do tych obliczeń, co ma szczególne znaczenie, gdy stopień wielomianu jest duży.

Metoda zwykła:
Liczba mnożeń = 0,5 n (n + 1)
gdzie n jest stopniem wielomianu

Metoda Hornera:
Liczba mnożeń = n

Stopień wielomianu	Liczba mnożeń metodą zwykłą	Liczba mnożeń metodą Hornera
2	3	2
3	6	3
5	15	5
10	55	10
20	210	20

KONIEC