Zespół Szkół nr 9
Im. Romualda Traugutta
Ul. Jedności 9
75 – 401 Koszalin

ALGORYTMIKA

Katarzyna Pinczak
Klasa II LB

Koszalin 2006

POJĘCIA

CO NAZYWAMY ALGORYTMIKĄ??

podstawowy dział informatyki poświęcony poszukiwaniom, konstruowaniu i badaniom algorytmów, zwłaszcza w kontekście ich przydatności do rozwiązywania problemów za pomocą komputerów. Autorem nazwy dziedziny jest D. Haral (Spirit of Computing).

CO TO JEST ALGORYTM??

- i*nform*. dokładny przepis wykonania w określonym porządku skończonej liczby operacji, pozwalający na rozwiązanie każdego zadania danego typu,

- *mat*. reguła przekształcania wyrażeń matematycznych poprzez powtarzanie tych samych działań na kolejno otrzymywanych wynikach działań poprzednich.

Nazwa algorytm wywodzi się od nazwiska perskiego matematyka ***Muhamed ibn Musy al-Chorezmi*** *(z Chorezmu), który ok. 820 r n.e. opisał pozycyjny system kodowania dziesiętnego liczb i sztukę liczenia w tym systemie. W XII w. Europie przetłumaczono jego książkę i rozpoczęto wykonywanie obliczeń metodą "pisemną".*

ALGORYTM ALGEBRAICZNY

przepis (zbiór instrukcji przekształcania) przetwarzania tzw. danych wejściowych krok po kroku według wskazanej instrukcji (reguły) przetwarzania w celu uzyskania wyników przetwarzania, tzw. danych wyjściowych. Dane, które uzyskuje się w kolejnych krokach przetwarzania nazywane są danymi pośrednimi.

ALGORYTM NIEALGEBRAICZNY

ciąg prostych zdań lub pytań uporządkowanych w logicznej kolejności (od najbardziej ogólnych do najbardziej szczegółowych) w taki sposób aby trzeba było czytać tylko te zdania, które odnoszą się do danego przypadku.

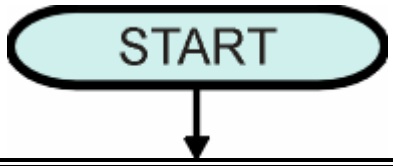
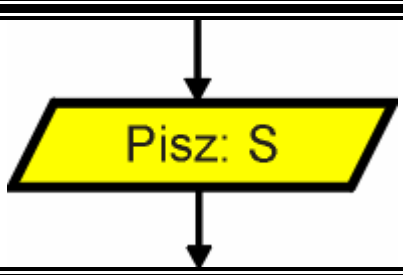
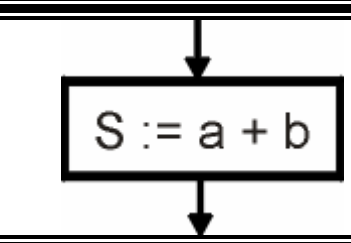
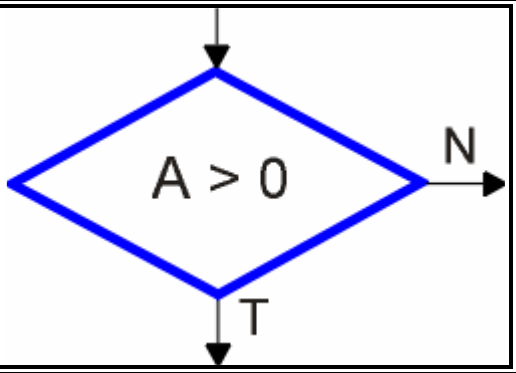
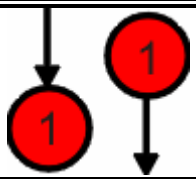
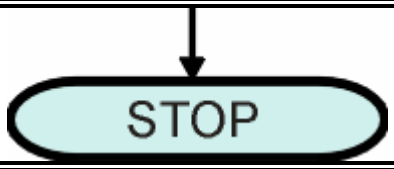
JAK ZAPISUJEMY ALGORYTMY?

Algorytmy można przedstawiać stosując:

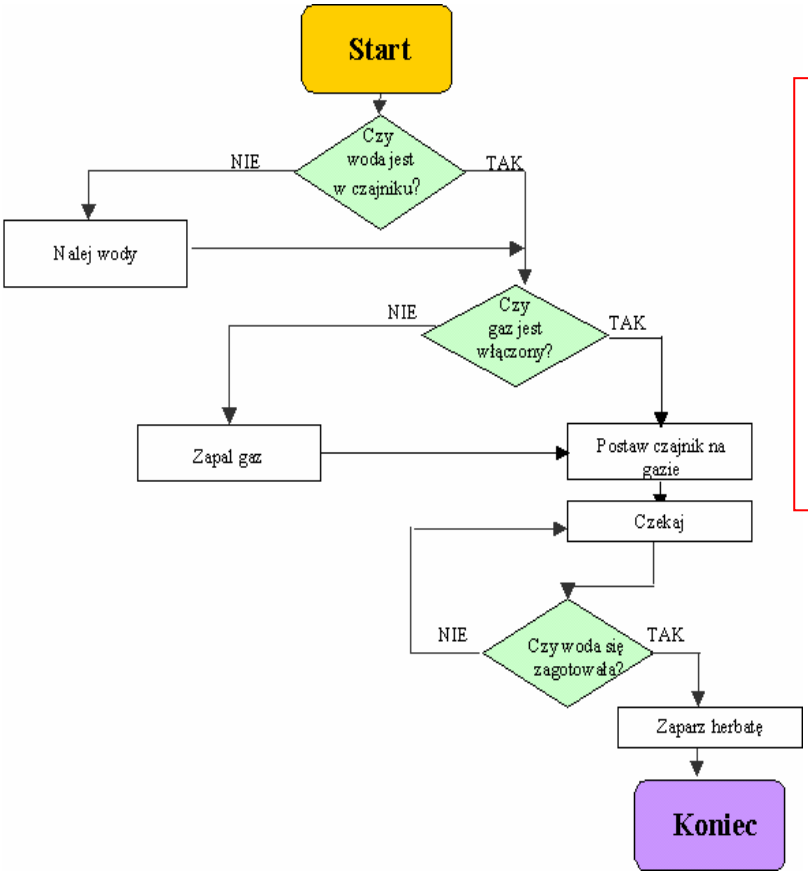
1. opis słowny
2. schemat blokowy
3. umowny strukturalny język programowania
4. język programowania wysokiego poziomu np. Pascal

1.Opis słowny - polega na logicznym i zrozumiałym dla odbiorcy przedstawieniu kolejnych czynności (akcji), jakie należy wykonać, aby osiągnąć zamierzony efekt. Przykładami takiego opisu algorytmu mogą być: przepis kulinarny, recepta wykonania leku, metoda rozwiązania zadania.

2.Schemat blokowy - czyli graficzna reprezentacja procedury lub programu sporządzana w celach poglądowych lub jako przedstawienie algorytmu zapisania w języku programowania.
Wykorzystuje się w nim, min. następujące symbole (elementy, bloki):

	**Początek** sieci działań (schematu). W schemacie może występować tylko jeden taki element.
	**Blok wejścia-wyjścia -** oznacza czynność wprowadzania danych i przyporządkowywania ich zmiennym używanym w dalszej części programu albo czynność wyprowadzania wyników obliczeń. Napis wewnątrz określa rodzaj czynności (np. czytaj, pisz) oraz nazwy zmiennych i stałych.
	**Element** (blok) **przetwarzania** (obliczeniowy) oznacza wykonanie operacji (lub zbioru operacji). Wewnątrz bloku określa się rodzaj czynności przetwarzania i ich argumenty.
	**Blok decyzyjny** (warunkowy, alternatywny, przełącznik) - oznacza element wyboru jednego z dwóch wariantów dalszego wykonywania programu. Wybór jest dokonywany na podstawie wyniku sprawdzenia warunku (będącego wyrażeniem logicznym) umieszczonego wewnątrz. Blok ten powinien zawsze posiadać dwa wyjścia opisane "T" (Tak, True) i "N" (Nie, False).
	**Łącznik wewnątrzstronicowy** - służy do łączenia odrębnych części schematu znajdujących się na tym samym arkuszu. Ułatwia zachowanie przejrzystości schematu. Komplementarne elementy oznacza się tym samym symbolem.
	**Zakończenie** wykonywania czynności.

Rysunek poniżej – *schemat blokowy parzenia herbaty*



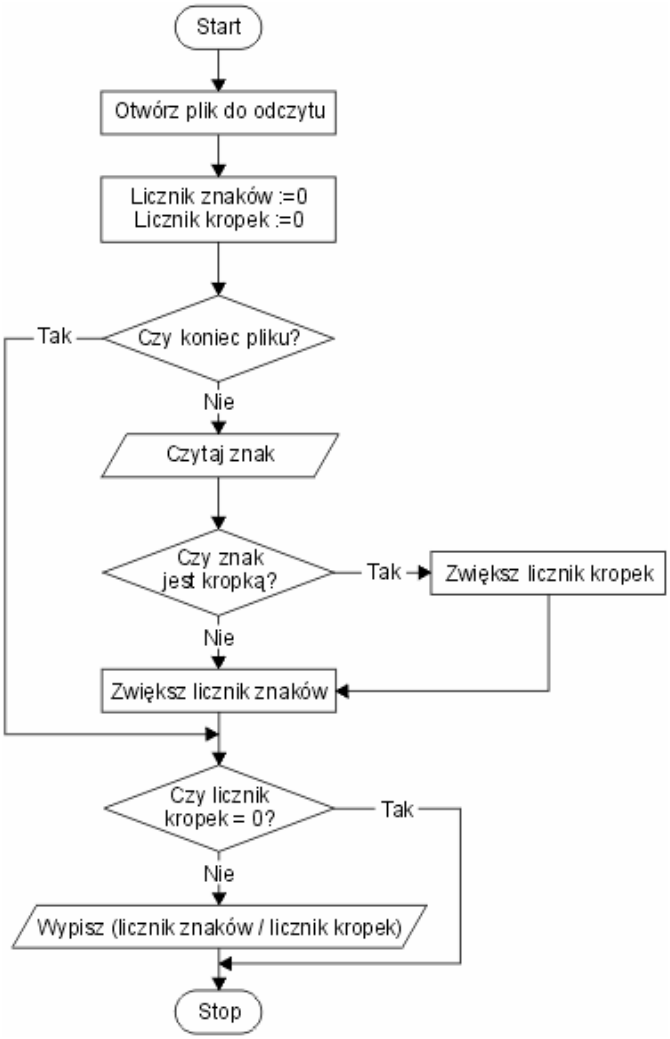
Myślę, że to można zrozumieć. Algorytm sprawdza, czy w czajniku jest woda, jeśli jej nie ma, to kieruje do kroku "Nalej wody". Następnie, kiedy woda na pewno jest w czajniku, należy się upewnić, czy włączyłeś gaz. Jeżeli nie, to należy to zrobić, a następnie postawić czajnik na palniku. Pozostaje czekać. Sprawdzamy, czy woda się zagotowała, jeśli nie, to czekamy dalej. Jeżeli woda się zagotowała, zalewamy herbatę. I koniec.

ZAPIS JEDNEGO ALGORYTMU W TRZECH RÓŻNYCH SPOSOBACH

Najprostszym sposobem jest słowne wyrażenie poszczególnych operacji w postaci punktów, np. tak:

1. *otwórz plik z tekstem do odczytu*
2. *wyzeruj licznik znaków*
3. *wyzeruj licznik kropek*
4. *jeśli koniec pliku, to idź do punktu 9*
5. *wczytaj kolejny znak z pliku*
6. *jeśli znak jest kropką, zwiększ licznik kropek o 1*
7. *zwiększ licznik znaków*
8. *idź do punktu 4*
9. *jeśli licznik kropek wynosi zero, idź do punktu 11*
10. *wypisz (licznik znaków/(licznik kropek))*
11. *STOP*

To samo można przedstawić w postaci graficznej, zwanej schematem blokowym lub z angielskiego *flowchart*.

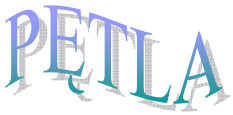


Rysunek po lewej stronie - *schemat blokowy obliczania średniej liczby znaków w zdaniu.*

Schemat postępowania można wreszcie zapisać w postaci tak zwanego pseudokodu, czyli imitacji "prawdziwego" języka programowania:

Start
twórz(plik);
znaki:= 0;
kropki:= 0;
dopóki nie koniec_pliku(plik)
start
czytaj (plik,znak) jeżeli znak = '.'
kropki := kropki+1;
znaki := znaki+1;
stop;
wypisz(znaki/kropki);
stop.

Każda z powyższych metod ma swoje wady i zalety. Zapis poszczególnych kroków w postaci punktów jest prosty, ale przy bardziej złożonych zadaniach staje się mało czytelny. Zapis w postaci schematu blokowego jest bardzo czytelny, ale pracochłonny w wykonaniu. Wreszcie zapis w pseudokodzie ma tę zaletę, że po przetłumaczeniu na język angielski staje się praktycznie gotowym tekstem programu.



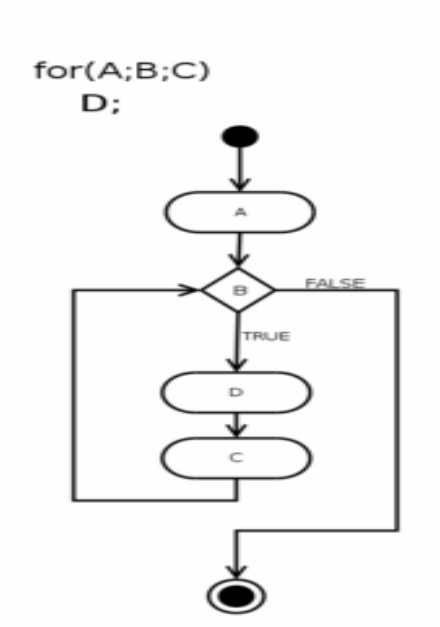
Pętla – ciąg instrukcji, z których ostatnia jest instrukcją skoku do pierwszej instrukcji ciągu. Pętla może być powtarzana dowolną,skończoną liczbę razy, co jest kontrolowane za pomocą zmieniającego się wewnątrz niej licznika lub zależy od badanego w pętli warunku. W szczególnych przypadkach lub wskutek błędu pętla może być powtarzana nieustannie. W językach wysokiego poziomu używa się trzech rodzajów pętli o tradycyjnie przyjętych nazwach:

- ***for*** (pętla „dla”)
- ***while*** (pętla „dopóki”)
- ***repeat*** albo ***do*** czyli pętla „powtarzaj”, inaczej „wykonuj”

Poniżej jest pokazana przykładowa pętla w języku Pascal:

```
var counter: integer;

begin
  for x:=1 to 10 do
   writeln('Linia numer ',x);
end.
```



Rysunek z lewej strony – diagram pętli *for* typu C (C jest językiem programowania stworzonym na początku lat siedemdziesiątych przez Dennisa Ritchie'ego do programowania systemów operacyjnych i innych zadań niskiego poziomu.)

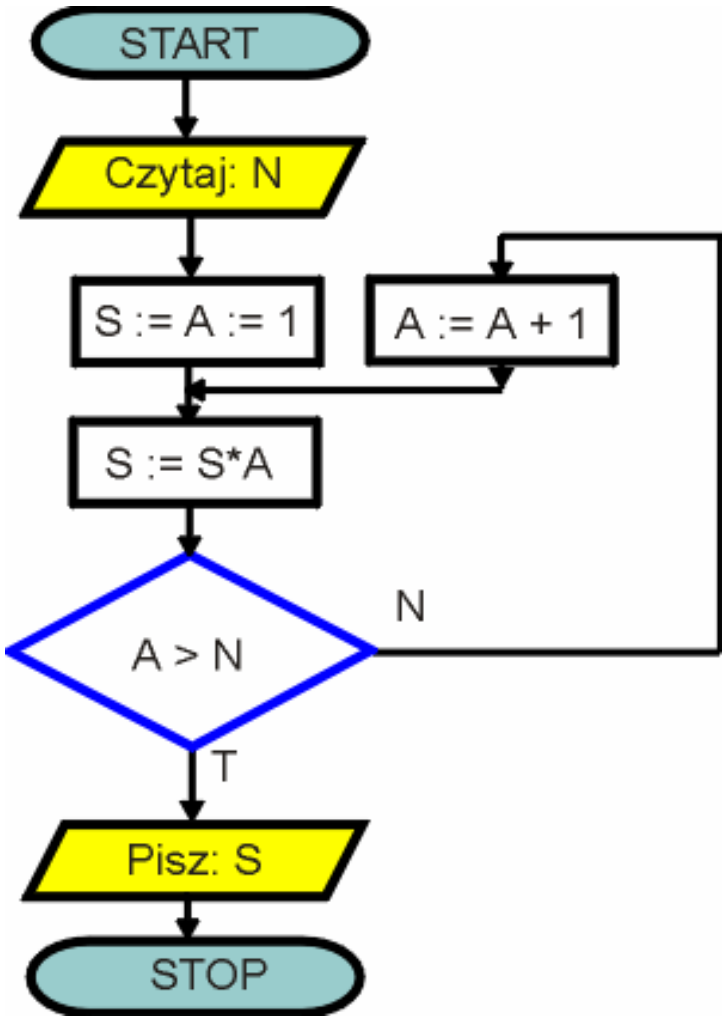
JAK POLICZYĆ SILNIE?

Istnieją zasadniczo dwa typy algorytmów rozwiązujących problem obliczenia silni liczby N.

1. **Algorytm rekurencyjny** – algorytm, który wywołuje sam siebie do rozwiązania tego samego problemu. Wykorzystuje się zależność: N!=(N-1)!*N (dla N>1).
2. **Algorytm cykliczny (iteracyjny) -** to powtarzanie (najczęściej wielokrotne) tej samej instrukcji.

Przedstawiony algorytm (cykliczny) oblicza iloczyn kolejnych liczb naturalnych, aż do osiągnięcia przez automatycznie zmieniany czynnik wartości N, zgodnie z zależnością:

N! = 1*2*3*4* ...*N



W matematyce funkcję *n!* (czyt. n-silnia) określoną na zbiorze liczb naturalnych definiuje się rekurencyjnie:

$$n! = \begin{cases} 1, & \text{gdy } n=0 \\ n*(n-1)!, & \text{gdy } n>0 \end{cases}$$

Dla *n=5*: 5! = 5*4! = 5*4*3! = 5*4*3*2! = 5*4*3*2*1*0! = 5*4*3*2*1*1 = 120. Gdyby nie wartość *0!=1* można by przyjąć, że silnia jest iloczynem liczb naturalnych nie większych od *n*. Na szczęście obliczając wartość funkcji podobnie jak wartość potęgi za pomocą pętli *For* można o tym zapomnieć:

```
Function Silnia(N : Byte) : LongInt;
Var
  I    : Byte;
  Wynik : LongInt;
Begin
  Wynik:=1;
  For I:=2 To N Do Wynik:=Wynik*I;
  Silnia:=Wynik;
End;
```

Ze względu na rekurencyjną definicję matematyczną nie wypada nie znać algorytmu rekurencyjnego wyznaczającego wartość funkcji w sposób wynikający z definicji: aby obliczyć *n!* musisz najpierw obliczyć *(n-1)!*:

```
Function Silnia(N : Byte) : LongInt;
Begin
  If N=0 Then Silnia:=1
  Else Silnia:=N*Silnia(N-1);
End;
```

Definicja ta jest dokładnym powieleniem definicji matematycznej - stąd warunek *N=0*, który z powodzeniem można zastąpić nierównością *N<2*

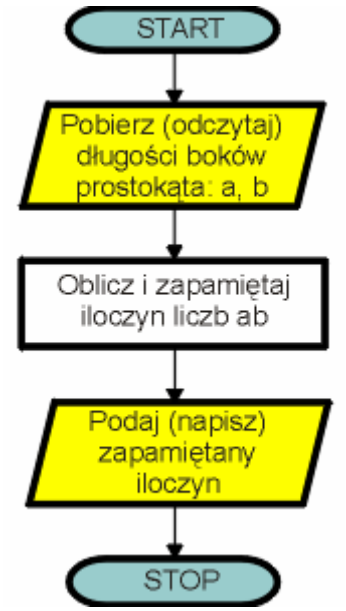
OBLICZANIE POLA PROSTOKĄTA

Stosując prosty przykład przedstawię uproszczone zasady tworzenia algorytmów. Na kolejnych stronach zostały przedstawione kolejne etapy rozbudowy algorytmu obliczania pola prostokąta. Rozpoczynając od struktury ogólnej poprzez zwiększanie poziomu szczegółowości przechodzimy do algorytmu stosunkowo rozbudowanego.

Obliczenie pola prostokąta jest dla każdego bardzo prostym zadaniem. Jest oczywiste, że pole prostokąta równe jest iloczynowi długości jego boków, czyli

P = ab

Na rysunku z prawej strony przedstawiono zbiór (sekwencję) prostych czynności, jakie należy wykonać, w określonej kolejności, aby obliczyć pole prostokąta. Jest to tzw. sieć działań inaczej schemat blokowy rozwiązania problemu, graficzna forma prezentacji algorytmu.



ALGORYTM NA OBLICZENIE POLA PROSTOKĄTA ZAPISANY UMOWNYM JĘZYKIEM PROGRAMOWANIA

Na rysunku nr 1 przedstawiono schemat blokowy z uproszczonym opisem poszczególnych bloków. Dla zwiększenia czytelności i przejrzystości schematów, zazwyczaj stosowane są takie opisy. Związane są one z tzw. metajęzykiem, czyli umownym językiem programowania. Poniżej przedstawiono ten algorytm zapisany w tym języku.

POCZĄTEK

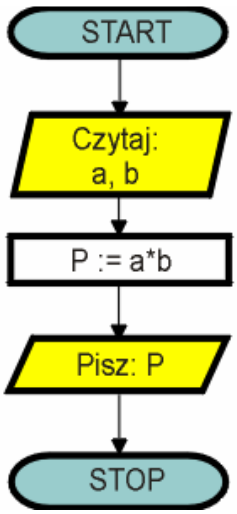
Czytaj(a,b)

P := a*b

Pisz(P)

KONIEC

Symbol **:=** oznacza czynność **przypisania** , np. w zapisie P:=a*b zmiennej P zostaje przypisana wartość iloczynu a*b (w zapisie szkolnym odpowiada to wyrażeniu P = ab).



Rys. 1

ALGORYTM NA OBLICZANIE POLA PROSTOKĄTA NAPISANY W JĘZYKU PASCAL

Algorytm rozwiązuje ten sam problem, lecz pobieranie danych wejściowych przedstawiono bardziej szczegółowo Przykład przedstawienia tego algorytmu w języku **Pascal**

```
PROGRAM Pole2;

Var a,b, P : Integer;

BEGIN

Write (' Podaj dlugość boku a ');

Readln(a);

Write (' Podaj dlugość boku b ');

Readln(b);

P:=a*b;

Write(P)

END.
```

Rys. 2

ALGORYTM NA OBLICZNIE POLA PROSTOKĄTA W JĘZYKU PASCAL – SPOSÓB DRUGI

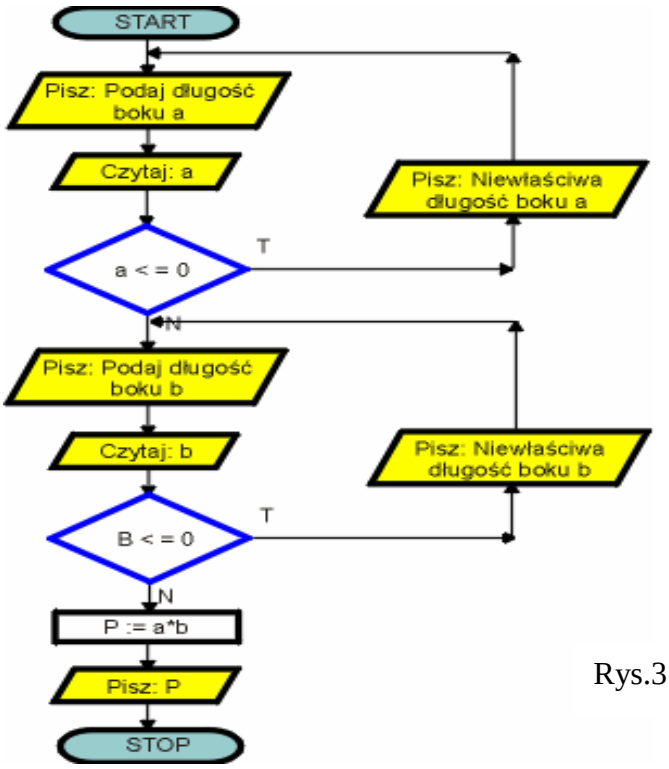
Algorytm rozwiązuje ten sam problem, lecz realizuje także **kontrolę pobieranych danych wejściowych**. Sprawdzanie danych wejściowych wykonują **bloki decyzyjne**.

Algorytm został przedstawiony na następnej stronie.

Drugi przykład w języku **Pascal**

```
PROGRAM Pole3;
Var a,b, P : Integer;
BEGIN
Write (' Podaj dlugość boku a ');
Readln(a);
If a <= 0 then
Begin
Writeln('Niewlaściwa długość boku a ');
Write (' Podaj dlugość boku a ');
Readln(a);
End;
Write (' Podaj dlugość boku b ');
Readln(b);
If b<=0 then
Begin
Writeln ('Niewłaściwa długość boku b ');
Write (' Podaj dlugość boku b ');
Readln(b);
End;
P:=a*b;
Write(P)
END.
```

Ciąg dalszy algorytmu



Rys.3

RODZAJE ALGORYTMÓW

Rozróżniamy następujące rodzaje algorytmów:

- Algorytm aproksymacyjny
- Algorytm bankiera
- Algorytm “czekaj albo giń”
- Algorytm Dijkstry
- Algorytm drugiej szansy
- Algorytm elekcji
- Algorytm Euklidesa
- Algorytm Floyda – Warshalla
- Algorytm głosowania
- Algorytm Grahama
- Algorytm Huffmana
- Algorytm IDEA
- Algorytm kryptograficzny
- Algorytm licytowany
- Algorytm LRU
- Algorytm LUC
- Algorytm MDS
- Algorytm niewywłaszczający
- Algorytm omiatający
- Algorytm o optymalnym czasie
- Algorytm OPT
- Algorytm piekarniany
- Algorytm planowania
- Algorytm plecakowy
- Algorytm probabilistyczny
- Algorytm przepisywania
- Algorytm przydziału ramek
- Algorytm Rabina – Karpa
- Algorytm rekurencyjny
- Algorytm rosyjskich wieśniaków
- Algorytm rotacyjny
- Algorytm rozproszony
- Algorytm równoległy
- Algorytm SHA
- Algorytm SJF
- Algorytm SKIPJACK
- Algorytm strusia
- Algorytm sympleks
- Algorytm tyrana
- Algorytm wielomianowy
- Algorytm windy
- Algorytm wywłaszczający

- Algorytm z nawrotami
- Algorytm zachłanny
- Algorytm zastępowania stron
- Algorytm “zrań albo czekaj”
- Algorytm zegarowy
- Algorytmy grafowe
- Algorytmy Kruskala i Prima
- Algorytmy planowania dysku
- Algorytmy przydziału pamięci
- Algorytmy przydziału procesora
- Algorytmy scentralizowane
- Algorytmy teorioliczbowe

Literatura:

- Szkolny słownik, komputer i internet, wydawnictwo europa
- www.wikipedia.pl
- Turbo Pascal. Programowanie. Tomasz M. Sadowisk , wydawnictwo Helion
- http://www.lo2.opole.pl/~dragosystems/algorytm.htm
- http://algorytmikaw.maturalny.org/