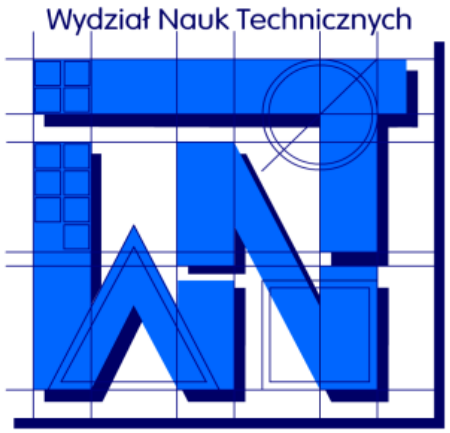


UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Języki Programowania

Algorytmy

Wojciech Sobieski

Olsztyn, 2001-2021

Algorytm

Algorytm – dokładny przepis podający sposób rozwiązania określonego zadania w skończonej liczbie kroków. Algorytm zapisany przy pomocy języka programowania jest programem.

Cechy Algorytmu:

- posiada dane wejściowe z dobrze zdefiniowanego zbioru
- musi działać poprawnie dla wszystkich zestawów danych z tego zbioru
- podaje wynik
- każdy krok algorytmu jest jednoznacznie określony
- jest skończony tzn. wynik musi zostać dostarczony po wykonaniu skończonej liczby kroków

Algorytmy mogą być **numeryczne**, operujące na liczbach (np. algorytm Euklidesa), lub **nienumeryczne**, operujące na obiektach innych niż liczby (np. sortowanie dokumentów).

Iteracja i rekurencja

Iteracja – metoda matematyczna polegająca na wielokrotnym kolejnym zastosowaniu tego samego algorytmu postępowania, przy czym wynik i-tej operacji stanowi dane wejściowe dla kolejnej, (i+1)-szej operacji.

Rekurencja (lub rekursja) – w programowaniu i w matematyce odwoływanie się funkcji do samej siebie. Np. poniższa definicja ciągu Fibonacciego jest rekursywna:

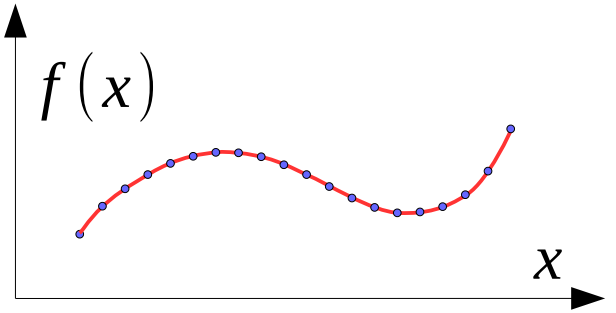
fib(0) = 0
fib(1) = 1
fib(n) = fib(n - 1) + fib(n - 2), dla n>=2

Obszary stosowania algorytmów

Metody numeryczne – metody rozwiązywania układów równań (zazwyczaj cząstkowych i nieliniowych), działające na danych dyskretnych i dające rozwiązanie przybliżone.

Najważniejsze metody numeryczne:

- Metoda Różnic Skończonych
- Metoda Elementów Skończonych
- Metoda Objętości Skończonych
- Metoda Elementów Dyskretnych
- Metoda gazu sieciowego Boltzmanna
- Metoda Cząstek Znaczonych
- Metoda Zanurzonego Brzegu



ciągła (kolor czerwony)
oraz dyskretna (kolor niebieski)
postać funkcji

Obszary stosowania algorytmów

Sortowanie – uporządkowanie zbioru danych względem pewnych cech charakterystycznych. Szczególnym przypadkiem jest sortowanie względem wartości każdego elementu, np. sortowanie liczb, słów itp.

Przykładowe algorytmy sortowania to:

- sortowanie bąbelkowe
- sortowanie przez zliczanie
- sortowanie przez wstawianie
- sortowanie przez wybieranie
- sortowanie przez kopcowanie
- sortowanie szybkie
- sortowanie kubełkowe
- sortowanie grzebieniowe

Obszary stosowania algorytmów

Kompresja – ogólnie działanie mające na celu zmniejszenie objętości czegoś (czyli zwiększenia gęstości), np. gazu (fizyka). Działaniem przeciwnym do kompresji jest dekompresja.

W informatyce chodzi o działania mające na celu zmniejszenie objętości informacyjnej danych, czyli wyrażenie zestawu danych za pomocą mniejszej ilości bitów.

Rodzaje kompresji:

- bezstratna lub stratna
- uniwersalna (tylko kompresja bezstratna) lub nakierowana na określony typ danych

Obszary stosowania algorytmów

Kryptografia – nauka zajmująca się układaniem szyfrów.

Wyróżniane są dwa główne nurty kryptografii:

- kryptografia symetryczna – to taki rodzaj szyfrowania, w którym tekst jawny ulega przekształceniu na tekst zaszyfrowany za pomocą pewnego klucza, a do odszyfrowania jest niezbędna znajomość tego samego klucza.

- kryptografia asymetryczna – to rodzaj kryptografii, w którym używa się zestawów dwu lub więcej powiązanych ze sobą kluczy, umożliwiających wykonywanie różnych czynności kryptograficznych.

Obszary stosowania algorytmów

Prosty szyfr podstawieniowy – szyfr, w którym każdy znak tekstu jawnego zastępowany jest przez dokładnie jeden, przyporządkowany mu znak szyfrogramu. Np. A-1, B-4, C-8 itd.

Szyfr homofoniczny – szyfr podstawieniowy, w którym każdej literze tekstu jawnego odpowiada inny zbiór symboli kryptogramu (homofonów). Liczba homofonów powinna być zależna od częstotliwości występowania danej litery w tekście naturalnym. Przy każdym szyfrowaniu litery wybierany jest losowo jeden z jej homofonów. W ten sposób zostaje spłaszczony histogram kryptogramu, a wielokrotne szyfrowanie tego samego tekstu daje za każdym razem inny wynik. Cechy te znacząco utrudniają kryptoanalizę opartą na statystyce tekstu.

Na przykład: Jeśli literze A odpowiada zbiór homofonów {4, 24, 54, 32}, a literze L {14, 81}, to słowo "ALA" może zostać zaszyfrowane jako (24, 81, 32), ale również jako (32, 14, 4).

Obszary stosowania algorytmów

Sztuczna inteligencja – technologia i kierunek badań informatycznych i psychologicznych. Jego zadaniem jest „konstruowanie maszyn, o których działaniu dałoby się powiedzieć, że są podobne do ludzkich przejawów inteligencji", jak to zostało zdefiniowane przez Johna McCarthy'ego, który w 1955 r. zaproponował ten termin.

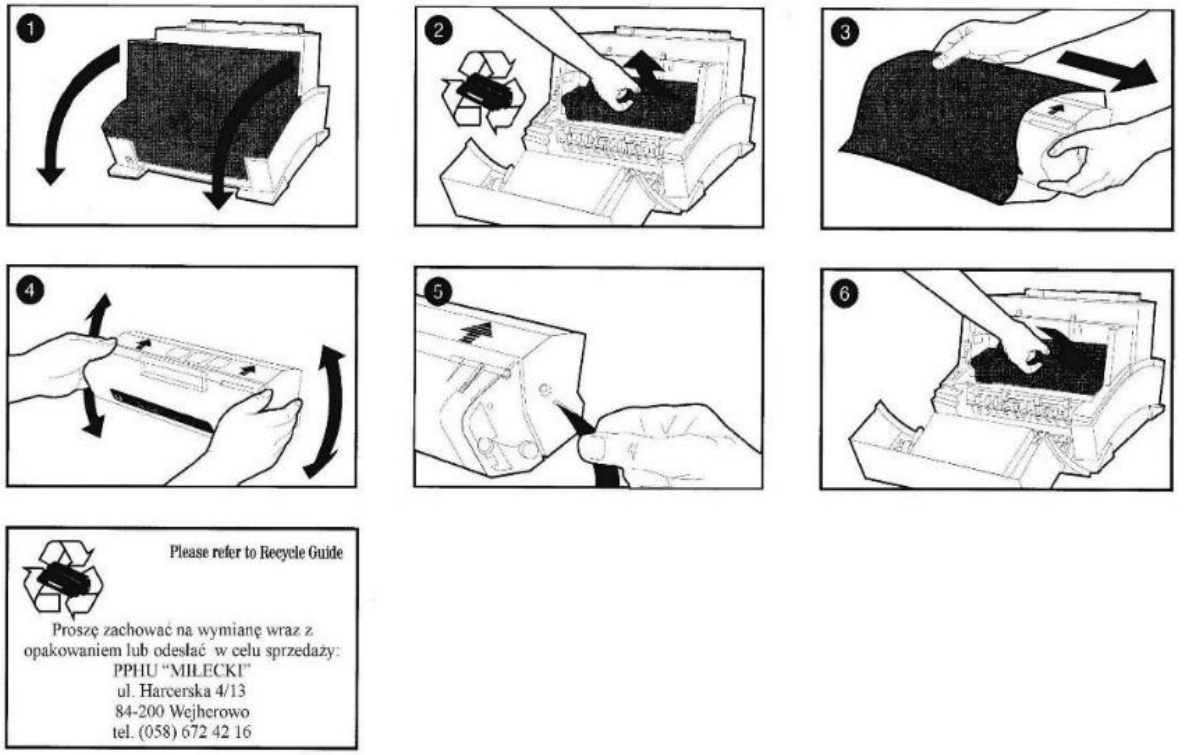
Dwa podejścia do sztucznej inteligencji:

- tworzenie całościowych modeli matematycznych analizowanych problemów i implementowanie ich w formie programów komputerowych, mających realizować konkretne cele

- tworzenia struktur i programów „samouczących się, takich jak modele sieci neuronowych oraz opracowywania procedur rozwiązywania problemów poprzez „uczenie" takich programów, a następnie uzyskiwanie od nich odpowiedzi na „pytania"

Zapis algorytmów

Algorytm opisany obrazkami – występuje szeroko w instrukcjach opisujących sposób montażu zabawek dla dzieci (np. klocki LEGO), modeli do sklejania, instrukcjach obsługi (np. telewizora, magnetowidu, lodówki).



Zapis algorytmów

Kruche ciasteczka

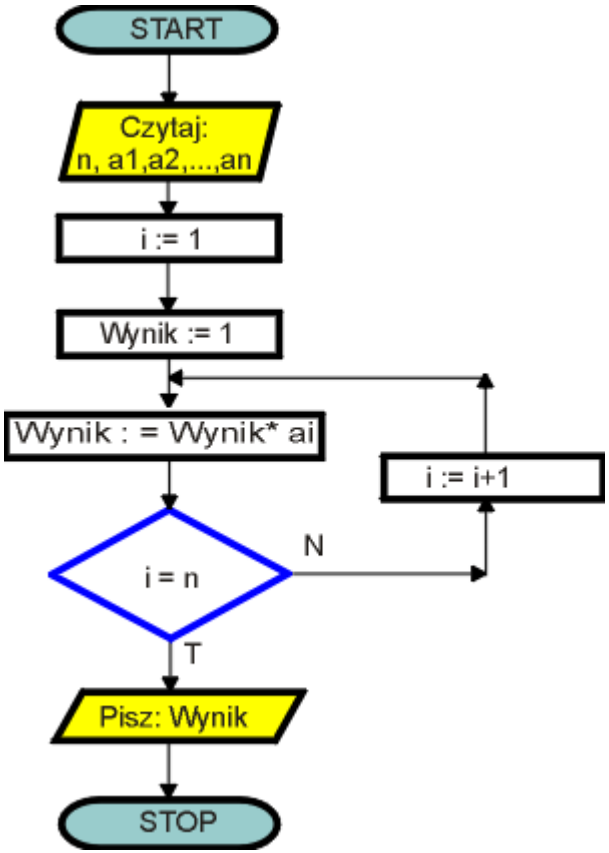
Składniki:
- 1 kg mąki,
- 1 kostka masła,
- 5 jajek(żółtka od białka oddzielamy),
- 1,5 szklanki cukru,
- śmietana,
- łyżeczka proszku do pieczenia.

Jak przyrządzić?
Żółtka ukręcić z cukrem. Następnie zagnieść wszystkie składniki na stolnicy- z wyjątkiem białek. Dodać tyle śmietany ile trzeba, by ciasto się połączyło(trzeba się przy tym trochę nagnieść i namęczyć ale warto! Ewentualnie jak nie ma śmietany można dodać mleka- też się uda). Ciasto rozwałkować na stolnicy i wykrawać ciasteczka. Można je posmarować białkiem, które zostało z jajek. Piec na złoty kolor, niezbyt długo- kilka minut - inaczej będą twarde. Piekarnik powinien mieć ok. 200 stopni C.

Algorytm opisany słownie – występuje we wszystkich instrukcjach obsługi, sprzętu domowego, aparatury naukowej, na lekcjach wielu przedmiotów w postaci opisu doświadczenia i w informatyce jako element poprzedzający właściwe programowanie.

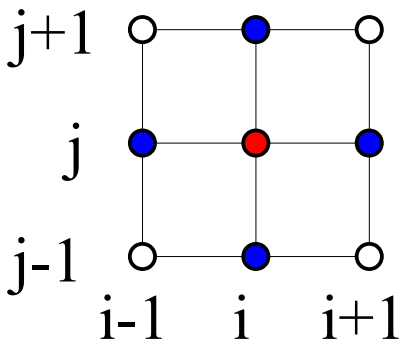
Zapis algorytmów

Algorytm opisany schematem blokowym – występuje głównie w nauczaniu elementów informatyki i służy do graficznego prezentowania i rozwiązywania problemu, powinien być poprzedzony opisem słownym. Schemat blokowy stanowi doskonałą bazę do stworzenie programu, w łatwy do zrozumienia sposób może powstać taki schemat. Graficznie ukazuje to co w programie jest najważniejsze, pokazuje zależności między kolejnymi poleceniami.



Zapis algorytmów

Algorytm opisany wzorem matematycznym – podstawowa forma zapisu algorytmów bazowych dla programów obliczeniowych.



równanie falowe:

$$\frac{\partial^2 U(x,y,t)}{\partial x^2}+\frac{\partial^2 U(x,y,t)}{\partial y^2}=\frac{1}{c^2}\cdot\frac{\partial^2 U(x,y,t)}{\partial t^2}$$

analog różnicowy:

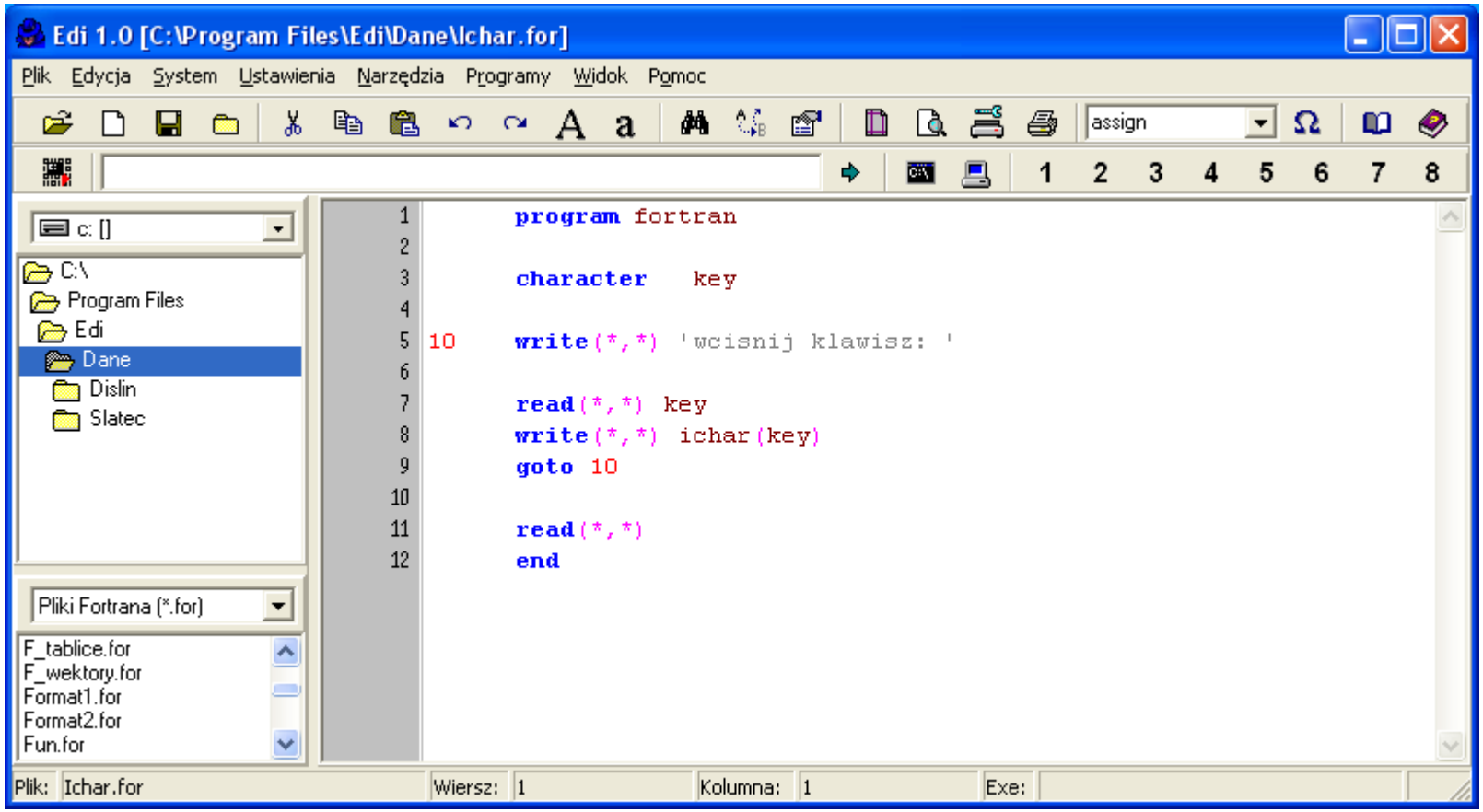
$$\frac{\partial^2 U}{\partial x^2}=\Delta_{x(5a)}U=\frac{U^n_{i+1,j}-2\cdot U^n_{i,j}+U^n_{i-1,j}}{\Delta x^2}$$

$$\frac{\partial^2 U}{\partial y^2}=\Delta_{y(5a)}U=\frac{U^n_{i,j+1}-2\cdot U^n_{i,j}+U^n_{i,j-1}}{\Delta y^2}$$

Analog 5-cio punktowy: wersja A.

Zapis algorytmów

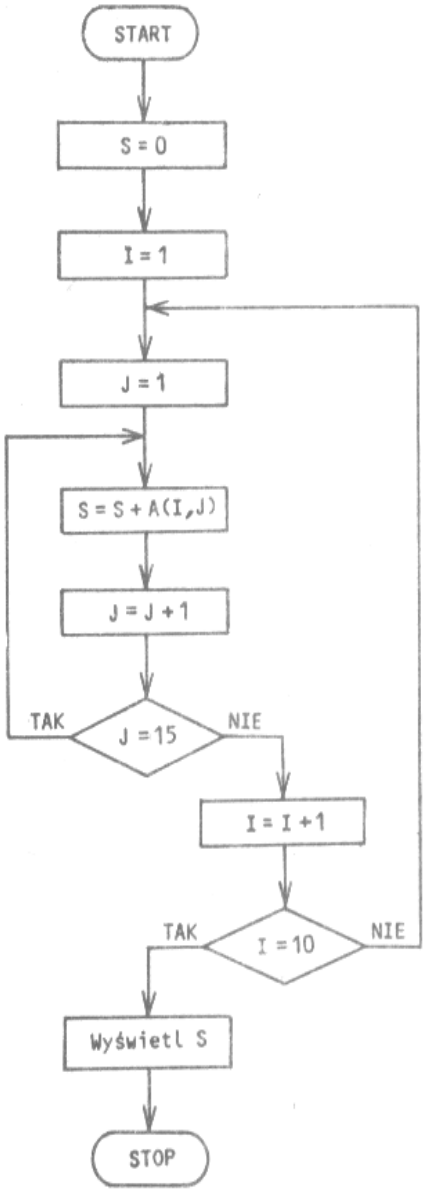
Algorytm opisany językiem programowania – (program) stanowi realizację projektu w konkretnym języku programowana, powinien być poprzedzony opisem słownym i schematem blokowym.



Diagramy

Diagram (schemat blokowy) – sposób przedstawiania algorytmów za pomocą odpowiednio opisanych figur geometrycznych oraz łączących je linii.

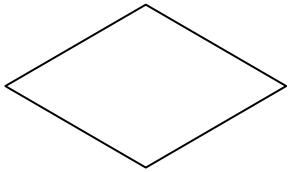
Zasadniczą zaletą schematów blokowych jest to, że graficznie prezentują one działanie programu, zarówno od strony występujących w nim działań, jak i ich kolejności.



Diagramy

Operacje na pamięci – operacje, w wyniku których ulega zmianie wartość, postać lub miejsce zapisu danych w pamięci operacyjnej komputera. Jeśli kilka operacji tworzy logiczną całość, to wszystkie one mogą być umieszczone w jednej skrzynce. Nie zaleca się jednak umieszczania tam zbyt dużej ilości operacji - nawet wtedy, kiedy są one powiązane ze sobą bezpośrednio – gdyż może to zmniejszyć czytelność schematu.

Operacje I/O – operacje wprowadzanie i wyprowadzanie danych do/z pamięci operacyjnej komputera.

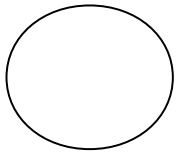


Operacje warunkowe – operacje prowadzące zawsze do konieczności rozważenia dwóch dróg: jednej (TAK) kiedy rozpatrywany warunek jest spełniony i drugiej, kiedy warunek nie jest spełniony (NIE).

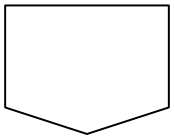
Proces zewnętrzny – proces określony poza programem i z tego powodu nie wymagający zdefiniowania w rozpatrywanym programie (podprogram).

Diagramy

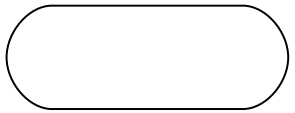
Kierunek – określa kierunek przepływu danych lub kolejność wykonywania działań.



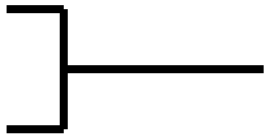
Łącznik stronicowy – wskazuje wejście lub wyjście z wyodrębnionych fragmentów schematu rozmieszczonych na tych samych stronach (arkuszach papieru).



Łącznik międzystronicowy – wskazuje wejście lub wyjście z wyodrębnionych fragmentów schematu rozmieszczonych na różnych stronach (arkuszach papieru).



Blok graniczny – oznacza miejsca rozpoczęcia, zakończenie lub przerwania działania programu.



Komentarz – służy do podawania dodatkowych informacji, niezbędnych do zrozumienia działania algorytmu.

Diagramy

Zasady tworzenia diagramów:

- schemat powinien być prosty i czytelny. W razie złożonego rozwiązania schemat należy podzielić na mniejsze części i zamieścić na osobnych arkuszach.
- w blokach niezbędne jest komentowanie zarówno zaprojektowanej operacji, jak i kolejności ich wykonania. Komentarze powinny być krótkie, lecz dokładnie wyjaśniające znaczenie opisywanych elementów.
- należy unikać rysowania przecinających się ścieżek sterowania. W razie konieczności lepiej jest wprowadzić odpowiedni łącznik, który pozwoli wyeliminować niektóre z linii.
- powinno się unikać zapisywania wprowadzanych operacji za pomocą instrukcji języków programowania.
- należy dokładnie numerować arkusze, na których został narysowany schemat blokowy.

Diagramy

Zasady tworzenia diagramów, cd.:

- trzeba liczyć się z możliwością wystąpienia konieczności poprawek do schematu, dlatego wskazane jest tak tworzyć arkusze, aby możliwe było naniesienie poprawek bez konieczności przerysowania całego schematu.
- należy unikać zarówno zbyt dużej szczegółowości jak i zbytniej ogólności schematów.
- nie należy umieszczać zbyt dużej liczby operacji w jednym bloku.
- operacja warunkowa JEŻELI zawsze prowadzi do konieczności rozważenia dwóch dróg, gdy warunek jest spełniony i gdy nie jest.
- operacja warunkowa CASE musi zawierać opis wszelkich możliwych przypadków zmiennej sterującej

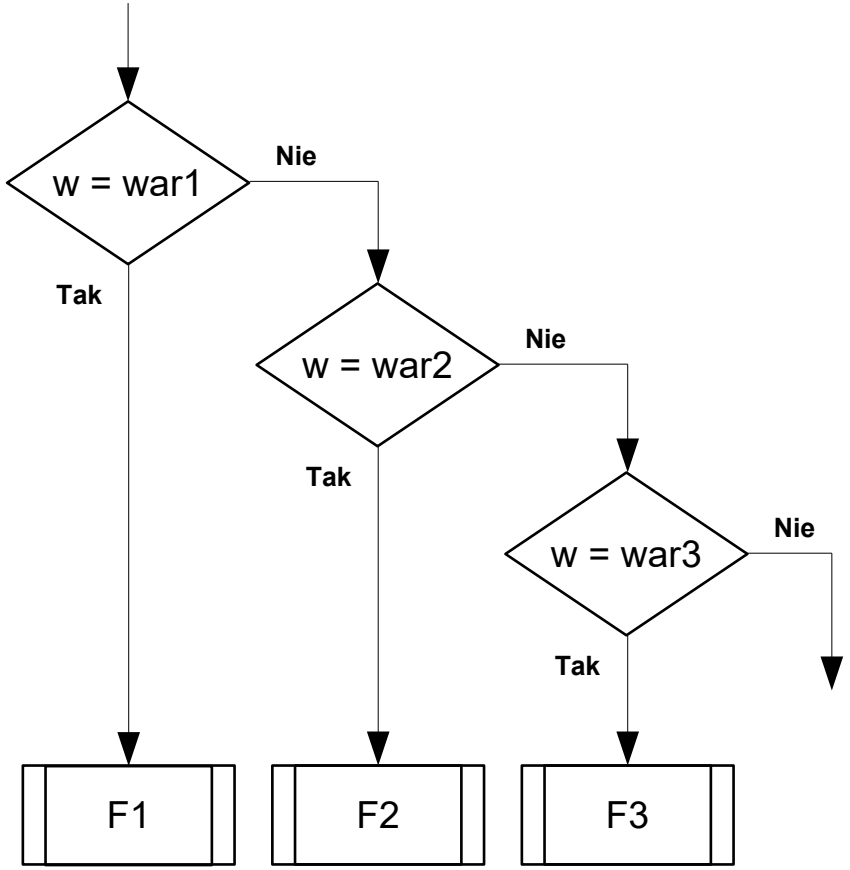
Etapy tworzenia algorytmów

Podstawowe etapy tworzenia algorytmów:

1. Sformułowanie zadania
2. Określenie danych wejściowych
3. Określenie celu, czyli wyniku
4. Poszukiwanie metody rozwiązania, czyli algorytmu
5. Przedstawienie algorytmu w postaci:
   - opisu słownego
   - listy kroków
   - schematu blokowego
   - jednego z języków programowania
6. Analiza poprawności rozwiązania
7. Testowanie rozwiązania i ocena efektywności przyjętej metody

Elementy algorytmów

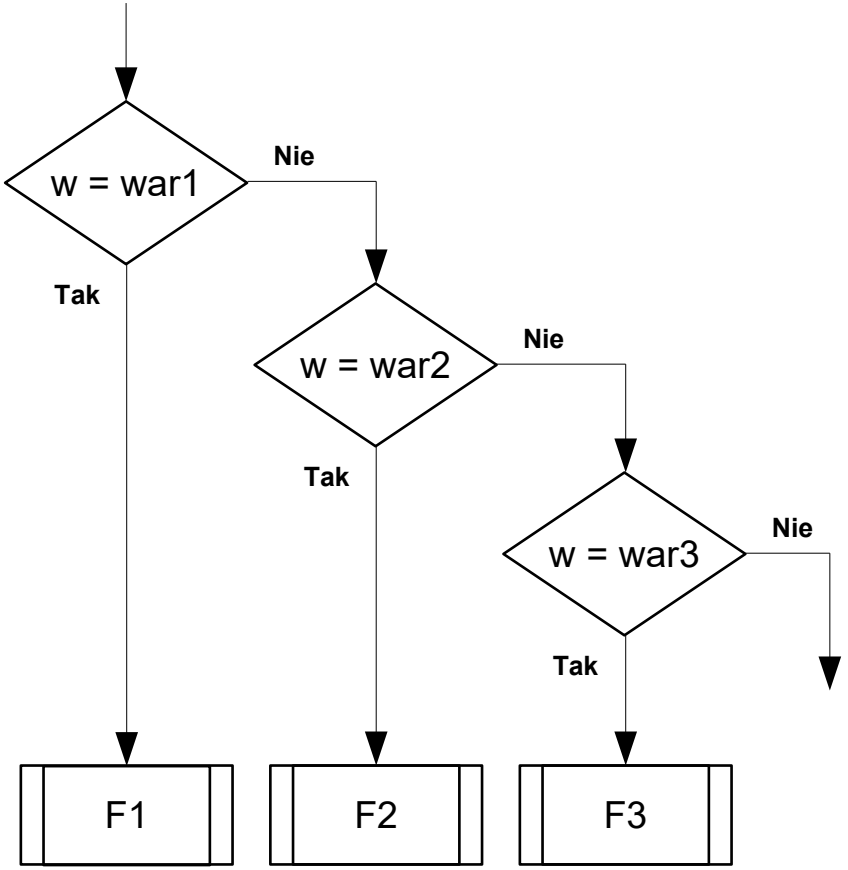
Instrukcja wyboru – diagram i przykład zapisu (Pascal):




```
case i of
    1     : ShowMessage('Jest 1');
    2..5 : ShowMessage('Jest 2-5');
    6,9   : ShowMessage('Jest 6,9');
else
          ShowMessage('Inne');
end;
```

Elementy algorytmów

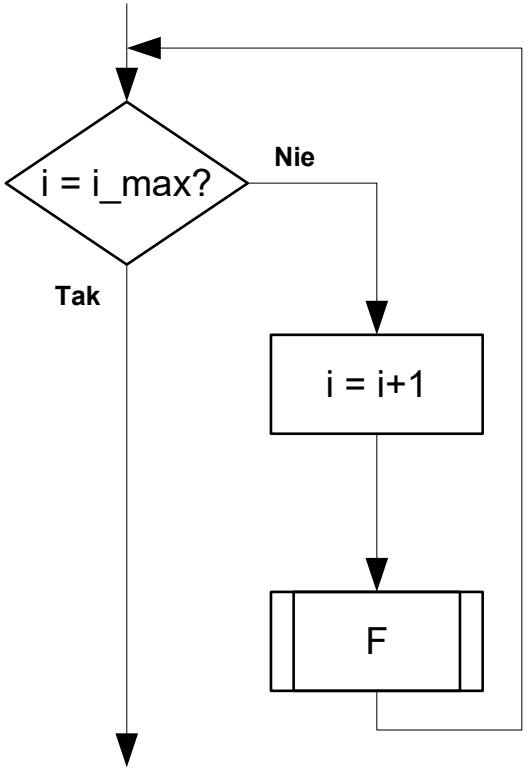
Instrukcja wyboru – diagram i przykład zapisu (Fortran):




```
select case (i)
case(1)
    write(*,*)  'Jest 1'
case(2:5)
    write(*,*)  'Jest 2-5'
case(6,9)
    write(*,*)  'Jest 6,9'
case default
    write(*,*)  'Inne'
end select
```

Elementy algorytmów

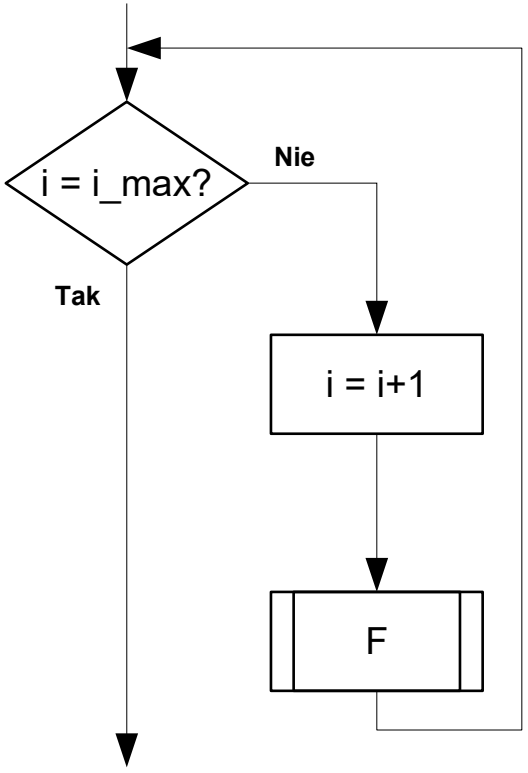
Instrukcja pętli o znanej liczbie powtórzeń – diagram i przykład zapisu (Pascal):




```
iMax:=5;
for i:=1 to iMax do
    begin
    Tablica[i] := 1;
    end;
```

Elementy algorytmów

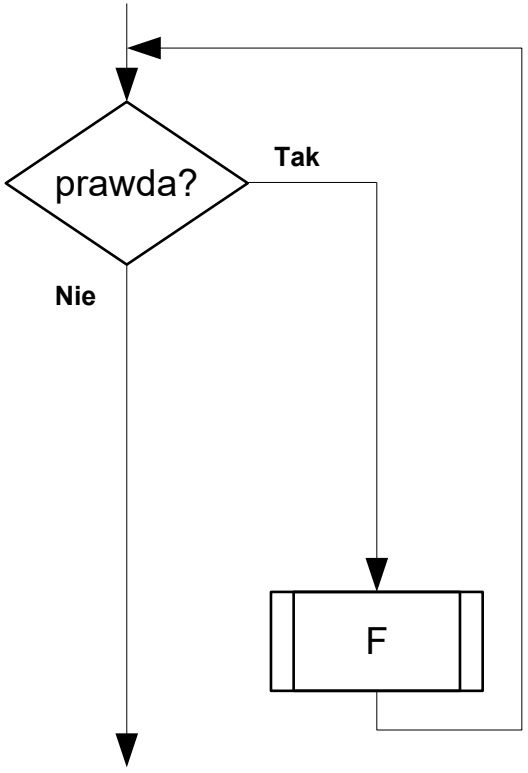
Instrukcja pętli o znanej liczbie powtórzeń – diagram i przykład zapisu (Fortran):




```
iMax=5
do i=1, iMax
    Tablica(i)=1
end do
```

Elementy algorytmów

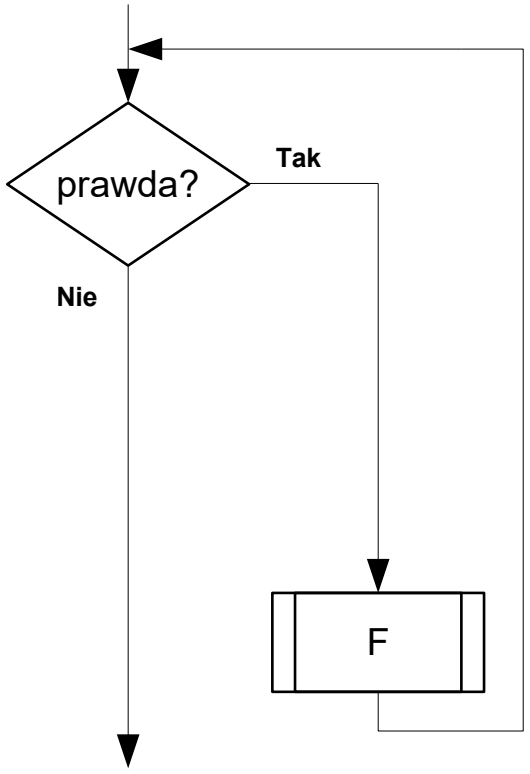
Instrukcja pętli o nieznanej liczbie powtórzeń z warunkiem na początku – diagram i przykład zapisu (Pascal):




```
i := 0;
iMax:=5;
while i < iMax do
   begin
   ShowMessage(IntToStr(i)+
   ' Mniej niz iMax!');
   i := i + 1;
   end;
```

Elementy algorytmów

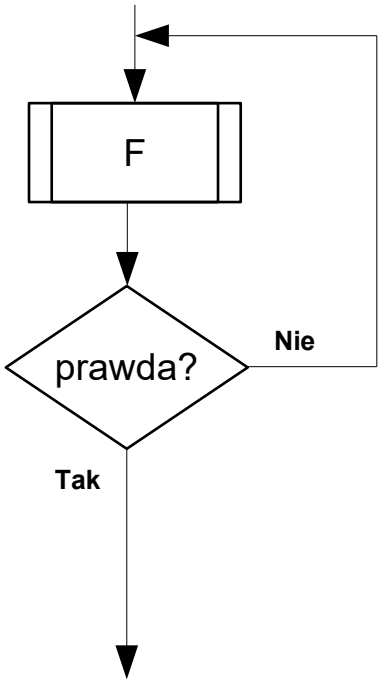
Instrukcja pętli o nieznanej liczbie powtórzeń z warunkiem na początku – diagram i przykład zapisu (Fortran):




```
i=0
iMax=5
do while (i .LT. iMax)
    write(*,*)  i,' Mniej niz iMax! '
    i=i+1
end do
```

Elementy algorytmów

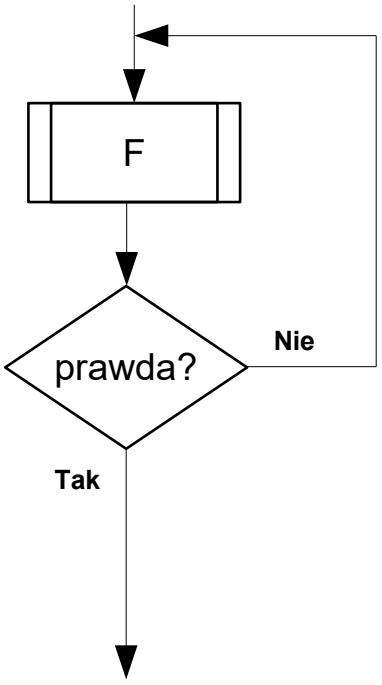
Instrukcja pętli o nieznanej liczbie powtórzeń z warunkiem na końcu – diagram i przykład zapisu (Pascal):




```
i := 0;
iMax:=5;
repeat
   ShowMessage(IntToStr(i)+
  ' Mniej niz iMax!');
   i := i + 1;
until i >= iMax;
```

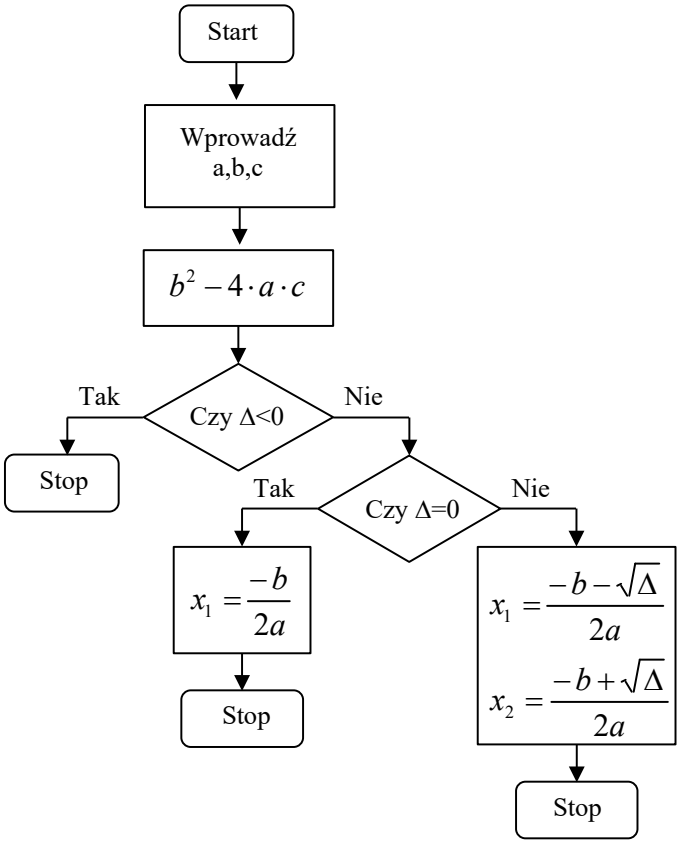
Elementy algorytmów

Instrukcja pętli o nieznanej liczbie powtórzeń z warunkiem na końcu – diagram i przykład zapisu (Fortran):

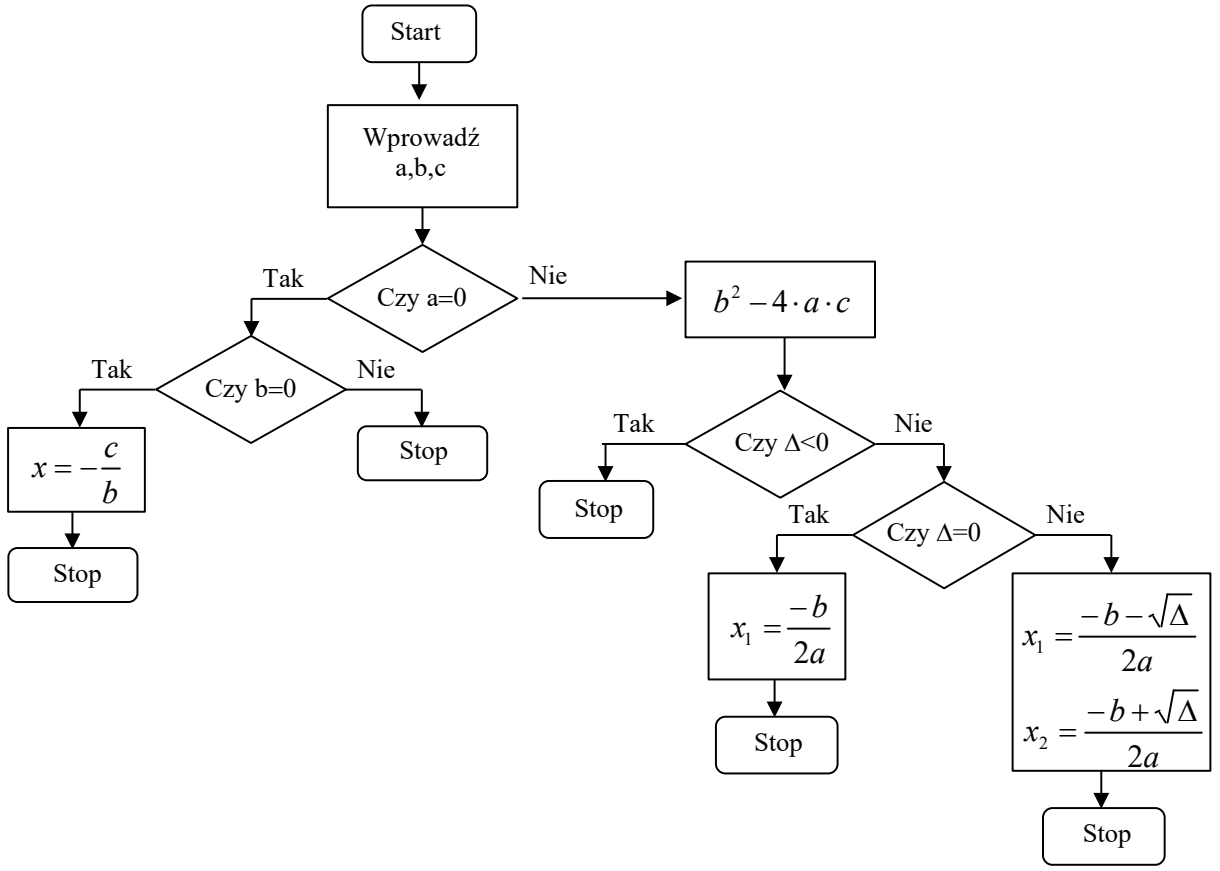



```
      i=0
      iMax=5
10    continue
         i=i+1
         write(*,*)  i,' Mniej niz iMax! '
      if (i .LT. iMax) goto 10
```

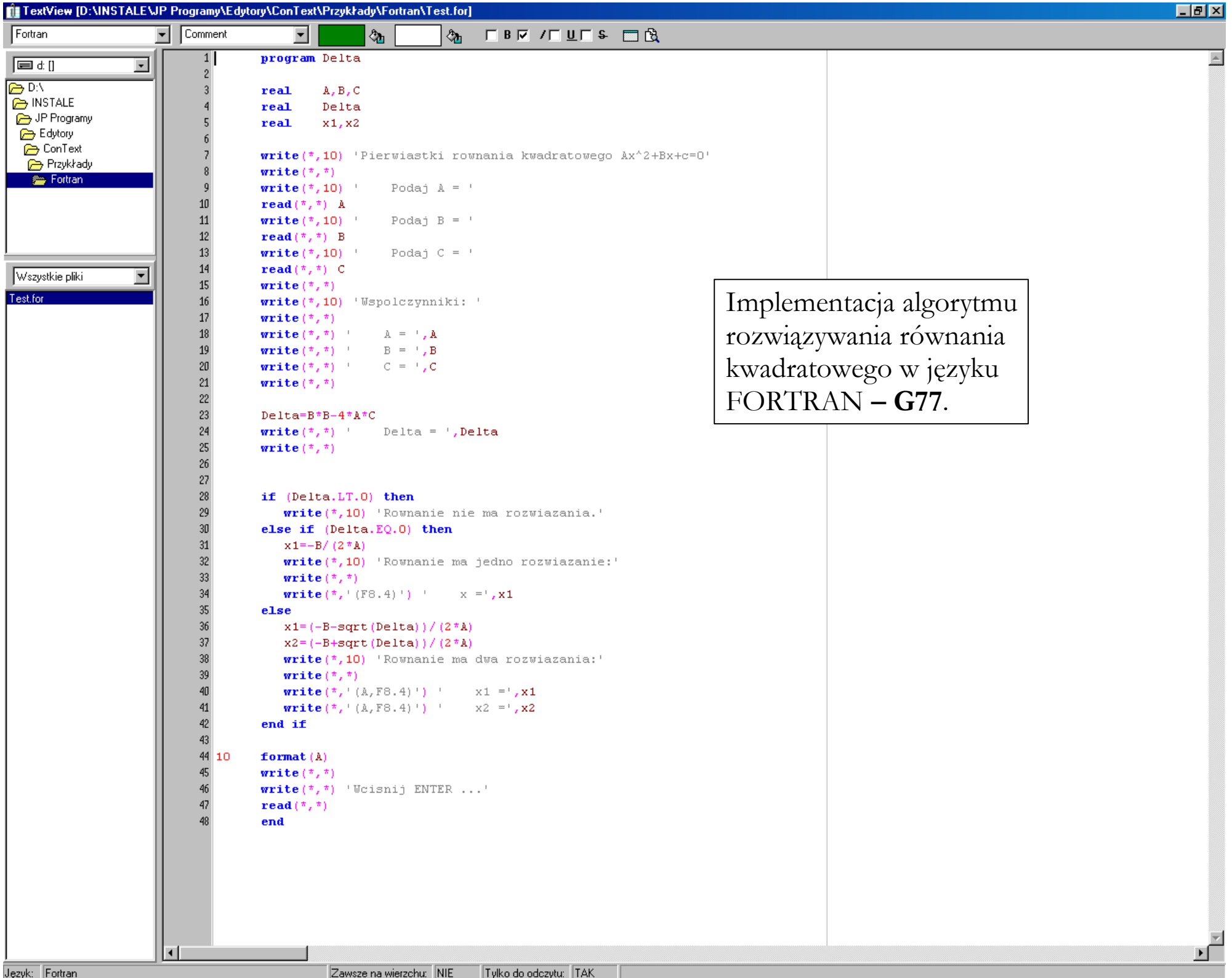
Przykłady algorytmów



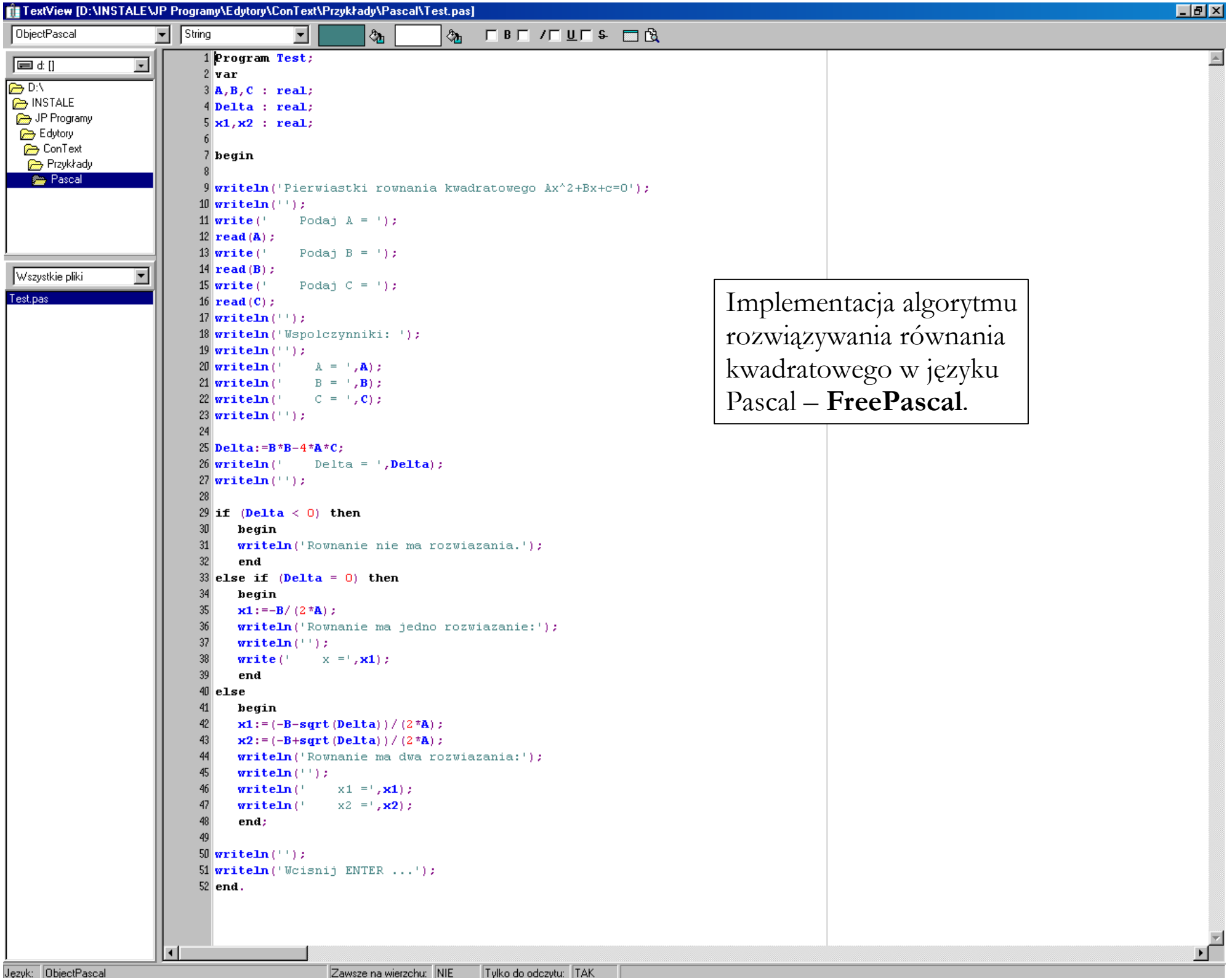
algorytm rozwiązania równania kwadratowego



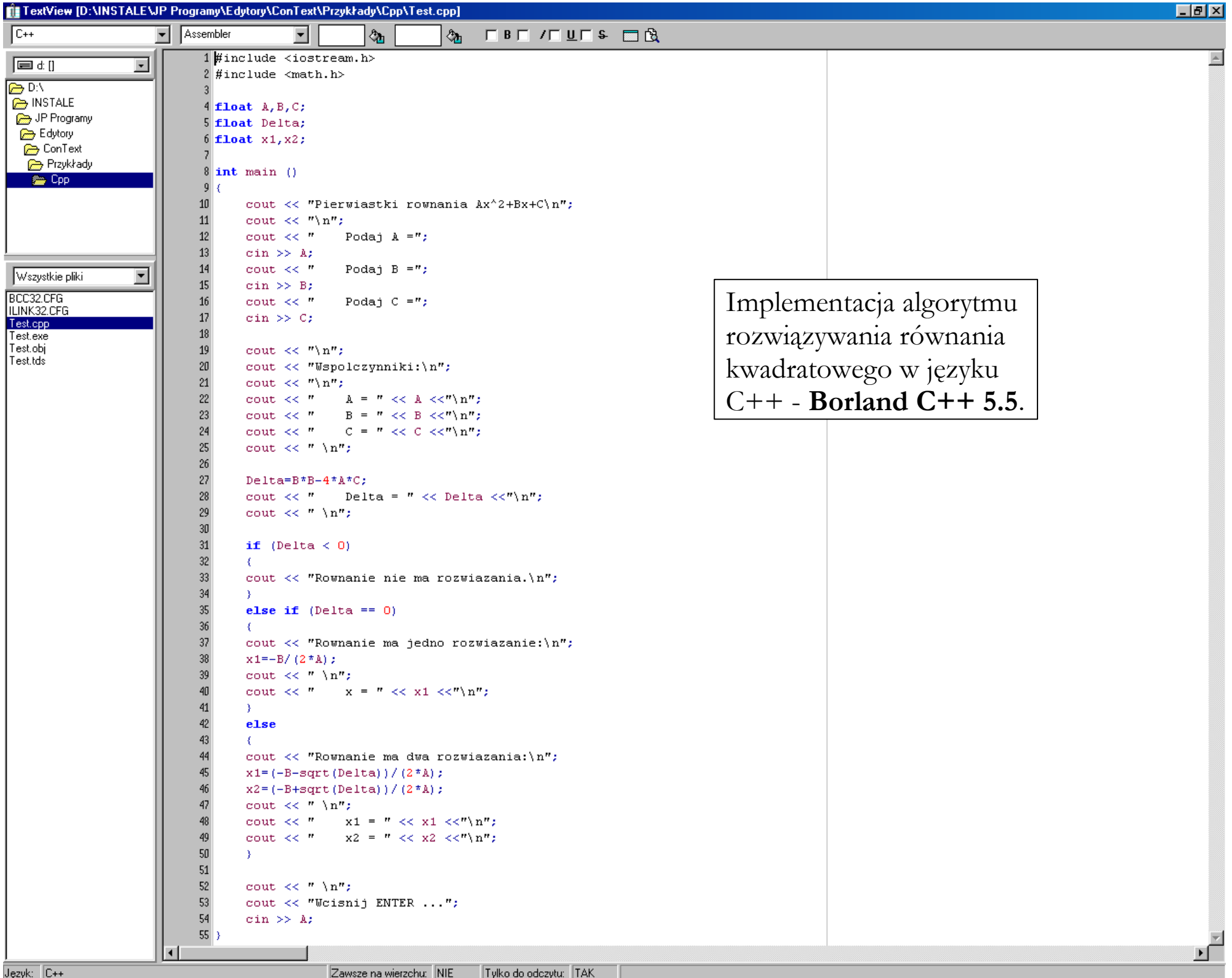
algorytm rozwiązania równania stopnia nie wyższego niż drugi



Implementacja algorytmu rozwiązywania równania kwadratowego w języku FORTRAN – **G77**.



Implementacja algorytmu rozwiązywania równania kwadratowego w języku Pascal – **FreePascal**.



Implementacja algorytmu rozwiązywania równania kwadratowego w języku C++ - **Borland C++ 5.5**.

Poprawność algorytmów

Praktyka programistyczna dowodzi, że nie da właściwie napisać programu, który by działał bezbłędnie:

**"*Jeżeli uważasz, że jakiś program komputerowy jest bezbłędny,
to się mylisz –
po prostu nie zauważyłeś jeszcze skutków błędu,
który jest w nim zawarty*".**

Rodzaje błędów

Błędy językowe – powstają w wyniku naruszenia składni języka programowania, którego używamy do zapisania algorytmu, np.:

zamiast	*for i:=1 to N*
jest	*for i:=1 do N*

Możliwe skutki i znaczenie:

- zatrzymanie kompilacji lub interpretacji z komunikatem lub bez
- przerwanie realizacji programu nawet jeżeli kompilator nie wykrył błędu
- są błędy niezbyt poważne i dość łatwe do naprawienia

Rodzaje błędów

Błędy semantyczne – wynikają z niezrozumienia semantyki używanego języka programowania, np. sądzimy, że po zakończeniu iteracji:

for:=1 to N do X[i]:=i

zmienna *i* ma wartość *N*, a nie *N+1*.

Możliwe skutki i znaczenie:

- program nie realizuje poprawnie algorytmu
- są to błędy trudne do wykrycia i potencjalnie groźne, ale są do uniknięcia przy większej wiedzy i starannym sprawdzeniu znaczenia używanych instrukcji

Rodzaje błędów

Błędy logiczne – wynikają ze złego planu rozwiązania problemu, np. stosujemy podczas przeszukiwania tekstu znak "." do określenia końca zdania, a nie przewidzieliśmy, że znak ten może wystąpić również w środku frazy, np.: Na rys. 4 pokazano ...

Możliwe skutki i znaczenie:

- algorytm przestaje być poprawnym rozwiązaniem zadania algorytmicznego
- dla pewnych zestawów danych wejściowych algorytm podaje wyniki niezgodne z oczekiwaniami
- procesor może nie być w stanie wykonać pewnych instrukcji (np. żądamy dzielenia przez 0)
- są to błędy bardzo groźne – mogą być trudne do znalezienia i pozostawać długo w ukryciu nawet w trakcie używania programu w postaci kodu

Rodzaje błędów

Błędy algorytmiczne – wynikają z wadliwie skonstruowanych struktur sterujących, np. niewłaściwych zakresów iteracji, niewłaściwych warunków arytmetycznych i logicznych, błędnego przeniesienia punktu sterowania, itd.

Możliwe skutki i znaczenie*:*

- algorytm dla pewnych dopuszczalnych danych wejściowych daje niepoprawny wynik
- wykonywanie programu realizującego algorytm jest przerywane w trybie awaryjnym
- proces realizujący algorytm nie kończy w normalnym trybie swego zadania

Optymalizacja

Programy komputerowe nie zawsze podlegają procesowi optymalizacji. Powszechna zasada mówi, że jeżeli jakiś program działa stabilnie i dostarcza poprawne wyniki w rozsądnym czasie, można uznać go za produkt gotowy. Często jednak względy praktyczne – szczególnie w dużych projektach – wymagają od programistów bardziej szczegółowego doboru rozwiązań.

Podstawowe kryteria optymalizacji:

- szybkość działania
- ilość zajmowanej pamięci operacyjnej
- ilość zajmowanej pamięci dyskowej
- bezpieczeństwo danych
- jakość kodu źródłowego

Optymalizacja

Szybkość działania

- stosowanie odpowiednich języków programowania
- stosowanie zoptymalizowanych algorytmów
- stosowanie gotowych bibliotek (często udoskonalanych przez lata)
- używanie nowoczesnych translatorów
- ograniczanie operacji wejścia-wyjścia
- ograniczanie zużycia pamięci
- pisanie kluczowych fragmentów kodu w asemblerze
- stosowanie programowania równoległego

Optymalizacja

Ilość zajmowanej pamięci operacyjnej

- ograniczanie liczby zmiennych (szczególnie indeksowanych)
- dynamiczny przydział pamięci
- odpowiedni dobór rozmiarów zmiennych indeksowanych
- przydział tych samych adresów pamięci różnym zmiennym

Ilość zajmowanej pamięci dyskowej

- przemyślana i zwięzła struktura danych
- stosowanie plików binarnych, a nie tekstowych
- kompresja danych

Optymalizacja

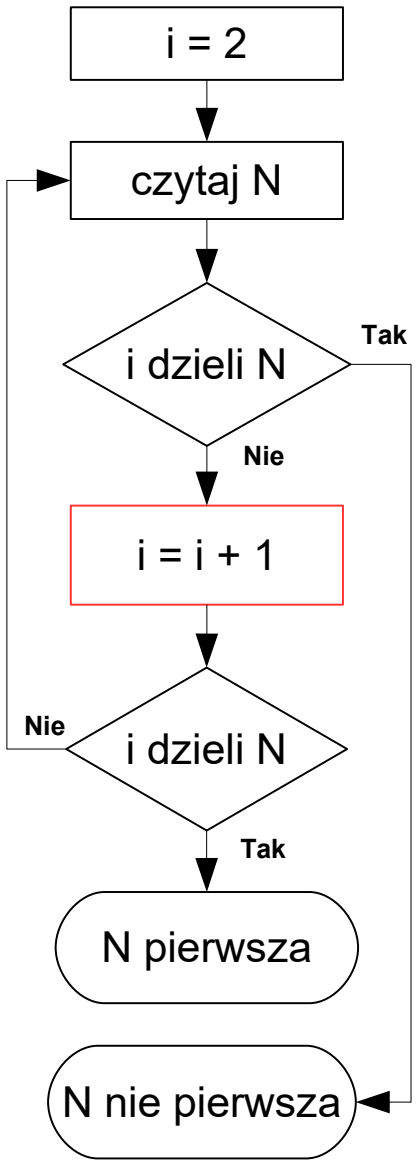
Bezpieczeństwo danych

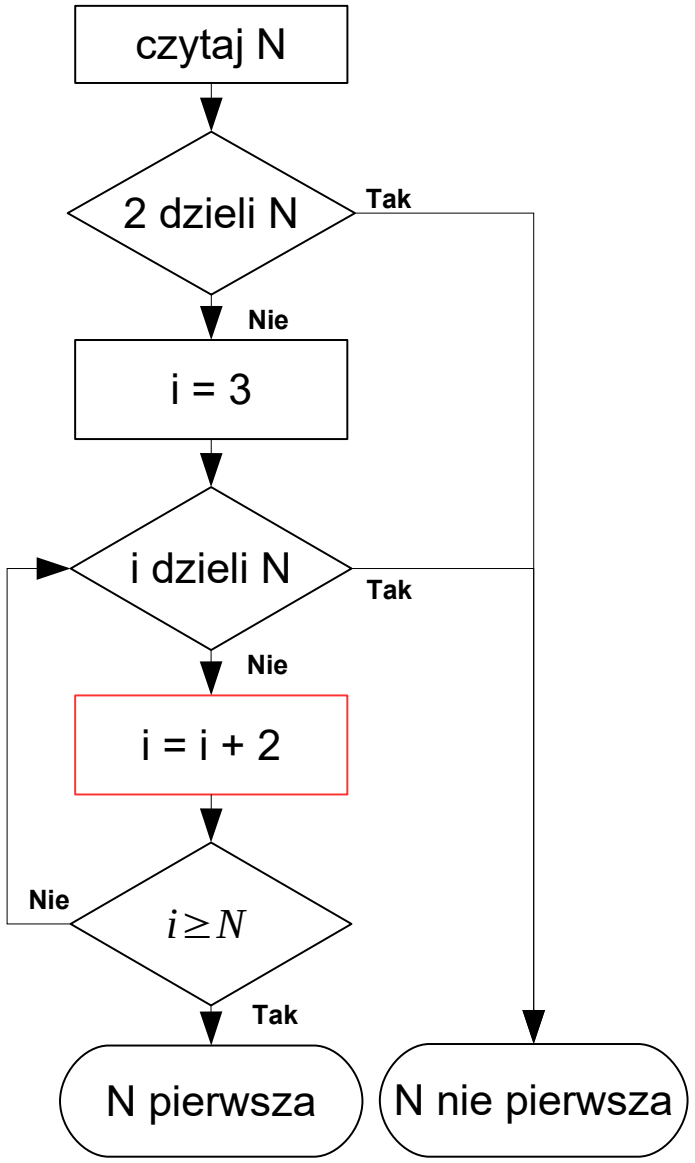
- stosowanie standardowych i przetestowanych formatów zapisu danych
- dokładnie przemyślane zasady zapisu-odczytu i współużytkowania plików
- stosowanie automatycznych modułów archiwizacji stosowanie narzędzi ułatwiających i przyspieszających odbudowę systemu po awarii

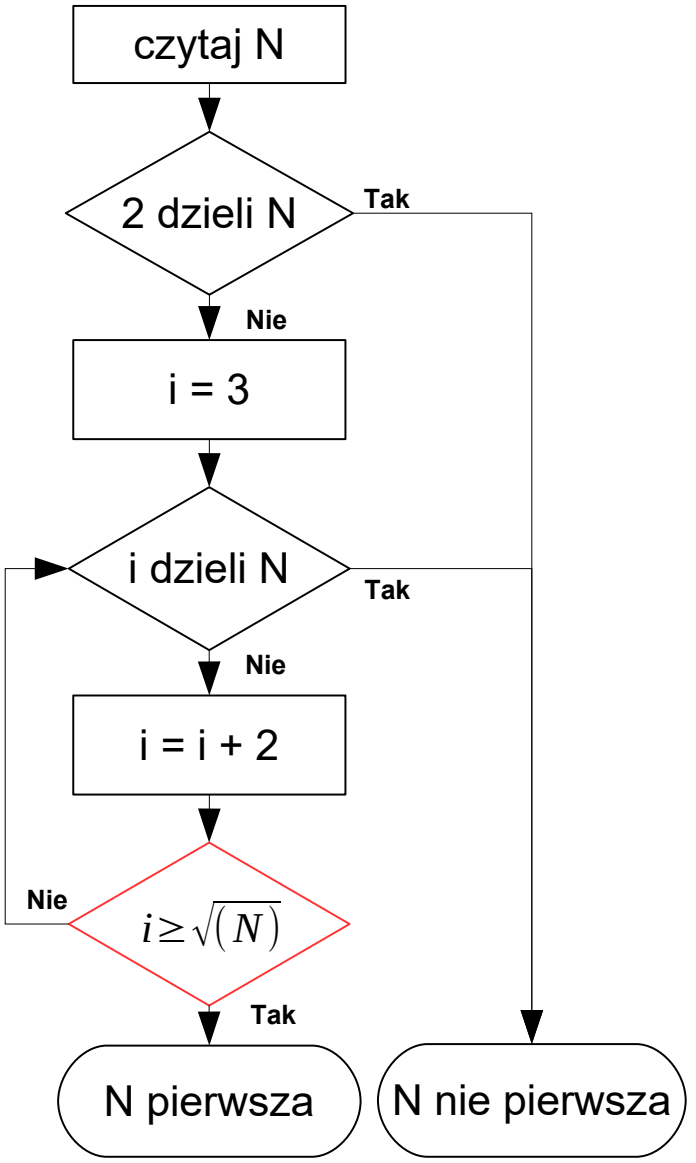
Jakość kodu źródłowego

- podział kodu na logiczne bloki realizujące określone działania
- stosowanie komentarzy
- stosowanie procedur i funkcji (nie wolno dublować zadań!)
- tworzenie dokumentacji kodu
- stosowanie uniwersalnych rozwiązań (niezależnych od systemu operacyjnego)

Przykład optymalizacji algorytmu







Wydział Nauk Technicznych

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Dziękuję

Wojciech Sobieski

Olsztyn, 2001-2021