METODY NUMERYCZNE
I PROGRAMOWANIE
(wykład + pracownia)

CEL prezentacja metod numerycznych

(dla IS – będzie to wstęp do metod numerycznych)

ilustrowana przykładami zastosowań w fizyce +

nauka zasad optymalnego programowania

ćwiczenia - laboratorium nauki programowania (w PASCALU lub w dowolnym innym języku) podstawowych metod numerycznych wykorzystywanych w fizyce ...

PLAN WYKŁADU

1. **Wstęp**
   - **architektura komputera i oprogramowania**
   - **systemy liczenia i reprezentacja informacji w komputerze**
   - **bardzo krótkie repetytorium języka PASCAL**
2. **Błędy obliczeń numerycznych**
3. **Metody numeryczne (analizy i algebry)**
   - **interpolacja**
   - **numeryczne całkowanie i różniczkowanie**
   - **rozwiązywanie równań**
   - **układy równań algebraicznych**
   - **poszukiwanie ekstremów**
   - **równania różniczkowe**
   - **zagadnienie własne dla macierzy – diagonalizacja, znajdowanie wybranych wartości i wektorów własnych**

- analiza fourierowska (transformaty)

- przegląd pozostałych zagadnień

<u>egzamin pisemny w formie testu</u>

dlaczego ciągle jeszcze PASCAL?

- **idealny do nauki programowania strukturalnego**
- **język dydaktyczny**

inne zajęcia komputerowe: FORTRAN, C, Delphi, LabView,
arkusze kalkulacyjne, programowanie równoległe,
C++ Builder-projektowanie aplikacji, aplety Java,
Pascal-II, bazy danych, systemy algebry
symbolicznej (Matlab, Mathematica, Maple),
komputerowe proj. obwodów,
systemy operacyjne, sieci komputerowe,
edytory tekstu i skład dokumentu...

...powtórka...

komputer ...

- **google (>37 mln),**
- **computer (>700 mln...)**

KOMPUTER: procesor (+zegar), rejestry
pamięć operacyjna
urządzenia zewnętrzne (peryferia)

(architektura von Neumanna – 1945) – odróżnienie od kalkulatorów i maszynek liczących
program i dane przechowywane w pamięci (kodowane w postaci liczb), wprowadzone z urządzeń peryferyjnych; procesor sekwencyjnie wykonuje zakodowane w programie rozkazy

procesor: część sterująca + część arytmetyczno-logiczna

wykonawca elementarnych działań (rozkazów*);*

***podstawowe* operacje arytmetyczne i logiczne,**

***transmisja danych do i z pamięci operacyjnej*;**

komunikacja z urządzeniami zewnętrznymi

Działanie komputera to realizacja tzw. uniwersalnej maszyny Turinga, stanowiącej pewien model matematyczny komputera jako przesuwającej się taśmy z zapisanymi na niej danymi oraz głowicy odczytującej dane z taśmy. Stan głowicy i stan odczytywanego pola determinuje kolejny krok działania komputera…

Procesor – ograniczona liczba elementarnych rozkazów
– tzw. l*ista rozkazów procesora*
Przed końcem lat 70. – architektura CISC procesorów (Complex Instruction Set Computers) – kilkaset instrukcji elementarnych,
następnie RISC (Reduced ..) kilkadziesiąt rozkazów elementarnych (rezultat analizy kodów w językach wysokopoziomowych)
Intel – CISC + rdzeń RISC; rozkazy CISC rozkładane na elementarne rozkazy (microops) realizowane przez rdzeń typu RISC

oprogramowanie:

1. **system operacyjny**
2. **aplikacje dodatkowe,**
3. **oprogramowanie użytkownika**
   program + dane (w pamięci w postaci binarnej)

Sposoby programowania:

- **programowanie (wykonanie) sekwencyjne (liniowe),**

proceduralne – oddzielnie dane i oddzielnie procedury „przetwarzające” dane;

cel: wykonanie pewnego z góry określonego zadania

PASCAL, FORTRAN95

- ***programy jednoprzebiegowe***
- ***programy interaktywne***

- **programowanie sterowane zdarzeniowo**

  jednoczesne, asynchroniczne wykonywanie wielu procesów – wielowątkowość ...

- **programowanie zorientowane obiektowo;**

  programowanie obiektowe

  główne elementy języka to - ***obiekty*** – zawierają jednocześnie dane i metody „obsługi” tych danych;

  C++, Java, Smalltalk, Perl

JĘZYKI PROGRAMOWANIA

... najpopularniejsze i historyczne....

podobnie jak języki naturalne: zbiór reguł semantycznych i syntaktycznych pozwalających budować wyrażenia a z nich tworzyć symboliczne zapisy algorytmów

- **<u>języki maszynowe</u> -**

binarna reprezentacja elementarnych rozkazów procesora:

- **zawartość komórki pamięci prześlij do rejestru procesora,**
- **wyzeruj zawartość komórki pamięci,**
- **dodaj binarnie zawartość 2 rejestrów,**

- **porównaj binarną zawartość rejestrów,**
- **instrukcje LOAD i STORE dla bezpośredniej komunikacji pamięci z procesorem**
- **...**

• **<u>asemblery</u> -**
symboliczna reprezentacja (~ 1:1) języka maszynowego

mnemoniki binarnych kodów (*opcodów*) operacji procesora

np.:

mov ah, 0E023h - wprowadź do rejestru AH liczbę szesnastkową E6023

mov ax, ah - przepisz do rejestru AX zawartość rejestru AH

.....

w obydwu przypadkach – składnia zależy od architektury procesora

• **<u>języki wyższych generacji</u> (algorytmiczne)**

*** zdania (=instrukcje) = wiele rozkazów asemblera**

*** konstrukcje języka odpowiadają typowym elementom algorytmów:**

pętle, mechanizmy decyzyjne, zmienne indeksowane, funkcje, moduły, procedury, dane o charakterze strukturalnym

Program tworzony w innych językach programowania niż asembler jest zwykle kompilowany do asemblera, a potem zamieniany na kod binarny przez *asembler*

kiedy korzystać z komputera? - elementarnie

- ***ręczne*** **rozwiązanie problemu jest zbyt trudne (b. długi czas)**

  *** wyznacznik dużej macierzy**

  *** całkowanie funkcji wielu zmiennych danej w postaci tablicy**

  *** odchylenie standardowe dyskretnego rozkładu ~10^6 punktów**

  *** znajdowanie wartości własnych i wektorów własnych dużych macierzy**

  *** rozwiązywanie układów równań różniczkowych cząstkowych**

  *** rozkład sygnału S(t) na częstości (transformata Fouriera)**

- **wielokrotne rozwiązanie (nawet prostego) problemu dla wielu różnych zestawów danych**

  *** wyszukiwanie elementu z dużych baz danych**

  *** modyfikacja dużej liczby rekordów w bazie danych**

  *** rozwiązywanie zagadnienia własnego dla macierzy hamiltonianu w różnych bazach funkcyjnych**

  *** poszukiwanie ekstremów funkcji wielu zmiennych**
  np. minimalizacja energii naprężeń dla układu krystalicznego złożonego z milionów atomów

PROGRAMOWANIE

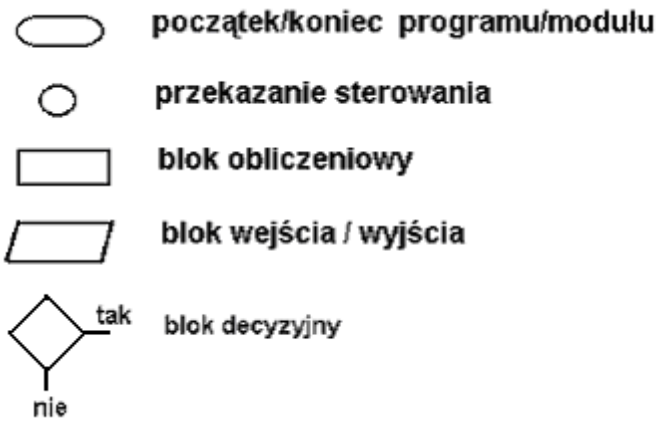
- **zdefiniowanie problemu**
- **szkic rozwiązania**
- **wybór algorytmu metody rozwiązania**
- **programowanie algorytmu (wybór języka)**
  (tworzenie źródła programu)
- **kompilacja programu (tłumaczenie na kod maszynowy)**
- **testowanie (uruchamianie i usuwanie błędów)**

ALGORYTM

sposób postępowania opisujący jak krok po kroku realizować wybraną metodę rozwiązania problemu

formy

- **graficzna (schemat blokowy – *flow charts*)**



- **pseudo-kod**
- **mieszane**

przykłady:

[A]. Wybór największej wartości spośród N wczytanych wartości

(1) wczytaj N wartości na tablicę $\{ a(i) \}_{i=1}^{N}$
(2) zmiennej *amax* przypisz wartość a(1)
(3) dla wszystkich *i* począwszy od 2 do N wykonaj:
(a) sprawdzaj czy a(i) > *amax*
TAK: zmiennej *amax* przypisz wartość a(i)
NIE: przejdź do następnego „i”

Uwaga: (3) to „pętla” (a) to warunek (instrukcja decyzyjna)

Inny przykład:

algorytm sortowania N wartości liczbowych

(od największej do najmniejszej)

- sortowanie przez wyszukiwanie największej wartości -

1. **wczytaj lub wylosuj N wartości**
2. **spośród N wartości wyszukaj największą (a_max)**
3. **przesuń a_max na początek listy**
4. **punkty (2,3) powtarzaj dla kolejnych list pomniejszonych o kolejno: jedną wartość, dwie, trzy, .. itd.**

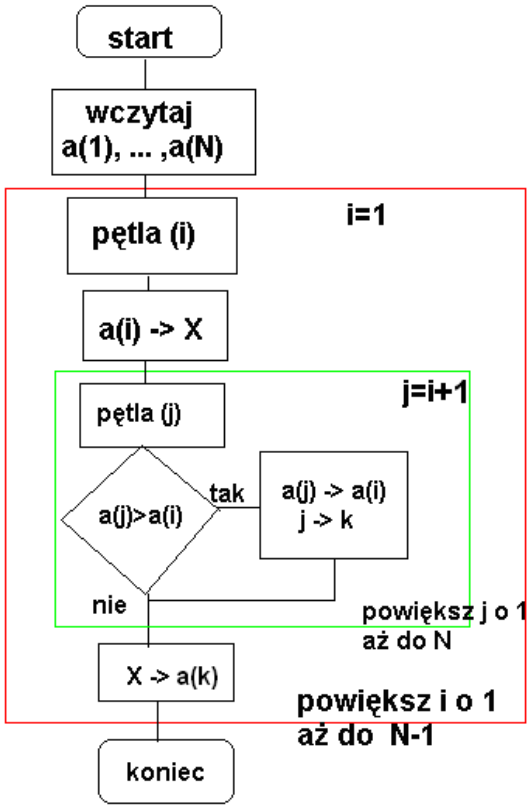
albo

[B]. algorytm sortowania N wartości liczbowych
(od największej do najmniejszej)

(1) wczytaj N wartości na tablicę (zmienną indeksowaną)
a(1), a(2), ... , a(N)
(2) rozpocznij *pętlę* po i=1 do N-1
(2.1) roboczej zmiennej X przypisz wartość a(i)
(2.2) rozpocznij *pętlę* po j od i+1 do N
(2.2.a) jeśli a(j)>a(i) to
- zapisz wartość a(j) w miejscu a(i)
- roboczej zmiennej k przypisz wartość j
(2.2) koniec *pętli* po j
(2.3) elementowi a(k) przypisz wartość X
(2) koniec *pętli* po i
(3) koniec

albo inaczej (w skrócie)

(1) wczytaj N wartości na tablicę (zmienną indeksowaną)
a(1), a(2), ... , a(N)
(2) rozpocznij *pętlę* po i=1 do N-1
(2.1) przywołaj procedurę [A] dla zbioru {a(i)-a(N)}
(2.2) na zmiennej L zapamiętaj miejsce wystąpienia amax
(2.2) elementowi a(L) przypisz wartość a(i)
(2.3) elementowi a(i) przypisz wartość amax
(2) koniec *pętli* po i
(3) koniec



w PASCALU

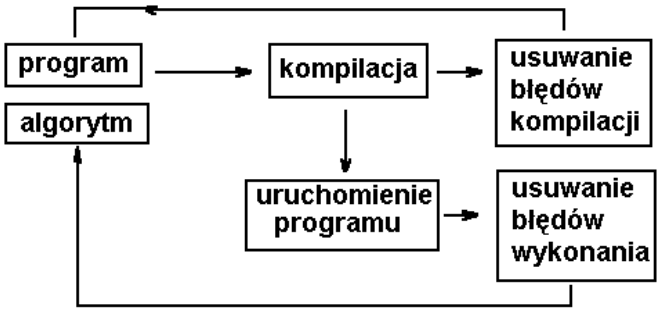
```
program sortowanie;
var x : real;
      i,j,N,k : 1..100;
     a:array[1..100] of real;
begin
read (N);
for i:=1 to N do read(a[i]);
for i:=1 to N-1 do
       begin
       X:= a(i);
       k=i;
       for j:=i+1 to N do if a(j)>a(i) then
                                    begin
                                    a(i):=a(j);
                                    k:=j;
                                    end;
       a(k):=X;
     end;
end.
```

w FORTRANIE (77)

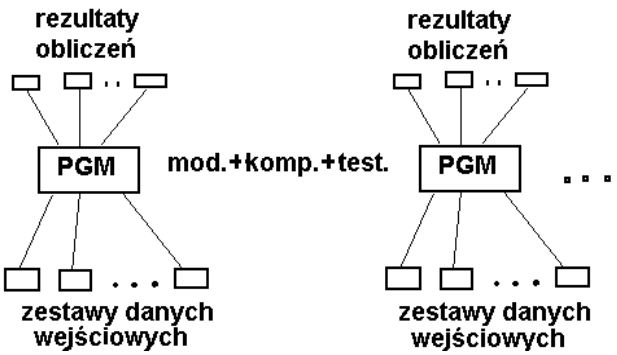
```
      dimension a(100)
      read (*,*) N
      read (*,*) (a(i),i=1,N)
      do i=1,N-1
      k=i
      X=a(i)
      do j=i+1,N
      if ( a(j).gt.a(i) ) then
                        a(i)=a(j)
                        k=j
      endif
      end do
      a(k)=X
      end do
      end
```

***

kolejny etap „programowania” to testowanie programu



wykorzystanie programu (PGM)



PROGRAMOWANIE STRUKTURALNE: -

odzwierciedlanie algorytmu (z jego skomplikowaną strukturą)

- **wykorzystanie *pętli* i *konstrukcji decyzyjnych***
  oraz danych o budowie strukturalnej
- **modularna postać programu**
  funkcje, procedury, podprogramy, obiekty
  biblioteki procedur

programowanie *zstępujące* i *wstępujące*

- **a) budowa szkieletu oprogramowania**
  dekompozycja problemu na mniejsze części
  b) wypełnianie modułami
  tworzonymi na potrzeby danego programu/problemu
- **konstrukcja bibliotek modułów o zunifikowanych metodach odwołań, z których budowane są dowolne programy i aplikacje**

Reprezentacja informacji w komputerze

(krótkie przypomnienie)

system liczenia = sposób reprezentacji wartości liczbowych za pomocą sekwencji skończonej liczby znaków

reprezentacja może być dokładna lub przybliżona

system dziesiętny

znaki: (cyfry) 0 1 2 3 4 5 6 7 8 9

inne A B C D E F G H I J

970 - JHA

system binarny

znaki: (bity) 0 1; (10011000111)

inne __ ---; (__ --- --- ___ ---)

- **każdy system liczenia oparty jest o wartość liczbową zwaną PODSTAWĄ SYSTEMU LICZENIA, będącą liczbą (ilością) znaków w systemie**
- **każdemu znakowi przyporządkowana jest jedna wartość liczbowa**
- **ciągi znaków reprezentują dowolne wartości liczbowe lub przybliżenia do nich**

jeśli w systemie o podstawie *p* reprezentacja liczby ma postać (ciąg symboli)

$$C_n C_{n-1} \ldots C_1 C_o , C_{-1} C_{-2} \ldots C_{-m}$$

to wartość liczbowa

$$L=C_n*p^n + \ldots C_o*p^o + \ldots C_{-m}*p^{-m}$$

w systemie dziesiętnym te znaki-symbole to – cyfry
w systemie binarnym – bity (0, 1)

„ **,** ” – oddziela części ułamkowe – „1” wyróżniona – $n^0 = 1$

max. L całkowita dodatnia reprezentowana na n+1 symbolach (bitach)

$$p^{n+1} - 1$$

najmniejszy ułamek reprezentowany na m symbolach (bitach)

$$p^{-m}$$

$L = C_n*p^n + \ldots C_o*p^o$ - podobne do wielomianu....

optymalizacja programowania...

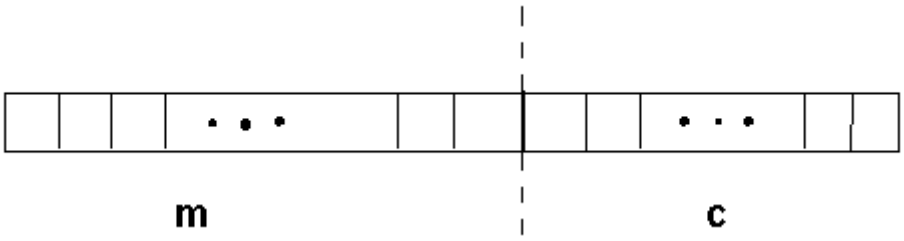
obliczenie L z definicji - $n^2 / 2 + n$ mnożeń

Schemat Hornera

$L=((..(C_n)*p + C_{n-1})*p \ . \ . \ . \ C_1)*p + C_o$

tylko n mnożeń

zmiennoprzecinkowa reprezentacja liczb (rzeczywistych)



mantysa | cecha

$$L = m * p^c$$

m - decyduje o dokładności obliczeń numerycznych

c - decyduje o rzędzie wielkości dostępnych liczb

postać znormalizowana: $p^0 > m >= p^{-1}$

w arytmetyce dziesiętnej: 1.0 – 0.1

w arytmetyce dwójkowej: 1.0 - 0.5

w arytmetyce szesnastkowej: 1.0 – 0.0625 ...

arytmetyka *znak-moduł*

najstarszy bit przeznaczony jest na przechowywanie znaku (0=+, 1= -)

- **utrudnienie w wykonywaniu operacji +/-**
- **(konieczność kontroli bitu znakowego)**

arytmetyka *uzupełnieniowa*

liczby ujemne reprezentowane są przez dopełnienia wartości bezwzględnej danej liczby do liczby większej (następnej) od możliwej do zapisania na danej ilości pozycji (bitów)

przykład: (czterobitowe słowo maszynowe)

3 rep. 0011

-3 rep. 1101 (10000 – 0011)

teraz operacje +/- wykonywane są na wszystkich bitach

3 + 3 0011

\+ 0011

0110 (=6)

3 – (-3) zamiast odejmowania dodajemy uzupełnienie dla liczby -3, (uzupełnieniem do –3 jest 3 tzn. 0011)

0011

\+ 0011

0110 (=6)

3 + (-3)

0011

\+ 1101

0000. (=0)

(dodatkowo zwiększa się o 1 zakres liczb ujemnych)

program REAL_INT (odwrócenie bajtów, przykład dla 7, -7)

(odwracanie kolejności bajów związane jest z przesyłaniem ich do rejestrów procesora i ustawianiem *„od prawej”*)

PASCAL – typ INTEGER – 2 bajty

liczba 7 - `0000000000000111`

uzupełnienie `1111111111111001`

gdyż ich suma daje `1000000000000000`

po odwróceniu bajtów `1111100111111111` **jako -7**