Fortran 90/95
wykład 2 & 3

Janusz Andrzejewski

13/11/11

PLAN

- Alfabet
- Typy danych
  - Typy wewnętrzne
    - Całkowite
    - Rzeczywiste
    - Zespolone
    - Logiczne
    - Znakowe
  - Typy definiowane

Alfabet czyli co wolno a co nie.

- Znaki alfanumeryczne
  - 26 liter alfabetu angielskiego, od A do Z
  - 10 cyfr, od 0 do 9
  - Znak podkreślenia, ”_”
- Znaki specjalne: = + - * / ( ) , . $ ' : (spacja) ! ” % & ; < > ?
- Znaki $ oraz ? nie mają specjalnego znaczenia
- Standard nie wymaga wspomagania dla małych liter
- Jeśli są małe litery, to duże i małe litery są utożsamiane. Fortran nie jest czuły na wielkość liter

Alfabet cd.

- Wszelkie nazwy (zmiennych, stałych itp.) muszą zaczynać się od litery i mogą składać tylko ze znaków alfanumerycznych

  Poprawne nazwy: a12, stol_kuchenny

  Niepoprawne nazwy: 12a, a23*, stol(kuchenny)

- Nazwą może być **dowolne** słowo, we Fortranie nie ma słów zarezerwowanych

- Nazwy zmiennych muszą być odseparowane przynajmniej jedną spacją od innych nazw, słów lub etykiet

  real x

  read 10

  30 **do** k=1, 3

Alfabet cd. cd.

- Opcjonalne spacje w słowach kluczowych:

block data	**double precision**	**else if**
end do	**end function**	**end block data**
end forall	**end module**	**end file**
end interface	**end subroutine**	**end if**
end select	**go to**	**end program**
end where		**end type**
select case		**in out**

- Każda lina może zawierac do 132 znaków
- Aż 39 dodatkowych linii kontynuacji może być użytych

```
x =                              &
    (-y+delta)                   &
     /(2.0*a)
```

Typy danych

- W Fortranie można zdefiniować i użuwać różnych typów
- Typ danych – zbiór wartości oraz operacji które są dozwolone np: wartości -2, -1, 0, 1, 2; operacje 1+1, 1-1
- W Fortranie mamy do dyspozycji:
  - Typy wbudowane (intrinsinc)
  - Typy definiowane
- Typy wbudowane
  - Typy liczbowe
  - Typ logiczny
  - Typ znakowy ('łańcuchy')

Typy liczbowe

- Typ całkowity – reprezentuje liczby całkowite w danym zakresie. Domyślny (default) zakres typ całkowity nie jest sprecyzowany, ale zwykle jest to od -2^(n-1) do 2^(n-1)-1, dla maszyn 32 bitowych daje to od -2 147 483 648 do 2 147 483 647
- Typ rzeczywisty – typ w którym można przedstawić liczby rzeczywiste. Standard mówi o 2 typach rzeczywistych:
  - Typ rzeczywisty pojedynczej precyzji

    REAL x

    REAL :: x2
  - Typ rzeczywisty o zwiększonej precyzji tzw. podwójnej precyzji

    DOUBLE PRECISION y
- Typ zespolony

  COMPLEX zpom

  COMPLEX :: zpom1

Typy nieliczbowe

- Typ logiczny – posiada tylko 2 wartości: **.true.** oraz **.false.**

  LOGICAL prawda
- Typ znakowy – pozwala na zapamiętanie ciągu znaków

  CHARACTER(12) ciag12

  CHARACTER(LEN=15) :: ciag15

Typ całkowity - parametryzacja

- Fortran 95 umożliwia określenie określenie zakresu dla liczby całkowitej, za pomocą funkcji wewnętrznej

  selected_int_kind(zakres), gdzie *zakres* określa liczby całkowite z przedziału (-10^zakres, 10^zakres).

- Można zdefiniować sobie różne rodzaje (**kind**) liczb całkowitych

  I1B = SELECTED_INT_KIND(2)

  I2B = SELECTED_INT_KIND(4)

  I4B = SELECTED_INT_KIND(9)

  I8B = SELECTED_INT_KIND(18)

- Funkcja **selected_int_kind** definuje tzw. rodzaj liczby całkowitej, do odczytu rodzaju liczby służy funkcja **kind(argument)**

- Jeśli nie da się znaleźć takiego rodzaju liczby który spełnia warunek, wówczas wartością funkcji jest -1

- rodzaj liczby nie koniecznie oznacza ilość bajtów potrzebnych do zapamiętania danej liczby

Typ całkowity – stałe nazwane i liczbowe

```
INTEGER, PARAMETER :: a=5
INTEGER, PARAMETER :: b=2*3, c=b/2
INTEGER, PARAMETER :: i1b=selected_int_kind(2)
INTEGER(i1b), PARAMETER :: d=3
```

Definicje stałych liczbowych:

- 23 - liczba 23 o domyślnym type całkowitym
- 23_i1b - liczba 23 o typie i1b liczby całkowitej
- 23_1 - bardzo niebezpieczne,

Typ całkowity

- Ogólna deklaracja typu całkowitego

  INTEGER [([KIND =] rodzaj)]

INTEGER x

INTEGER (KIND=dlugi) Caly

INTEGER (SHORT) :: wybor

INTEGER(KIND=selected_int_kind(5)) :: aa

- Operacje zdefiniowane na typie całkowitym
  - Arytmetyczne: +(dodawanie) -(odejmowanie) *(mnożenie) /(dzielenie) **(potęgowanie)
  - Relacji: <, <=, ==, /=, >= >
  - Jednoargumentowe (unary) -(minus) +(plus)
- Jeżeli wszystkie operandy są typu całkowitego, wynikiem wyrażenia jest typ całkowity

Typ rzeczywisty

- Do określenia rodzaju (kind) liczby rzeczywistej służy funkcja

  selected_real_kind([p] [, r]), gdzie p – oznacza precyzję liczby rzeczywistej, r – zakres liczby rzeczywistej.

- W przypadku gdy nie da się znaleźć odpowiedniego typu danych funkcja ta zwraca wartośći:
  - -1 – gdy nie ma odpowiedniej precyzji
  - -2 – gdy nie ma odpowiedniego zakresu
  - -3 - gdy nie ma odpowiedniej precyzji i zakresu

Przykłady:

selected_real_kind(5, 10)

selected_real_kind(p=10)

selected_real_kind(r=20)

Typy rzeczywiste - przykłady

```
integer, parameter ::long=selected_real_kind(9, 99)
  DOUBLE PRECISION   dx, dy
  REAL         rx, ry
  real :: a, b
  real(long)::  c, d
  real(8)  :: x, y  ! Ryzykownie, najprawdopodobniej jest to double
                    !precision


   A=1.0
   B=1.0e0;   B1=1.0d0;
  c=2.0_long
  X=4.0_8   ! ryzykownie
```

REAL cd.

- Ogólna definicja typu rzeczywistego

  REAL [([KIND=] rodzaj)]

REAL x, y

REAL(KIND=wyoska) :: x1, x2

REAL(selected_real_kind(8,70)) :: aa

- Operacje zdefiniowane na typie rzeczywistym
  - Arytmetyczne: +(dodawanie) -(odejmowanie) *(mnożenie) /(dzielenie) **(potęgowanie)
  - Relacji: <, <=, ==, /=, >= >
  - Jednoargumentowe (unary) -(minus) +(plus)
- Jeżeli jeden z operandów jest typu rzeczywistego, wynik wyrażenia jest typu rzeczywistego

Typ COMPLEX

Typ COMPLEX składa się z 2 licz rzeczywistych, więc do określenia typu COMPLEX używa sie tego samego sposobu co do określenia typu dla licz rzeczywistych

```
 COMPLEX  cz

 DOUBLE COMPLEX  dz, zz

integer, parameter :: lzsp=selected_real_kind(9, 99)

complex :: z1

complex(lzsp) ::  z2


  Z1=(1.0, 2.0)

  z2=(1.0_lzsp, 3.0_lzsp)
```

COMPLEX

- Ilość rodzajów (kind) liczb zespolonych musi być taka sama jak ilość rodzajów liczb rzeczywistych
- Ilość pamięci potrzebnej na przechowywanie liczby zespolonej musi być dwa razy większa niż ilość pamięci potrzebnej na przechowywanie liczby rzeczywistej o tym samym rodzaju co liczba zespolona

  COMPLEX [([KIND=] rodzaj)]
- Operacje zdefiniowane na typie zespolonym
  - Arytmetyczne: +(dodawanie) -(odejmowanie) *(mnożenie) /(dzielenie) **(potęgowanie)
  - Relacji: ==, /=
  - Jednoargumentowe (unary) -(minus) +(plus)
- Jeżeli jeden z operandów jest typu zespolonego, wynik wyrażenia jest typu zespolonego

Kilka funkcji związanych z modelem liczb dostępnych w implementacji

- **digits(x)** - dla x całkowitych albo rzeczywistych, zwraca ilość cyfr znaczących reprezentowanych dla argumentu typu x
- **range(x) –** dla x całkowitych, rzeczywistych lub zespolonych, zwraca zakres liczby reprezentowanej przez argument x
- **huge(x) –** dla x całkowitych lub rzeczywistych, zwraca najwiekszą liczbę w modelu reprezentowanym przez argument x
- **tiny(x)** - dla x rzeczywistej, zwraca najmniejsza dodatnią liczbę reprezentowaną przez argument x
- **precision(x)** - dla x rzeczywistej lub zespolonej, ilość cyfr znaczących w zapisie dziesiętnym dla licz reprezentowanych przez argyment x
- **maxexponent(x), minexponent(x)** - dla x rzeczywistych, największy lub najmniejszy wykładnik dla liczb reprezentowanych przez argument x

Typ logiczny

- Służy do zapamiętywania wartości wyrażeń logicznych
- Ma tylko dwie wartości prawdę - **.true.** oraz fałsz **.false.**

  LOGICAL :: prawda, klamstwo

  prawda=.true.

  klamstwo=.false.
- Kompilator musi posiadać implementację tylko jednego domyślnego typu logicznego
- Nie ma żadnej wewnętrznej funkcji która by pozwalała na wybór rodzaju typu logicznego
- Ogólna deklaracja typu logicznego

  LOGICAL [([KIND=] rodzaj)]

Typ znakowy

- Służy do przechowywania napisów

  CHARACTER(len=dlugosc) :: zdanie

  - okresla zmienna *zdanie* które może zapamiętać dowolny ciąg znaków o długości *dlugosc*

  CHARACTER(LEN=23) :: zdanie23

  CHARACTER(24) :: zdanie24

  CHARACTER*25 :: zdanie25

  zdanie23='Ala ma kota !!'

  zdanie24=”I don't want”

CHARACTER

- Kompilator musi posiadać implementację tylko jednego domyślnego typu znakowego
- Możliwe deklaracje
  - CHARACTER (dlugosc[, [KIND=]rodzaj])
  - CHARACTER(LEN=dlugosc [, KIND=rodzaj])
  - CHARACTER(KIND=rodzaj [, LEN=dlugosc])
  - CHARACTER*dlugosc
- Jeśli pominiemy długość, wtedy domyślną długością jest 1
- Jeśli długość ma wartość ujemną, wtedy założoną długością jest zero

CHARACTER cd.

- Do wybory rodzaju typu znakowego służy funkcja

  selected_char_kind(nazwa**)**, gdzie w szczególności nazwa

  - DEFAULT – domyślny zbiór znaków, tyle co KIND('A')
  - ASCII – zbiór znaków ASCII
  - ISO_10646 – międzynarodowy zbiór znaków – 4 bajtowy, w którym każdy z międzynarodowych znaków ma swoją reprezentację
- W przypadku gdy danego zbioru znaków nie zaimplementowano, funkcja selected_char_kind zwraca wartość -1

Typy danych a pamięć

- Domyślny typ rzeczywisty musi zajmować tyle pamięci ile domyślny typ całkowity
- Typ DOUBLE PRECISION musi zajmować dwa razy więcej pamięci niż domyślny typ rzeczywisty
- Typ complex zajmuje dwa razy więcej pamięci niż taki sam typ rzeczywisty
- Domyślny typ logiczny musi zajmować tyle samo pamięci co domyślny typ całkowity
- Ilość pamięci potrzebnej do przechowywania znaku nie musi być taka sama jak dla domyślnego typu całkowitego

Typ danych definiowany

W Fortranie 90/95 można definiować własne nowe struktury danych.

Postać:

```
type nazwa_SD
    typ_zmiennej :: nazwa_zmiennej
     ...
end type nazwa_SD
```

Przykład:

```
type osoba
    character(len=20) :: imie
    real  :: wiek
    integer :: id
end type osoba
```

Przykład

Deklaracja zmiennych o własnym typie danych

```
type(osoba)  :: me
```

Operacje na strukturach danych.

Dostęp do poszczególnych pól SD poprzez operator **„%”**

```
me%imie=”Janusz”
me%wiek=podaj_wiek()
me%id+9   ! jako wyrażenie
```

Dziękuję za uwagę