FORTRAN 90
wykład 6

Janusz Andrzejewski

06/12/11

PLAN

Struktury decyzyjne

Instrukcje

- GO TO
- CONTINUE
- STOP

Struktury decyzyjne

```
[name:] IF (wyrażenie_logiczne) THEN
        [blok instrukcji]
END IF  [name]
```

Np:

```
IF (A .GT. 0.0) THEN
        PRINT*,'A jest dodatnie'
END IF
```

W przypadku gdy WL ma Wartość .TRUE. Wykonywany jest ciąg instrukcji, w przeciwnym razie wykonywana jest następna instrukcja po END IF

Postać uproszczona:

```
IF (wyrażenie_logiczne) jedna_intrukcja
```

Np:

```
IF (A .GT. 0.0) PRINT*,'A jest dodatnie'
```

Struktury decyzyjne

```
[name:] IF (wyrazenie_logiczne) THEN
      [blok_instrukcji1 ]
ELSE [name]
       [blok_instrukcji2]
END IF [name]
```

Ten ciąg instrukcji jest wykonywany gdy WL ma wartość .TRUE.

Ten ciąg instrukcji jest wykonywany w przypadku gdy WL ma wartość .FALSE.

np.

```
    IF (A .GT. 0.0) THEN
          PRINT*,'A jest liczba dodatnia'
    ELSE
          PRINT*,'A jest liczba ujemna lub zerem'
    END IF
```

Struktury decyzyjne

```
IF (wyr_log1) THEN
        [ciag_instr1]
ELSE IF (wyr_log2) THEN
        [ciag_instr2]
ELSE
        [ciag_instrN]
END IF
```

Ten fragment bloku instrukcji warunkowej może być powtórzony dowolną ilość razy

Ten fragment bloku instrukcji warunkowej może się znajdować tylko raz, wykonywany jest tylko w przypadku gdy żaden z warunków nie jest spełniony

UWAGA:

W tym wyrażeniu tylko jeden ciąg instrukcji może być wykonany. Jeśli prawdziwe jest wyr_log1 to wykona się tylko ciag_instr1, a następnie sterowanie zostanie przekazane do pierwszej instrukcji po END IF. W przypadku gdy żaden w wyrażeń logicznych nie jest prawdziwy, wykona sie ciąg instrukcji będących po pojedynczym słowie kluczowym ELSE

Struktury decyzyjne

```
IF (A .GT. 0.0) THEN
      PRINT*,'A jest dodatnie'
ELSE IF (A .LT. 0.0) THEN
       PRINT*,'A jest ujemne'
ELSE
        PRINT*,'A jest rowne zero'
END IF
```

Konstrukcja CASE

```
[name:] SELECT CASE (expr)
  [  CASE selector [name]
        blok_instrukcji ]
 END SELECT [name]
```

Ten fragment konstrukcji CASE może być powtórzony dowolną ilość razy

Wyrażenie **expr** musi być skalarem typu znakowego, logicznego lub całkowitego
Wartość wyrażenia jest obliczana na początku konstrukcji

Selektor:

- Musi być takiego samego typu jak **expr**
- W przypadku typu znakowego długość expr oraz selektora mogą się różnić, ale rodzaj (KIND) już nie
- W przypadku typu logicznego i całkowitego rodzaj (KIND) wyrażenia i selektora mogą się różnić
- Wartości selektorów muszą być wzajemnie różne, tak aby tylko obliczona wartość wyrażenia pasowała do nie więcej niż jednego selektora

Postacie selektora

- **case(1) –** podanie pojedynczej wartości
- **case(low:high)** - podanie zakresu zmiennej. Wartość **low** albo **high** może być pominięta, ale nie obie naraz
- **case(1,2, 7, 10:17) –** można podawać kilka nieprzykrywających się zakresów
- **case default –** jest równoważnemu określeniu wszystkich pozostałych wartości wyrażenia które nie zostały zdefiniowane przez poprzednie wartości selektorów

Przykłady

```
Integer  :: number, n_sign
Read(*,*) number
celect case (number)
case(:-1)
    n_sign=-1
case(0)
    n_sign=0
case(1:)
     n_sign=1
end select
```

```
select case (ch)
case('c', 'd', 'r')
      ch_type=.true.
case('i':'n')
       int_type=.true.
case default
     real_type=.true.
end select
```

Etykiety

Etykiety (ang. labels) ciąg maksymalnie 5 cyfr, z których przynajmniej jedna musi być rożna od zera, umieszczonych w kolumnach od 1 do 5. Etykiety służą jako metoda do odwoływania się do danej części programu.

UWAGI:

Każda instrukcja w Fortranie może posiadać etykietę.

Tylko etykiety instrukcji czynnych oraz etykieta do instrukcji **FORMAT** może służyć jako etykieta dzięki której można się odwołać, ale i od tej reguły są wyjątki

Początkowe zera oraz spacje w etykiecie są ignorowane

W danym segmencie (FUNCTION, SUBROUTINE) nie mogą pojawić się dwie takie same etykiety, natomiast w różnych segmentach mogą być takie same etykiety

Instrukcja skoku bezwarunkowego

Postać

```
GO TO etykieta
```

np:

```
 25   READ(*,*) a
       PRINT*,'Wartosc a = ',a
      GO TO 25
```

UWAGI:

- Instrukcja GO TO można wyjść z dowolnego miejsca w programie, ale nie do dowolnego miejsca w programie można wejść.
- Instrukcją GO TO można poruszać się tylko w ramach tego samego segmentu, nie można przeskakiwać pomiędzy segmentami

GO TO – dalsze uwagi

UWAGI (ciąg dalszy):

- Instrukcją GO TO nie można wejść do instrukcji do „środka” instrukcji warunkowej IF ... THEN lub innych „bloków”
- Instrukcją GO TO nie można wejść do „środka” instrukcji cyklicznej (pętle)

```
10      IF (A .GT. 1.0) THEN
11           PRINT*,'A jest dodatnie'
12      ELSE IF (A .GT. 5.0 .AND. A .LT. 10.0) THEN
13           PRINT*,'A jest z przedzialu od 5 do 10'
14      ELSE
15             PRINT*,'A jest jakies dziwne'
16       END IF
```

Instrukcja CONTINUE

Instrukcja **CONTINUE** jest instrukcją pustą tzn. instrukcją która nic nie robi, służy jako instrukcja która może mieć etykietę oraz jako instrukcja kończąca inne instrukcje.

```
01 0    CONTINUE
        PRINT*,'PODAJ A'
        READ(*,*)A
        PRINT*,'A wynosi ',A
        GO TO 10
```

Instrukcja STOP

Instrukcja STOP przerywa działanie programu. Składnia

STOP

STOP liczba

STOP lancuch

Liczba – dowolna liczba, składająca się nie więcej niż z 5 cyfr

Instrukcja STOP poza zatrzymaniem się programu, spowoduje wyświetlenie sie w przypadku obecności liczby lub lancucha. Instrukcja ta może znajdować sie w głównym segmencie, ale także może znajdować się w dowolnie innym segmencie SUBROUTINE lub FUNCTION

Dziękuję za uwagę

Obliczenia cykliczne - pętle

Obliczenia cykliczne są to obliczenia które są wykonywane pewną ilość razy przez ten sam kod programu. Ilość iteracji pętli, może być ustalona na początku pętli, może także zależeć od bieżącej sytuacji tzn. za każdą iteracją ustalane jest czy pętla ma się dalej kontynuować czy nie.

Typy pętli

- Pętla typu while-do
- Pętla typu do-while
- Pętla typu do