FORTRAN-77 управляющие операторы

Михайленко К. И.

Башгосуниверситет Институт механики Уфимского научного центра РАН

(4)

Пример программы

```
prog11.f
```

```
program prog11
implicit none
integer a, b, N
read *, a, b
if (a .gt. b) then
 N = a
else
 N = b
end if
print *, 'Наибольшее из введённых чисел:', N
end
```

б операций сравнения

"равно"	.EQ.	A .EQ. $B \Leftrightarrow a == b$
"не равно"	.NE.	A .NE. B \Leftrightarrow a != b
"меньше"	.LT.	A .LT. B \Leftrightarrow a < b
"меньше или равно"	.LE.	A .LE. B \Leftrightarrow a <= b
"больше"	.GT.	A .GT. B \Leftrightarrow a $>$ b
"больше или равно"	.GE.	A .GE. B \Leftrightarrow a $>=$ b

б операций сравнения

"равно"	.EQ.	A .EQ. $B \Leftrightarrow a == b$
"не равно"	.NE.	A .NE. B \Leftrightarrow a != b
"меньше"	.LT.	A .LT. B \Leftrightarrow a < b
"меньше или равно"	.LE.	A .LE. B \Leftrightarrow a <= b
"больше"	.GT.	A .GT. B \Leftrightarrow a $>$ b
"больше или равно"	.GE.	A .GE. B \Leftrightarrow a $>=$ b

б операций сравнения

"равно"	.EQ.	A .EQ. B \Leftrightarrow a == b
"не равно"	.NE.	A .NE. B \Leftrightarrow a != b
"меньше"	.LT.	A .LT. B \Leftrightarrow a < b
"меньше или равно"	.LE.	A .LE. B \Leftrightarrow a <= b
"больше"	.GT.	A .GT. B \Leftrightarrow a $>$ b
"больше или равно"	.GE.	A .GE. B \Leftrightarrow a $>=$ b

б операций сравнения

"равно"		A .EQ. $B \Leftrightarrow a == b$
"не равно"	.NE.	A .NE. B \Leftrightarrow a != b
"меньше"	.LT.	A .LT. B \Leftrightarrow a < b
"меньше или равно"	.LE.	A .LE. B \Leftrightarrow a <= b
116		
"больше"	. G'1' .	A .GT. B \Leftrightarrow a > b

б операций сравнения

"равно"		A .EQ. $B \Leftrightarrow a == b$
"не равно"	.NE.	A .NE. B \Leftrightarrow a != b
"меньше"	.LT.	A .LT. B \Leftrightarrow a < b
"меньше или равно"	.LE.	A .LE. B \Leftrightarrow a <= b
"больше"		A .GT. B \Leftrightarrow a > b
"больше или равно"	.GE.	A .GE. B \Leftrightarrow a $>=$ b

б операций сравнения

"равно"	.EQ.	A .EQ. B \Leftrightarrow a == b
"не равно"	.NE.	A .NE. B \Leftrightarrow a != b
"меньше"	.LT.	A .LT. B \Leftrightarrow a < b
"меньше или равно"	.LE.	A .LE. B \Leftrightarrow a <= b
"больше"	.GT.	A .GT. B \Leftrightarrow a > b
"больше или равно"	.GE.	A .GE. B ⇔ a >= b

б операций сравнения

"равно"	.EQ.	A .EQ. $B \Leftrightarrow a == b$
"не равно"	.NE.	A .NE. B \Leftrightarrow a != b
"меньше"	.LT.	A .LT. B \Leftrightarrow a < b
"меньше или равно"	.LE.	A .LE. B \Leftrightarrow a <= b
"больше"	.GT.	A .GT. B \Leftrightarrow a > b
"больше или равно"	.GE.	A .GE. B ⇔ a >= b

Условные операторы

Операторы цикла "Малые"операторы Логические выражения Структурный оператор IF Оператор ELSE IF Логический условный оператор IF

отрицание ("не")	.NOT.	.NOT. L (унарная операция)
"или"	.OR.	
II N II	. AND.	
эквивалентность	.EQV.	
неэквивалентность	.NEQV.	

Условные операторы Операторы цикла "Малые" операторы

Логические выражения Структурный оператор IF Оператор ELSE IF Логический условный оператор IF

отрицание ("не") "или"	.NOT.	.NOT. L (унарная операция)
II N II	.AND.	
эквивалентность	.EQV.	
неэквивалентность	.NEQV.	

Условные операторы Операторы цикла "Малые" операторы

Логические выражения Структурный оператор IF Оператор ELSE IF Логический условный оператор IF

отрицание ("не")	.NOT.	.NOT. L (унарная операция)
"или"	.OR.	
"u"	.AND.	
эквивалентность	.EQV.	
неэквивалентность	.NEQV.	

Условные операторы Операторы цикла

Логические выражения Структурный оператор IF Оператор ELSE IF "Малые"операторы Логический условный оператор IF

отрицание ("не")	.NOT.	.NOT. L (унарная операция)
"или"	.OR.	
"u"	.AND.	
эквивалентность	.EQV.	
неэквивалентность	.NEQV.	

Условные операторы Операторы цикла

"Малые"операторы

Логические выражения Структурный оператор IF Оператор ELSE IF Логический условный оператор IF

отрицание ("не")	.NOT.	.NOT. L (унарная операция)
"или"	.OR.	
"u"	.AND.	
эквивалентность	.EQV.	
неэквивалентность	.NEQV.	

Результаты логических операций

L1 =	.TRUE.	.TRUE.	.FALSE.	.FALSE.
L2 =	.TRUE.	.FALSE.	.TRUE.	.FALSE.
.NOT. L1	.FALSE.	.FALSE.	.TRUE.	.TRUE.
L1 .OR. L2	.TRUE.	.TRUE.	.TRUE.	.FALSE.
L1 .AND. L2	.TRUE.	.FALSE.	.FALSE.	.FALSE.
L1 .EQV. L2	.TRUE.	.FALSE.	.FALSE.	.TRUE.
L1 .NEQV. L2	.FALSE.	.TRUE.	.TRUE.	.FALSE.

Порядок выполнения логических операций

- .NOT.
- 2 .AND.
- .OR.
- .EQV. и .NEQV.
- Порядок операций может быть изменён посредством скобок ()

Порядок выполнения логических операций

- ① .NOT.
- 2 .AND.
- .OR.
- EQV. и .NEQV.
- Порядок операций может быть изменён посредством скобок ()

Общая форма структурного IF

```
IF (логическое выражение) THEN операторы при .TRUE. ELSE операторы при .FALSE. END IF
```

```
из prog11.f

if (a .gt. b) then

N = a

else

N = b

end if
```

Общая форма структурного IF

```
IF (логическое выражение) THEN операторы при .TRUE.
ELSE операторы при .FALSE.
END IF
```

```
if (a .gt. b) then

N = a

else

N = b

end if
```

Сокращённая форма структурного IF

IF (логическое выражение) THEN операторы при .TRUE.

END IF

```
read *, a, b
print *, 'Max:', N
```

Сокращённая форма структурного IF

IF (логическое выражение) THEN операторы при .TRUE.

END IF

```
prog12.f
        program prog12
        implicit none
        integer a, b, N
        read *, a, b
        N = b
        if (a .gt. b) then
          N = a
        end if
        print *, 'Max:', N
        end
```

Вложенные операторы IF

$$f(x) = \begin{cases} 0, & x < 0; \\ x, & 0 \le x < 1; \\ x^2, & 1 \le x < 10; \\ 2x^3, & x \ge 10 \end{cases}$$

Вложенные операторы IF

$$f(x) = \begin{cases} 0, & x < 0; \\ x, & 0 \le x < 1; \\ x^2, & 1 \le x < 10; \\ 2x^3, & x \ge 10 \end{cases}$$

```
if (x .lt. .0) then
 y = 0.
else
 if (x .lt. 1.) then
   y = x
 else
   if (x .lt. 10.) then
     y = x**2
   else
     y = 2. * x**3
   end if
  end if
end if
```

Оператор ELSE IF

$$f(x) = \begin{cases} 0, & x < 0; \\ x, & 0 \le x < 1; \\ x^2, & 1 \le x < 10; \\ 2x^3, & x \ge 10 \end{cases}$$

```
if (x .lt. .0) then
  y = 0.
else if (x .lt. 1.) then
  y = x
else if (x .lt. 10.) then
  y = x**2
else
  y = 2. * x**3
end if
```

Логический условный оператор IF

IF (логическое выражение) оператор при .TRUE.

Логический условный оператор IF

IF (логическое выражение) оператор при .TRUE.

```
prog13.f
     program prog13
     implicit none
     integer a, b, N
```

read *, a, b

$$N = b$$

if (a .gt. b) $N = a$

print *, 'Наибольшее из введённых чисел:', N

end

Ограничения

- Нельзя передавать управление на операторы ELSE, ELSE IF, END IF
- Нельзя передавать управление извне внутрь условного оператора

Пример программы

```
prog14.f
        program prog14
        implicit none
        integer i, a, sum
        sum = 0
        do 10 i = 1, 10
          read *, a
          sum = sum + a
     10 continue
        print *, 'Сумма 10 чисел:', sum
        end
```

Условные операторы Операторы цикла "Малые"операторы Оператор цикла DO ... END DO Вложение операторов цикла DO ... END DO Вложение операторов цикла DO Оператор цикла DO WHILE

Оператор цикла DO

DO метка параметр = нач, кон тело цикла метка последний оператор цикла

Оператор-пустышка CONTINUE

Условные операторы Операторы цикла "Малые"операторы Оператор цикла DO ... END DO Вложение операторов цикла DO ... END DO Вложение операторов цикла DO Оператор цикла DO WHILE

Оператор цикла DO

DO метка параметр = нач, кон тело цикла метка последний оператор цикла

Оператор-пустышка CONTINUE

Оператор цикла DO Форма оператора цикла DO ... END DO Вложение операторов цикла Общая форма оператора цикла DO Oператор цикла DO WHILE

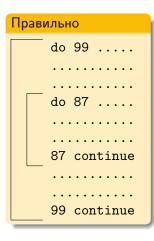
Oператор цикла DO ... END DO (Fortran 90)

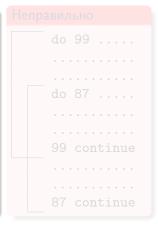
```
DO параметр = нач, кон
тело цикла
END DO
```

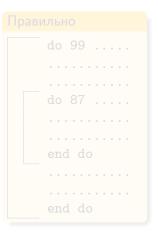
Оператор цикла DO Форма оператора цикла DO ... END DO Вложение операторов цикла Общая форма оператора цикла DO

Оператор цикла DO WHILE

Вложение операторов цикла



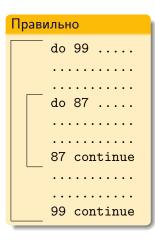


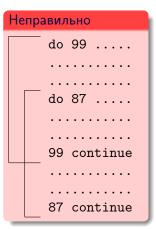


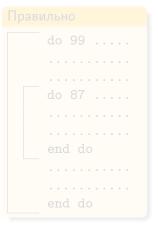
Оператор цикла DO Форма оператора цикла DO ... END DO Вложение операторов цикла Общая форма оператора цикла DO

Оператор цикла DO WHILE

Вложение операторов цикла

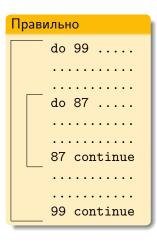


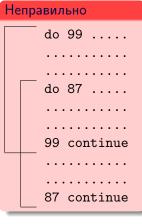


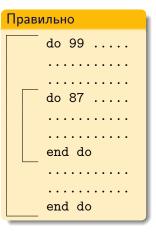


Оператор цикла DO
Форма оператора цикла DO ... END DO
Вложение операторов цикла
Общая форма оператора цикла DO
Оператор цикла DO WHILE

Вложение операторов цикла







Общая форма оператора DO

DO метка параметр = нач, кон, шаг тело цикла метка CONTINUE

DO параметр = нач, кон, шаг тело цикла END DO

LIND DO

Количество проходов цикла:

$$N = \operatorname{int}\left(rac{\mathsf{KOH} - \mathsf{HaY}}{\mathsf{War}} + 1
ight)$$

 Параметр цикла допустимо использовать в теле цикла; значение параметра цикла менять нельзя.

```
do i = 1, 20
   j = i**2
   print *, i, j
end do
```

Значения нач, кон, шаг вычисляются перед началом цикла; полученные значения хранятся на протяжении всего цикла.

```
1 6
2 7
3 8
4 9
5 10
```

• На выходе из цикла параметр цикла сохраняет последнее присвоенное значение.

6

Параметр цикла сначала получает очередное значение, и лишь потом производится проверка, не превышено ли значение $\kappa o \mu$.

Значение параметра цикла после выхода из цикла:

параметр
$$=$$
 нач $+N*$ шаг

- Нельзя передавать управление в тело цикла извне.
- О Цикл можно размещать внутри блоков IF, ELSE, ELSE IF, но недопустимо размещение начала цикла в одном блоке, а окончания в другом.
- Аналогично для обратной ситуации размещения условного оператора в теле цикла.
- Рекомендация Fortran 90: недопустим нецелый пераметр цикла.
 - Стандарт Fortran 95: запрещён нецелый пераметр цикла.

Оператор цикла DO WHILE (Fortran 90)

DO WHILE (логическое выражение) тело цикла

END DO

Выполняется, пока *погическое выражение* имеет значение .TRUE.

Оператор безусловного перехода

GOTO метка

- Нельзя передавать управление внутрь цикла (но можно использовать GOTO в пределах тела цикла).
- Нельзя передавать управление внутрь блоков условного оператора (но можно использовать GOTO в пределах отдельного блока).
- Нельзя давать метки операторам ELSE, ELSE IF, END IF

Оператор временного останова

PAUSE символьная константа

Способ возобновления работы программы определяется компилятором.

```
pause.f
    program esuap
    pause 'Пауза в программе'
    print *, 1
    end
```

Оператор временного останова

```
$ g77 -o pause.g77 pause.f
$ ifort -o pause.if pause.f
$ ./pause.g77
PAUSE Пауза в программе statement executed
To resume execution, type go. Other input will terminate the job.
go
Execution resumes after PAUSE.
$ ./pause.if
Пауза в программе
PAUSE prompt>
$ ./pause.if
Пауза в программе
PAUSE prompt> end
$
```

Оператор завершения работы

STOP символьная константа

STOP целая константа (число)