

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
```

6 операций сравнения

"равно"	.EQ.	$A \text{ .EQ. } B \Leftrightarrow a == b$
"не равно"	.NE.	$A \text{ .NE. } B \Leftrightarrow a \neq b$
"меньше"	.LT.	$A \text{ .LT. } B \Leftrightarrow a < b$
"меньше или равно"	.LE.	$A \text{ .LE. } B \Leftrightarrow a \leq b$
"больше"	.GT.	$A \text{ .GT. } B \Leftrightarrow a > b$
"больше или равно"	.GE.	$A \text{ .GE. } B \Leftrightarrow a \geq b$

Результат операции сравнения — значение типа LOGICAL

6 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.	$A .GE. B \Leftrightarrow a >= b$

Результат операции сравнения — значение типа LOGICAL

6 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.	$A .GE. B \Leftrightarrow a >= b$

Результат операции сравнения — значение типа LOGICAL

6 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.	$A .GE. B \Leftrightarrow a >= b$

Результат операции сравнения — значение типа LOGICAL

6 операций сравнения

"равно"	.EQ.	$A \text{ .EQ. } B \Leftrightarrow a == b$
"не равно"	.NE.	$A \text{ .NE. } B \Leftrightarrow a \neq b$
"меньше"	.LT.	$A \text{ .LT. } B \Leftrightarrow a < b$
"меньше или равно"	.LE.	$A \text{ .LE. } B \Leftrightarrow a \leq b$
"больше"	.GT.	$A \text{ .GT. } B \Leftrightarrow a > b$
"больше или равно"	.GE.	$A \text{ .GE. } B \Leftrightarrow a \geq b$

Результат операции сравнения — значение типа LOGICAL

6 операций сравнения

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

Результат операции сравнения — значение типа LOGICAL

6 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.	$A .GE. B \Leftrightarrow a >= b$

Результат операции сравнения — **значение типа LOGICAL**

5 логических операций

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

5 логических операций

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

5 логических операций

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

5 логических операций

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

5 логических операций

отрицание ("не")	.NOT.	.NOT. L (унарная операция)
"или"	.OR.	
"и"	.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

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

Порядок выполнения логических операций

- ❶ .NOT.
- ❷ .AND.
- ❸ .OR.
- ❹ .EQV. и .NEQV.
- ❺ **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
```

из prog11.f

```
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

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
```

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

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

Вложенные операторы IF

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

```
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

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

```
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
```

Оператор цикла DO

DO *метка* *параметр* = *нач*, *кон*

тело цикла

метка **последний оператор цикла**

Оператор-пустышка CONTINUE

Оператор цикла DO

DO *метка* *параметр* = *нач*, *кон*

тело цикла

метка последний оператор цикла

Оператор-пустышка CONTINUE

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

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

Вложение операторов цикла

Правильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
87 continue  
.....  
.....  
99 continue
```

Неправильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
99 continue  
.....  
.....  
87 continue
```

Правильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
end do  
.....  
.....  
end do
```


Вложение операторов цикла

Правильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
87 continue  
.....  
.....  
99 continue
```

Неправильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
99 continue  
.....  
.....  
87 continue
```

Правильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
end do  
.....  
.....  
end do
```

Вложение операторов цикла

Правильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
87 continue  
.....  
.....  
99 continue
```

Неправильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
99 continue  
.....  
.....  
87 continue
```

Правильно

```
do 99 .....  
.....  
.....  
do 87 .....  
.....  
.....  
end do  
.....  
.....  
end do
```

Общая форма оператора DO

DO метка параметр = нач, кон, шаг

тело цикла

метка CONTINUE

DO параметр = нач, кон, шаг

тело цикла

END DO

Количество проходов цикла:

$$N = \text{int} \left(\frac{\text{кон} - \text{нач}}{\text{шаг}} + 1 \right)$$

Замечания

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

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

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

Замечания

- ② Значения *нач*, *кон*, *шаг* вычисляются перед началом цикла; полученные значения хранятся на протяжении всего цикла.

```
N = 5
do i = 1, N
  N = N + 1
  print *, i, N
end do
```

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

Замечания

- 3 На выходе из цикла параметр цикла сохраняет последнее присвоенное значение.

```
do i = 1, 5  
  N = N + 1  
end do  
print *, i
```

6

Параметр цикла сначала получает очередное значение, и лишь потом производится проверка, не превышено ли значение *кон*.

Значение параметра цикла после выхода из цикла:

$$\text{параметр} = \text{нач} + N * \text{шаг}$$

Замечания

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

Оператор цикла DO WHILE (Fortran 90)

DO WHILE (*логическое выражение*)

тело цикла

END DO

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

Оператор безусловного перехода

GOTO метка

- ❶ Нельзя передавать управление внутрь цикла (но можно использовать GOTO в пределах тела цикла).
- ❷ Нельзя передавать управление внутрь блоков условного оператора (но можно использовать GOTO в пределах отдельного блока).
- ❸ 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.
  1
$ ./pause.if
Пауза в программе
PAUSE prompt>
      1
$ ./pause.if
Пауза в программе
PAUSE prompt> end
$ _
```

Оператор завершения работы

STOP *символьная константа*

STOP *целая константа (число)*