FORTRAN-77

массивы

Михайленко К. И.

Башгосуниверситет
Институт механики Уфимского научного центра РАН

(5)

Пример программы

prog15.f

```
program prog15
implicit none
integer i, N, mark, best
dimension mark(100)
real sum, avrg

read *, N
sum = .0
do i = 1, N
  read *, mark(i)
  sum = sum + real(mark(i))
end do
avrg = sum/real(N)
best = 0
do i = 1, N
  if (mark(i) .gt. avrg) best = best + 1
end do
print *, 'Средняя оценка:', avrg
print *, 'Оценок выше средней:', best
end
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)

3. REAL A(4)
4. REAL A(0:2)
5. REAL A(-2:2)

Описание массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)
3. REAL A(4)
4. REAL A(0:2)
5. REAL A(-2:2)

1

```
INTEGER a
DIMENSION a(10)
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)
3. REAL A(4)
4. REAL A(0:2)
5. REAL A(-2:2)

1

```
      INTEGER a
      DIMENSION a(10)
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)
3. REAL A(4)
4. REAL A(0:2)
5. REAL A(-2:2)

2

```
INTEGER a(10)
```

Размеры массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)

2

```
INTEGER a(10)
```

3. REAL A(4)
4. REAL A(0:2)
5. REAL A(-2:2)

3

a_1, a_2, a_3, a_4

A(1), A(2), A(3), A(4)

Размеры массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)

2

```
INTEGER a(10)
```

3. REAL A(4)
4. REAL A(0:2)
5. REAL A(-2:2)

4

a_0, a_1, a_2

A(0), A(1), A(2)

Размеры массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)

2

```
INTEGER a(10)
```

3. REAL A(4)
4. REAL A(0:2)
5. REAL A(-2:2)

5

$a_{-2}, a_{-1}, a_0, a_1, a_2$

A(-2), A(-1), A(0), A(1), A(2)

Индекс массива

1. *type Name*
   DIMENSION *Name*(*size*)
2. *type Name*(*size*)

2

```
INTEGER a(10)
```

3. `REAL A(4)`
4. `REAL A(0:2)`
5. `REAL A(-2:2)`

5

$a_{-2}, a_{-1}, a_0, a_1, a_2$

`A(-2), A(-1), A(0), A(1), A(2)`

В качестве индекса может быть использовано любое арифметическое выражение, возвращающее значение целого типа

6

```
A(I*int(sin(X))/2)
```

Ввод массива

1. Определение через оператор DATA
2. Чтение из стандартного потока ввода (считываются данные для всех элементов массива)
3. Поэлементное чтение
4. Поэлементное чтение в цикле
5. Поэлементное чтение, организованное через неявный цикл

```
double precision Array(-1:1)

data Array / 1.d0 1.5d0, 2.d0 /
data Array / 3 * 1.d0 /

read *, Array

read *, Array(1), Array(-1)

do i = -1, 1
  read *, Array(i)
end do

read *, (Array(i), i=-1, 1)
```

Ввод массива

1. Определение через оператор DATA
2. Чтение из стандартного потока ввода (считываются данные для всех элементов массива)
3. Поэлементное чтение
4. Поэлементное чтение в цикле
5. Поэлементное чтение, организованное через неявный цикл

```
double precision Array(-1:1)

data Array / 1.d0 1.5d0, 2.d0 /
data Array / 3 * 1.d0 /

read *, Array

read *, Array(1), Array(-1)

do i = -1, 1
  read *, Array(i)
end do

read *, (Array(i), i=-1, 1)
```

Ввод массива

1. Определение через оператор DATA
2. Чтение из стандартного потока ввода (считываются данные для всех элементов массива)
3. Поэлементное чтение
4. Поэлементное чтение в цикле
5. Поэлементное чтение, организованное через неявный цикл

```
double precision Array(-1:1)

data Array / 1.d0 1.5d0, 2.d0 /
data Array / 3 * 1.d0 /

read *, Array

read *, Array(1), Array(-1)

do i = -1, 1
  read *, Array(i)
end do

read *, (Array(i), i=-1, 1)
```

Ввод массива

1. Определение через оператор DATA
2. Чтение из стандартного потока ввода (считываются данные для всех элементов массива)
3. Поэлементное чтение
4. Поэлементное чтение в цикле
5. Поэлементное чтение, организованное через неявный цикл

```
double precision Array(-1:1)

data Array / 1.d0 1.5d0, 2.d0 /
data Array / 3 * 1.d0 /

read *, Array

read *, Array(1), Array(-1)

do i = -1, 1
  read *, Array(i)
end do

read *, (Array(i), i=-1, 1)
```

Ввод массива

1. Определение через оператор DATA
2. Чтение из стандартного потока ввода (считываются данные для всех элементов массива)
3. Поэлементное чтение
4. Поэлементное чтение в цикле
5. Поэлементное чтение, организованное через неявный цикл

```
double precision Array(-1:1)

data Array / 1.d0 1.5d0, 2.d0 /
data Array / 3 * 1.d0 /

read *, Array

read *, Array(1), Array(-1)

do i = -1, 1
  read *, Array(i)
end do

read *, (Array(i), i=-1, 1)
```

Ввод массива

1. Определение через оператор DATA
2. Чтение из стандартного потока ввода (считываются данные для всех элементов массива)
3. Поэлементное чтение
4. Поэлементное чтение в цикле
5. Поэлементное чтение, организованное через неявный цикл

```
double precision Array(-1:1)

data Array / 1.d0 1.5d0, 2.d0 /
data Array / 3 * 1.d0 /

read *, Array

read *, Array(1), Array(-1)

do i = -1, 1
  read *, Array(i)
end do

read *, (Array(i), i=-1, 1)
```

Вывод массива

1. Запись в стандартный поток вывода (печатаются все элементы массива)
2. Поэлементный вывод
3. Поэлементный вывод в цикле
4. Поэлементный вывод с неявным циклом

```
double precision Array(-1:1)

print *, Array

print *, Array(1), Array(-1)

do i = -1, 1
  print *, Array(i)
end do

print *, (Array(i), i=-1, 1)
```

Вывод массива

1. Запись в стандартный поток вывода (печатаются все элементы массива)
2. Поэлементный вывод
3. Поэлементный вывод в цикле
4. Поэлементный вывод с неявным циклом

```
double precision Array(-1:1)

print *, Array

print *, Array(1), Array(-1)

do i = -1, 1
  print *, Array(i)
end do

print *, (Array(i), i=-1, 1)
```

Вывод массива

1. Запись в стандартный поток вывода (печатаются все элементы массива)
2. Поэлементный вывод
3. Поэлементный вывод в цикле
4. Поэлементный вывод с неявным циклом

```
double precision Array(-1:1)

print *, Array

print *, Array(1), Array(-1)

do i = -1, 1
  print *, Array(i)
end do

print *, (Array(i), i=-1, 1)
```

Вывод массива

1. Запись в стандартный поток вывода (печатаются все элементы массива)
2. Поэлементный вывод
3. Поэлементный вывод в цикле
4. Поэлементный вывод с неявным циклом

```
double precision Array(-1:1)

print *, Array

print *, Array(1), Array(-1)

do i = -1, 1
  print *, Array(i)
end do

print *, (Array(i), i=-1, 1)
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size1*,..,*sizeN*)
2. *type Name*(*size*,..,*sizeN*)
3. $N \leqslant 7$

```
1
      REAL b, c
      DIMENSION b(10, 5), c(-3:0, 2, 0:125)
```

```
2
      REAL b(10, 5), c(-3:0, 2, 0:125)
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size1*,..,*sizeN*)
2. *type Name*(*size*,..,*sizeN*)
3. $N \leqslant 7$

1
```
      REAL b, c
      DIMENSION b(10, 5), c(-3:0, 2, 0:125)
```

2
```
      REAL b(10, 5), c(-3:0, 2, 0:125)
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size1*,..,*sizeN*)
2. *type Name*(*size*,..,*sizeN*)
3. $N \leqslant 7$

1
```
      REAL b, c
      DIMENSION b(10, 5), c(-3:0, 2, 0:125)
```

2
```
      REAL b(10, 5), c(-3:0, 2, 0:125)
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size1*,..,*sizeN*)
2. *type Name*(*size*,..,*sizeN*)
3. $N \leqslant 7$

1

```
REAL b, c
DIMENSION b(10, 5), c(-3:0, 2, 0:125)
```

2

```
REAL b(10, 5), c(-3:0, 2, 0:125)
```

Описание массива

1. *type Name*
   DIMENSION *Name*(*size1*,..,*sizeN*)
2. *type Name*(*size*,..,*sizeN*)
3. $N \leqslant 7$

1

```
REAL b, c
DIMENSION b(10, 5), c(-3:0, 2, 0:125)
```

2

```
REAL b(10, 5), c(-3:0, 2, 0:125)
```

Хранение двумерного массива

Массив в Fortran хранится по столбцам!

(медленнее всего изменяется последний индекс)

```
DIMENSION a(3,4)
```

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

A(1,1)	A(1,2)	A(1,3)	A(1,4)
A(2,1)	A(2,2)	A(2,3)	A(2,4)
A(3,1)	A(3,2)	A(3,3)	A(3,4)

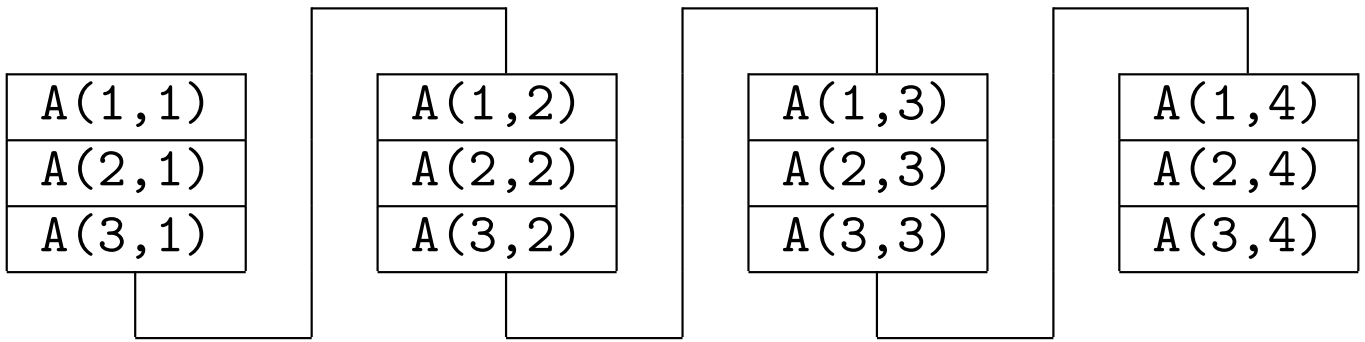
Хранение двумерного массива

Массив в Fortran хранится по столбцам!

(медленнее всего изменяется последний индекс)

```
DIMENSION a(3,4)
```

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$



Хранение двумерного массива

Массив в Fortran хранится по столбцам!

(медленнее всего изменяется последний индекс)

```
DIMENSION a(3,4)
```

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

A(1,1)	A(2,1)	A(3,1)	A(1,2)	A(2,2)		···		A(2,4)	A(3,4)

Хранение трёхмерного массива

Массив в Fortran хранится по столбцам!

(медленнее всего изменяется последний индекс)

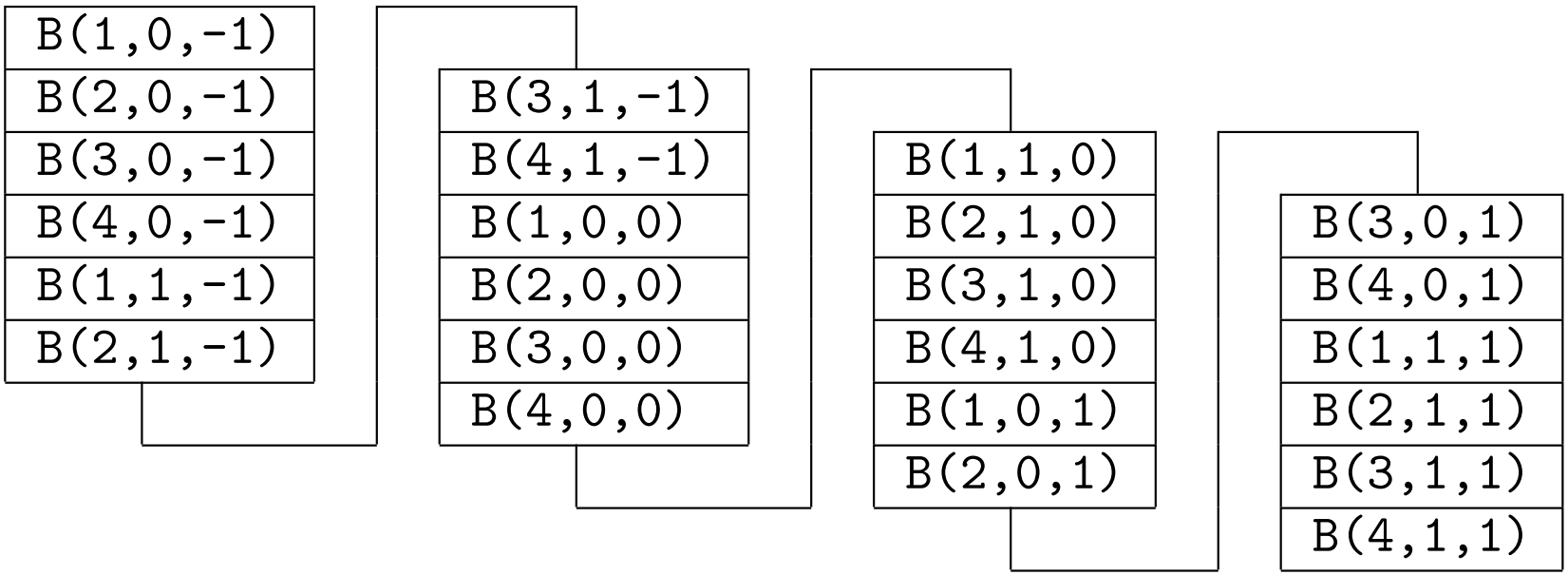
```
DIMENSION B(4,0:1,-1:1)
```

Хранение трёхмерного массива

Массив в Fortran хранится по столбцам!

(медленнее всего изменяется последний индекс)

```
DIMENSION B(4,0:1,-1:1)
```



Хранение массива

Имя массива — это указатель

Индекс элемента массива задаёт смещение от начала массива

В Фортране не отслеживаются границы массива

```
DIMENSION C(2,4)
```

C(1,1)	C(1,2)	C(1,3)	C(1,4)
C(2,1)	C(2,2)	C(2,3)	C(2,4)

C(1,1)	C(2,1)	C(1,2)	C(2,2)	C(1,3)	C(2,3)	C(1,4)	C(2,4)
C(1,1)	C(2,1)	C(3,1)	C(4,1)	C(5,1)	C(6,1)	C(7,1)	C(8,1)

То есть `C(6,1)` совпадает с `C(2,3)`

Допустима адресация вида `C(-2,1)` или `C(6,3)`
но результат (возвращаемое значение) непредсказуем;
возможна ошибка попытки обращения
к "чужой"области памяти (Segmentation fault)

Хранение массива

Имя массива — это указатель

Индекс элемента массива задаёт смещение от начала массива

В Фортране не отслеживаются границы массива

```
DIMENSION C(2,4)
```

C(1,1)	C(1,2)	C(1,3)	C(1,4)
C(2,1)	C(2,2)	C(2,3)	C(2,4)

C(1,1)	C(2,1)	C(1,2)	C(2,2)	C(1,3)	C(2,3)	C(1,4)	C(2,4)
C(1,1)	C(2,1)	C(3,1)	C(4,1)	C(5,1)	C(6,1)	C(7,1)	C(8,1)

То есть C(6,1) совпадает с C(2,3)

Допустима адресация вида C(−2,1) или C(6,3)
но результат (возвращаемое значение) непредсказуем;
возможна ошибка попытки обращения
к "чужой"области памяти (Segmentation fault)

Хранение массива

Имя массива — это указатель

Индекс элемента массива задаёт смещение от начала массива

В Фортране не отслеживаются границы массива

```
DIMENSION C(2,4)
```

C(1,1)	C(1,2)	C(1,3)	C(1,4)
C(2,1)	C(2,2)	C(2,3)	C(2,4)

C(1,1)	C(2,1)	C(1,2)	C(2,2)	C(1,3)	C(2,3)	C(1,4)	C(2,4)
C(1,1)	C(2,1)	C(3,1)	C(4,1)	C(5,1)	C(6,1)	C(7,1)	C(8,1)

То есть C(6,1) совпадает с C(2,3)

Допустима адресация вида C(−2,1) или C(6,3)
но результат (возвращаемое значение) непредсказуем;
возможна ошибка попытки обращения
к "чужой"области памяти (Segmentation fault)

Хранение массива

Имя массива — это указатель

Индекс элемента массива задаёт смещение от начала массива

В Фортране не отслеживаются границы массива

```
DIMENSION C(2,4)
```

C(1,1)	C(1,2)	C(1,3)	C(1,4)
C(2,1)	C(2,2)	C(2,3)	C(2,4)

C(1,1)	C(2,1)	C(1,2)	C(2,2)	C(1,3)	C(2,3)	C(1,4)	C(2,4)

C(1,1)	C(2,1)	C(3,1)	C(4,1)	C(5,1)	C(6,1)	C(7,1)	C(8,1)

То есть `C(6,1)` совпадает с `C(2,3)`

Допустима адресация вида `C(-2,1)` или `C(6,3)`
но результат (возвращаемое значение) непредсказуем;
возможна ошибка попытки обращения
к "чужой"области памяти (Segmentation fault)

Хранение массива

Имя массива — это указатель

Индекс элемента массива задаёт смещение от начала массива

В Фортране не отслеживаются границы массива

```
DIMENSION C(2,4)
```

C(1,1)	C(1,2)	C(1,3)	C(1,4)
C(2,1)	C(2,2)	C(2,3)	C(2,4)

C(1,1)	C(2,1)	C(1,2)	C(2,2)	C(1,3)	C(2,3)	C(1,4)	C(2,4)

C(1,1)	C(2,1)	C(3,1)	C(4,1)	C(5,1)	C(6,1)	C(7,1)	C(8,1)

То есть `C(6,1)` совпадает с `C(2,3)`

Допустима адресация вида `C(-2,1)` или `C(6,3)`
но результат (возвращаемое значение) непредсказуем;
возможна ошибка попытки обращения
к "чужой"области памяти (Segmentation fault)

Хранение массива

Имя массива — это указатель

Индекс элемента массива задаёт смещение от начала массива

В Фортране не отслеживаются границы массива

```
DIMENSION C(2,4)
```

C(1,1)	C(1,2)	C(1,3)	C(1,4)
C(2,1)	C(2,2)	C(2,3)	C(2,4)

C(1,1)	C(2,1)	C(1,2)	C(2,2)	C(1,3)	C(2,3)	C(1,4)	C(2,4)

C(1,1)	C(2,1)	C(3,1)	C(4,1)	C(5,1)	C(6,1)	C(7,1)	C(8,1)

То есть `C(6,1)` совпадает с `C(2,3)`

Допустима адресация вида `C(-2,1)` или `C(6,3)`
но результат (возвращаемое значение) непредсказуем;
возможна ошибка попытки обращения
к "чужой"области памяти (Segmentation fault)

Печать транспонированной матрицы

prog16.f

```
      program prog16
      implicit none
      integer i, j, M, N
      real a(10,10)

      read *, M, N
      if ((M .lt. 1) .or. (M .gt. 10) .or.
     +    (N .lt. 1) .or. (N .gt. 10)) stop

      print *, 'Исходная матрица размера ', M, 'x', N, ':'
      do i = 1, M
        read *, (a(i,j), j=1,N)
      end do

      print *, 'Транспонированная матрица размера ', N, 'x', M, ':'
      do j = 1, N
        print *, (a(i,j), i=1,M)
      end do

      end
```

Печать транспонированной матрицы

```
$ g77 -o prog16 prog16.f
$ ./prog16
4 2
 Исходная матрица размера  4x 2:
1 2
3 4
5 6
7 8
 Транспонированная матрица размера  2x 4:
  1.  3.  5.  7.
  2.  4.  6.  8.
```