

Программирование

Лекция 6 Массивы

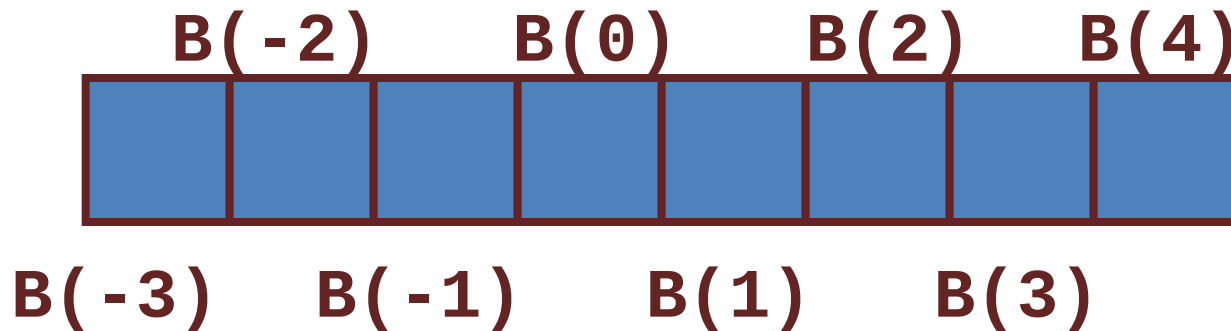
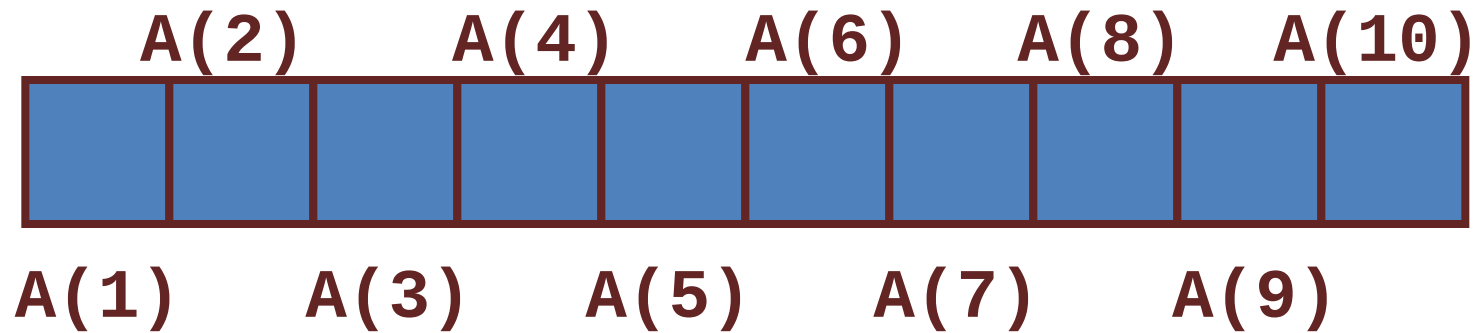
Петров Александр Владимирович
Фёдоров Станислав Алексеевич

**(по материалам Веренинова Игоря
Андреевича с изменениями и
дополнениями на Fortran 08 и UML)**

Сентябрь 2013

Одномерные массивы

Вектора, последовательности



Одномерные массивы

Объявление и инициализация

```
real A(10)
real V(1:10), W(-5:15), S(0:90)
real, dimension(10) :: R = 2 ! все элементы 2

! границы задаются через константы
integer, parameter :: N = 10
complex :: B(-N:N) = (0.0, 0.0) ! обнуление
integer :: C(10) = (/2, 6, 3, 2, 1, 4, 5, 6, 7, 8/)
integer :: D(1:7) = [6, 7, 2, 1, 9, 0, 3]

! массив констант
integer, parameter :: INDX(4) = [12, 86, 75, 9]
```

Одномерные массивы

Конструктор массива и присваивание

real A(10)

A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0] ! конструктор

A = (/0, 0, 0, 0, 0, 2, 2, 2, 2, 2/)

A = 0 ! обнуление массива

A = (/ (0, k=1, 5), (2, k=6, 10) /)
! циклический список

A(1) = -2 ! присваивание элементу

A(3) = 2*A(1)+A(5)

Одномерные массивы

Операции над элементами массивов
(массив как обычная переменная)

1. Сложить два вектора

```
real A(10), B(10), C(10)
```

```
C = A+B
```

2. Перемножить элементы вектора $C(i) = A(i) * B(i)$

```
real A(10), B(10), C(10)
```

```
C = A*B
```

Одномерные массивы

Операции над элементами массивов

3. Возведение в степень $C(i) = A(i)**B(i)$

```
real A(10), B(10), C(10)
```

```
C = A**B
```

4. Вычисление процедур от элементов массива

```
real A(10), B(10)
```

```
call random_number(A)
```

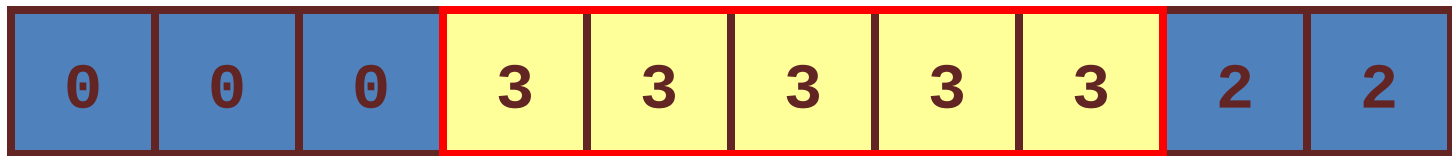
```
B = sqrt(A)
```

Одномерные массивы

Обращение к группе элементов
(сечение массива)

1. Индексный триплет (все параметры
необязательны)

нижняя граница : верхняя граница : шаг



$A(4:8)=3$

Одномерные массивы



$A(:5)=5$



$A(7:)=5$

Одномерные массивы



$A(1:10:2)=5$



$A(: : 3)=5$

Одномерные массивы

2. Векторный индекс - одномерный массив, содержащий номера избранных элементов массива.

A =

3	1	8	7	4	5	2	0	1	6
1	2	3	4	5	6	7	8	9	10

V =

1	2	5	7	8	9
---	---	---	---	---	---

$A(V) = 0$ Применяем векторный индекс

A =

0	0	8	7	0	5	0	0	0	6
---	---	---	---	---	---	---	---	---	---

Одномерные массивы

Ввод / вывод (экран)

```
program read_array
integer A(5)

! данные вводятся через пробел / перевод строки
read(*,*,ERR = 100) A  ! 1 2 3 4 5

write(*,*) A           ! 1 2 3 4 5

write(*,*) (A(i),i = 1,2)  ! 1 2

write(*,*) A(::2)         ! 1 3 5
stop

100 stop "Ошибка при чтении данных"

end
```

Одномерные массивы

Ввод / вывод (файл)

```
program read_array_file  
integer A(5)
```

```
open(1,file = "D:\DATA\ARR.txt",ERR = 100) ! ввод из файла  
read(1,*,ERR = 101) A  
close(1)
```

```
open(2,file = "D:\DATA\RES.txt",ERR = 102) ! вывод в файл  
write(2,*,ERR = 103) A(1:4)  
close(2)  
stop
```

```
100 stop "Ошибка при открытии файла"  
101 stop "Ошибка при чтении данных"  
102 stop "Ошибка при создании файла"  
103 stop "Ошибка при записи данных"  
end
```

Двумерные массивы

A =

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

Объявление

```
real A(2,4)    ! 2 строки, 4 столбца
```

```
real A(1:2,1:4), B(-1:100, -1:200)
```

```
real, dimension(2,4) :: A
```

```
integer, parameter :: Mi = 2, Mj = 4
```

```
real, dimension :: A(Mi,Mj)
```

Двумерные массивы

Двумерный массив хранится в памяти **по столбцам**

Инициализация

```
real :: A(2,4) = [2,5,7,9,0,1,4,8]
```

$$A = \begin{bmatrix} 2 & 7 & 0 & 4 \\ 5 & 9 & 1 & 8 \end{bmatrix}$$

```
real :: A(2,4) = 0      ! обнуление массива
```

Двумерные массивы

Операции над элементами массивов
(массив как обычная переменная)

1. Сложить две матрицы

```
real A(10,10), B(10,10), C(10,10)
```

```
C = A+B
```

2. Перемножить элементы матриц

```
C(i,j) = A(i,j)*B(i,j)
```

```
real A(10,10), B(10,10), C(10,10)
```

```
C = A*B
```

Двумерные массивы

Использование сечений

1. Обнулить первый столбец матрицы

0		
0		
0		
0		



$A(:, 1) = 0$

2. Обнулить первую строку матрицы

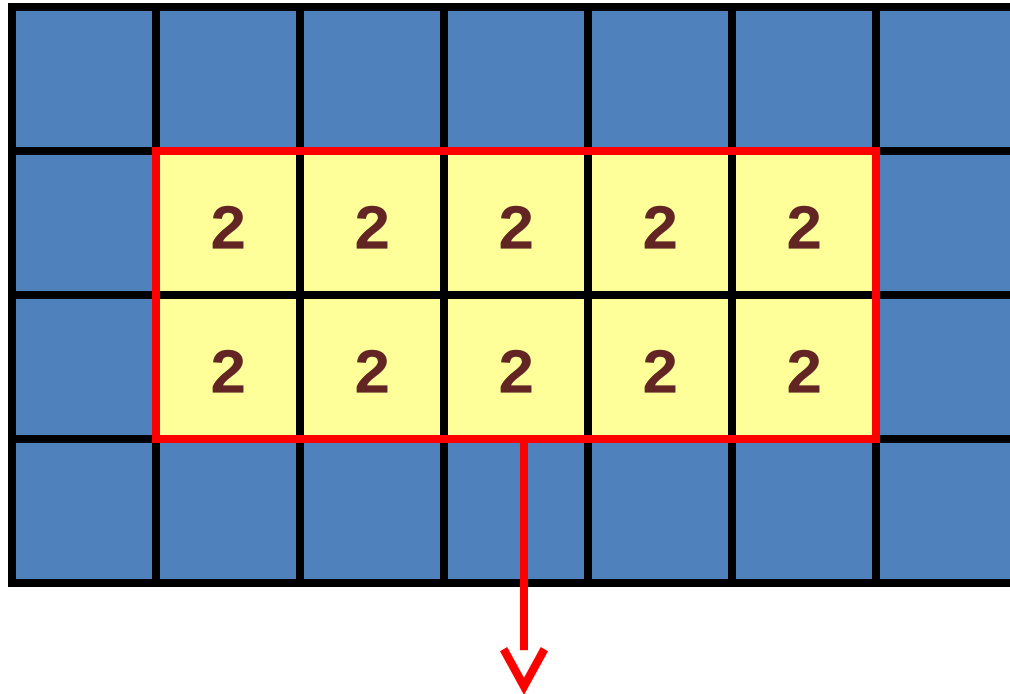
0	0	0

$\rightarrow A(1, :) = 0$

Двумерные массивы

Использование сечений

3. Присвоить подматрице значения



$$A(2:M_i-1, 2:M_j-1)=2$$

Двумерные массивы

Ввод / вывод (экран)

```
program read_array_2D
integer A(3,3)
```

```
! данные вводятся по столбцам (формат хранения в памяти)
```

```
read(*,*,ERR = 100) A  ! 1 1 1    2 2 2    3 3 3
```

```
! выведем данные по строкам
```

```
do i = 1,3
  write(*,*) (A(i,j),j = 1,3)
end do
stop
```

```
100 stop "Ошибка при чтении данных"
end
```

Двумерные массивы

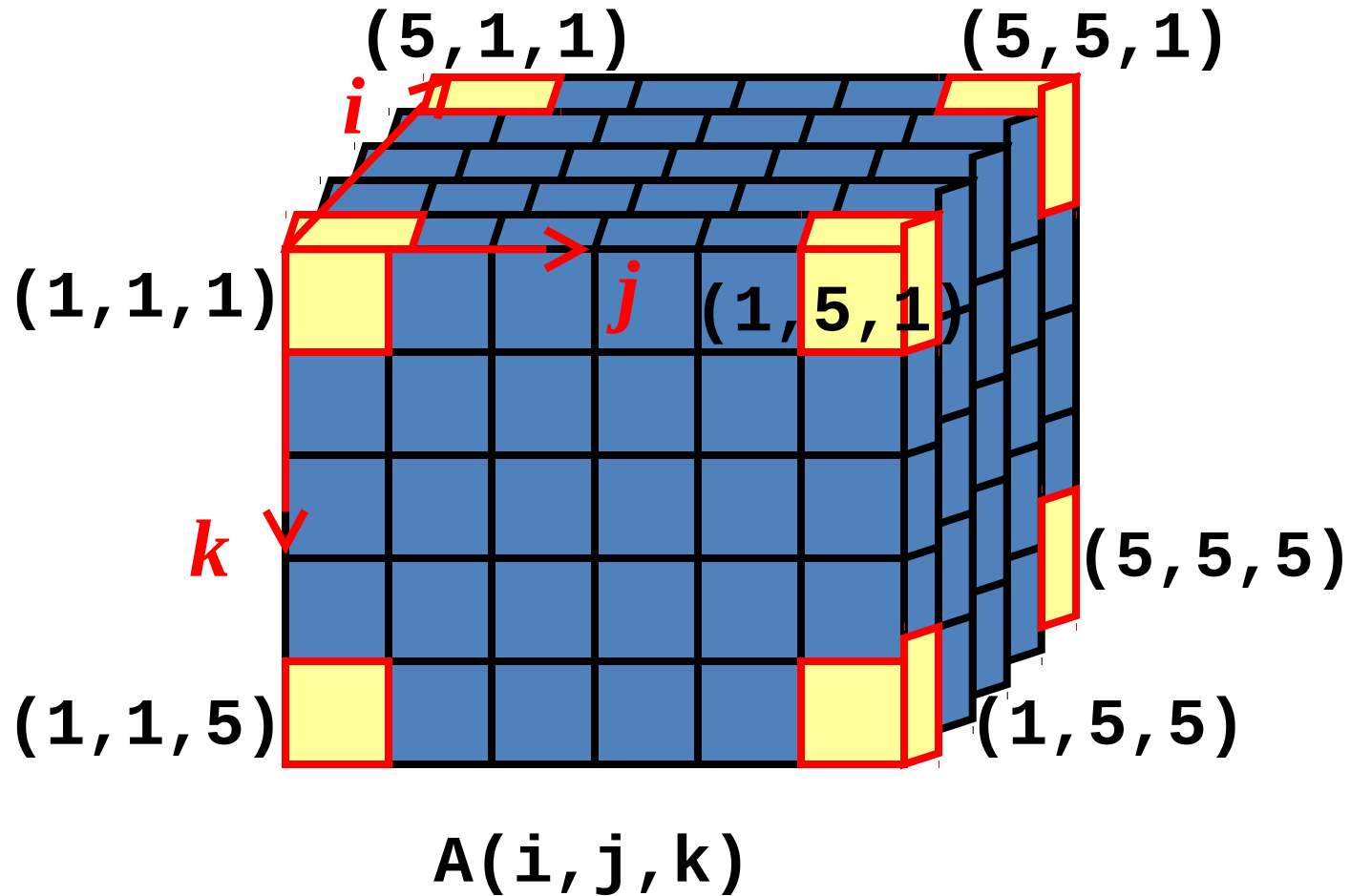
Ввод / вывод (файл)

```
program read_array_2D_file
integer A(5,5)

! данные вводятся по столбцам
open (1, file = "D:\DATA\ARR_2D.txt", ERR = 100)
read (1, *, ERR = 100) A
close(1)

! запишем нижний треугольник в файл
open (1, file = "D:\DATA\RES_2D.txt", ERR = 100)
do i = 1,5
    write(1, *, ERR = 100) A(i,1:i)
end do
close(1)
stop
100 stop "Ошибка при работе с файлами"
end
```

Трёхмерные массивы



Трёхмерные массивы

Объявление и инициализация

```
real A(10,20,30)
```

```
real A(1:10,1:20,1:30)
```

```
real, dimension :: A(10,20,30)
```

```
integer, parameter :: Mi = 10, Mj = 20
```

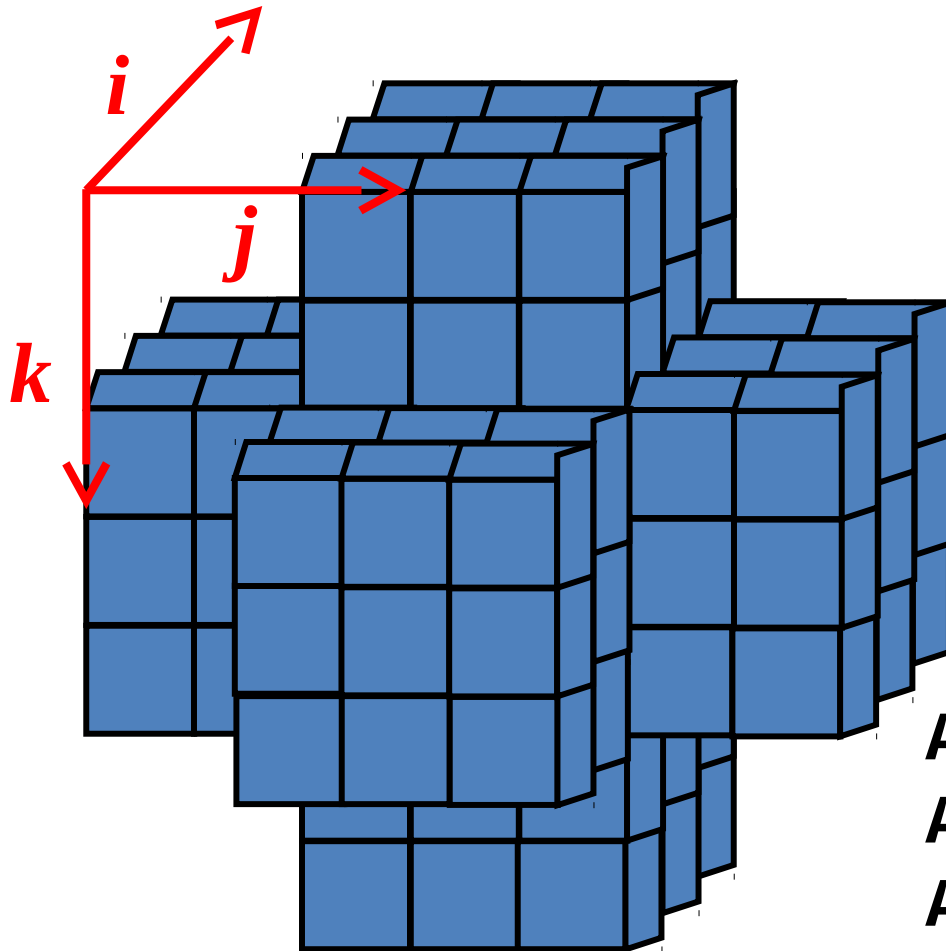
```
real, dimension(Mi,Mj,Mi) :: A
```

```
logical :: L(10,10,10) = .TRUE.
```

```
character :: A(20,3,3) = 'Q'
```

Трёхмерные массивы

Использование сечений



$A = 0$! обнуление

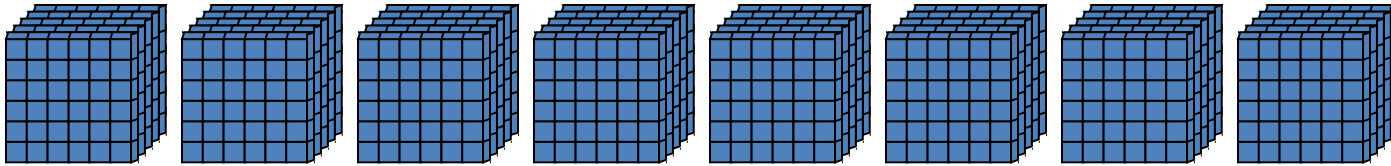
$A(:, 3:5, 3:5) = 1$! i

$A(3:5, :, 3:5) = 1$! j

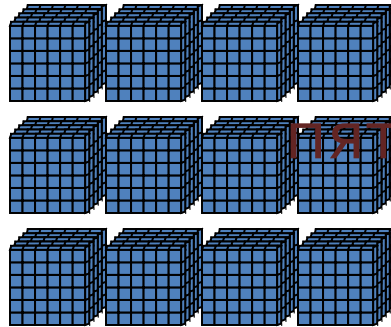
$A(3:5, 3:5, :) = 1$! k

Многомерные массивы

четырехмерный массив



`real(8) :: S(8,5,6,6) = 0.5d0`



пятимерный массив

`complex(16) :: D5(3,4,5,6,6) = (0.0q0, 0.0q0)`

Терминология

Ранг – число измерений массива

Размер – число элементов массива

Форма – ранг и протяженность вдоль каждого измерения

Согласованность – ранг, форма и размер совпадают

```
real A(3, 5, 8) ! ранг = 3  
                ! размер = 3x5x8 = 120  
                ! форма  = (3, 5, 8)
```

```
real C(-1:1, 0:4, 1:8) ! A и C согласованные
```


Динамические массивы

Размер массива задается во время работы программы

```
real A(10,20) ! статический массив  
real, allocatable :: B(:, :) ! динамический
```

Оператор **allocate** выделяет память под массив

Оператор **deallocate** освобождает память

Функция **allocated** выполняет проверку размещения массива

Динамические массивы

```
real, allocatable :: A(:, :)  
integer ERR_ALLOC  
  
! создали массив 50x50  
allocate(A(50,50), STAT=ERR_ALLOC)  
  
if (ERR_ALLOC/=0) stop "Allocation ERROR"  
  
print *, allocated(A)    ! T  
deallocate(A)            ! освободили память  
print *, allocated(A)    ! F
```

Динамические массивы

```
complex(16) A( -20:20, -30:30)
complex(16), allocatable :: CHILD(:, :)
integer ERR_ALLOC
```

```
! унаследовали границы и форму массива A
allocate(CHILD, SOURCE=A, STAT=ERR_ALLOC)
if (ERR_ALLOC/=0) stop "Allocation ERROR"
```

```
CHILD( -20, 30)=(11.0q0, 2.0q0)
print *, allocated(CHILD) ! T
```

```
deallocate(CHILD) ! освободили память
```

Оператор where

Эффективное выборочное присваивание
(выгодная замена связки **do---if**)

where (логическое выражение-массив)

операторы присваивания массивов

elsewhere

операторы присваивания массивов

end where

Оператор where

```
program use_where
```

```
integer :: A(7)=[1, -3, 4, -5, -6, -7, 0]
```

```
where (A<0)
```

```
    A=-A
```

```
elsewhere
```

```
    where (A==0)
```

```
        A=5
```

```
    elsewhere
```

```
        A=A**2
```

```
    endwhere
```

```
end where
```

```
write(*, "(7i4)") A !      1      3     16      5      6      7      5
```

```
end
```

Оператор forall

Выборочное присваивание только
для некоторой части массива.

forall (спецификация триплета, выражение
маска)

операторы присваивания массивов

end forall

```
program use_forall
```

```
integer :: A(20) = [1,0,0,0,0,0,1,2,3,4, &  
                   0,0,0,0,0,0,1,1,1,1]
```

```
forall (i = 1:10, A(i) == 0)
```

```
    A(i) = 5
```

```
end forall
```

```
write(*,"(20i2)") A ! 1 5 5 5 5 5 1 2 3 4 0 0 0 0 0 0 1 1 1 1
```

```
end
```

Оператор forall

Оператор **forall** не равносильен оператору цикла

- В цикле оператор присваивания выполняется при каждом вызове.
- В **forall** сначала полностью вычисляется правая часть оператора присваивания, а затем результат присваивается массиву.

Оператор forall

```
program forall_NE_do
```

```
integer, parameter :: M = 5
```

```
integer a(M)
```

```
  a = 1
```

```
  do k = 2, M
```

```
    a(k) = a(k-1)+1
```

```
  end do
```

```
  write(*, "(10(i4))") a      ! 1  2  3  4  5
```

```
  a = 1
```

```
  forall (k = 2:M) a(k) = a(k-1)+1
```

```
  write(*, "(10(i4))") a      ! 1  2  2  2  2
```

```
end
```


Процедуры обработки массивов

Вычисления в массиве

ALL, ANY

COUNT

MAXLOC, MINLOC

MAXVAL, MINVAL

PRODUCT

SUM

Преобразование

MERGE

Переформирование

RESHAPE

Справочные

ALLOCATED

IS_CONTIGUOUS

Вектора и матрицы

DOT_PRODUCT

MATMUL

TRANSPOSE

Граница, форма, размер

LBOUND, UBOUND

SHAPE, SIZE

Упаковка и распаковка

PACK

UNPACK

Построение

SPREAD

Сдвиг массивов

CSHIFT, EOSHIFT