Обзор языка Фортран – 90

Лекция летней школы
параллельного программирования 2008

Куликов И.

Содержание

Простейший ввод/вывод

Выведем приветствие на языке Фортран:

```
program Hello
    print *,"Hello, World!"
end
```

Первая строка содержит оператор **program**, который задаёт имя программы **Hello**.

Вторая строка содержит оператор **print** для печати на стандартный вывод строки **“Hello, World!”**.

Третья строка содержит оператор **end**, указывающий конец программы.

Попробуем прочитать со стандартного вывода первые пять символов строки и вывести в форматированном виде:

```
      program print5symbol
          implicit none
          character*8 str
          read(*,'(a5)') str
          write(*,100) str
100       format("Your 5 simbol: ",a5)
      end
```

Во второй строке **implicit none** указано, что все переменные требуют описания (по умолчанию переменные простого типа можно не описывать). В третий строке объявлена строка из 8 символов. В четвёртой строке со стандартного ввода читается оператором **read** 5 символов в строку **str**, затем в пятой строке выводится оператором **write** на стандартный вывод форматированная строка.

Типы данных

Целочисленный тип:

integer a

Формат ввода/вывода: **‘(i#)’**

– длина числа

Логический тип данных:

logical a

Формат ввода/вывода: **‘(L)’**

Строковый тип данных

character*12 str

Формат ввода/вывода: **‘(a#)’**

– длина строки

Вещественный тип данных:

real a

double precision b

Формат ввода/вывода: **‘([e|f]#.$)’**

e или **f** – обычный или экспонециальный способ вывода

– длина числа

$ – количество знаков после запятой

Комплексный тип данный:

complex a

double complex b

Формат ввода/вывода: **‘(2[e|f]#.$)’**

e или **f** – обычный или экспонециальный способ вывода

– длина числа

$ – количество знаков после запятой

Пример программы работы с типами данных

```
program printtype
        implicit none
        character*13 :: str="Hello, World!"
        integer :: i = 1234
        real :: a = 12.4e-2
        double precision :: b = 0.1d0
        double complex :: c = (1.0d0,1.0d0)
        logical :: d = .TRUE.
        write(*,'(a14)') str
        write(*,'(i4)') i
        write(*,'(f5.3)') a
        write(*,'(e9.3)') b
        write(*,'(2f5.2)') c
        write(*,'(L)') d
  end
```

Основные математические функции

****** – возведение в степень

abs – модуль числа

cos – косинус

sin – синус

tan – тангенс

acos – арккосинус

asin – арксинус

atan – арктангенс

sqrt – квадратный корень

exp – экспонента

log – натуральный логарифм

log10 – десятичный логарифм

mod – остаток от деления

max – максимум

min – минимум

floor – наибольшее целое, меньшее или равное аргументу

ceiling – наименьшее целое, большое или равное аргументу

random_seed – инициализация генератора случайных чисел

random_number – генерация случайного числа

.and. – логическое И

.or. – логическое ИЛИ

Для двойной точности перед названием математической функции указывается приставка **d**

Оператор условия

Конструкция **IF**

```
if (Логическое_выражение) then
    Операторы_1
else
    Операторы_2
endif
```

Пример:

```
...
if ( k < 0 ) then
    write(*,*) “k<0”
else
    write(*,*) “k>=0”
endif
...
```

Организация циклов

Конструкция **DO**:

```
do переменная = начальное значение, конечное значение, шаг
    тело цикла
enddo
```

Конструкция **DO WHILE**:

```
do while (логическое_условие)
    тело цикла
enddo
```

Бесконечный цикл:

```
do
    тело цикла
enddo
```

Примеры:

```
…
do i = 2, 10, 2
    write(*,*) i  ! вывод 2, 4, 6, 8, 10
enddo
…
i = 2
do while (i<=10)
    write(*,*) i
    i = i + 2
enddo
…
do
    print *,”Hello!”
enddo
...
```

Массивы

Описание статического N-мерного массива и массива с фиксированными первым и последним индексом:

type name_array(dim_1, dim_2, ..., dim_N)

type name_array(start_1 : end_1, ..., start_ N : end_ N)

Пример:

```
program array
    implicit none
    integer, parameter :: N = 5
    integer a(N)
    integer i
    do i=1,N
        a(i) = i
        print *,a(i)
    enddo
end
```

Организация многомерного массива

Логическое представление матрицы:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

Представление матрицы в памяти:

$$A = \left\{ a_{1,1}, a_{2,1}, a_{3,1}, a_{1,2}, a_{2,2}, a_{3,2}, a_{1,3}, a_{2,3}, a_{3,3} \right\}$$

Пример работы с многомерными массивами

```
program array2d
    implicit none
    integer, parameter :: N=5, M=4
    integer a(N,M)
    integer i,j
    do i=1,N
        do j=1,M
            a(i,j) = i*j
            write(*,"(i4,\)") a(i,j) ! вывод элементов в строку
        enddo
        write(*,*)
    enddo
end
```

Непроцедурные операции с массивами

Присваивание массиву одного значения:

array = value

Умножение матрицы на число:

array1 = array2 * value

Поэлементное сложение, вычитание, умножение матриц:

array1 = array2 + array3

array1 = array2 – array3

array1 = array2 * array3

Пример операций с сечениями массивов

```
program array2d
    implicit none
    integer, parameter :: N=5, M=4
    integer a(N,M)
    integer i,j

    a = 0              ! зануление массива
    a(2:4,:) = 1       ! заполнение со 2-й по 4-й строки единицами
    do i=1,N
        do j=1,M
            write(*,"(i4,\)") a(i,j)     ! вывод в строку
        enddo
        write(*,*)
    enddo
end
```

Оператор where

Схема оператора **Where** (как):

```
where (логическое выражение)
    операторы присваивания массивов
elsewhere
    операторы присваивания массивов
end where
```

Оператор **Where** требуется, когда элементам массива, удовлетворяющим некоторым условиям, следует присвоить определённые значения.

Пример работы с оператором Where

```
program use_where
    implicit none
    integer, parameter :: N=5
    integer a(N), b(N)
    integer i
    a = (/1, -2, 7, 8, -10/)
    b = (/1, 2, 3, 4, 5/)
    where(a < 0)
        b = b**2
    elsewhere
        b = 0
    endwhere

    do i=1,N
        write(*,"(i4,\)") b(i)
    enddo
    write(*,*)
end
```

Оператор Forall

Схема оператора **Forall** (для всех):

forall (спецификация триплета, выражение маска)

операторы присваивания

end forall

Оператор **forall** требуется при выборочном присваивании массивам значений. Разница между циклом **do** и оператором **forall** есть принципиальная разница, которая заключается в том, что в цикле оператор присваивания выполняется при каждом вызове, а в **forall** сначала полностью вычисляется правая часть оператора присваивания и затем результат присваивается массиву.

Пример работы с оператором Forall

```
program use_forall
    implicit none
    integer, parameter :: N=5
    integer a(N), b(N)
    integer i
    a = (/1, -2, 7, 8, -10/)
    b = (/1, 2, 3, 4, 5/)
    forall(i=1:N, a(i)<0)
        b(i) = 0
    endforall
    do i=1,N
        write(*,"(i4,\)") b(i)
    enddo
    write(*,*)
end
```

Функции для работы с массивами

matmul – функция перемножения двух матриц целого, вещественного, комплексного и логического типов.

maxval – функция, находящая максимальное значение в массиве.

minval – функция, находящая минимальное значение в массиве.

sum – находит сумму элементов массива.

transpose – функция транспонирования матрицы.

Динамические массивы

Схема объявления динамического массива:

```
type, allocatable :: array(:1, ..., :N)
```

Выделение памяти для динамического массива:

```
allocate(array(M1, ..., MN))
```

Работа с динамическим массивом:

```
array(index1, ..., indexN)
```

Освобождение памяти

```
deallocate(array)
```

Пример работы с динамическими массивами

```
program dynarray
   implicit none
   integer, allocatable :: a(:)
   integer, parameter :: M = 5
   integer i
   allocate(a(M))
   forall(i=1:M)
      a(i) = 1
   endforall
   do i=1,M
      write(*,"(i4,\)") a(i)
   enddo
   write(*,*)
   deallocate(a)
end
```

Функции, подпрограммы, модули, интерфейсы

Схема описания функции:

```
type function f_name(аргументы)
   тело функции
   f_name = результат
end function f_name
```

Схема описания подпрограммы:

```
subroutine p_name(аргументы)
   тело подпрограммы
end subroutine p_name
```

Схема описания модулей:

```
module m_name
   описание типов, констант, переменных
end module
```

Схема описания интерфейсов:

```
interface
   описание подпрограмм и функций
end interface
```

Пример работы с процедурами, функциями, модулями

```
! Описание модуля
module array
   double precision, allocatable :: a(:)
end module

! Описание функции
double precision function summa_array(a)
   double precision, dimension (:) :: a
   summa_array = sum(a)
end function summa_array

! Описание процедуры
subroutine destroy()
   use array
   implicit none
   deallocate(a)
end subroutine destroy
```

```fortran
! Программа
program funcmodproc
   use array
   implicit none
   integer, parameter :: M = 5
   double precision dsumm
   ! Описание интерфейса
   interface
      double precision function summa_array(a)
         double precision, dimension (:) :: a
      end function summa_array
   end interface

   allocate(a(M))
   a = 1.d0
   dsumm = summa_array(a)
   print *, dsumm
   call destroy()
end
```

Динамические структуры данных

Схема описания ссылки:

```
type, pointer :: ptr
```

Схема описания цели:

```
type, target :: goal
```

Пример программы:

```
program struct
   integer, pointer :: ptr
   integer, target :: goal
   goal = 100
   ptr => goal
   print *,ptr
end
```

Работа с файлами

Открытие файла:

```
open(номер_файла, file=имя_файла)
```

Чтение из файла:

```
read(номер_файла, формат) переменные
```

Запись в файл:

```
write(номер_файла, формат) переменные
```

Закрытие файла:

```
close(номер_файла, формат)
```

Пример работы с файлами

```
program copy
   implicit none
   integer i
   open(100,file="1.txt")
   open(200,file="2.txt")
   do while(.not. eof(100))
      read(100,'(i1)') i
      write(200,'(i1)') i
   enddo
   close(100)
   close(200)
end
```

Литература

1. Артёмов И.Л. Fortran: Основы программирования. – М.: Диалог-МИФИ, 2007. – 304 с.
2. www.cs.mtu.edu/~shene/COURSES/cs201/NOTES/fortran.html