1. Теория.

1.1. Where.

Оператор where позволяет присваивать некоторое значение (выполнить некоторые операции) только тем элементам, значения которых удовлетворяют некоторым условиям. К примеру:

where(b>0) b=2\*b

Данная запись увеличит вдвое все положительные b. Данную операцию можно сделать и с помощью цикла и ветвления, однако как и для сечения массива оператор where в фортране работает быстрее чем аналогичные записи, за счет создания массива-маски.

В случае более сложного присваивания, или нескольких операций можно писать:

where(b>0)

b=2\*b

a=b

end where

Оба присваивания осуществляются только для положительных b.

Также возможно записать:

where(b>0)

b=2\*b

elsewhere

b=b/2

end where

В итоге все положительные значения вдвое увеличатся, все отрицательные вдвое уменьшатся.

Условия могут быть и сравнением двух массивов. Если а и b одинаковой формы:

where(a>b)

a=b

elsewhere

b=a

end where

Даст новый массив, составленный из меньших элементов двух массивов.

Помимо этого есть возможность сделать множественные elsewhere.

where(a>0)

a=1

elsewhere(a<0)

a=-1

elsewhere

a=0

end where

Количество конструкций elsewhere неограниченно, если завязано на одну переменную. К примеру:

where(a>65)

b=1

elsewhere(a<-8)

b=-1

elsewhere(a>2)

c=1

elsewhere(a<0)

c=-1

elsewhere(a==0)

d=1

elsewhere

d=-1

end where

Работать будет. Хотя сложно разобраться как.

Также важно отметить работу при изменении переменной на которую накладывается условие:

a=(/-10,-20,0,4,-5,60,70,-3,2,11/)

where(a>0)

a=-a

elsewhere(a<-10)

a=100

end where

На выходе даст (/-10,100,0,-4,-5,-60,-70,-3,-2,-11/). Числа -60 и -70 хоть и удовлетворяют условию elsewhere не превращаются в 100, так как работа осуществляется с помощью двух массивов масок и исходного массива.

А

where(a>0)

a=-a

elsewhere(a>10)

a=100

end where

Не даст ни одного значения 100, так как приоритет имеет первое присваивание.

1.2. ForAll.

Forall также является оператором, использующем массив-маску. При этом он, пожалуй ближе всего в обычным циклам.

forall(i=3:99:2) a(i)=a(i-2)+5

Данная конструкция для всех элементов от 3 до 100 с шагом 2 совершит следующее присваивание. При этом данное присвоение будет отличаться от аналогичного в цикле. Там будут использоваться нарастающие значения, тогда как для forall используются для присвоения первоначальные значения массива а. Шаг в триплете может быть отрицательный:

forall(i=97:1:-2) a(i)=a(i+2)+5

Допускается написание:

name forall(i=97:1:-2)

a(i)=a(i+2)+5

b(i)=a(i-2)

end forall name

Параметр name является необязательным. b будут присваиваться значения после присвоения а в рамках forall. Присвоение каждого элемента в forall можно сделать только один раз. Оператор forall работает и с массивами разной формы/размера при аккуратном использовании.

Примеры использований:

forall(i=1:100,j=1:100) a(i,j)=(i+j)\*b(i,j)

из-за использования в присвоении индексов нельзя сделать с помощью сечений массива или where.

forall(i=1:100) a(i,i)=0

дает быстрый и простой способ получить главную диагональ.

forall(i=1:n) a(i)=sum(x(1:i))

forall(i=1:n) b(i)=product(x(1:i))

Дает суммы и факториалы чисел. Хорошо подходит для расчета рядов.

1.3. Автоматические массивы.

В рамках работы процедуры могут быть заданы локальные массивы, размеры которых меняются при разных вызовах процедуры или перенимают форму/размер внешних к процедуре массивов.

subroutine swap(a,b)

real a(:),b(:) !перенимающие форму массивы

real c(size(a)) ! автоматический массив, чья форма зависит от а

c=a

a=b

b=c

end subroutine swap

Данная процедура меняет значения массивов а и b, при этом автоматический массив c даже не появляется в основной программе.

Вызов процедуры в программе выглядит следующим образом:

interface

subroutine swap(a,b)

real a(:),b(:)

end subroutine swap

end interface

call swap(x,y)

Задание явного интерфейса процедуры требуется по причине использования перенимающих форму массивов. Так, их можно заменить на массивы заданной формы:

subroutine swap(a,b,m,n)

integer m,n ! индексы, задающие форму массивов а и b

real a(m\*n),b(m\*n) !массивы заданной формы, вместо (m\*n) может быть, к примеру, (m,n) или (m:n)

real c(size(a)) ! автоматический массив, чья форма зависит от а

c=a

a=b

b=c

end subroutine swap

Если для перенимающего форму массива важна нижняя граница (по умолчанию она будет равна 1), то надо объявить перенимающий форму массив не так a(:),b(:), а так: a(0:),b(10:). Форма массива будет определена внешним к процедуре массивом, но при этом отчет будет идти от указанной границы.

Также можно использовать перенимающие размер массивы. Если в процедуре надо использовать массив другой формы но того же размера. Впрочем, тут минусов больше чем плюсов. Для перенимающего размер массива надо задать одну или несколько (для многомерных массивов) верхних границ \*.

real x(8)

call assume(x)

…

subroutine assume(a)

real a(2,2,\*)

В результате массив а переняв размер будет выглядеть как массив a(2,2,2).

Для x(12) это будет массив a(2,2,3). А вот для x(7) это будет массив a(2,2,2), у которого нет элемента (2,2,2), но есть, к примеру, (1,1,2). Обращение к элементу (2,2,2) вызовет ошибку или непредсказуемый результат. Так, если:

real x(7), b

x=5, b=-2

call assume(x)

write(\*,\*) b ! даст не -2, а 15

subroutine assume(a)

real a(2,2,\*)

a(2,2,2)=15

end

Так как следующей ячейкой памяти после заполнения x было b. И перенимающий размер массив а решил что при обращении к следующему элементу надо взять следующую ячейку памяти.

Если ячейка памяти будет пустой, тогда при компиляции возникнет ошибка.

Из-за такого рода ошибок и не рекомендуется кроме крайних случаев использовать перенимающие размер массивы.

2. Задание.

1.1. Where.

Продемонстрировать применение where вместо циклов. Показать применение elsewhere и множественных elsewhere.

1.2. For All.

Продемонстрировать применение forall вместо циклов. Показать достоинства и недостатки работы с массивом-маской. Использовать forall для задач где непринимы сечения массивов.

1.3. Автоматические массивы.

Написать процедуру с использованием автоматического массива, массива перенимающего форму и массива перенимающего размер.