1. Теория.

1.1. Модули и встраиваемые функции.

Существует 3 типа дополнительных программных единиц. Модуль, подпрограмма, функция.

Синтаксисы:

module pm

integer :: a, b, c ! данные переменные доступны везде где вызывается модуль

real, private :: e ! данная переменная доступна только в модуле

contains

subroutine plus(a,b,c)

c=a+b

e=1

end subroutine plus

subroutine minus(a,b,c)

c=a-b

e=-1

end subroutine minus

end module pm ! модуль представляет собой отдельный файл f90

program n1

use pm

integer :: d, mult

a=5

b=8

call plus(a,b,d)

print \*, d

call minus(a,b,d)

print \*, d

d= mult(a,b)

print \*, d

end program n1

integer function mult(a,b)

integer :: a, b

mult=a\*b

end function mult

Данная программа демонстрирует использование всех программных единиц.

Все переменные заданные в модуле без атрибута private являются глобальными. Они доступны в программе, где используется этот модуль, и ее подпрограммах.

Для определения правил передачи данных между переменными при использовании модулей и процедур используется атрибут intent. Он может обладать параметрами: in, out, inout, необходимыми для обмена информацией между модулем и основной программой. Если переменная описана как in – она не может быть изменена в процедуре. out – такие параметры всегда должны быть не определенны на входе в процедуру/модуль. inout – изменяемые параметры, которые определены сразу.

integer, intent (in) a, b

real, intent (inout) c

Функция должна иметь результирующую переменную (совпадающую с названием функции), именно она является результатом вызова функции. Можно разделить название и результирующую переменную: function mult(a,b) result(d). Требуется при этом описать переменную d и в функции и в программе.

При записи процедур можно использовать определение переменных с атрибутом optional. Тогда переменные с этим атрибутом будут использоваться в процедуре только если они переданы в нее.

1.2. Рекурсивные функции. Include.

Можно создать процедуры с прямой или косвенной рекурсией. Пример:

program fact

integer n, ifact

n=5

print \*, ifact(n)

recursive function ifact(n) result(fav) ! для рекурсивных процедур с прямой рекурсией имя функции и результирующая переменная должны отличаться обязательно

integer fav

integer, intent(in) :: n

if (n<=1) then

fav=1

else

fav=n\*ifact(n-1) ! вызывает процедуры пока n не станет меньше 1

end if

end

Для косвенных рекурсивных процедур запись такая же, но имя функции может совпадать с результирующей переменной. Если не объявить префикс recusive то обращение к самой процедуре в процедуре вызовет только ошибку.

Можно добавить текстовый файл в программе вместо строки используя:

include ‘имя файла’

В файле должен быть исходный код программы на языке Фортран.

При применении компилируется текст файла вместо строки с оператором include.

В файле также может использоваться оператор include. Таких вложений может быть до 10. В некотором роде такое использование является устаревшим, но может также пригодиться при некоторых задачах.

1.3. Return. Entry

Оператор return позволяет сделать несколько точек выхода из процедуры. Entry, напротив, позволяет сделать вход во внешнюю процедуру или модуль.

Например, если надо прекратить выполнение процедуры когда некоторая переменная становится отрицательной, то можно записать:

if (j<0) return

Так же возможен выход в определенное место программы, а не в момент вызова процедуры.

call alre(a, \*10, \*20)

print \*, 0

go to 40

10 print \*, -1

go to 40

20 print \*, 1

40 end

subroutine alre(a, \*10, \*20)

integer a

if (a .eq. 0) return ! возврат на строку после обращения к процедуре

if (a < 0) return 1 ! возврат на метку 10

return 2 !возврат на метку 20

end

Пример программы с несколькими входами в процедуру:

if (a .ge. 0) then

call vsign

else

call negative

end if

end

subroutine vsign

write (\*,\*) ‘>=0’

return

entry negative

write (\*,\*) ‘<0’

return

end

Надо помнить то такие возвраты, как и переходы по меткам с go to ухудшают структуру программы, потому пользоваться ими следует только в случае крайней необходимости.

2. Задание.

1.1. Модули и встраиваемые функции.

Написать программу использующую одновременно модуль, подпрограмму и функцию

1.2. Рекурсивные функции. Include.

Написать пример рекурсивной процедуры. Показать использование оператора include.

1.3. Return. Entry

Использовать операторы входа и выхода при написании процедуры.