Программирование

Лекция 12
Производные типы данных

Петров Александр Владимирович
Фёдоров Станислав Алексеевич

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

Октябрь 2013

Пример (1)

```
type :: wheel
    integer :: spokes
    real :: diameter, width
    character(len=15) :: material
end type wheel
```

```
type(wheel) :: w1
```

Определения

Запись - это переменная производного типа данных (структурного типа данных).

Производный тип данных (структура) – это одно или несколько объявлений переменных (как правило, разного типа), сгруппированных под одним именем. Структура должна вводиться в разделе объявлений программы.

Пример (2)

```
type :: bicycle
    character(len=80) :: description(100)
    type(wheel) :: front, back
    real, allocatable, dimension(:) :: times
    integer, dimension(100) :: codes
end type bicycle
```

Замечания

- Входящие в состав производного типа переменные называются его компонентами.
- По умолчанию объявленный в модуле производный тип доступен в любой программной единице, использующей модуль.
- При определении компонента производного типа могут быть использованы атрибуты: pointer, allocatable и dimension.

Присваивание (1)

- Оператор type, как и другие операторы объявления данных, предшествует исполняемым операторам.
- Запись является составной переменной. Для доступа к компоненту записи используется селектор компонента – символ процента (%)

```
type(bicycle) :: mine, yours
    yours = mine
mine%front = yours%back
```

Присваивание является единственной встроенной операцией

Присваивание (2)

```
type :: fred
    real :: x
end type fred

type :: joe
real :: x
end type joe

type(fred) :: a
type(joe) :: b

a = b ! ошибка
```

Конструктор производного типа (1)

```
type circle
    real :: x, y, radius
    logical :: filled
end type circle

type(circle) :: a

a = circle(1.23, 4.56, 2.0, .false.)
```

Fortran 2003:

```
a = circle(x = 1.23, y = 4.56, radius = 2.0, filled = .false.)
```

Конструктор производного типа (2)

Переменную производного типа можно определить, применив конструктор производного типа:

```
имя-типа (список выражений)
```

Конструктор может быть использован для инициализации записей в операторах объявления записей, в операторе **DATA**, в операторе присваивания, в выражениях (если выполнена перегрузка операций) и в качестве фактического параметра процедуры.

Аналогичный конструктор используется и для генерации констант производного типа:

```
имя-типа (список константных выражений)
```

Инициализация по умолчанию

```
type :: circle
    real :: x = 0.0, y = 0.0, radius = 1.0
    logical :: filled = .false.
end type circle

type(circle) :: a, b, c

a = circle(1.23, 4.56, 2.0, .true.)
```

или

```
a = circle(x = 1.23, y = 4.56)
```

Частные типы

```
module mammals
    type, private :: kangaroo
        real :: weight, length
    end type kangaroo

    real, private :: koala

contains

. . .

end module mammals
```

kangaroo, как и переменная **koala**, не доступны вне модуля **mammals**

Скрытые компоненты

```
module mammals
    type :: kangaroo
        private
        real :: weight, length
    end type kangaroo
contains
. . .
end module mammals
```

kangaroo доступен вне модуля **mammals**, а его компоненты **(weight, length)** нет

Присваивание значений компонентам записи

```
type item_d ! Описание заказанной вещи
    character(20) color, size ! Название, цвет, размер
    real(4) price ! Цена
end type
type order ! Описание заказа
    integer(4) ordnum ! Номер заказа
    type(item_d) item(10)
end type

type(order) cur_ord
cur_ord = order(1200, item_d('white', 'S', 35.25))

! Присвоим значение отдельному компоненту записи
cur_ord%ordnum = 1300 ! Изменим код покупателя

! Присвоим значение компоненту записи - элементу массива:
cur_ord%item(2)%color = 'blue' ! Изменим цвет второй вещи заказа

! Присвоим значение всему массиву - компоненту записи:
cur_ord%item%color = 'none'
```