



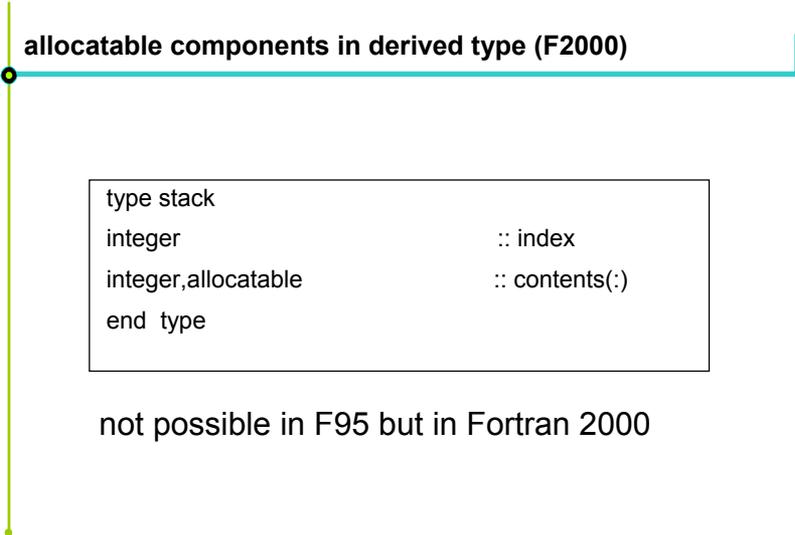
from Fortran-90  
to Fortran-95

Uwe Kuester  
kuester@hls.de



Uwe Küster

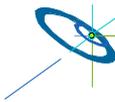
Höchstleistungsrechenzentrum Stuttgart



## allocatable components in derived type (F2000)

```
type stack
integer                :: index
integer,allocatable    :: contents(:)
end type
```

not possible in F95 but in Fortran 2000



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart



## F95: Pure functions: example part 1 (digital fortran compiler)

```
PURE INTEGER FUNCTION MANDELBROT(X)
  COMPLEX, INTENT(IN) :: X
  COMPLEX              :: XTMP
  INTEGER              :: K
  ! Assume SHARED_DEFS includes the declaration
  ! INTEGER ITOL
  USE SHARED_DEFS

  K = 0
  XTMP = -X
  DO WHILE (ABS(XTMP) < 2.0 .AND. K < ITOL)
    XTMP = XTMP**2 - X
    K = K + 1
  END DO
  MANDELBROT = K
END FUNCTION MANDELBROT
```



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: Pure functions: example part 2

```
COMPLEX              :: xx
INTEGER              :: I,J,M,N

.....
FORALL (I = 1:N, J = 1:M)
  xx = COMPLX((I-1)*1.0/(N-1), (J-1)*1.0/(M-1))
  A(I,J) = MANDELBROT(xx)
END FORALL
```



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: pure procedures

- procedures must be pure if they are
- referenced in a forall construct
- referenced in a specification statement
- passed as an actual argument to a pure procedure
- referenced in the body of a pure procedure (including those referenced by a defined operator or defined assignment)



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: elemental procedures

- elemental procedures are pure procedures with only scalar arguments
- they have the prefix **elemental**
- must not contain IO statements or stop
- variables cannot be
  - **save** or in a **data statement**
- global variables must not be changed
- dummy arguments
  - must have an **intent**
  - must be scalar
  - are not allowed to have the pointer attribute
  - cannot appear in a specification statement
  - cannot be dummy procedures
- **elemental procedures may be called with conforming arrays**



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## f95: elemental procedures, example function

- example

```
elemental function vip_calc(x, y)
  real :: vip_calc
  real, intent(in) :: x, y
  vip_calc = 3*x + y
end function vip_calc
```

! A call to vip\_calc with scalar arguments  
x = vip\_calc(1.1, 2.2)

! The result of this call is an array  
! conformable with a(1:n) and b(1:n)



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## f95: elemental procedures, example subroutine

- example subroutine

```
elemental pure subroutine source (a, b, c)
  real, intent(in) :: a, b
  real, intent(out) :: c
  ...
end subroutine source
```

```
real, dimension (2,2,3) :: s, t
real q
call source (q, s, t)
```



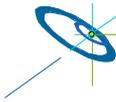
Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: forall

- forall ( v1=l1,u1:s1, ... , vn=ln:un:sn, mask )  
  a(e1, ... , em) = right\_hand\_side  
  ! calls of pure functions  
end forall
- loop indices are evaluated before calculation
- loop indices may be evaluated in any order
- each statement is completed before the next will be executed
- an array may appear only once on the left side
- all indices v1,v2,...,vn must appear in the array assignments on the left side
- pointer assignments are allowed



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: forall .. end forall is a parallel do .. enddo

where ( A /= 0.0 ) B = 1.0 / A

where ( A /= 0.0 )

B = 1.0 / A

endwhere

forall( i = 1:n, j = 1:n, a(i, j) .ne. 0.0 ) b(i, j) = 1.0 / a(i, j)

forall( i = 1:n, j = 1:n )

  where( a(i, j) .ne. 0.0 ) b(i, j) = 1.0 / a(i, j)

end forall



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: forall generalizes array syntax

- ! more general than array syntax:  
 $\text{forall}(i = 1:n, j = 1:n) \quad h(i, j) = 1.0 / \text{real}(i + j - 1)$
- this loop cannot be formulated in array syntax



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

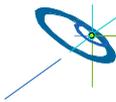
H L R I S 

## F95: forall ... end forall different from do ... enddo

- ! no recursion!
- ! Jakobi, not Gauss-Seidel

```
forall( i = 2:n-1, j = 2:n-1 )  
  d(i, j) = 0.25*(c(i, j + 1) + c(i, j - 1) + c(i + 1, j) + c(i - 1, j)) - c(i, j)  
  c(i, j) = c(i, j) + eps*d(i, j)  
end forall
```

first statement is evaluated completely before the second



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: masked elsewhere

- where (condition\_1)  
...  
elsewhere(condition\_2)  
...  
elsewhere  
...  
end where



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: minloc and maxloc with dim

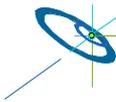
- the two-dimensional array  
$$A = \begin{array}{cccc} 0 & -5 & 8 & -3 \\ 3 & 4 & -1 & 2 \\ 1 & 5 & 6 & -4 \end{array}$$

- gives the following values

$\text{maxloc}(A)$  is (/ 1, 3 /)

$\text{maxloc}(A, \text{dim}=1)$  is (/ 2, 3, 1, 2 /)

$\text{maxloc}(A, \text{dim}=2)$  is (/ 3, 2, 3 /)



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

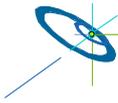
H L R I S 

## F95: initialization of pointers

- pointer can be initialized in specification and in derived types

```
real, pointer, dimension(:) :: vector => null()

type my_type
  real, pointer, dimension(:) :: vector => null()
end type my_type
```



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: default initialization of derived type components

- 

```
type my_type
  real :: value = -1.
  integer :: col
end type my_type
```



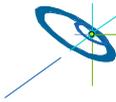
Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: IEEE arithmetic: plus zero and minus zero

- plus zero and minus zero are identical in
  - relational operations
  - in all intrinsics except sign
  - in the scalar expression in the arithmetic if
- `sign(x,y)` is
  - +`abs(x)` if `y > 0`
  - +`abs(x)` if `y` is plus zero
  - `abs(x)` if `y` is minus zero
  - `abs(x)` if `y < 0`.



Uwe Küster

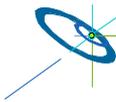
Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: intrinsic `cpu_time`

- the new intrinsic `cpu_time` delivers the cpu time in seconds

```
real :: time_in_seconds
call cpu_time(time_in_seconds)
```



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## F95: automatic deallocation of allocatable arrays

- allocatable are deallocated at the end of their scoping unit

```
subroutine allo(nmax,value)
  real,allocatable,dimension(:) :: array

  allocate(array(nmax))

  ...

  value=...

end subroutine allo
```



Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

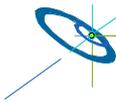
## F95: zero field edit descriptor

- An edit descriptor with zero field width indicates that as few columns as possible should be used to produce the result.
- On input, w must not be zero. On output, if it is zero, the processor selects the field width to be used; otherwise, it specifies the field width.

```
price = 44.23
print "(a, f0.2, a)", "The price is $", price, "."
```

- produces the output:

The price is \$44.23.



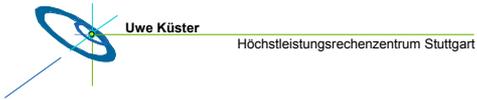
Uwe Küster

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

## used Fortran links

- <http://www.compaq.com/fortran/docs/lrm/dflrm.html>
- <http://asd-www.larc.nasa.gov/lectures/f95/>
- <http://www.nsc.liu.se/~boein/f77to90/f95.html>



end

