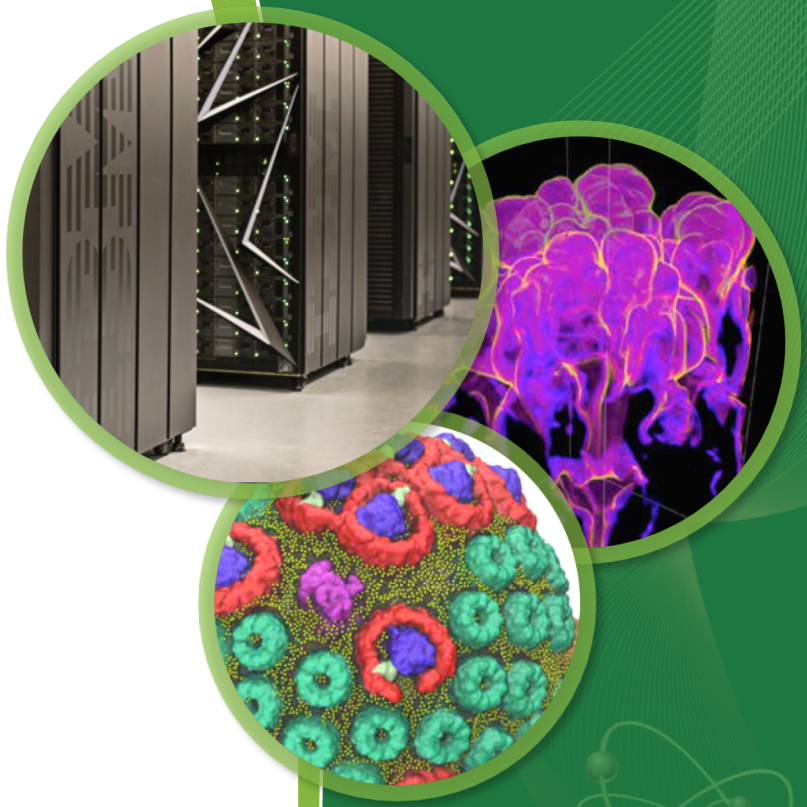
Introduction to FORTRAN

Bronson Messer

(after a presentation by Suzanne Parete-Koon)

OLCF

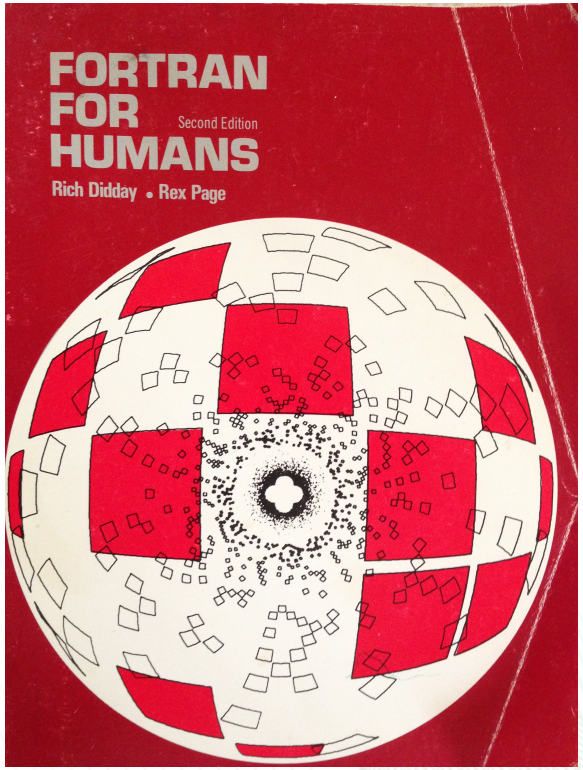


ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

"Fortran changed the terms of communication between humans and computers." ~ *New York Times*

- FORmula TRANslation – developed by IBM in the 1950s.
- Still widely used today. ~50% of OLCF production simulation codes (and these are, in many cases, the largest consumers of cycles as well).
- Fortran compilers can produce highly optimized executables.
- Fortran has true multidimensional arrays!
  - This is important for science – vectors, matrices, tensors…



Fortran basics

- Program structure
- Variables
- Loops
- Selection
- Arrays

Program Structure

```
Program program name
Variable declarations
Executable statements
[Subprograms]
End program name
```


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Basics

- First statement in code is program statement
  - Followed by program name

    program *myprog* (first line in source code)
  - Suggestion: give the source file the same name as the program

    myprog.f90 (name of source file)
- Last statement is a corresponding end program myprog

  (myprog optional)
- Language is *not* case sensitive (PROGRAM myProg works)
- Single blank space serves as delimiter
- But white space (multiple consecutive blanks) is ignored (*program myprog* is the same as *program myprog*)

Hello World Fortran

hello.f90

```
program hello
  write(*,*)“Hello World”
end program hello
```

write(*,*) – means write in the

default format, to STDOUT (the screen).

To compile:

```
[gfortran] hello.f90
```

To run:

```
./a.out
```

Variables FORTRAN

FORTRAN supports six different data types:

- Integer !32 bits
- Real !32
- Double precision (`REAL*8`) !64 bits
- Character
- Complex
- Logical

Variable Declaration Syntax

Type :: variable name

- Integer :: x
- Real :: fraction
- Character (len= 3) :: three_letter_word

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Hello+ (World) in Fortran

cp hello.f90 hello+.f90

vi hello+.f90

```
program hello
    implicit none
    integer:: x
    character (len=12):: phrase
        x=10
        phrase="hello world!"
        write(*,*) phrase, x
    end program hello
```

Compile `gfortran hello+.f90`

To run `./a.out`

`implicit none`: Your best friend

- In the 1950s computers only had a few KB of memory
- Programs needed to be as short as possible to fit
- Fortran variable types were implicit- you did not have to declare them.
  - All variables starting with i, j, k, l, m and n, if not declared, are of the INTEGER type by default.
- One side-effect: Typos are not caught by the compiler

```
numberyears=nubmeryear+1
```

ALWAYS use “implicit none”

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Comments

- Everything following a ! is a comment and will be ignored by the compiler

```
!This program demonstrates the basics
program hello
    implicit none          ! No implicit variables
    integer:: x            ! Number of iterations
    character (len=12):: phrase
        x=10
        phrase="hello world!"
        write(*,*) phrase, x   ! Write to screen
    end program hello          !End program
```


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Arithmetic Operations

- + Addition `z=y+x`
- - Subtraction `y=z-x`
- * multiplication `z=y*x`
- / Division `y=z/x`
- ** Exponentiation `three_squared= 3**2`

- Operator priority
  - ** is the highest; * and / are the next, followed by + and –
- Use () to ensure the desired priority

```
age=20+7*(h-2)
```


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Fortran loops

- `do loop` syntax

```
integer :: index
   . . .
do index=min,max
      operation(index)
enddo
```

```
Integer:: I
 Real  :: a
 a=1.01
 do i=1,10
     a=a+i
 enddo
write(*,*) a
```


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Hello++ World Fortran

cp hello+.f90 hello++.f90

vi hello++.f90

```
program hello
implicit none
integer:: x, i
character (len=12):: phrase
x=10
phrase="hello world!”
do i=1,x
   write(*,*) phrase, i
 enddo
end program hello
```

Vi Cheat sheet

To start ***vi hello++.f90***
To write ***i***
Delete
  if in write mode ***delete***
  if not in write mode ***x***
To stop writing ***esc***
Save ***:w***
Exit ***:q***

To compile `gfortran hello+.f90`

To run:`./a.out`

Hello World++ Fortran

```
suzanne@titan-ext6:~/crashcourse/fortran> ftn hello++.f90
suzanne@titan-ext6:~/crashcourse/fortran> ./a.out
 hello world!           1
 hello world!           2
 hello world!           3
 hello world!           4
 hello world!           5
 hello world!           6
 hello world!           7
 hello world!           8
 hello world!           9
 hello world!          10
suzanne@titan-ext6:~/crashcourse/fortran>
```


OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Selection FORTRAN

Syntax for `if` statements

IF (logical-expression) THEN

statements-1

ELSE

statements-2

END IF

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Fortran Selection

```
if (x < 10)then
  write(*,*) “low”
else
 write(*,*) “high”
Endif
```

As close to C syntax as can be imagined…

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Fortran Subroutines

Subroutine(arguments)
 body
end subroutine

Program mainpr
 call subroutine(par1)
 do something with par1
End mainpr

```
subroutine square (i,isquare)
  integer, intent(in)  :: i
  integer, intent(out) :: isquare
  isquare = i**2
end subroutine square

program sq
  implicit none
  integer :: i,isq,icub
  i = 4
  call square(i,isq)
  print*,"i,i^2=",i,isq,icub
end program sq
```

Array Fundamentals

- An array is a collection of data of the *same* type.
- Syntax:

type, DIMENSION(shape, shape) :: name1,name2,name3
or
type, DIMENSION:: name1(shapeA,shapeB), name2(shapeC, shapeD),…

- The rank shown above is 2.
- A three-dimensional array would have, e.g., (shape,shape,shape).
- Fortran90 can handle up to rank 7.
- The shape is the number of elements in that dimension.

Array Fundamentals

- There is one more attribute, extent, that allows you to control where the indices start. The default is to start at 1 (C indexing starts at 0).
- Modern FORTRAN does also have dynamic memory allocation

```
INTEGER ERR

INTEGER, ALLOCATABLE :: A(:), B(:)

...

ALLOCATE(A(10:25), B(SIZE(A)), STAT=ERR)  ! A is invalid as an
argument to function SIZE
```

Array Fundamentals

- One-dimensional
  - Real, dimension(3) :: A ! A 1D floating point array with thee elements
  - integer, dimension (5) :: B ! A 1D integer array with 5 elements
- Two-dimensional
  - Real, dimension(2,2):: A ! A 2D array, (2 by 2)
  - Integer, dimension(2,3):: B ! A 2D array (2 by 3)

  - Not covered here, but arrays in Fortran can be allocated, after they are declared.
  - Integer, dimension(x,y):: B ! A 2D array x and y can be set later in the program.

How arrays are stored in memory

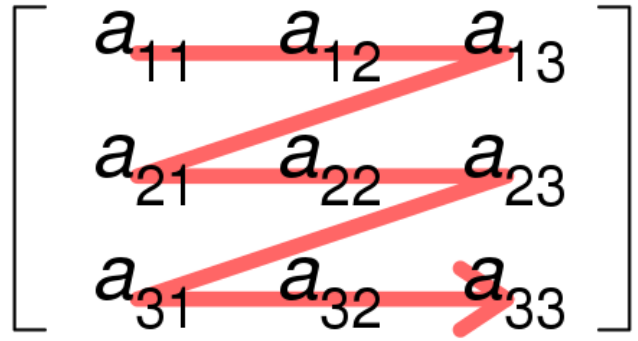
C

A[0][0] a11	A[0][1] a12	A[0][2] a13	A[0][3] a21	A[1][0] a22	A[1][1] a23	. . .

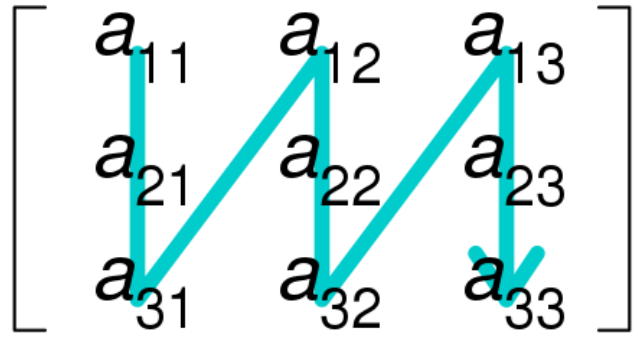
Fortran

A(1,1)	A(2,1)	A(3,1)	A(1,2)	A(2,2)	A(3,2)	. . .

Row-major order



Column-major order



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

array.f90

```
PROGRAM ARRAYTEST

INTEGER, PARAMETER :: COLSIZE = 10000
INTEGER, PARAMETER :: ROWSIZE = 20000
INTEGER :: array(ROWSIZE, COLSIZE)
INTEGER :: i
INTEGER :: j


DO  j = 1, COLSIZE

 DO i = 1, ROWSIZE

  array(i, j) = j*1.7*i

 END DO

END DO

END PROGRAM
```

```
> time ./a.out
```

github.com/bronson79/fortranTut.git

array_fortran.f90

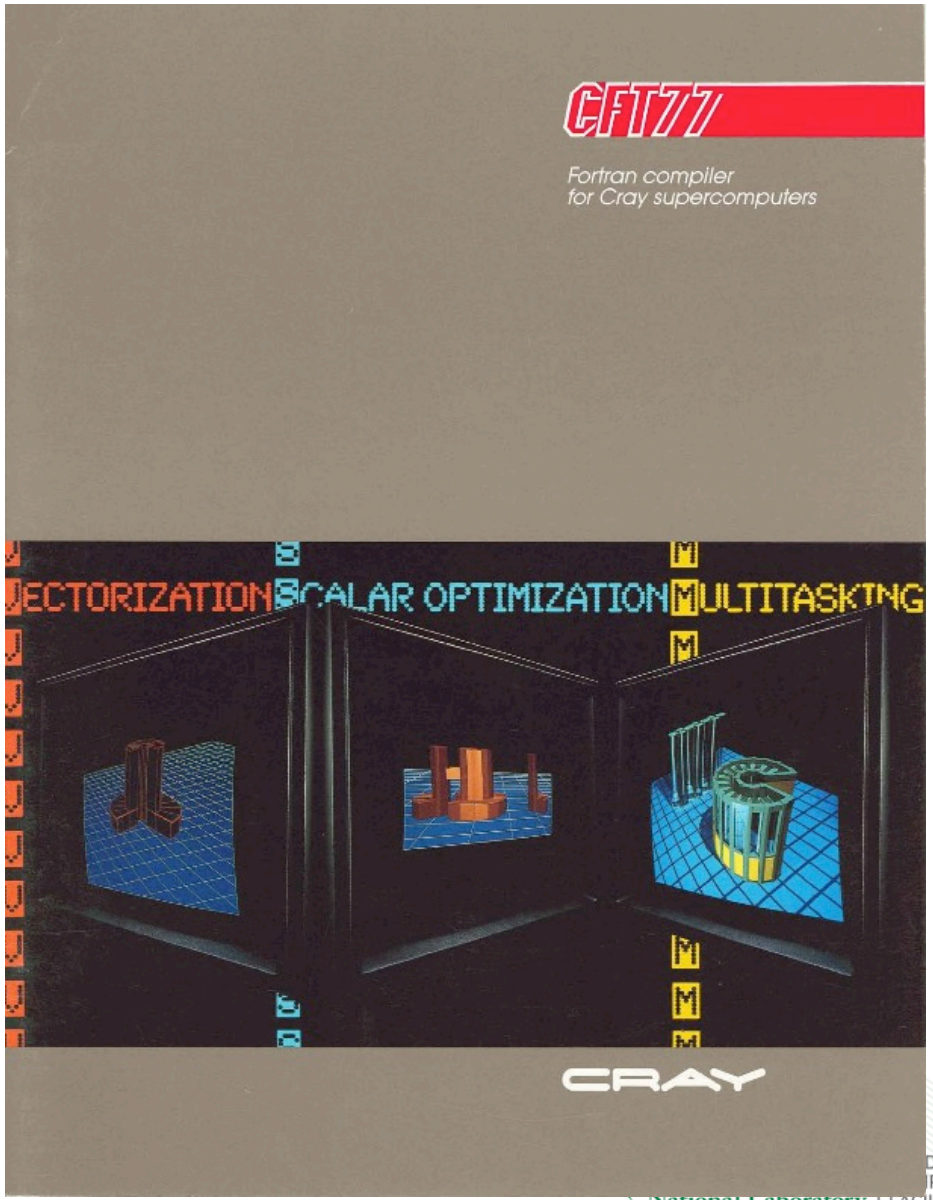
```
PROGRAM ARRAYTEST

INTEGER, PARAMETER :: COLSIZE = 10000
INTEGER, PARAMETER :: ROWSIZE = 20000
INTEGER :: array(ROWSIZE, COLSIZE)
INTEGER :: i
INTEGER :: j


DO i = 1, ROWSIZE

 DO  j = 1, COLSIZE

  array(i, j) = j*1.7*i

 END DO

END DO

END PROGRAM
```

```
> time ./a.out
```

github.com/bronson79/fortranTut.git

Questions?


FORTRAN
INPUT: REASON • OUTPUT: PLEASURE
Distributed by
EDUCATIONAL DESIGN, INC.
47 West 13th Street
New York, NY 10011


CFT77
Fortran compiler
for Cray supercomputers
ECTORIZATION
CALAR OPTIMIZATION
ULTITASKING
CRAY