Introduction To Fortran 90

CS 471/ Math 471
Brian T. Smith, UNM
Andrew C. Pineda, UNM

History

- Defined and implemented by Backus and his IBM team in mid 1950s
- Designed to demonstrate that a high level language for scientific computation could be efficient -- generate efficient object code
- Maintained its efficiency emphasis was well as portability, particularly for numeric computation, throughout its development

History Cont'd

- Standardized in 1966 by ANSI
- First programming language to be standardized by an official body
- Revised 1978 -- Fortran 77 by ANSI
- Rev.1992 -- Fortran 90 by ANSI and ISO
- Revised 1997 -- Fortran 95 by ISO
- Technical development of Fortran 2K now

References

- Programming Languages, Concepts and Constructs, Ravi Sethi, Addison-Wesley
- Programming Texts
  – The Fortran 90 Programming Guide, Adams, ...
  – Essential Fortran 90 and 95, Meissner
  – The Programming Language F, Brainerd, ...

Fortran 90 Reference Books

- The Handbook on Fortran 90, Adams, ..., McGraw-Hill
- The Handbook on Fortran 95, Adams, ..., Springer-Verlag
- Fortran Top 90, Adams ..., the top 90 features of Fortran 90

What Is A Programming Language?

- A tool for instructing machines
- A means of communicating between programmers
- A vehicle for expressing high-level designs
- A notation for algorithms
- A way of expressing relationships between concepts

A Programming Language?

- A tool for experimentation
- A means of controlling computerized devices
- A quote from Stroustrup, 1994
  - NOT a collection of “neat” features

What is Fortran’s Niche?

- Efficiency of execution of the object code
  - extensive freedom to rearrange code provided the result is mathematically equivalent
  - specifications only on what the statements of the language mean
  - no (intentional) constraints on how the language is to be implemented

Fortran’s Niche Cont’d

- Portability of source code
  - standardization
  - consistency with floating point hardware stnds
- Major support for scientific (numer) comp
  - extensive intrinsic library
  - extensive numeric precision control

Numeric Precision Control

- Multiple floating point precisions
- Complex numeric data types (2 precisions)
- Numeric inquiry and parameterizations
- Consistent with IEEE binary fl. pt. stnd
- Manipulation of numeric parts of floating pointer numbers

Fortran -- An Imperative (Procedural) Language

- Other imperative languages
  - C, Ada, Algols, Pascal, ...
- Other kinds of languages
  - functional languages (e.g. Lisp)
  - data flow languages (e.g. SISAL, Cantata)
  - object-oriented languages (e.g. Smalltalk, C++)
  - logic programming languages (e.g. Prolog)
  - specification languages (e.g. Ada, Z)

Characteristic of Imper. Lang.

- Emphasis on executable statements
- Emphasis on data, types, values, operators
- Emphasis on procedures (functions and subroutines) as a means of extension
  - to perform new operations or actions not intrinsic in the language

Imperative Language Cont'd

- Emphasis on:
  - efficiency
  - low level
  - systems and operations
  - imposes no discipline on interfaces between operations, procedures, parts

Programming Vs Natural Languages

- Both are a means of communicating, human to machine, human to human
- Both have a structure:
  - a natural language has a grammar and a set of conventions or rules for expressing thoughts
  - a programming language has a STRICT grammar and set of rules

Comparison Cont'd

- Both have a means for defining words
  - a natural language has a dictionary and usage manual
  - a programming language has a semantic definition of statements but essentially allows the user to define the words (e.g. data objects, operations, procedures)

Comparison Cont'd

- Each statement in a procedural programming language specifies an action or specifies the environment in which the action takes place.

Do Loops in Fortran 90

- It is a construct, that is, has some structure
  - a heading statement
  - a terminating statement
  - a block of executable statements
- Alternative forms of heading statement
  - do
  - do while(<exp>)
  - do <iteration_clause>

English Vs. Fortran

English	Fortran
• Book	Complete program
• Chapters	Subprogram
• Paragraphs	Parts of a program
• Title of chapter	Subprogram header
• Intro. paragraph	Specification part
• Middle paragraphs	Execution part
• Sections	Internal procedures

English Vs. Fortran Cont'd

- Summary paragraph — End statement
- Sentences — Statements
  - assertive — -- spec. statements
  - question — -- executable stmts
- Words — Tokens
  - defined in dictionary — -- identifiers, oper; -- keywords

English Vs. Fortran Cont'd

- Delimiters — Delimiters
  - periods, commas parens, question marks, blanks — -- periods, commas, parens, ...
- Parts of words — Parts of tokens
  - letters, apostrophe, digits — -- letters, digits, underscores
- Side (margin) notes — Comments

English Vs. Fortran Cont'd

- References — References
  - see ... — -- CALL statements
  - [ref] — -- Include statements; -- USE statements

Comparison Summary

- For a natural language, you learn rules for constructing sentences, paragraphs, sections, chapters, books, etc. and memorize meanings of words to construct the other constructs.
- In a programming language, you learn rules for constructing statements, subprograms, and programs, memorize semantics of individual statements, and construct and manipulate data objects with values to create a program.

An Example

- A simple Fortran program

```
program  hello                    ! Header statement

  character(10)  T                ! Specification part
  intrinsic  date_and_time

  call  date_and_time( time = T )     ! Exec. part
  print *, 'Hello, world: the time is ', T(1:2), ":", &
           T(3:4)

end program  hello                ! End statement
```


Do Loops Cont'd

- Iteration clause of the form:
  <integer_variable> = start, end [, stride]
  - e.g. I = 1, N
    J = N, 1, -1
    K = N+3, 2*N+10, L+3
- Terminating statement
  enddo
- No labels necessary

Do Loops Cont'd

- Ways to control a loop iteration:
  - EXIT statement
    - terminate the innermost loop and exit to the first statement after the enddo for this loop
  - CYCLE statement
    - terminate the current iteration of the innermost loop and start the next iteration at the top of the loop body
- Construct names to identify loops by name

Use of Construct names

- Exit and cycle from loops specified by name. For example:

```
outer: do i = 1, N
      inner:  do j = 1, M
                  ...
                  if( converge)  exit   inner
                  ...
                  cycle  inner
               enddo  inner
            enddo  outer
```


Exercise on Loops

- Write a Fortran 90 program to multiply a matrix A by a matrix B of order 3 using method 5 (by columns) and store the result in matrix C. Check your result by comparing matrix C with matrix CC defined by the statement
  CC = matmul(A,B)
- Test your program on the following matrices

Matrix Examples

$$A = \begin{bmatrix} -1.1 & .22 & -.33 \\ 8.4 & -5.556 & 6.6 \\ -.77 & 18 & -19.9 \end{bmatrix} \quad B = \begin{bmatrix} -1.1 & -2.2 & -3.3 \\ 5.5 & -2.2 & 1.1 \\ 1.1 & 1.1 & 1.1 \end{bmatrix}$$

- Repeat the test with the slightly different matrices -- replace the first row of A and the first column of B with 2^{19} times the first row of A and first column of B respectively. Replace the do loop DO K = 1, N with the DO loop DO K = N, 1, -1. Does it make any difference mathematically? Numerically?