FORTRAN, Part 4

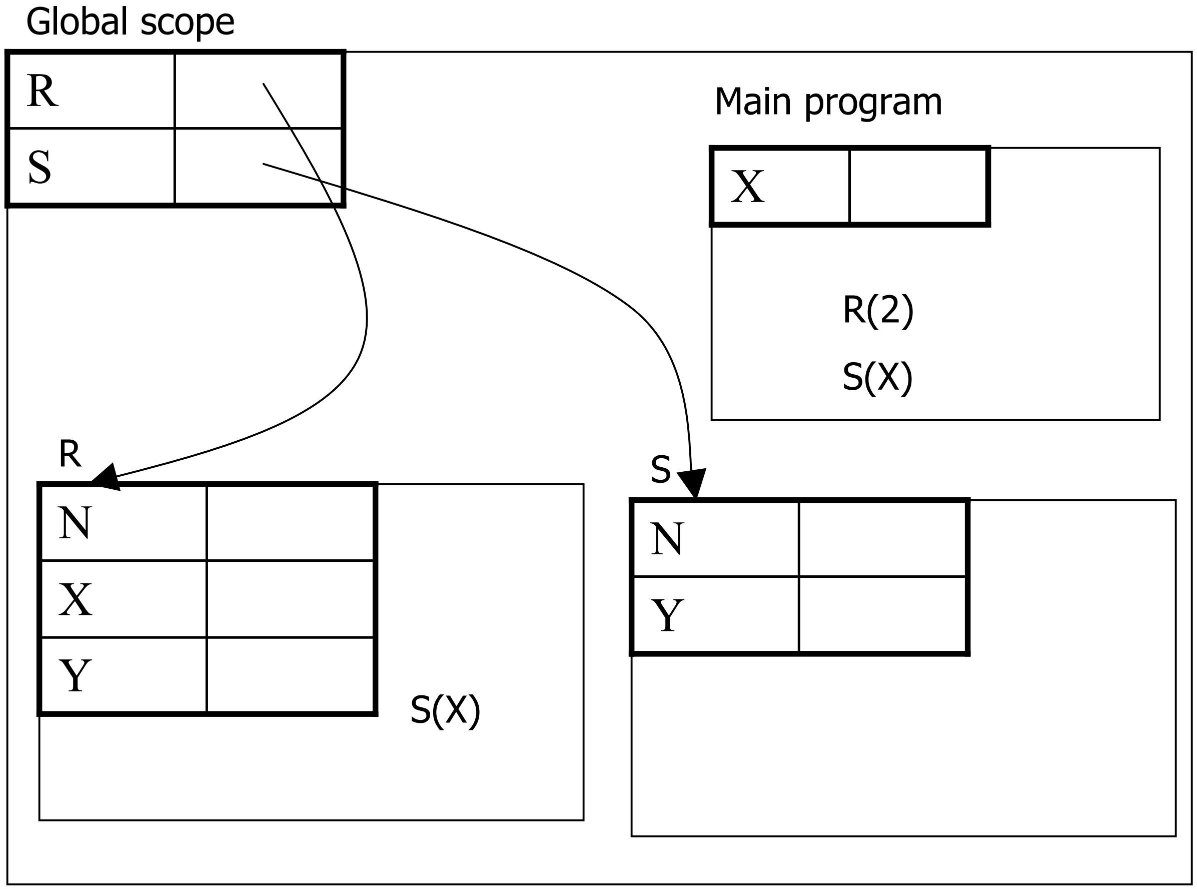
CS4100

February 21, 2011

SCOPE

- Scope of a binding of a name
  - Region of program where binding is visible
- In FORTRAN
  - Subprogram names GLOBAL
    - Can be called from anywhere
  - Variable names LOCAL
    - To subprogram where declared

Contour Diagram



Once we have subprograms…

- We need to find a way to share data
  - Parameters
    - Pass by reference
    - Pass by value-result
      - Caller copies value of actual to formal variable
      - On return, caller copies result value to actual
        - Omit for constants or expressions as actuals

Once we have subprograms…

- Share Data With Just Parameters?
  - Cumbersome, and hard to maintain
  - Produces long list of parameters
  - If data structure changes, there are many changes to be made
  - Violates information hiding

Sharing Data

- FORTRAN’s solution:
- COMMON blocks allow more flexibility
  - Allows sharing data between subprograms
  - Scope rules necessitation this
- Consider a symbol table

```
SUBROUTINE ARRAY2 (N, L, C, D1, D2)
COMMON /SYMTAB/ NAMES(100), LOC(100), TYPE(100)
...
SUBROUTINE VAR (N, L, C)
COMMON /SYMTAB/ NAMES(100), LOC(100), TYPE(100)
```

COMMON Problems

- Tedious to write
- Unreadable
- Virtually impossible to change AND
- COMMON permits aliasing, which is dangerous
  - If COMMON specifications don’t agree, misuse is possible

Aliasing

- The ability to have more than one name for the same memory location
- Very flexible!

```
COMMON /B/ M, A(100)

COMMON /B/ X, K, C(50), D(50)
```

EQUIVALENCE

- Since dynamic memory allocation is not supported, and memory is scarce, FORTRAN has EQUIVALENCE

```
DIMENSION INDATA(10000), RESULT(8000)
EQUIVALENCE INDATA(1), RESULT(8)
```

- Allows a way to explicitly alias two arrays to the same memory

EQUIVALENCE

- This is only to be used when usage of INDATA and RESULT do not overlap
- Allows access to different data types (float as if it was integer, etc.)
- Has same dangers as COMMON

DESIGN: Syntactic Structures

- Languages are defined by lexics and syntax
  - Lexics
    - Way to combine characters to form words or symbols
    - E.g. Identifier must begin with a letter, followed by no more than 5 letters or digits
  - Syntax
    - Way to combine symbols into meaningful instructions
- Syntactic analysis:

  Lexical analyzer (scanner)

  Syntactic analyzer (parser)

Fixed Format Lexics

- Still using punch-cards!
- Particular columns had particular meanings
- Statements (columns 7-72) were free format

Columns	**Purpose**
1-5	Statement number
6	Continuation
7-72	Statement
73-90	Sequence number

Blanks Ignored

- FORTRAN ignored spaces (not just white spaces)
- Thisisveryunfortunate!

```
DIMENSION INDATA(10000), RESULT(8000)
D I M E N S I O N I N D A T A (1 0 0 0 0), R E S U L T (8000)
DIMENSIONINDATA(10000),RESULT(8000)
```

- Lexing and parsing such a language is very difficult

Blanks Ignored

- In combination with other features, it promoted mistakes

```
DO 20 I = 1. 100
DO 20 I = 1, 100
DO20I = 1.100
```

- Variable DO20I is unlikely, but . and , are next to each other on the keyboard…

No Reserved Words

- FORTRAN allows variable named IF

```
DIMENSION IF(100)
```

- How do you read this?

```
IF (I - 1) = 1 2 3
IF (I - 1) 1, 2, 3
```

- The compiler does not know what `IF (I - 1)` will be
  - Needs to see , or = to decide

Algebraic Notation

- One of the main goals was to facilitate scientific computing
  - Algebraic notation had to look like math
  - (-B + SQRT(B**2 – 4*AA*C))/(2*A)
  - Very good, compared to our pseudo-code
- Problems
  - How do you parse and execute such a statement?

Operators Need Precedence

- $b^2 – 4ac == (b^2) – (4ac)$
- $ab^2 == a(b^2)$
- Precedence rules
  1. Exponentiation
  2. Multiplication and division
  3. Addition and subtraction
- Operations on the same level are associated to the left (read left to right)
- How about unary operators (-)?

Some Highlights

- Integer type is overworked
  - Integer
  - Character strings
  - Addresses
- Weak typing
- Combine the two and we have a security loophole
  - Meaningless operations can be performed without warning

Some Highlights

- Arrays
  - Only data structure
  - Data constructor
  - Static
  - Limited to three dimensions
  - Restrictions on index expressions
  - Optimized
  - Column major order for 2-dimensional
  - Not required to be initialized