Restructuring RELAP5-3D

2005 RELAP5 International Users Seminar

Dr. George L. Mesina & Joshua M. Hykes

September 7, 2005
Outline

• Purpose
• FORTRAN 90 programming
• Conversion Methodology
• Measurements
Purpose

• Convert interwoven logic flow paths (spaghetti) to structured blocks of coding
• Improvements (according to computer industry) gained by structuring the code.
  – Easier to read and understand
  – Less time required for code development
  – Reduced debugging time
  – Reduced cost for maintenance
• These will lead to greater robustness
Definition: Structured Programming

- From General Services Administration, Federal Standard 1037C (Telecom Glossary 2000)

• A **technique** for organizing and coding computer programs in which a **hierarchy of modules** is used, each having a **single entry** and a **single exit point**, and in which **control** is passed **downward** through the structure with **no unconditional branches** to higher levels of the structure.

There are three types of flow **control**:

• Sequential
• Test (**if** and **case**)
• Iteration (**loop**)
Definition of a “Block of Code”

• A module or block of code is a group of consecutive lines of code and/or smaller blocks that have:
  – A single entry point at the top
  – A single exit point
  – Execution or control passes downward through consecutive statements or blocks

• Examples

<table>
<thead>
<tr>
<th>Structured</th>
<th>Unstructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read (IN, FMT) A</td>
<td>Read (IN, FMT) A</td>
</tr>
<tr>
<td>B = A/3.14159265</td>
<td>10 B = A/3.14159265</td>
</tr>
<tr>
<td>Write (OUT) B</td>
<td>Write (OUT) B</td>
</tr>
</tbody>
</table>

• The second example has more than one entry point.
Flowcharts of Structured Blocks

- **Sequential**

  - STMT 1
    - STMT 2
    - STMT 3

- **Iteration / Loop**

  - Until
    - Body Block
  - While
    - Body Block

- **If / Case**

  - IF
    - Block T
    - Block F
  - CASE
    - Block 1
    - Block N
    - Block Default

One entry
One exit
Structured Programming

• Essentially, there are:
  – No GO TO statements (multiple entry)
  – No multiple returns (multiple exit)

• For loops, special structured GO TO statements:
  – EXIT – leave loop immediately when condition occurs and resume execution with statement after end-of-loop
  – CYCLE – leave iteration of loop immediately and resume execution with loop’s test statement
FOR_STRUCT

- FOR_STRUCT is a commercial software package for structuring unstructured code
  - Applies to FORTRAN IV, FORTRAN 66, and FORTRAN 77
    - Does not work on FORTRAN 90 code.
- Reformats code it restructures, for example:
  - Uniform spacing conventions
  - Uniform indentation
  - Resequencing of line labels
FOR_STRUCT Restructuring

**REPLACES**
if (.not. condition) go to

| if (.not.condition) go to 10  
| Block 1  
| go to 20  
| 10 Block 2  
| 20 continue |

**WITH**
if (condition) then

| if (condition) then  
| Block 1  
| else  
| Block 2  
| endif |

Arithmetic IF
Computed GO TO

IF-THEN-ELSE-ELSEIF
CASE
FOR_STRUCT Restructuring

REPLACES
• Do-loop continue statements
• Jump to end of iteration
• Jump out of loop
• Backwards go to
• Multiple returns in a subroutine

WITH
end do statement
cycle statement
exit statement
do while statement *
case statement and a single return

* Only if it is an actual loop.
FOR_STRUCT Limitations

• Some coding is so complex that FOR_STRUCT only partially restructures it.

• FOR_STRUCT cannot process pre-compiler directives.
  – #IFDEF and #INCLUDE

• FOR_STRUCT cannot process FORTRAN 90 code.
Overcoming FOR_STRUCT limits

• Partially restructuring
  – Applying FOR_STRUCT to its own output further restructures complex code.
  – We used 3 iterations.

• Pre-compiler directives
  – After applying pre-compiler, any coding that was removed is not restructured.
  – Restructure file several times with different flags active.
  – Recombine carefully.
Methodology: Complexity Control

• Files vary in complexity with:
  – Size of file
  – The number of different IFDEFS
  – The number of IFDEF branches
  – Nesting of IFDEFS

• Sorted files according number of IFDEFS and then according to size.
  – Process files from least complexity to greatest
  – Develop means to overcome each difficulty as it occurs.
Methodology: Work in stages

- **Stage 1 – Prepare file**
  - Prepare to apply CPP and FOR_STRUCT.
- **Stage 2 – Process file**
  - Apply CPP and FOR_STRUCT
- **Stage 3 – Post-processing file**
  - Essentially, undo the preparations
Stage 1: Preparing a file

- Replace F90 derived-type variables with dummy variables.
- Associate an index number with each IFDEF.
- Make “commented copies” of IFDEFS and INCLUDES.
- Append DEFINE heading(s) to file, usually creating multiple files.
  - Combinations of DEFINEs depend on:
    - Nesting
    - Mutually exclusive options
Preparing a file: Example

Original File

```c
ix = vlm(mi)%vctrls
#ifndef int32
   iip = ishft(is23(ix),-30)
#endif

C Set indexes in tables
11  if (s(ix) .ge. a(iip)) go to 10
    iip = iip - 1
    go to 11
10  continue
```

Prepared File

```c
ix = dummy1avctrls
Converted #ifndef 4.0.0.0 int32
#endif

C Set indexes in tables
11  if (s(ix) .ge. a(iip)) go to 10
    iip = iip - 1
    go to 11
10  continue
```
Stage 2: Processing a file

• Preprocess the file(s) with CPP
  – Expands INCLUDES
  – Eliminates some conditional code
• Run FOR_STRUCT iteratively on each file.
• Troubleshoot errors by manually changing the input or output file.
  – Usually involves moving anENDIF into or out of an IFDEF block
Processing a file: Example

After CPP

```c
ix = dummy1vctrls
Converted #ifndef 4.0.0.0
C~LIT_ON
Converted #endif 4.0.0.0
C~LIT_OFF
Set indexes in tables
11 if (s(ix) .ge. a(iip)) go to 10
  iip = iip - 1
  go to 11
10 continue
```

After FOR_STRUCT

```c
ix = dummy1vctrls
Converted #ifndef 4.0.0.0
C~LIT_ON
Converted #endif 4.0.0.0
C~LIT_OFF
Set indexes in tables
  do while (s(ix).lt.a(iip))
    iip = iip - 1
  end do
```

Note, the code protected with “#ifndef int32” was eliminated by CPP.
Methodology: Post Processing

- Substitute F90 variables for dummy variables.
- Combine files into one complete file.
  - Use IFDEF indexes to match blocks of code.
  - Verify the number of IFDEFs did not change.
- Uncomment the commented copies of IFDEFS and INCLUDES.
- Delete the included files.
- Fix the undesirable formatting details that FOR_STRUCT predictably produces.
- Run small test set; ensure output remains same.
Post Processing a file: Example

After FOR_STRUCT

ix = dummy1vctrls
Converted ifndef 4.0.0.0 int32
C~LIT_ON
Converted endif 4.0.0.0
C~LIT_OFF
C Set indexes in tables
   do while (s(ix).lt.a(iip))
      iip = iip - 1
   end do

After Post Processing

ix = vlm(mi)%vctrls
ifndef int32
   iip = ishft(is23(ix),-30)
endif
C Set indexes tables
   do while (s(ix).lt.a(iip))
      iip = iip - 1
   end do
Results

• 443 files in the RELAP subdirectory restructured.
  – 53 files need no restructuring.
• For the 443 restructured files:
  – Avg # GOTOs/subroutine
    • Before: 10.6, After: 5.4
  – Max # GOTOs in any subroutine
    • Before: 213, After: 146
  – Max # labels in any subroutine
    • Before: 210, After: 48