# §2.3-2.4: Problem Solving, Documentation

*devo*

15 Sep 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

***Reminders:***

***journals in folder***

# Review of 2.2

- Components of a baby Modula-2 program
- Modules
- Reserved words
- Library tools (what are some we know already?)
- Identifiers (what are some legit examples?)
- Strings, quoting, newlines
- Structure of a program module (railroad diagram)
  - MODULE, IMPORT, BEGIN, END.

# What's on for today (2.3-2.4)

- Steps to problem solving: WADES in more detail

- Analyze the problem: write, ask appropriate, rewrite
- Plan and revise a solution
- Data tables and I/O
- Pseudocode
- Implement in Modula-2 code
- Compile, link, and run (several times)
- Check output against specifications
- Documentation

# Steps to solving a problem


- Steps are an expansion of ***WADES***:

- Analyze the problem
- Plan a solution
- Write down your data tables and I/O
- Refine your solution (several times)
- Execute your plan (code) and evaluate the results

# Analyze the problem

- Step 1: Write the problem out
  - "Write a program that prints out a user-specified number of hash marks (#)."
- Step 2: Ask whether a computer is appropriate
  - Other ways to solve the problem?
- Step 3: Rewrite the problem in your own words
  - Given: number of hash marks to print
  - To do: print hash marks
  - Result: a string of hash marks, e.g., #######
  - Formula: none needed

# Plan and refine a solution

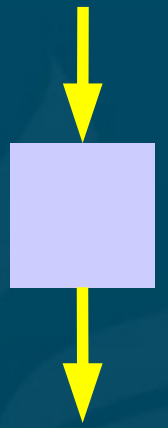
- Step 4: Re-use previous work where possible
  - Our program has input and output, so we will use the STextIO and SWholeIO libraries.
- Step 5: Break the problem into smaller steps
  - Input: read in desired number of hash marks
  - Computation: none
  - Output: print out hash marks

# Further refinements

- Second refinement:
  - Input:
    - Ask user for desired number of hash marks
    - Input response and assign to a cardinal variable
  - Computation:
    - Initialize a cardinal counter to zero
  - Output:
    - While the counter is less than the desired number of hash marks:
      - Print a hash mark
      - Increment the counter

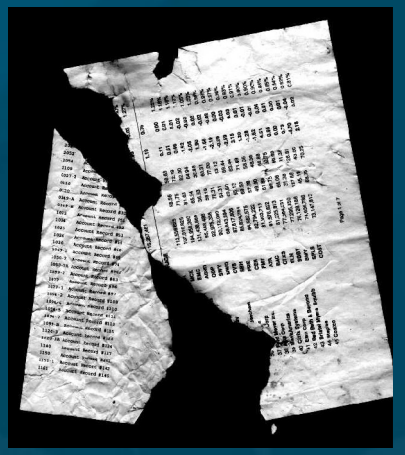
# Data tables and I/O

- Step 6: List all variables and imports (data table)
  - Variables: NumberOfHashes, counter: cardinals
  - Imports:
    - From STextIO: WriteString, WriteLn
    - From SWholeIO: ReadCard
- Step 7: List required input (precondition) and expected output (postcondition)
  - Input: A cardinal number ≥ 0, e.g. 6
  - Output: A string of hashes, e.g. "######"



# Refining the solution

- Step 8: Pseudocode
  - Print "How many hashes do you want printed?"
  - Read user input into NumberOfHashes
  - counter <---- 0
  - While (counter < NumberOfHashes)
    - Print "#"
    - counter <----- counter + 1



# Write the Modula-2 code

- Step 9: Modula-2 code (syntax matters here)
  - MODULE HashMarks;
  - FROM STextIO IMPORT
    - WriteString, WriteLn;
  - FROM SWholeIO IMPORT
    - ReadCard;
  - VAR
    - NumberOfHashes, counter: CARDINAL;
- (continued next page)

# Modula-2 code, cont.

- BEGIN
  - WriteString (“How many hashes do you want? ”);
  - ReadCard (NumberOfHashes);
  - counter := 0;
  - WHILE counter < NumberOfHashes
    - DO
      - WriteString (“#”);
      - counter := counter + 1;
    - END;
  - WriteLn;
- END Hashmarks.

# Execution and evaluation

- Step 10: Compile, link, run
  - First run:
    - How many hashes do you want? 4
    - ####
  - Second run:
    - How many hashes do you want? 0
    - (no output)
- Step 11: Check against specifications
  - Does program print the right number of hashes? No one-off errors?
  - What about weird input: 0, -1, 120, 5.3, abc?

# Documentation

- Document your thinking at every step, *even the ideas that didn't work!*
  - Programmer's diary: log of everything
- External documentation: outside the program
  - User manual:
    - What user input is required
    - What the user should expect the program to output
    - No details about program internals
- Internal documentation: within the program
  - Descriptive variable/module names
  - Comments in the code
  - Online help for the user

# Examples of internal documentation

- Good variable name: NumberOfHashes
  - Bad variable name: x, num, i
- Comments: (* in Modula-2 *)
  - (* loop NumberOfHashes times *)
  - WHILE counter < NumberOfHashes
    - DO
      - WriteString ("#"); (* print just one # *)
      - counter := counter + 1;
    - END;
- Online help:
  - "Enter 'h' for online help."

# Review of today (2.3-2.4)

- Analyze the problem: write, ask appropriate, rewrite
- Plan and revise a solution:
  - Reuse modules, break into smaller steps
- Data tables and I/O:
  - variables/imports, pre/postcondition
- Pseudocode
- Implement in Modula-2 code
- Compile, link, and run (several times)
- Check output against specifications
- Documentation

# TODO items

- Homework due tomorrow (Fri):
  - §1.11 # 35
- Reading: through §2.5 for Fri
- Quiz ch2 next Mon
- Lab 1 due next MTW in lab section
  - Short writeup ok