§2.5: Variables in Modula-2

devo

16 Sep 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

1) HW by lab section
2) journals in folder

TRINITY WESTERN UNIVERSITY

http://cmpt14x.seanho.com/

Announcements

- Running programs from the command line

Review of 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

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."

What's on for today (2.5)

- Variables
  - Names vs. values
  - Assignment operator
- Strongly typed
  - Declaring vs. initializing
- Standard identifiers
  - cf. reserved words

Variables: names and values

- A Modula-2 variable is a name for a memory location, the contents of which can be changed by a program.
  - VAR
    - NumberOfApples: CARDINAL;
- The assignment operator := is the means by which the name on the left is given the value on the right.
    - NumberOfApples := NumberOfApples + 1;

Strong typing

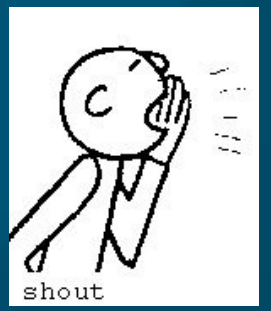
- Modula-2 is a strongly-typed language:
  - Before you can use a variable, you must declare it along with its type
  - x := 5;
    - What type is x? Cardinal? Integer? Real? Character?
  - VAR
    - x, y : REAL;
    - ch : CHAR;
- No ambiguity: we declare its type ahead of time
- Can't change type or assign a value of different type:
  - x := “Hello World!”;

Examples of type

- VAR
  - card1, card2 : CARDINAL;
  - real1, real2 : REAL;
  - char1, char2 : CHAR;
- Which are correct:
  - card1 := 30; Okay!
  - real1 := 3.0; Okay!
  - real2 := real1 + 2.6; Okay!
  - card2 := real1; Not okay!
  - char1 := “h”; Okay!
  - char2 := “hi”; Not okay!
  - card1 := char1 + 10; Not okay!

Declaring vs. initializing

- Declaring a variable tells the compiler what type it is:
  - VAR
    - NumberOfApples : CARDINAL;
- Its value is undefined until it is initialized:
  - BEGIN
    - NumberOfApples := 5;
- Always remember to both declare and initialize your variables before using them



Standard identifiers

- Similar to reserved words:
  - CARDINAL, INTEGER, REAL, CHAR, etc.
  - Don't use them for your variable names
  - A standard identifier is a name built-in to the notation. It is written all-caps.
- But not the same as reserved words:
  - Reserved words are structural punctuation
  - You can redefine standard identifiers (but you shouldn't!)
    - VAR CARDINAL : INTEGER; (* ick! *)

Review of today (2.5)

- Variables
  - Names vs. values
  - Assignment operator
- Strongly typed
  - What are examples of legal/illegal assignment?
  - Declaring vs. initializing
- Standard identifiers
  - cf. reserved words
  - Examples?

TODO items

- Quiz ch2 next Mon
- Reading: through §2.9 for Mon
- Lab 1 due next MTW in lab section
  - Short writeup ok
- Homework due next Wed: 2.14 # 33

- Have a good weekend and don't forget to do your quiet times and journal!