§2.6, 2.7, 2.9, 2.10: Literals, Constants, Expressions, and Reals

devo

19 Sep 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

*1) **journals** in folder*
*2) **quiz** today*

Announcements

- HWs, journals from last week handed back
- Clarification on Lab1:
  - Time-and-a-half for overtime hours:
  - e.g., 10hrs ---> 8hrs*rate + 2hrs*rate*1.5

Review of 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?

Quiz ch2 (3 questions, 20 marks, 10 minutes)

- Mark each of the following 6 strings with “ok” or “not ok” for being an identifier:

  StudentRecord **10thAnniversary**

  monthly budget **WriteString**

  z **twu.ca**

- What does this code snippet output?

```
WriteString (“Hello”);
WriteString (“World”);
```

- Write a complete Modula-2 program that reads a character from the user and prints it back to the screen.
  - Don't worry about “pausing” at the end of the program
  - Hint: remember the import and declaration blocks

Quiz ch2 answers (#1, 2)

- Mark each of the following 6 strings with “ok” or “not ok” for being an identifier: *(1pt each)*
  - StudentRecord Ok!
  - monthly budget Not!
  - z Ok!
  - 10thAnniversary not!
  - WriteString Ok!
  - twu.ca not!
- What does this code snippet output? *(2pts)*
  - WriteString (“Hello”);
  - WriteString (“World”);
- HelloWorld

Quiz ch2 answers (#3)

- Write a complete Modula-2 program that reads a character from the user and prints it back to the screen.
  (12pts: MODULE(1), IMPORT(2), VAR(3), ReadChar(3), WriteChar(2), END(1))
  (-1 for using WriteString instead of WriteChar)

```
MODULE EchoChar;
FROM STextIO IMPORT
    ReadChar, WriteChar, WriteString, WriteLn;
VAR
    ch : CHAR;
BEGIN
    WriteString ( "Please type a character and press enter: " );
    ReadChar ( ch );
    WriteString ( "Thank you.  You typed: " );
    WriteChar ( ch );
    WriteLn;
END EchoChar.
```

What's on today (2.6, 2.7, 2.9, 2.10)

- Literals
- Constants, how to initialize
- Operators on CARDINAL/INTEGERs
- Operators on REALs
- Type conversions among CARDINAL, INTEGER, REAL

Literals

- A literal is an entity specified by its value, rather than by a pre-declared name.
  - NumApples := NumApples + 15;
  - WriteString("Hello World!");
- Names vs. values:
  - Technically, 15 is a name
    - Its name is an encoding of its value
- Literals have an implied type(s):
  - myCardinal := 15;
  - myReal := 15.0;

Constants

- A constant in Modula-2 is an identified value that cannot be later re-assigned in a program.
- You have to initialize a constant at the same time you declare it:
  - CONST
    - MyAppetiteForApples = 20;
  - VAR
    - ApplesInBin : CARDINAL;
- Type is implicit from the initialization
  - 20: either CARDINAL or INTEGER

Constance Bennett, 1904-1965, actress

Expressions

- An expression is a combination of
  - Literals, constants, and variables,
  - Using appropriate operations (by type)

    12 – 7

    NumApples * 4

- A few operators we'll look at:
  - For CARDINAL/INTEGER:
    + - * / DIV MOD REM
  - For REAL: + - * /

Basic operators on CARDINALs/INTEGERs

- We've already seen + and -
- Multiplication is designated by * (asterisk/star)
- Division is a little more complicated with CARDINAL / INTEGER:
  - Result must also be a CARDINAL or INTEGER
  - / just gives integer part of the quotient:
    - 13 / 3 ----> 4
    - -13 / 3 ----> -4
  - DIV reduces dividend to next lower multiple:
    - 13 DIV 3 ----> 4
    - -13 DIV 3 ----> -15 DIV 3 ----> -5

More on division: REM and MOD

The Mod Squad

- REM is remainder of a / division,
  MOD is remainder of a DIV division
  - (a / b) * b + (a REM b) ----> a
  - (a DIV b) * b + (a MOD b) ----> a
- / and DIV produce same results for positive input
  thus also REM == MOD for positive input

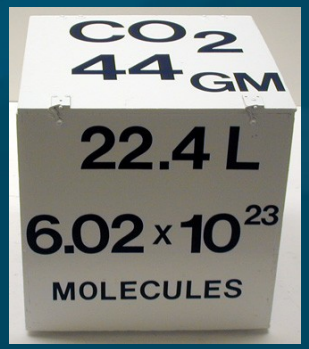
```
13 REM 3 ---> 1          13 MOD 3 ----> 1
-13 REM 3 ---> -1        -13 MOD 3 ---> 2
13 REM 0 ---> error (divide by zero)
```

Basic operators on REALs

- REAL numbers: must have decimal point
  - Optionally has scale factor (scientific notation)

    2.0, -5.62900, 6.022E23, 52.4E-04
- Operators: +, -, *, / (no DIV, REM, or MOD)
- No surprises here; everything works pretty much as you'd expect it

Amedeo Avogadro, 1776-1856, chemist



Initializing constants with expressions

- You can use certain expressions to initialize constants:

```
CONST
    numDogs = 10;
    numEars = numDogs * 2;
    numPaws = numDogs * 4;
```

- The expressions cannot include any variables

Precedence

- + and – are additive operators
- *, /, DIV, REM, and MOD are "muloperators"
- All muloperators have higher precedence than all additive operators
- Within the same precedence level, work left to right:

```
1 + 2 * 3 – 4 DIV 5 + 6
        ----> 1 + (2*3) – (4 DIV 5) + 6
```

Type conversions

- Okay (assignment compatible):

```
int1 := card1;
```

- Not okay (not expression compatible):

```
int1 := card1 + int2;
```

- But we can make it okay by converting the value in card1 from CARDINAL type to INTEGER type:

```
int1 := VAL (INTEGER, card1) + int2;
```

- Converting from INTEGER to CARDINAL is okay as long as the value is within range:

```
card1 := VAL (CARDINAL, int1) + card2;
```

Type conversions on REALs

- FLOAT converts INTEGER/CARDINAL to REAL:

```
real1 := FLOAT (card1) + real2;
```

- TRUNC converts from REAL to CARDINAL (in range):

```
card1 := TRUNC (real1) + card2;
```

- INT converts from REAL to INTEGER:

```
int1 := INT (real1) + int2;
```

- TRUNC and INT just drop the fractional part; they do not round off to the nearest integer

```
FLOAT (int1)      ==      VAL (REAL, int1)
TRUNC (real1)     ==      VAL (CARDINAL, real1)
INT (real1)       ==      VAL (INTEGER, real1)
```

Review of today (2.6, 2.7, 2.9, 2.10)

- Literals
- Constants (how to initialize?) (= vs. :=)
- Operators on CARDINAL/INTEGERs
  - /, DIV, REM, MOD
- Operators on REALs
- Type conversions among CARDINAL, INTEGER, REAL:
  - VAL, FLOAT, TRUNC, INT

TODO items

- Lab 1 due today/tomorrow/Wed in lab section
  - Short writeup ok
- Homework due Wed: 2.14 # 33
- Reading: through §2.11 for Wed