§4.3–4.7: Functions

devo

30 Sep 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

1) journals in folder
2) hw: #17, 36

Review of 4.1–4.2

- Procedures:
  - No parameters
  - Read-only parameters
  - Writeable parameters
  - Both kinds of parameters
  - Formal vs. actual parameters
  - Scope

What's on for today (4.3-4.7)

- Value vs. variable parameters
- Pre-/post-conditions (predicates)
- Function procedures

- Standard helper functions
- Naming conventions
- Debugging tips

Value and variable parameters

- Two kinds of parameters in procedures:

```
PROCEDURE CubeVolume (
    width : REAL;
    VAR volume : REAL );
....
CubeVolume (w, v);
```

- The first is a value parameter (call-by-value):
  - When the procedure is invoked, the value of the actual parameter (w) is copied into the formal parameter (width)
- Second is a variable parameter (call-by-reference):
  - Actual parameter (v) and formal parameter (volume) both are aliases for the same memory location

Value vs. variable param: example

```
PROCEDURE DoubleInc (x : REAL; VAR y : REAL);
BEGIN
    INC (x);
    INC (y);
END DoubleInc;

VAR
    myX, myY : REAL;
BEGIN
    myX := 10.0;
    myY := 20.0;
    DoubleInc (myX, myY);
```

$\boxed{\begin{array}{l} x = 11.0 \\ y = 21.0 \end{array}}$

$\boxed{\begin{array}{l} myX = 10.0 \\ myY = 21.0 \end{array}}$

***myX* is not modified!**

Predicates: pre-/post-conditions

```
PROCEDURE ASCIIChar (
    ascii : CARDINAL; VAR ch : CHAR );
BEGIN
    ascii := VAL (CHAR, ch);
END ASCIIChar;
```

- Value parameter ascii needs to be <128: either
  - State preconditions clearly in comments:

```
(* Convert from an ASCII code to a character *)
(* pre: ascii < 128
   post: ch = the character with ASCII code ascii *)
```

  - Or put error-checking code in the procedure

Error-handling example

```
PROCEDURE ASCIIChar (
    ascii : CARDINAL; VAR ch : CHAR );
(* Convert from an ASCII code to a character
    pre: none
    post: ch = the character with ASCII code ascii;
    ch is not modified if ascii is out of range *)
BEGIN
    IF ascii < 128
        THEN
            ascii := VAL (CHAR, ch);
        ELSE
            WriteString ("ASCIIChar: input must be < 128!");
            WriteString (" output unchanged");
        END;
END ASCIIChar;
```

Functions

- Functions are a type of procedure that returns a value:
  - CAP ('g') returns 'G'
  - PROCEDURE CubeMe (x : REAL) : REAL;

```
BEGIN
   RETURN x * x * x;
END CubeMe;
```

  - mySphereVol := (4.0/3.0)*Pi* CubeMe (radius);
- Functions must return a value (of proper type)
- Can have multiple RETURN statements; first one encountered ends execution of the function

Invoking functions

- A function call is not a complete statement:
  - Not OK: CubeMe (radius);
  - OK: volume := CubeMe (radius);
- Functions can have no arguments, but must still use parentheses when invoking:

```
PROCEDURE GetLowercaseChar() : CHAR;
    VAR ch : CHAR;
    BEGIN
        ReadChar (ch);
        RETURN (ch);
    END GetLowercaseChar;
userInput := GetLowercaseChar();
```

Standard helper procedures

- ABS (real1 : REAL) : REAL
  - Also works with INTEGERS
  - Test if two REAL numbers are equal:
    - Set epsilon to some small number, say 1E-5:
    - IF ABS(real1 – real2) < epsilon
- CHR (ascii : CARDINAL) : CHAR
  - Converts from ASCII code to character
- INC, DEC
  - Works with any scalar type:
    CARDINAL, INTEGER, REAL, LONGREAL, etc.

Standard helpers, cont.

- CAP (ch: CHAR) : CHAR
  - Convert to uppercase
- FLOAT (n) : REAL; LFLOAT (n) : LONGREAL
- INT (n) : INTEGER; TRUNC (n) : CARDINAL
  - Type conversion on scalar types
- ODD (n) : BOOLEAN (no EVEN)
  - Works on INTEGER, CARDINAL types
- MAX (type), MIN (type)
  - Maximum/minimum values for a scalar type
  - MAX (INTEGER) = 4294967295

Some notes on choosing names

- Variables and constants: numApples, myInput
  - Nouns
  - Begin with lowercase
  - Use capitals to separate words
- Procedures/functions: PrintUsage, ComputeVolume
  - Verbs
  - Begin with uppercase

Some debugging tips

- Do hand-simulation on your code
- Use WriteChar/Card/Real liberally
- Double-check for off-by-one errors
  - Especially in counting loops
- Try a stub program
  - General program structure of full program
  - Skip over computation/processing
    - Use dummy values for output
- Check out the debugger in Stony Brook

Review of today (4.3–4.7)

- Value vs. variable parameters: a.k.a.
  - call-by-value vs. call-by-reference
- Pre-/post-conditions (predicates): two choices:
  - Specify in documentation/comments
  - Code to check input for validity
- Function procedures
- Standard helper functions
- Naming conventions
- Debugging tips

TODO items

- Lab3 due next week:
  - §4.11 # (33 / 34 / 41) (choose one)
  - Full writeup!
- Quiz ch4: next Mon
- Reading: through §5.2.2 for Mon
- Midterm ch1-4: one week from today
  - (same day as MATH123 calc midterm)