§4.8–5.2: Recursion, Enumerations

devo

3 Oct 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

1) journals in folder
2) quiz ch4 today

Announcements

- Midterm ch1-4 this Friday in-class
  - Includes material in text not covered in class!
  - Expect questions similar to quizzes
  - Bring blank sheets of paper
  - Closed book/notes/laptop/phone/calc
  - Review on Thu
- Thanksgiving next Mon: no M lab section
- CMPT140 final W-Th 26-27Oct in-class
- CMPT145 final W 14Dec 2-4pm Neu13
- Student Alumni dinners

Student Alumni Dinners

Are you tired of Caf food?
What's the 'real' world like after graduation?

You are invited for a FREE, casual dinner at a Trinity Alumna's home to talk about your area of interest!

These dinners take place on various nights throughout the year.
If interested please sign up now!

student alumni network

Review of 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

Quiz ch4 (3 questions, 20 marks, 10 minutes)

- Describe in your own words the difference between value parameters and variable parameters.
- Write a Modula-2 procedure Swap that swaps the values of its two REAL parameters
  - e.g., if x=1.0 and y=2.0, then after invoking Swap (x, y), we should have x=2.0 and y=1.0.
- Write a function procedure SortPair that swaps the values of its two REAL parameters iff the first is greater than the second. The function should return TRUE iff a swap has been performed.
  - e.g. If x=1.0 and y=2.0, then
    - SortPair (x, y) should not change the values of x or y, and should return FALSE
    - SortPair (y, x) should result in y=1.0, x=2.0, and return TRUE

Quiz ch4 answers (#1-2)

- Value vs. variable parameter:
  - Value param: “read-only”, value from actual parameter is copied into formal parameter at invocation
  - Variable param: “writeable”, formal parameter is an alias to actual parameter
- PROCEDURE Swap (VAR x, y : REAL);

```
VAR
    temp : REAL;
BEGIN
    temp := x;
    x := y;
    y := temp;
END Swap;
```

Quiz ch4 answers (#3)

- PROCEDURE SortPair (VAR x, y : REAL) : BOOLEAN;

```
VAR
    temp : REAL;
BEGIN
    IF x > y
        THEN
            temp := x;
            x := y;
            y := temp;
            RETURN TRUE;
        END;
    RETURN FALSE;
END SortPair;
```

What's on for today (4.8-5.2)

- Recursion
  - Tail recursion, using loops instead
- Enumeration types
  - Ordinal types
- Subrange types
  - Expression, assignment compatibility

Recursion

- A recursive procedure invokes itself:
  - Factorial(n) = n! = 1 * 2 * 3 * ... * (n–1) * n

```
PROCEDURE Factorial (n : CARDINAL) : CARDINAL;
BEGIN
    IF n <= 1
        THEN
            RETURN 1;
        ELSE
            RETURN n * Factorial (n - 1);
        END;
END Factorial;
```

- Note that Factorial() invokes itself with n–1, not n
  - Otherwise it'd end up in an infinite recursion!

Uses of recursion

- Often a solution can be implemented using iteration (loops) instead of recursion:
  - Tail recursion: when self-invocation happens only at the end of the procedure
  - Recursion uses more CPU resources than iteration (procedure stack)
- But some problems are more clearly, elegantly solved using recursion
  - Fibonacci sequence; Towers of Hanoi example in the text

User-defined types

- Modula-2 allows us to define our own types in addition to the built-in types we've been using so far:
  - Atomic types
    - Scalar types
      - **Real types (REAL, LONGREAL)**
      - **Ordinal types**
        - **Whole number types (INTEGER, CARDINAL)**
        - **Enumerations (5.2.1)**
        - **Subranges (5.2.2)**
  - Structured (aggregate) types
    - Arrays (5.3)
      - **Strings (5.3.1)**
    - Sets (9.2-9.6)
    - Records (9.7-9.12)

today

Wed

Enumeration types

```
TYPE
    DayName = (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
VAR
    today : DayName;
BEGIN
    today := Mon;
```

- We could have used CARDINALs instead (and indeed the underlying implementation does)
  - But the logical semantic of today's type is a DayName type, not a CARDINAL
- Can be thought of as Sun=0, Mon=1, Tue=2, ...

Working with enumeration types

- INC and DEC work on enumerated types:

```
today := Mon;
INC (today);
```

  - But cannot increment/decrement past bounds:

```
today := Sat;
INC (today);          (* run-time error *)
```

- Cannot mix with cardinal types:

```
today := Mon + 1;   (* expression incompatible *)
```

- Comparison does work:

```
IF today < Thu
```

Enumerations are ordinal types

```
TYPE
    DayName = (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
VAR
    today : DayName;
    todayNum : CARDINAL;
BEGIN
    today := VAL (DayName, 2);          (* Tue *)
    todayNum := ORD (today);            (* 2 *)
    today := VAL (DayName, 7);          (* range error *)
```

- CHAR is also an ordinal type
- BOOLEAN can be thought of as an ordinal type:

```
TYPE BOOLEAN = (FALSE, TRUE);
```

Subranges

- Another kind of user-defined type is a subrange:

```
TYPE
    DayName = (Sun, Mon, Tue, Wed, Thu, Fri, Sat);
    WeekdayName = [Mon .. Fri];
    WeekdayName = DayName [Mon .. Fri];      (* alt. form *)
BEGIN
    weekday := Sat;                          (* error *)
    num := ORD (Mon);                        (* 1, not 0 *)
    weekday := VAL (WeekdayName, 1)          (* Mon, not Tue *)
```

  - Ordinal number of a subrange is same as host type

Subrange compatibility

- Subranges are expression compatible if the base types match exactly
- Subranges are assignment compatible if the base types are assignment compatible

```
TYPE
    TenCards = CARDINAL [1 .. 10];
    TenInts = INTEGER [1 .. 10];
    FiveCards = CARDINAL [1.. 5];
BEGIN
    tenInt := tenCard;                  (* ok *)
    tenInt := tenCard + fiveCard;       (* ok *)
    tenInt := tenCard + tenInt;         (* not expr. comp. *)
```

Review of today (4.8–5.2)

- Recursion
  - Tail recursion, using loops instead
- Enumeration types
  - Ordinal types
- Subrange types
  - Expression, assignment compatibility

TODO items

- Lab3 due tonight/tomorrow/Wed:
  - §4.11 # (33 / 34 / 41) (choose one)
  - Full writeup!
- Homework: §4.11 #19 due Wed (next class)
- Midterm ch1–4: this Friday!
  - (same day as MATH123 calc midterm)
- Reading: through §5.3 for Wed