§10.5-10.8: Program Control

- *devo*

14 Nov 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- ***journals** in folder*

What's on for today (10.5-10.11)

- Local modules
  - Import and export of items from modules
  - Qualified export
- Program control
  - General LOOP and EXIT
  - RETURN
  - HALT
  - FINALLY

Local modules

- Modules can contain:
  - IMPORTs
  - VAR/TYPE declarations
  - PROCEDURE declarations
  - A body (BEGIN ... END)
    - Also called the module's initialization section (c.f. library IMP-lementation module body)
  - Other modules!
- A local module is a construct for encapsulation:
  - Defines a scope of visibility for items

Local module example

```
MODULE Parent;
VAR
    pVar1, pVar2 : REAL;

    MODULE Child;
        IMPORT pVar1;
        EXPORT cVar1;
        VAR
            cVar1, cVar2 : REAL;
    END Child;

BEGIN
END Parent.
```

Visible:
pVar1, pVar2
cVar1

Visible:
pVar1,
cVar1, cVar2

All vars exist and keep value throughout Parent module, even when not visible!

Qualified export

- Cannot export two variables of same name to same scope of visibility:

```
MODULE Parent;
VAR myVar : REAL;
MODULE Child;
    EXPORT myVar;          (* bad!  (compile error) *)
    VAR myVar : REAL;
END Child;
```

- Solution: use qualified export:

```
EXPORT QUALIFIED myVar;    (* inside Child module *)
```

- Parent module is forced to use qualified form:

```
Child.myVar := 5.0;
```

Example: sibling modules

```
MODULE Parent;

    MODULE Child1;
        EXPORT c1var;
        VAR c1var: REAL;
    END Child1;

    MODULE Child2;
        IMPORT Child1;
        VAR c2var: REAL;
    END Child2;

END Parent.
```

Visible in Child1:
c1var

Visible in Child2:
c1var (aka Child1.c1var),
c2var

Visible in Parent:
c1var (aka Child1.c1var)

Summary of (10.5-10.7)

- Local modules
- Import and export of items from modules
- Qualified export

- The book covers more details in 10.6-10.7; please read them on your own:
  - Dynamic modules
  - Opaque types
  - Make sure you understand the Fibonacci example

Outline of (10.8-10.11)

- Program control:
  - Exiting loops:
    - General LOOP and EXIT
  - Exiting procedures and modules:
    - RETURN
  - Exiting the whole program:
    - HALT
  - Cleanup and module termination:
    - FINALLY

Loop control: LOOP and EXIT

- We know three kinds of loops:
  - WHILE/DO, REPEAT/UNTIL, FOR
- There is also a general LOOP that allows EXITs:

```
LOOP
    (* statement sequence *)
    IF WantToExit THEN
        EXIT
    END;
END;
```

- EXIT jumps to end of corresponding LOOP
- WHILE and UNTIL can be written as LOOPs

Program control: RETURN

- We've used RETURN with function procedures to finish the procedure and return a value
- RETURN can also be used (without a return value) in regular procedures or modules:

```
IF CAP (ch) = "Q" THEN
    RETURN
END;
```

  - Terminates execution and transfers control back to calling context
- Normal program termination: via RETURN or just running through to the end

Program control: HALT

- HALT is abnormal termination:
  - Aborts main program from anywhere:
    - Can be within procedure
  - Rarely used (only for extreme errors)
  - An error message is usually printed

- HALT vs. RETURN:
  - Abnormal vs. normal termination
  - RETURN only terminates current procedure/module; HALT aborts main program

Program termination: FINALLY

- Modules may have an optional FINALLY clause at the end of the body:

```
MODULE MyModule;
    (* imports, type/var/procedure declarations *)
BEGIN
    (* main body of module *)
FINALLY
    (* Cleanup and safe termination of program *)
END MyModule.
```

- RETURN in module body jumps to FINALLY clause
- Use FINALLY to close files, print closing message to user, “pause”, etc.

HALT and FINALLY

- When a HALT or run-time error is encountered, the FINALLY clause of every module whose body has started to execute is run
- Could have multiple modules in use:
  - Their FINALLY clauses are run in reverse order
    - Order depends on order of IMPORTs
- Distinguish whether you got to a FINALLY clause via HALT or RETURN (or normal termination):

```
TERMINATION.HasHalted() : BOOLEAN;
```

Review of today (10.5-10.11)

- Local modules
- Import and export of items from modules
- Qualified export

- General LOOP and EXIT
- RETURN
- HALT (vs. RETURN?)
- FINALLY

TODO items

- Lab 7 due today/tomorrow/Wed:
  - 9.14 #(37 + 38) / (40 + 41)
- Paper topic due Wed
- Reading: through §10.14 for Wed
- HW due Fri: 9.14 #30
- Quiz ch10 Fri
- Midterm ch8-10 Wed 23Nov (next week)