# §9.11-10.4: RndFile, Scope

•*devo*

7 Nov 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

***Reminders:***

- ***journals in folder***

## Review of last time (9.7-9.10)

- Records
  - Defining record types
  - Fields
  - Initializing record variables
  - WITH
- Using records and arrays
  - Example: Class of students
- Output of aggregate data

## Summary of today (9.11-10.4)

- RndFile: random-access files
  - OpenOld/OpenClean, NewPos/SetPos
- Scope, visibility, blocks
- Rules of thumb about variables/parameters
- Procedure variables

# Recap of raw record I/O

- Storing an array of records in raw binary form:

```
FROM StreamFile IMPORT
    Open, Close, ChanId, OpenResults, read, write, old, raw;
IMPORT RawIO;
VAR
    cmpt14x : ARRAY [0..29] OF Student;
BEGIN
    Open (cid, "test.out", write+raw+old, res);
    RawIO.Write (cid, cmpt14x);
    Close (cid);
```

- Reading is similar:

```
    Open (cid, "test.out", read+raw, res);
    RawIO.Read (cid, cmpt14x);
```

# I/O to just the nth record

- If we want to access the nth record from disk,
  - Read in entire array:

    ```
    RawIO.Read (cid, cmpt14x);
    cmpt14x [n-1];
    ```
  - Or read one record at a time:

    ```
    FOR idx := 1 TO n DO
        RawIO.Read (cid, currentStudent);
    ```
  - Or index into random-access file
- Each record has fixed length: SIZE (Student):
- Set position in file to (n-1) * SIZE (Student)
  - Just read in the record we need

# RndFile: random-access files

- RndFile is analogous to StreamFile/SeqFile
- NewPos makes a file position (type: FilePos) at a given number of chunks after a starting point:
    - NewPos (cid: ChanID; chunks : INTEGER; chunkSize: CARDINAL; from: FilePos): FilePos;
- SetPos applies a FilePos to a given channel:

```
FROM RndFile IMPORT
    OpenOld, Close, ChanId, OpenResults, read, write, raw;
OpenOld (cid, “file.bin”, read+raw, res);
top := StartPos (cid);
recordSize := SIZE (Student);
pos := NewPos (cid, n, recordSize, top);
SetPos (cid, pos);
RawIO.Read (cid, currentStudent);
```

## Summary of I/O drivers

- StreamFile: restricted streams
  - Open, Close
- SeqFile: rewindable streams
  - OpenRead, OpenWrite, OpenAppend, Close
  - Reread, Rewrite
- RndFile: random-access files
  - OpenClean, OpenOld, Close
  - StartPos, CurrentPos, EndPos, NewPos
  - SetPos

# Ch10: Scope and visibility

- Lots of places for variables:
  - Global (VAR block in MODULE)
  - Local to procedure (VAR block in PROCEDURE)
  - Parameters in a procedure
  - In libraries: DEF vs. IMP
- A variable is visible to a part of a program if it is available for use there
- A variable's scope is where it is visible
- Different languages have different scope rules

## Blocks

- An M2 block is the declaration+body of a module or procedure:
  - VAR, CONST, TYPE, IMPORT
  - BEGIN … END
- Parameters and things declared within a block are local to that block, and global to any other blocks enclosed within that block:

```
MODULE Parent;
VAR globalVar : REAL;
    PROCEDURE Child (param: REAL);
    VAR localVar : REAL;
END Child;
```

# Side-effects and global variables

- Be careful about unintended side-effects:

```
VAR counter : CARDINAL;
```

```
PROCEDURE LoopOne ();
BEGIN
    counter := 1;
    WHILE counter <= 10 DO
        do stuff
        INC (counter);
    END;
END LoopOne;
```

```
PROCEDURE LoopTwo ();
BEGIN
    counter := 1;
    WHILE counter <= 20 DO
        do stuff
        LoopOne;
        INC (counter);
    END;
END LoopTwo;
```

- Solution: local counter for each procedure

# Rules of thumb for parameters

- Always choose the correct kind of parameter: value or VAR
- Minimize the use of global variables:
  - Declare a variable in the smallest scope allowable
- Avoid reusing variable names:
  - Local variable hides global var of same name
- Use functions to return results rather than VAR parameters, as much as possible
- Don't use VAR parameters in functions

## Procedure variables

- We can assign and pass procedures around just like any other variable:

```
TYPE
    RectCalcType = PROCEDURE (REAL, REAL): REAL;
VAR
    RectCalc : RectCalcType;
PROCEDURE Perimeter (w, h: REAL): REAL;
PROCEDURE Area (w,h: REAL): REAL;

BEGIN
    RectCalc := Perimeter;
    RectCalc := Area;
```

## Summary of today (9.11-10.4)

- RndFile: random-access files
  - OpenOld/OpenClean, NewPos/SetPos
- Scope, visibility, blocks
- Rules of thumb about variables/parameters
- Procedure variables

## TODO items

- Lab 7 due today/tomorrow/Wed:
  - 8.13 #(53 / 60 / 62)
- HW due  Wed: 9.14 #30
- Quiz ch9 Wed
- Reading: through §10.7 for Wed