§10.12: Exceptions

16 Nov 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- ***journals** in folder*
- ***Paper** topic*

Review of last time (10.5–10.11)

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

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

What's on for today (10.12)

- Exceptions: another level of error handling
  - Raise/handle (a.k.a. throw/catch)
  - EXCEPT clause:
    - Do nothing, RETURN, RETRY
  - Built-in exceptions: M2EXCEPTION:
    - M2Exceptions, IsM2Exception(), M2Exception()
  - Standard library exceptions: e.g., IOChan:
    - ChanExceptions, IsChanException(),ChanException()
  - User defined exceptions: how to raise/handle
  - Exceptions and termination

Options for error handling

- Use a combination of these:
  - Ask the user to be nice:
    - User manual, precondition comments, WriteString
  - Print an error message to screen
  - Set a result flag:
    - As a parameter: Open(...., res)
    - Accessible via separate function: ReadResults()
  - Panic and die (HALT)
  - Raise an exception: DIVIDE_BY_ZERO

Exceptions

- When an exception is raised (thrown),
  - execution of the current procedure stops, and
  - Control jumps to the nearest exception handler (catches the exception)
- The exception handler can cleanup and either
  - Pass control on to next outer handler, or
  - RETURN back to whoever called the procedure that raised the exception, or
  - RETRY the procedure from the start
- If the exception reaches outermost level, an error message is automatically generated

EXCEPT clause

- Any procedure or module body can have an exception handler: EXCEPT clause
  - FINALLY clause can have its own EXCEPT:

```
MODULE MyModule;
BEGIN
    (* module body may raise an exception *)
EXCEPT
    (* handler for exceptions raised in body *)
FINALLY
    (* finalization clause *)
EXCEPT
    (* handler for exceptions raised in FINALLY clause *)
END MyModule.
```

M2 built-in exceptions

- Built-in exceptions are just an enumeration:

```
M2EXCEPTION.M2Exceptions =
    (indexException, rangeException, ... );
```

- An exception handler can check if an exception has been raised:

```
IF M2EXCEPTION.IsM2Exception() THEN ...
```

- And find out which exception has been raised:

```
IF M2EXCEPTION.M2Exception() = rangeException
```

Example: handling div-by-0

```
MODULE DivByZero;
FROM M2EXCEPTION IMPORT
    M2Exceptions, M2Exception, IsM2Exception;
VAR myInt : INTEGER;
BEGIN
    myInt := 5 / 0;
EXCEPT
    IF IsM2Exception() AND
        (M2Exception() = wholeDivException)
        THEN
            WriteString ("Divided by zero, but we're okay.");
            RETURN;          (* exception is cleared *)
        END;
END DivByZero.
```

Standard library exceptions

- Some standard libraries define their own exceptions similarly to the built-in exceptions:

```
MODULE IOChan;
TYPE ChanExceptions =
    (wrongDevice, notAvailable, skipAtEnd, ... );
PROCEDURE IsChanException() : BOOLEAN;
PROCEDURE ChanException() : ChanExceptions;
```

- Handle these exceptions in similar way

User-defined exceptions

- You can also define your own exceptions:
  - Define an enumeration type:

```
TYPE MyEx = (Goodness, Badness, Ugliness);
```

  - Register with the M2 exception system:

```
FROM EXCEPTIONS IMPORT
    ExceptionSource, AllocateSource;
VAR mysrc : ExceptionSource;
AllocateSource (mysrc);
```

- Raise your exception with EXCEPTIONS.RAISE:

```
RAISE (mysrc, ORD (Badness), "The sky has fallen!");
```

Handling your own exceptions

- EXCEPTIONS pseudomodule:
  - IsExceptionalExecution (): BOOLEAN;
    - Check if any exception has been raised
  - IsCurrentSource (mysrc): BOOLEAN;
    - Check if the exception was raised by this source
  - CurrentNumber (mysrc): CARDINAL;
    - Which exception was raised (ORD of enumeration type)
  - GetMessage (string);
    - Get the string message associated with the exception that was raised

User-defined exceptions example

```
FROM EXCEPTIONS IMPORT ....
TYPE MyEx = (Goodness, Badness, Ugliness);
VAR
    mySrc : ExceptionSource;
    msg : ARRAY [0..30] OF CHAR;
BEGIN
    AllocateSource (mySrc);
    RAISE (mySrc, ORD (Badness), "Sky has fallen!");
EXCEPT
    IF IsExceptionalExecution() AND IsCurrentSource(mySrc)
        THEN
            GetMessage (msg);
            WriteString (msg);
        END;
```

User-defined exception helpers

- It may be useful to provide some helper functions with your user-defined exceptions:
  - PROCEDURE IsMyEx () : BOOLEAN;

```
RETURN IsExceptionalExecution() AND
    IsCurrentSource (mySrc);
```

  - PROCEDURE MyException () : MyEx;

```
RETURN VAL (MyEx, CurrentNumber (mySrc));
```

- Analogous to TYPE M2Exceptions, IsM2Exception(), and M2Exception()

EXCEPT and FINALLY

- All module and procedure bodies can have EXCEPT clauses
  - The FINALLY clause can also have an EXCEPT!
- Exceptions raised in the body get handled by the body's EXCEPT clause
- Unhandled exceptions result in termination (go to FINALLY clause)
- The FINALLY clause may raise more exceptions;
  - They get handled in FINALLY's EXCEPT clause

Summary of today (10.12)

- Exceptions: another level of error handling
  - Raise/handle (a.k.a. throw/catch)
  - EXCEPT clause:
    - Do nothing, RETURN, RETRY
  - Built-in exceptions: M2EXCEPTION:
    - M2Exceptions, IsM2Exception(), M2Exception()
  - Standard library exceptions: e.g., IOChan:
    - ChanExceptions, IsChanException(),ChanException()
  - User defined exceptions: how to raise/handle
  - Exceptions and termination

TODO items

- HW due Fri: 9.14 #30
- Quiz ch10 Fri
- Reading: through end of §10 for tomorrow
- Lab #9 next week: 10.15 #(44 / 49)
- Midterm ch8–10 Wed 23Nov (next week)