§11.4–11.9: Variant Records

18 Nov 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- ***journals** in folder*
- ***Homework** due*
- ***Quiz** today*

What's on for today (11.4–11.9)

- Paper topics:
  - Choose case-studies for evidence to support your thesis
  - Make sure your main point (thesis) is very clear
  - Visit the Writing Centre in Douglas 2nd floor
- Constructors: Type { list }
  - Set constructors
  - Array constructors
  - Record constructors
- Variant records

Quiz ch10: 5 questions, 20 marks, 10 min

- What keyword delimits a termination clause?
- Describe all the differences between HALT and RETURN
- What are the 3 steps needed to define and throw your own exception?
- For each module (Child1, Child2, Parent) on the next page, name all visible variables.
- Write a complete module that throws a wholeDivException exception and handles it
  - Hint: use M2Exceptions, IsM2Exception(), M2Exception() in the M2EXCEPTION library

Quiz ch10 #4

```
MODULE Parent;
VAR pvar: REAL;
MODULE Child1;
    EXPORT c1var;
    VAR c1var: REAL;
END Child1;

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


END Parent.
```

Visible in Child1:

Visible in Child2:

Visible in Parent:

Quiz ch10 answers: #1-3

- Keyword delimits termination clause? FINALLY
- HALT vs. RETURN:
  - HALT: abnormal termination, exits out of whole program (jumping to FINALLY clause), usually prints error message to user
  - RETURN: normal termination, only exits current procedure/module
- Define and throw your own exception:
  - Declare enumeration type for exceptions
  - AllocateSource()
  - RAISE your exception

Quiz ch10 answers: #4

```
MODULE Parent;
VAR pvar: REAL;
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)
pvar

Quiz ch10 answers: #5

- Throw wholeDivException and handle it:

```
MODULE ExceptionExample;
FROM M2EXCEPTION IMPORT
    M2Exceptions, M2Exception, IsM2Exception;
VAR myInt : INTEGER;
BEGIN
    myInt := 5 / 0;          (* throws wholeDivException *)
EXCEPT
    IF IsM2Exception() AND
        (M2Exception() = wholeDivException) THEN
        (* could do more here *)
        RETURN;
    END;
END ExceptionExample;
```

Constructors

- M2 allows us to initialize aggregate variables directly with constructors:
  - Specify the aggregate type name, followed by a list of entries in curly braces: Type { list }
- Set constructors:

```
TYPE
    Apple = (Ambrosia, Fuji, Gala, Rome);
    AppleSet = SET OF Apple;
bagOfApples := AppleSet {Fuji, Ambrosia, Gala};
```

- Array constructors
- Record constructors

Array constructors

- Constructors work for arrays, too:

```
TYPE
    Vector7D = ARRAY [0..6] OF REAL;
myVec := Vector7D {2.3, 0.9, -1.5, 6.7, 1.2, 0.8, 5.4};
```

- All entries in the array must be given:

```
myVec := Vector7D {2.3, 0.9};        (* error! *)
```

- As a shortcut, BY indicates repetition:

```
myVec := Vector7D {1.0 BY 6};   (* set all entries *)
```

- Multidimensional arrays:

```
myMatrix := Matrix2x3 { {1.0 BY 3} BY 2 };
```

Record constructors

- We can also initialize records in the same way:

```
TYPE
    Student = RECORD
        name : String20;
        ID : CARDINAL;
    END;
    Class = ARRAY [0..29] OF Student;
thisStudent := Student { “Jane Doe”, 12345 };
wholeClass := Class { thisStudent BY 30 };
```

- Equivalent to:

```
thisStudent.name := “Jane Doe”;
thisStudent.ID := 12345;
```

Variant records

- Variant record types can adapt:

```
TYPE
    FruitType = (Apple, Pear, Tomato);
    Fruit = RECORD
        colour : Colour;
        CASE type : FruitType OF
            Apple :
                sweetness : REAL |
            Pear :
                crunchiness : REAL |
            Tomato :
                bobness : CARDINAL;
        END;
    END;
myFruit := Fruit { red, Apple, 0.5 };
```

Summary of today (11.4–11.9)

- Constructors: Type { list }
  - Set constructors
  - Array constructors
  - Record constructors
- Variant records

- Read on your own:
  - CASE statement
  - Pragmas
  - Tips for program efficiency

TODO items

- Lab #9 next week: 10.15 #(44 / 49)
- Reading: through §11.9 for Mon
- Midterm ch8–10 next Wed

- Get cracking on your paper!