§5.1-5.3: Enumerations, Arrays

devo

5 Oct 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

1) journals in folder
2) hw due 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

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. *)
```

Comparisons work for scalar types

- Scalar types include real types and all ordinal types
- Ordinal types include whole number types and all enumerations and subranges
- Examples:
  - IF (today >= Monday) AND (today <= Friday)
  - WHILE (ch >= 'A') AND (ch <= 'Z')

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)

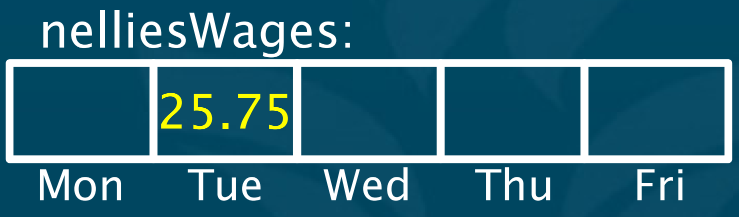
today

Wed

Arrays

- An array is a collection of objects with the same type that is indexed by an ordinal type
- Array types can be declared using TYPE:

```
TYPE
    CharArray = ARRAY [0 .. 20] OF CHAR;
    Weekdayname = [Mon .. Fri];
    WageArray = ARRAY WeekdayName OF REAL;
VAR
    myName, yourName : CharArray;
    nelliesWages : WageArray;
BEGIN
    myName [0] := 'S';
    nelliesWages [Tue] := 25.75;
```



Using arrays

- We can access individual entries in an array:
  - myName [1] := yourName [0];
- We cannot index an array out of bounds:
  - myName [2000] := 'a'; (* out of range *)
  - nelliesWages [Sat] := 10.0; (* out of range *)
- We can assign whole arrays of the same type:
  - myName := yourName;
- We can't do comparisons on whole arrays:
  - IF myName = yourName (* invalid *)
  - IF myName < yourName (* invalid *)

Anonymous array types

- We can declare a variable to be an array without explicitly declaring an array type:
  - VAR

    myWages : ARRAY Weekdayname OF REAL;
- This type is called an anonymous array type
- In M2, anonymous types are not compatible with named types (recall nelliesWages is a WageArray):
  - myWages := nelliesWages (* type mismatch *)
- Functions also may not use an anonymous type as a return type:
  - PROCEDURE GetWages() : WageArray; (* ok *)
  - PROCEDURE GetWages() : ARRAY WeekdayName of REAL; (* not *)

Strings

- In M2, strings are just arrays of CHARs!
  - TYPE

```
String = ARRAY [0 .. 10] OF CHAR;
LongString = ARRAY [0 .. 80] OF CHAR;
Paragraph = ARRAY [1 .. 10] OF LongString;
```

  - VAR

```
string1, string2 : String;
ch : CHAR;
para : Paragraph;
```

- Note that our String types have fixed length
- String and LongString are different types
  - Hence not assignment/expression compatible

Using strings (more detail in ch7)

- We can use arrays of CHAR wherever we can use literal strings:
  - string1 := “Hello!”;
  - string2 := string1;
  - WriteString (string1);
- CHARs can be assigned to strings:
  - string1 := ch;
- We can input strings from the user:
  - ReadString (string1);
- But be careful of exceeding the string length!
  - string1 := “Hello World!”;   (* too long! *)

Review of today (5.1-5.3)

- Enumeration types
  - Ordinal types
- Subrange types
  - Expression, assignment compatibility
- Array types
  - How to declare an array type
  - How to declare a variable of array type
  - How to use and access arrays
  - Strings

TODO items

- Midterm ch1-4: this Friday!
  - (same day as MATH123 calc midterm)
  - Review in-class tomorrow morning
- Lab4 next Tue/Wed: 5.11 #(26 or 28 or 32)
  - M-lab section can turn it in up to a week late
- Quiz ch5 postponed until Fri 14Oct
- Reading: through §5.5 for Wed 12Oct