

§3.4-3.8: Repetition (Loops)

devo

26 Sep 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- 1) *journals* in folder
- 2) *quiz* today

Announcements

- Clarification on Lab2 #3.45:
 - You may require the user to input in whatever format suits you
 - You should allow the user to choose an output format from the three formats shown

Review of 3.1-3.3

- Statement **sequences**; use of semicolon (;)
- Forms of the **IF** statement:
 - IF – THEN – END
 - IF – THEN – ELSE – END
 - IF – THEN – ELSIF – THEN – ELSE – END
- **Boolean** expressions:
 - =, <, >, <=, >=, <> (#)
 - AND (&), OR, NOT (~)
 - **Precedence**
 - **Shortcut** semantics

Quiz ch3 (# questions, 20 marks, 10 minutes)

- (8pts) **Evaluate** the following Boolean expressions, or if they give an error, indicate **why**:
 - $(3 + 5 < 9) \text{ AND } (14 \text{ MOD } 3 = 2)$
 - $7/3 = 2 \text{ OR } 5 > 3$
 - $(6 < 4) \& (2 / (4 - 4) = 0)$
 - $\sim 12 \# 4$
- (3pts) What is **wrong** with this loop? How would you **fix** it?

REPEAT

counter := 9;

statement sequence;

counter := counter - 1;

UNTIL counter < 0;

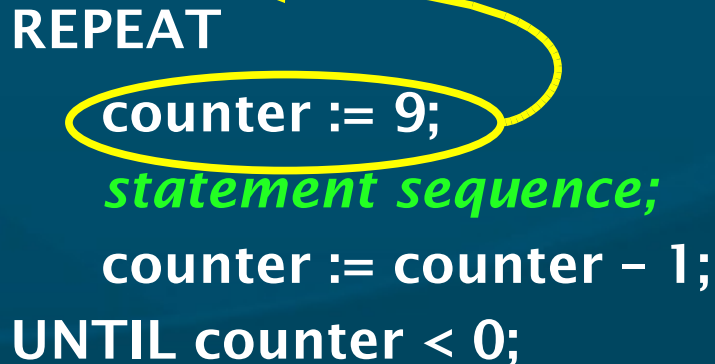
- (9pts) Write a **program** to convert inches to centimetres or vice versa, depending on user input.

Quiz ch3 answers (#1-2)

- **Evaluate** the following Boolean expressions, or if they give an error, indicate **why**: (2pts each)
 - $(3 + 5 < 9) \text{ AND } (14 \text{ MOD } 3 = 2)$ **TRUE**
 - $7/3 = 2 \text{ OR } 5 > 3$ **incorrect operands: 2 OR 5**
 - $(6 < 4) \ \& \ (2 / (4 - 4) = 0)$ **FALSE** (no divide-by-0 err)
 - $\sim 12 \ \# \ 4$ **incorrect operands: NOT 12**

- What is **wrong** with this loop? How would you **fix** it? (3pts)

```
REPEAT
  counter := 9;
  statement sequence;
  counter := counter - 1;
UNTIL counter < 0;
```



Quiz ch3 answers (#3, p.1/3)

- Write a **program** to convert inches to centimetres or vice versa, depending on user input.

```
MODULE InchToCm;
```

```
FROM STextIO IMPORT
```

```
    WriteString, WriteLn, SkipLine, ReadChar;
```

```
FROM SRealIO IMPORT
```

```
    ReadReal, WriteFixed;
```

```
CONST
```

```
    cmInAnInch = 2.54;           (* conversion factor *)
```

```
VAR
```

```
    inputLength : REAL;         (* input from user *)
```

```
    inputIsCm : CHAR;          (* kbd response from user *)
```

Quiz ch3 answers (#3, p.2/3)

BEGIN

```
WriteString ("This program converts between inches ");  
WriteString ("and centimetres.");  
WriteLn;  
WriteString ("Please enter a length (without the units): ");  
ReadReal (inputLength);  
SkipLine;  
  
WriteString ("By default, inches are assumed.");  
WriteLn;  
WriteString ("Type 'c' if this length is in cm instead: ");  
ReadChar (inputIsCm);  
SkipLine;
```

Quiz ch3 answers (#3, p.3/3)

```
WriteFixed (inputLength, 2, 0);
```

```
IF inputIsCm = 'c'
```

```
    THEN (* convert cm => in *)
```

```
        WriteString ("cm =");
```

```
        WriteFixed (inputLength / cmInAnInch, 2, 0);
```

```
        WriteString ("in");
```

```
    ELSE (* convert in => cm *)
```

```
        WriteString ("in =");
```

```
        WriteFixed (inputLength * cmInAnInch, 2, 0);
```

```
        WriteString ("cm");
```

```
    END;
```

```
WriteString(". Have a nice day!");
```

```
WriteLn;
```

```
END InchToCm.
```


What's on for today (3.4-3.8)

- **Loops**: WHILE, REPEAT
- **Sentinel** variables
- Loop **counters**
- Using mathematical **closed forms** instead of loops
- An example of problem-solving
 - **Stub** program
 - Using **ReadResult()** to test the previous Read operation

WHILE loops

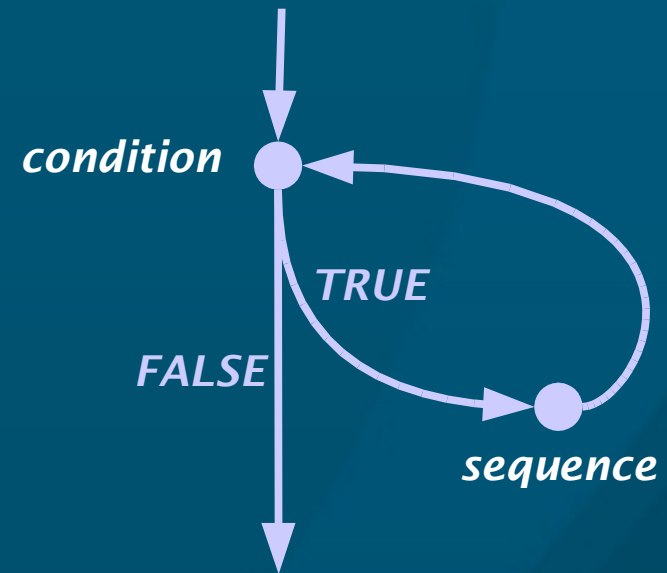
WHILE *condition*

DO

statement sequence

END;

- As with IF, *condition* is a **Boolean expression**:
 - It is evaluated **once** before entering the loop,
 - And **re-evaluated** each time through the loop:
 - Top-of-loop testing
- *Statement sequence* is run only if *condition* evaluates to TRUE

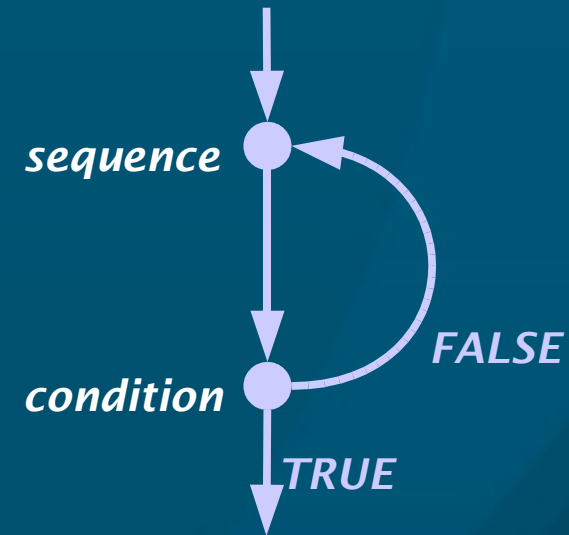


REPEAT loops

REPEAT

statement sequence

UNTIL *condition*;



- As with WHILE, *condition* is a **Boolean expression**:
 - It is evaluated each time through the loop, **after** the *statement sequence* is executed
 - Bottom-of-loop testing
- *Statement sequence* is run as long as *condition* evaluates to FALSE

Sentinel variables

- A **sentinel variable** controls whether a loop continues: the loop only exits when the sentinel variable has a certain value

REPEAT

WriteString (“Math quiz: 2+2=”);

ReadCard (answer);

SkipLine;

UNTIL **answer = 4;**

- Sentinel variable is **answer**
- Sentinel value is **4**

Counting loops

- A common form of loop uses a **counter**:

```
counter := 1;  
WHILE counter <= max  
  DO  
    sum := sum + counter;  
    counter := counter + 1;  
  END;
```

- What if we need to **prematurely** exit this loop?

```
WHILE counter <= max  
  DO  
    IF NeedToExitLoopEarly THEN  
      counter := max + 1;  
    END; ....
```

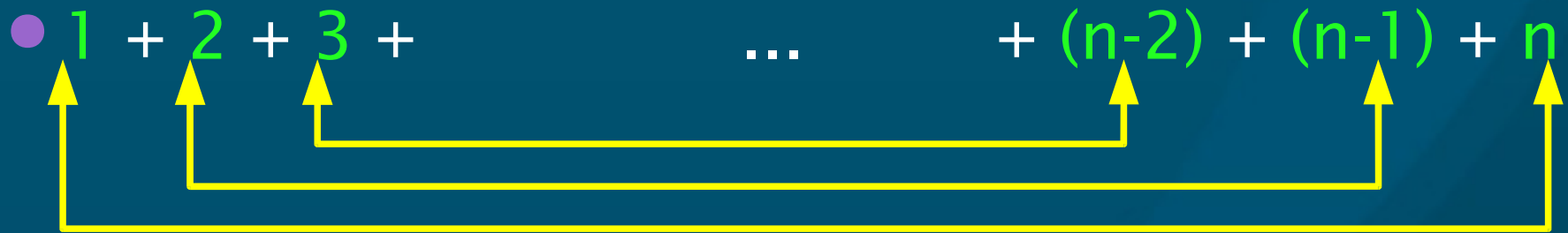
Closed forms instead of loops

- Sometimes with a bit of thought we can replace a loop with a single **mathematical equation**:
 - “Work smarter, not harder”
- Example: Add the first n integers >0

```
sum := 0;
counter := 1;
WHILE counter <= n
  DO
    sum := sum + counter;
    counter := counter + 1;
  END;
```

Closed form solution

- But observe the pattern:



- Each pair makes $n+1$; there are $n/2$ pairs:

- Closed form solution:

$$\text{sum} := n * (n+1) / 2;$$

- ◆ (If n is type CARDINAL, does the $/$ cause problems?)

A fun example: ROT13

- **Task:** Translate characters into **ROT13** one line at a time
 - **ROT13:**
 - ◆ Treat each **letter** A-Z as a **number** between 1-26,
 - ◆ Add **13** to the number and wrap-around if necessary
 - ◆ Convert back to a **letter**
 - ◆ Preserve **case**
 - ◆ Leave all non-letter characters alone
 - e.g., ROT13 ('a') = 'n', ROT13 ('P') = 'C',
ROT13 ('#') = '#'

ROT13: Problem restatement

- **Input:**
 - A sequence of **letters**, ending with a newline
- **Computation:**
 - Convert letter to **numerical** form
 - Add **13** and wrap-around if necessary
 - Convert back to **letter** form
- **Output:**
 - Print **ROT13**'d character to screen

ROT13: convert letters to numbers

- How do we convert from a letter character to a **numerical** code?
 - Try VAL (CARDINAL, char1): **testbed** program

```
ReadChar (char1);
WriteString ("The numerical ASCII code for ");
WriteChar (char1);
WriteString (" is");
WriteCard (VAL (CARDINAL, char1), 0);
WriteLn;
```
- ASCII codes: 'A' = **65**, 'Z' = **90**, 'a' = **97**, 'z' = **122**
- Convert back with VAL (CHAR, card1)

ROT13: Pseudocode

- Print **intro** to the user
- Repeat:
 - Read a character the user typed
 - Convert to **ASCII** numerical code
 - If character is an **uppercase** letter,
 - ◆ Add **13** to code
 - ◆ If code is now beyond 'Z', subtract 26 (**wrap-around**)
 - Else if character is a **lowercase** letter,
 - ◆ Add **13** to code
 - ◆ If code is now beyond 'z', subtract 26 (**wrap-around**)
 - Convert numerical code back to **character** and print
- **Until** current character is **newline**

How to test if upper/lower case?

- Our pseudocode involves a test if the character is an uppercase letter or lowercase letter
- How to do that?

```
ascii := VAL (CARDINAL, ch);  
IF (ascii >= VAL (CARDINAL, 'A')) AND  
   (ascii <= VAL (CARDINAL, 'Z'))  
THEN  
    (* uppercase *)
```

ROT13: Stub program pseudocode

- Repeat:
 - Read a character the user typed
 - Convert to ASCII numerical code
 - Convert back to character
 - Print ASCII code and converted character
- Until current character is **newline**

- This **stub** program allows us to test the char<->ASCII **conversion** process and the interactive **keyboard** reading
- Tackle the **ROT13** processing later

ROT13: Stub program code

```
MODULE Rot13Stub;
```

```
FROM STextIO IMPORT
```

```
    ReadChar, WriteChar;
```

```
FROM SWholeIO IMPORT
```

```
    WriteCard;
```

```
FROM SIOResult IMPORT          (* needed to test for newline *)
```

```
    ReadResult, ReadResults;
```

```
VAR
```

```
    ch : CHAR;                (* user input *)
```

```
    ascii : CARDINAL;        (* numerical code corresponding to ch *)
```

ROT13: Stub program code, p.2

```
BEGIN
```

```
  ReadChar (ch);
```

```
  WHILE ReadResult() <> endOfLine (* Read until end of line *)
```

```
    DO
```

```
      ascii := VAL (CARDINAL, ch);
```

```
      WriteCard (ascii, 0);
```

```
      WriteChar (VAL (CHAR, ascii));
```

```
      ReadChar (ch);
```

```
    END;
```

```
END Rot13Stub.
```

- Sample input: hiya<newline>
- Sample output: “ 104h 105i 121y 97a”

ROT13: Full program code

```
MODULE Rot13;
FROM STextIO IMPORT
    ReadChar, WriteChar, SkipLine;
FROM SIOResult IMPORT
    ReadResult, ReadResults;
CONST
    ASCIIA = VAL (CARDINAL, 'A');      (* ASCII code for 'A' *)
    ASCIIZ = VAL (CARDINAL, 'Z');
    ASCIIa = VAL (CARDINAL, 'a');
    ASCIIz = VAL (CARDINAL, 'z');
VAR
    ch : CHAR;                          (* user input *)
    ascii : CARDINAL;                   (* numerical code corresponding to ch *)
```


ROT13: Full program code, p.2

```
BEGIN
```

```
  ReadChar (ch);
```

```
  WHILE ReadResult() <> endOfLine    (* Read until end of line *)
```

```
    DO
```

```
      ascii := VAL (CARDINAL, ch);
```

```
      IF (ascii >= ASCIIA) AND (ascii <= ASCIIZ) (* uppercase *)
```

```
        THEN
```

```
          ascii := ascii + 13;
```

```
          IF (ascii > ASCIIZ)    (* wrap-around *)
```

```
            THEN
```

```
              ascii := ascii - 26;
```

```
            END;
```

ROT13: Full program code, p.3

```
ELSIF (ascii >= ASCIIa) AND (ascii <= ASCIIz) THEN
    ascii := ascii + 13;                (* lowercase *)
    IF (ascii > ASCIIz)                (* wrap-around *)
        THEN
            ascii := ascii - 26;
        END;
    END;
```

```
WriteChar (VAL (CHAR, ascii));
ReadChar (ch);
```

```
END;
```

```
END Rot13.
```

ROT13: Results and analysis

- Input: `hiya`
 - Output: `uvln`
- Input: `uvln`
 - Output: `hiya`
- Input: `Hello World! This is a longer example.`
 - Output: `Uryyb Jbeyq! Guvf vf n ybatre rknzcyr.`
- **Generalizations/**extensions?
 - Handle multiple lines one line at a time?
 - ◆ How to quit?

Review of today (3.4-3.8)

- Loops: WHILE, REPEAT
- Sentinel variables
- Loop counters
- Using mathematical closed forms instead of loops
- ROT13 example:
 - Stub program
 - Using ReadResult() to test the previous Read operation

TODO items

- **Lab2** due this MTW: §3.14 # (38 / 45)
 - Choose either #38 or #45
 - Short writeup okay
- **Homework:** §3.14 #17 (hand in on Fri)
- **Reading:** through §4.2 for Wed