

ENGG1003 - Friday Week 2

More Flow Control and Examples

Brenton Schulz

University of Newcastle

March 7, 2019

DO ... WHILE

- ▶ Same as WHILE except executes *at least once*
- ▶ The condition is tested at the end
- ▶ Loops repeats if condition is TRUE
- ▶ Syntax:

```
DO
    stuff
WHILE condition
```

FOR Loops

- ▶ A FOR loop executes a given number of times
- ▶ Used when the number of loop repeats is known *before* entering the loop
 - ▶ Repeat count could be “hard coded” as a number
 - ▶ Could also be a variable
- ▶ Can be easier to read than WHILE
- ▶ Example pseudocode syntax:

```
FOR x = 1 to 10  
    Do something ten times  
ENDFOR
```

- ▶ The *loop variable* is automatically incremented

BREAK Statements

- ▶ Sometimes you want to exit a loop *before* the condition is re-tested
- ▶ The flow-control mechanism for this is a BREAK statement
- ▶ If executed, the loop quits
- ▶ BREAKs typically go inside an IF to control their execution

Loop `continue` Statements

- ▶ A `continue` causes execution to jump back to the loop start
- ▶ The *condition* is tested before reentry

FOR Example 1

- ▶ Two equivalent ways to implement the $\cos()$ series from before are:

NB: $|tmp|$ means “absolute value of tmp”.

```
BEGIN
  INPUT x
  sum = 0
  FOR k = 0 to 10
    tmp =  $\frac{(-1)^k x^{2k}}{(2k)!}$ 
    sum = sum + tmp
    IF |tmp| < 1e-6
      BREAK
    ENDIF
  ENDWHILE
END
```

```
BEGIN
  INPUT x
  tmp = 1
  k = 0
  sum = 0
  WHILE (k < 10) AND (|tmp| > 1e-6)
    tmp =  $\frac{(-1)^k x^{2k}}{(2k)!}$ 
    sum = sum + tmp
    k = k + 1
  ENDWHILE
END
```

FOR Example 2 - Factorials

- ▶ Use FOR to count from 2 to our input number
- ▶ Keep a running product as we go

```
BEGIN
  INPUT x
  result = 1
  FOR k = 2 TO x
    result = result * k
  ENDFOR
END
```

- ▶ Is this algorithm robust? What happens if:
 - ▶ $x = -1$
 - ▶ $x = 1$
 - ▶ $x = 0$ (**NB:** $0! = 1$ because *maths*)

GOTO

- ▶ There exists a GOTO flow control mechanism
 - ▶ Sometimes also called a *branch*
- ▶ It “jumps” from one line to a different line
- ▶ It exists for a purpose
- ▶ That purpose does not (typically) exist when writing C code
 - ▶ C *supports* a `goto` statement
 - ▶ It results in “spaghetti code” which is hard to read
 - ▶ Don't use it in ENGG1003
- ▶ You *can* use GOTO in ELEC1710

Increment Example

```
1 #include <stdio.h>
2 int main() {
3     int x = 0;
4     int y = 0;
5     int z = 0;
6     y = ++x + 10;
7     printf("Pre-increment: %d\n", y);
8     y = z++ + 10;
9     printf("Post-increment: %d\n", y);
10    return 0;
11 }
```

Listing 1: increment.c

Pre/post-inc/decrements have many applications,
more details in coming weeks.

Binary Nomenclature

- ▶ The value range is a result of the underlying binary storage mechanism
- ▶ A single binary digit is called a *bit*
- ▶ There are 8 bits in a *byte*
- ▶ In programming we use the “power of two” definitions of kB, MB, etc:
 - ▶ 1 kilobyte is $2^{10} = 1024$ bytes
 - ▶ 1 Megabyte is $2^{20} = 1048576$ bytes
 - ▶ 1 Gigabyte is $2^{30} = 1073741824$ bytes
 - ▶ (Advanced) These numbers look better in hex: 0x3FF, 0xFFFFF, etc.

Binary Nomenclature

- ▶ Observe that kilobyte, Megabyte, Gigabyte, etc use scientific prefixes
- ▶ These *normally* mean a power of 10:
 - ▶ kilo- = 10^3
 - ▶ Mega- = 10^6
 - ▶ Giga- = 10^9
 - ▶ ...etc (see the inside cover of a physics text)
- ▶ Computer science stole these terms and re-defined them

Binary Nomenclature

- ▶ This has made some people *illogically angry*
- ▶ Instead, we can use a more modern standard:
 - ▶ 2^{10} bytes = 1 kibiByte (KiB)
 - ▶ 2^{20} bytes = 1 Mebibyte (MiB)
 - ▶ 2^{30} bytes = 1 Gibibyte (GiB)
 - ▶ ...etc
- ▶ Generally speaking, KB (etc) implies:
 - ▶ powers of two to *engineers*
 - ▶ powers of ten to *marketing*
 - ▶ The number is smaller
 - ▶ Hard drive manufacturers, ISPs, etc like this

Unambiguous Integer Data Types

- ▶ Because the standard `int` and `long` data types don't have fixed size unambiguous types exist
- ▶ Under OnlineGDB (ie: Linux with `gcc`) these are defined in `stdint.h` (`#include` it)
- ▶ You will see them used commonly in embedded systems programming (eg: Arduino code)
- ▶ The types are:
 - ▶ `int8_t`
 - ▶ `uint8_t`
 - ▶ `int16_t`
 - ▶ `...etc`

Code Blocks in C

TODO

Variable Scope

TODO

#define Constants

TODO

for (; ;) Loops

TODO