Lecture 2

Flow-Charts

Tutorial Introduction
Flowcharts

• See handout
Handout Exercise #1

• Draw a flowchart to find the sum of first 50 natural numbers.

```
START
SUM = 0
N = 0
N = N + 1
SUM = SUM + N
N=50?
Yes
PRINT SUM
END
No
```

ENGE 220 – Digital Logic Design
Handout Exercise #2

• Draw a flowchart to find the largest of three numbers A, B, and C.

START

READ A B C

IS B>C ?

No

PRINT B

Yes

PRINT C

IS A>B ?

No

PRINT C

Yes

PRINT C

IS A>C ?

No

PRINT C

Yes

PRINT A

END
Handout Exercise #3

- Draw a flowchart for computing factorial N (N!)
  - Where N! = 1 × 2 × 3 × …… N.
Microsoft Visio

• More on this in the lab – we will install it and attempt to reproduce a flowchart in Visio.
• hello_world.c
  – C Programs consist of **functions** and **variables**
  – **Functions** contain **statements**
  – **main** is a special **function**.
  – **function** calls can contain **arguments**
  – a **string** is a sequence of characters
    • “hello, world\n” is a **character string** or **string constant**
  – There are special “escape” characters
    • \n represents a **newline** character (return)
    • \t is the tab
    • \b is a backspace
    • \" for the double quote
    • \\\n      is the backslash itself
    • \g (my favorite) is the system bell – I don’t know if it works in cygwin
• hello_world_1.c Produces identical output.
Handout #5 - Variable Expressions

• fahr_cells.c
  – Uses the formula C = (5/9)(F-32) to convert Fahrenheit temperatures to their Celsius equivalents.
  – *Comments* are of two types
    • /* .... */ a multiline comment
    • // a single line comment (this does not work in some older compilers)
  – Variables should be declared before being used. If they are not declared, they will default to an int datatype. It is not advisable to use default datatyping in good code.
  – A *declaration* consists of the datatype followed by the variable name
    • int joe;
Data Types

- datatype.c
- There are four basic datatypes in C
  - char, int, float, double
- There are some qualifiers
  - short, long, long long, unsigned
- And a special one
  - void
- Char is one byte
- Int can be two or four bytes depending on the system
- short int are generally two bytes
- long ints are generally four bytes
- long long ints are generally eight bytes
- floats are four bytes
- doubles are eight bytes
  - floats and doubles are defined by an IEEE specification.
  - Do a google search on IEEE floating point to find a good definition.
  - More on these later.
#include <stdio.h>

main ()
{
	printf("Size of char is %d\n", sizeof(char));
	printf("Size of short is %d\n", sizeof(short));
	printf("Size of int is %d\n", sizeof(int));
	printf("Size of long is %d\n", sizeof(long));
	printf("Size of long long is %d\n", sizeof(long long));
	printf("Size of float is %d\n", sizeof(float));
	printf("Size of double is %d\n", sizeof(double));
}
Loops

• C provides several looping constructs
  – while
  – for
  – do while

• The looping loop on a statement
  – statements can be one line
  – or multiple statements within a block
    • {.... }
Style

• Recommended practice is one statement per line
• Braces can be anywhere on a line, but folks get pesky.
  – Some insist on putting the opening brace under the statement
    while (c > 9)
    {
      \em statement;
      \em statement;
      \em statement;
    }
  – I tend to put the opening brace at the then of the statement
    while (c > 9) {
      \em statement;
      \em statement;
    }
Braces

• I use braces for every loop to assist in future debugging.
  
  ```c
  while (c > 9) {
    statement;
  }
  ```

• Some (the book) will omit braces for single statement loops.
  
  ```c
  while (c > 9)
    statement;
  
  or
  while (c > 9) statement;
  ```
Arithmetic

- Integer division *truncates* \( \frac{5}{9} = 0 \)
- This can be dealt with in different manners
  - use double division \( (5.0/9.0) \) and then truncate
  - `fahr_cels_1.c`
  - more on this later
Arithmetic – what are the values of the following statements?
- $7/4 = 1$
- $14/3 = 4$
- $14.0/3 = 4.67$
- $14/3.0 = 4.67$
- $14.0/3.0 = 4.67$
- `int i = 7/4; 1`
- `int i = 14/3; 4`
- `int i = 14.0/3; 4`
- `float f = 7/4; 1 (the right hand side is still integer)`
- `float f = 14.0/3; 4.67`
- `float f = 14.0 * (3/2); 14 (3/2 is integer and truncates to 1)`
### printf

- printf is not part of the C language, it is a basic function provided in the standard C library (stdlib).
- As Howard used to say …
  - type man printf
• fahr_cels_2.c
• for (initial statement; condition; concluding statement)
  – A for loop will execute the initial statement upon entry (ONCE!)
  – It will then check the condition. If the condition evaluates to TRUE (1), then it will execute the block.
  – Upon completion of the block, it will execute the concluding statement
  – It then returns to the evaluation of the condition and repeats until the condition is false
  – When a false condition is met, the program continues with the statement after the end of the for loop block

  for (fahr = 0; fahr <= 300; fahr = fahr + 20) {
    statement;
  }
While or For?

• Can do equivalent with either one.
  – For is typically used when the number of iterations is known before running the code.
Symbolic Constants

- fahr_cels_3.c
- The C preprocessor is so integrated with C and is considered basically a part of the language.
- C preprocessor directives begin with #
  - `#include`
  - `#define`
- `#define` directives do a text search/replace of the define token with the value
- `#define LOWER 0`
  - This directive finds all instances of LOWER in the body of the file being compiled and replaces the word LOWER with 0.
    - The preprocessor is case sensitive and will only replace complete word matches – LOWER is different than LOWERING
    - To avoid a replacement, the token can be placed in quotes. This also prevents any string contents from being processed – including those in printf statements