GNG1106
Fundamentals of Engineering Computation

Instructor: **Hitham Jleed**

University of Ottawa

Fall 2023 ~

Outline

First Program

```
#include <stdio.h>
/* This is our first program */
int main()
{
            printf("Hello World!\n");
            return 0;
}
```

Comments

- A block of code that begins with `/*` and ends with `*/` is a comment block.
- Anything written in a comment block is ignored by the compiler and does not generate anything in the executable code.
- A comment block may span multiple lines, if needed.
- Comments may also be written using "`//`", placed anywhere on a line
  - Anything written to its right in the same line is treated as a comment and ignored by the compiler.
  - This is actually C++ syntax.
- Comments are written for humans, not for computer or compilers.
- Comments are meant to serve as the internal documentation of the program.

Preprocessing Directives

- All lines that begin with # are preprocessing directives
- These commands are processed before compiling the C code.
  - e.g., `#include<stdio.h>`
  - In this case, the directive tells the compiler to include a file named `stdio.h` into the code before compilation. This file contains standard I/O (Input/Output) function headers and other definitions related to standard I/O.
- There is no `;` after a preprocessor directive.

The `main` Function

- The `main` function is required for every program.
- Usually `int main()` (more recommended) or `void main()` is used as the function header.
- The code enclosed in the pair $\{\cdots\}$ of braces is the instruction block.
- A program in C begins by executing line by line the instruction block.
  - Each instruction in the block must end with semi-colon "`;`"
  - When the header of the `main()` function is "`int main()`", the ending instruction in the block should be "`return 0;`".
  - However this command should not be included if the header is "`void main()`".
- C is case-sensitive: the compiler treats `ABC` and `abc` as different things.

printf

- The line
  `printf("Hello World\n");`
  calls a standard I/O function `printf()`, which prints out the desired message to the screen.
  - standard output ("`stdout`") = screen
  - standard input ("`stdin`") = keyboard
- The message to be written is included in a pair of double quotes.
- The message together with the quotation marks is the input to the function `printf()`
- "`\n`" is called an "escape sequence"; it positions the cursor at the beginning of the next line after the message is printed.

Outline

- The basic object in any computer language is a variable.
- A variable stores a value.
- A variable is essentially a group of memory units (ie, byte).
- The type of a variable implies:
  - how much memory it takes to store a value of this type, (e.g. it takes less memory to store an integer value than a real value), and
  - how the value of the variable is encoded into bytes (and hence how the bytes should be interpreted into a value).
- Every variable has an address.

Highlight

The memory can be thought logically as an array of bytes, each having an address. If we think of each byte as a house, the memory is a street lined up with houses, and street number of a house is the address of the byte.

- Before a variable is used, it must be declared, or "created".
- A variable is declared using syntax like this:
  `int x;`
  - This declaration means: "create" an `int`-typed variable whose name is "`x`". There are other types, e.g., `float, double`, etc.
    - `int` stands for "integer"
  - More accurately, with this command, the program requests the operating system (OS) to allocate a block of 4 bytes (noting that int-typed values are represented by 4 bytes) and reserve it for storing the value of `x`.
  - If this command is successfully executed, the OS will also remember the address of the first byte in the reserved block (this address is also referred to as the "address of `x`") for future access of the variable `x`.
  - For example, if later the program wants to get the value of `x`, the OS will, based on the address of `x`, find the block of 4 bytes, and convert it to a number.

- Variable assignment is done using syntax like this:
  `x=3;`
  where we have assumed that the variable `x` has been declared to have int type.
  - The "=" sign in this line does not mean "equals" or "is equal to".
  - It means: assign value 3 to variable `x`.
  - The execution of this command involves
    1. converting the value 3 to an 4-byte representation (assuming `x` has been declared as `int`) , and
    2. storing the 4 bytes to the block of memory reserved for `x`.
- If that memory reserved for x previously has stored some value, the value will be replaced by the (4-byte representation of the) new value, 3.

Variable Name

- A variable name is not allowed to begin with a number.
- Reserved words are not allowed to be used as variable names.
  - e.g., `if`, `else`, `while` ... cannot be used as variable names.
- Certain characters reserved as operators are not allowed in a variable name.
  - e.g., `+`, `-`, ... are reserved to represent addition and subtraction etc; they cannot be used as variable names.
- Declare two variables having the same type: e.g. `float a, b;`
- Declare a variable and initialize its value at the same time: e.g., `int y=5;`
- Use descriptive variable names! e.g, use `student_1` instead of `s1`
  - It will be much easier for YOU to remember what it means
  - It will be much easier for OTHERS to understand your code

Outline

- ```
  printf("%d", 100);
  printf("%f", 101.3);
  printf("%d", integer_1);
  printf("%f", real_1);
  ```
- The `%d` is the conversion specifier used to print out an integer and the `%f` is the specifier used to print out a real number.
- The conversion specifier must match the data type.
- A conversion specifier always begins with a `%`
- The specifier must be enclosed in a pair of double quotes
- A conversion specifier is like a "space holder" for the object, in the to-be-printed message.
  ```

- More than one "object" can be printed using a single `printf`
  - For example:
    `printf("%d pounds and %d feet tall", weight, height);`
  - If the values of variables `weight` and `height` are 140 and 7, the line prints:
    `140 pounds and 7 feet`

  The arguments to printf are separated by a comma
- To print % as part of the message, use two percent symbols %%.

Outline

- Data can be entered from the keyboard (a.k.a, standard input, or `stdin`) using the standard C function `scanf`, e.g.

```
scanf("%d", &integer_1);
scanf("%f", &real_1);
scanf("%d%f'', &integer_1, &real_1);
```

- `scanf` requires a conversion specifier corresponding to the type of data to be read.
- An ampersand `&` must be placed before the variable into which the data will be read.

Coding Demonstration

https://github.com/hjleed/GNG1106_Archive/tree/main/week2_codes