CSCI 2132 – Software Development
Introduction to C

January 30, 2017

Dalhousie University

Meng He

Background

- Originally invented as a language for writing operating systems and other system software
  - Denis Ritchie
- C optimizes for machine efficiency at the expenses of increased implementation and debugging time
- A central difficulty in C programming: programmers must do their own memory management
- C assumes that you know what you are doing

Writing a Simple Program

hello.c: first C program by K&R

```
#include <stdio.h>

int main(void) {
  printf("hello, world\n");
  return 0;
}
```

Includes information about C’s standard I/O library

No arguments

The “main” program

A function from the standard I/O library to produce formatted output

Returns 0 as its exit code

From Text to Executable

- Three Steps
  - Preprocessing (by a preprocessor): modifies the program by obeying directives
  - Compiling (by a compiler): translates the modified program into object code (machine instructions)
  - Linking (by a linker): combines the object code with additional code needed to yield a complete executable program
- gcc automatically executes three steps
- Shell scripts, python, perl – interpretation: slower, easier to modify

The General Form of a Simple Program

```
directives

int main(void) {
   statements
}
```

Directives

- Commands intended for preprocessors
- Always begin with a #
- One line long
- No semicolon at the end
- #include <stdio.h>: the content of stdio.h to be included into the program before it is compiled
- stdio.h: an (actual) header file
  - /usr/include/stdio.h

Functions

- Building blocks from which C programs are constructed
- A function is essentially a series of statements that have been grouped together and given a name
- Library functions: functions provided as part of the C implementation, e.g. printf
- Main function: the function that is called automatically when the program is executed
- int main(void) means that main returns an integer value, and does not take any command-line arguments

Statement

- A command to be executed when the program runs
- Must end with a semicolon

Printing Strings

- printf can display a string literal – a series of characters enclosed in " "
- Newline character: \n
- Examples
  - printf("hello, ");
    printf("world\n");   =   printf("hello, world\n");
  - printf("hello, \nworld\n");   This prints:

```
hello,
world
```

Comments

- /* comments (one or more lines) */
- Examples

  -
    ```
    /* Name: hello.c
     * Purpose: prints hello, world
     * Authors: K&R
     */
    ```
- C99: // comments (to the end of the line)

Variables

- Types
  - Each variable must have a type
- Examples
  - int: integers
  - float: floating-point numbers

Declarations

- Variables must be declared before use
- Syntax: type name;
- Examples
  - int height;
  - float profit;
- In C89 or earlier, in any function, all the declarations must precede statements
- No such restrictions in C99
- Some operators: =, +, -, *, /, %

Printing Variables

- Printing an integer
  - printf("Height: %d\n", height);

- Printing a floating-point number
  - printf("Profit: %f\n", profit);
    - 6 digits after decimal point
  - printf("Profit: %.2f\n", profit);
    - 2 digits after decimal point

Initialization

- Most variable do not have default values when declared

- Declaration & initialize
  - int height = 8;
  - float profit = 1030.56f;

Reading Input: scanf

- Reading an int value
  - scanf("%d", &height);
- Reading a float value
  - scanf("%f", &profit);

Defining Names for Constants

- Macro definition (directives)
  - `#define PI 3.14159f`
- The preprocessor modifies the program by replacing each macro by the value it represents
- A macro definition does not define a variable
- The value of a macro can be an expression
  - `#define RECIPROCAL_OF_PI (1.0f / 3.14159f)`
- If the expression contains operators, place it in ( )
- Reason: `float pi = 1.0f / RECIPROCAL_OF_PI;`
  `float pi = 1.0f / (1.0f / 3.14159f);`
- Convention: uppercase in macro names

Identifiers

- Names for variables, functions, macros, etc.
- May contain letters, digits and underscores
- Must begin with a letter or underscore

Example

- Suppose that you are a cashier working in a retail store
- When a customer pays a certain amount for a product of a certain price, before HST,
- You want to calculate the balance to be returned to the customer
- Design
  - Read price, payment, calculate, print the result
  - HST can be defined as a macro

Code

```c
#include <stdio.h>

#define HST 0.15f

int main(void) {
  float price, payment, balance;

  printf("Enter price: ");
  scanf("%f", &price);

  printf("Enter payment: ");
  scanf("%f", &payment);

  balance = payment - price * (1 + HST);
  printf("Balance to be returned to customer: %.2f\n", balance);

  return 0;
}
```