C: compilation; types, operators; output; control structures

COSC 208, Introduction to Computer Systems, 2022-01-25

Announcements

- First lab tomorrow and Thursday

Outline

- Warm-up
- Hello, C: types; operators
- output
- Control structures

Warm-up

- Q1: *What are the main components of a computer system?*
  - Central processing unit (CPU)
  - Random access memory (RAM)
  - Secondary storage devices — e.g., hard disk drive, solid state drive
  - Input/output (I/O) ports
  - Operating system (OS)
- *What is the role of each of the components?*
  - CPU — executes assembly instructions
  - RAM — stores program data (code, variables, etc.) at runtime
  - Secondary storage — stores program data (code, documents, etc.) persistently
  - I/O ports — receives user input (e.g., keyboard, mouse, audio); provides output to users (e.g., text, graphics, audio)
  - OS — manages hardware resources

Hello, C

- Write a C program that prints "Hello, C!"

```c
#include <stdio.h>
// visual studio code demo, including errors ;)
int main() {
    printf("Hello, C!\n");
    return EXIT_SUCCESS:
}
```

- *How do you compile the program?*

```
$ clang -g -Wall -o hello hello.c
```

- *How do you run the program?*

```
$ ./hello
```

Primitive types

- Most common types
  - `char` — 1 byte (8-bits); integers -128...127; integers 0...127 correspond to ASCII characters
  - `int` — usually 4 bytes (32-bits); integers -2 billion...2 billion
  - `long` — usually 8 bytes (64-bits); integers -9 quintillion...9 quintillion
- Add `unsigned` in front of these types for a range from 0 to 255, 4 billion, or 18 quintillion, respectively

Operators

- *How would I increment the number stored in a variable `x` by `1`?*
  - `x = x + 1`
  - `x += 1`
  - `x++`
  - `++x`
- Q2: *What is the output of this program?*

```c
int main() {
    int x = 1;
    int y = 2;
    x = x+5;
    printf("%d ", x);
    x = y*2;
    printf("%d ", x);
    x *= 5;
    printf("%d ", x);
    printf("%d ", x--);
    printf("%d ", x);
    printf("%d ", --x);
    printf("%d", x);
}
```

```
6 4 20 20 19 18 18
```

- Q3: *What is the output of this program?*

```c
int main() {
    int x = 5;
    int y = x/2;
    int z = x%2;
    printf("%d %d\n", y, z);
}
```

```
2 1
```

- Q4: *What is the output of this program?*

```c
int main() {
    int x = 5;
    char y = 'A';
    y = y + 5;
    printf("%c %d\n", y, y);
}
```

```
F 70
```

Output

- What is the syntax for `printf`?
    - `printf(FORMAT_STRING, VALUES, ...);`
    - `FORMAT_STRING` is a string constant (sequence of characters surrounded by double quotes) that may optionally include format specifiers
    - Format specifiers define how to convert a value to a string
        - `%d` decimal (i.e., base 10) number
        - `%c` character
        - `%x` hexadecimal (i.e., base 16) number
        - `%s` string — more on this next week
    - After the format string, include a value for each format specifier
        - A compile error will occur if the number of format specifiers does not match the number of values
        - A compile warning will occur if the value type does not match the format specifier
- Q5: *Assume the variables `year`, `month` and `day` contain the parts of a date. Use `printf` to output the data (e.g., `2022-1-26`)*

```c
printf("%d-%d-%d\n", year, month, day);
```

- Q6: *Assume the variables `length` and `width` contain the dimensions of a sports field/court. Use `printf` to output the dimensions (`94ft x 50ft`)*

```c
printf("%dft x %dft\n", length, width);
```

- Q7: *Assume the variables `first` and `last` contain a patient's first and last initial, and the variables `systolic` and `diastolic` contain the patient's blood pressure readings. Use `printf` to output the patient's initials and blood pressure (e.g., `A.G. 115/70`)*

```c
printf("%c.%c. %d/%d\n", first, last, systolic, diastolic);
```

Q8: *Write a program that prints the number of days, hours, and minutes in a week.*

```c
#include <stdio.h>
int main() {
    int days = 7;
    int hours = days * 24;
    int minutes = hours * 60;
    printf("1 week = %d days = %d hours = %d minutes\n", days, hours, minutes);
}
```

Control structures

- Control structures in C have the same syntax as control structures in Java.
    - Conditionals

      ```c
      if (/* BOOLEAN EXPRESSION */) { // Exactly one
          /* STATEMENTS */
      }
      else if (/* BOOLEAN EXPRESSION */) { // Zero or more
          /* STATEMENTS */
      }
      else { // Zero or one
          /* STATEMENTS */
      }
      ```

    - For loops

      ```c
      for (/* INTITIALIZER */; /* CONTINUATION CONDITION */; /* INCREMENT */) {
          /* STATEMENTS */
      }
      ```

    - While loops

      ```c
      while (/* CONTINUATION CONDITION */) {
          /* STATEMENTS */
      }
      ```

- Curly braces are optional if the body of a conditional, for loop, or while loop is only one line
    - But, you should **always** include them to make the code easier to read and reduce the likelihood of future errors

Practice

- Q9: *Write a program that flips a coin: call `random()` to generate a random number, and print `heads` if the number is even and `tails` if the number is odd.*

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    int num = random();
    if (num % 2 == 0) {
        printf("heads\n");
    } else {
        printf("tails\n");
    }
}
```

- Q10: *Write a program that prints all even numbers from 1 to 100 using a for loop.*

```c
#include <stdio.h>
int main() {
    for (int i = 2; i <= 100; i+=2) {
        printf("%d\n", i);
    }
}
```

~ OR ~

```c
#include <stdio.h>
int main() {
    for (int i = 0; i <= 100; i++) {
        if (0 == i % 2) {
            printf("%d\n", i);
        }
    }
}
```

- Q11: *Write the same program using a while loop.*

```c
#include <stdio.h>
int main() {
    int i = 2;
    while (i <= 100) {
        printf("%d\n", i);
        i+=2;
    }
}
```

Extra practice

- QA: *Write a program that prints out the powers of 2 from 2 through 2048.*

```c
#include <stdio.h>
#define MAX 2048
int main() {
    int i = 2;
    while (i <= MAX) {
        printf("%d ", i);
        i *= 2;
    }
    printf("\n");
}
```

- QB: *Write a program that prints all numbers from 1 to 100, except:*
  - *If the number is divisible by 3 then print `Three`*
  - *If the number is divisible by 5 then print `Five`*
  - *If the number is divisible by 3 and 5, print `Both`*

```c
#include <stdio.h>
int main() {
    for (int i = 1; i <= 100; i++) {
        if (i % 3 == 0 && i % 5 == 0) {
            printf("Both\n");
        }
        else if (i % 5 == 0) {
            printf("Five\n");
        }
        else if (i % 3 == 0) {
            printf("Three\n");
        }
        else {
            printf("%d\n", i);
        }
    }
}
```

- QC: *Write a program that prints every letter of the alphabet in upper and lower case: `AaBbCcDd...YyZz`*

```c
#include <stdio.h>
int main() {
    for (char upper='A'; upper <= 'Z'; upper++) {
        char lower = upper - 'A' + 'a';
        printf("%c%c", upper, lower);
    }
    printf("\n");
}
```