COMP1511: Programming Fundamentals

L. Cheung

February 18, 2026

Contents

1 Lecture 1

1.1 What is COMP1511

- Introduction to programming
- Learning how to write precise instructions to operate computers
- Assumption of no programming knowledge

1.2 Introduction to C

```c
#include <stdio.h>

int main(void) {
        printg("Hello world!"\n);
        return 0;
}
```

2 Lecture 2

2.1 How does a computer remember things?

- Computer memory is a big pile of on-off switches, called bits (a choice between a 1 or a 0)
- These bits are usually bunched into sets of 8, a byte
- When executing code, the CPU processes the instructions and performs basic arithmetic, but the RAM keeps track of all the data needed in those instructions and operations

2.2 Variables

- A variable is a certain allocation of bits that can be used to store information
  - int → integer, a whole number
    - A whole number, no fractions or decimals
    - Most commonly uses 32 bits (4 bytes)
    - Exactly 2^{32} different values
    - Finite range
  - char → a single character
    - The character holds an ASCII value, allowing characters to be read as integers
    - Lowercase letters are only 32 numbers away from their uppercase variant (flipping one bit)
    - Enclosed using single apostrophes
  - double → floating point number
    - A double-sized floating point number (64 bits, hence double the size of integers)
    - Floating point means the point can be anywhere in the number
- Names are a description of what the variable is
- C is case sensitive, "ansWer" ≠ "answer"
- C also reserves some words (eg. "return", "int", "double")
- Variables are printed using a format specifier (eg. %d formats the variable in base 10, %lf formats it in a double, %c formats it as a character)

- When printing variables, they appear in the order of the specifiers given
  - eg. `printf("Age: &d\n Name: %d\n", name, age)`

```c
#include <stdio.h>

int main(void) {
        double grade = 99.9;
        int age = 18;                          // dcc throws an error for unused variables
        char first_initial = "H";
        grade = 97.5;
        age = 67;
        printf("%d\n", age);        // "%d" formats the variable as a decimal
}
```

2.3 Input

- `scanf` can be used to get an input

```c
#include <stdio.h>

int main(void) {
        int age;
        printf("Enter your age: ")
        scanf("&d", &age);
        printf("Your age is %d!\n", age);
        return 0;
        }
```

- The `&` symbol tells `scanf` the address of the variable in memory and where to place the given value
- Inserting a space before the specifier in `ells i` to ignore all preceding whitespace

2.4 Constants

- Constants are usually defined at the start of the script using `#define CONST VALUE`

2.5 Maths

- Very familiar functions
  - Adding +
  - Subtracting -
  - Multiplication *
  - Division /

```c
#include <stdio.h>

int main(void) {
        int age = 12;
        age = age + 15 * 3;
        printf("\%d", age);
        return 0;
}
```

- BODMAS applies to maths in C

- Math can be done to characters since they are just integers
- Adding two large integers may roll over the maximum value and produce a very small or negative number (dcc will throw warning if this occurs)
- There is no infinite precision when encoding a number (eg, a third cannot be represented in binary)
- C will maintain variable types when doing arithmetic
- Integers will drop whatever fraction exists, ie. rounding down
- % is called the modulus and will provide the remainder from the division between two integers, eg. `5 % 3 = 2` since $\frac{5}{3} = 1$ rem 2