COMP1511: Programming Fundamentals

L. Cheung

February 23, 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) {
        printf("Hello world!"\n);
        return 0;
}

// Output: Hello world!
```

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
}

// Output: 67
```

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;
        }

// Output: Entered number
```

- 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\n", age);
        return 0;
}

// Output: 57
```

- 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

```c
#include <stdio.h>

int main(void) {
        int number = 15;
        int new_number = number % 4;
        printf("%d\n", new_number);
        return 0;
}

// Output: 3
```

3 Lecture 3

3.1 If Statements

- Using if statements, a program can branch between sets of instructions depending on a condition
- Can be used where a decision problem is a question with a yes/no answer

```c
if (condition) {
        do something;
        do something else;
}
```

Example.

```c
#include <stdio.h>

int main(void) {
        int number;
        printf("what is your favourite number: ");
        scanf("%d", &number);
        if (number == 15) {
                printf("That's Henry's birthday!\n")
        }
        return 0;
}
```

- If statements can be chained using `else if` statements

Example.

```c
#include <stdio.h>
if (condition_one) {
        do something
} else if (condition_two) {
        do different_something
} else {
        do another_different_something
}
```

3.2 Operators

3.2.1 Relational Operators

- Relational operators work with pairs of numbers:
  - < less than
  - > greater than
  - <= less than or equal to
  - >= greater than or equal to
  - == equals
  - != not equal to
- All these result in 0 if false and 1 if true

3.2.2 Logical Operators

- Between two expressions:
  - && AND: if both expressions are true then the condition is true
  - || OR: if an of the two expressions are true the the condition is true
- In front of an expression:
  - ! NOT: reverse the expression

Important!!!

Any value that is NOT 0 is TRUE

Example.

```c
#include <stdio.h>

int main(void) {
        int order = 66;
        if (order > 60 && order % 2 == 1) {
                printf("The Jedi are safe\n")
        }
        return 0;
}
```

- Brackets can be used to group logic statements to apply an order of operations
  - Eg. `if ((condition_one && condition_two) || condition_three)`

3.2.3 Example Problem

We have decided to run a competition to see how many free energy drinks were given out at O- Week. Students that guess the right number of free energy drinks win! You get told whether your guess was less than, more than or the winning guess :) Extend the problem - if you are within 5 of the correct number, you win the guessing game.

```c
#include <stdio.h>

int main(void) {
        int guess;
        printf("Guess how many energy drinks were given out at O-Week: ");
        scanf("%d", &guess);
        int answer = 94;
        if (guess == answer) {
                printf("Congratulations!\n");
        } else if (answer - guess <= 5 && answer - guess >= -5) {
                printf("So close, but no cigar\n");
        } else if (guess < answer) {
                printf("Too small!\n");
        } else {
                printf("That guess is too big!\n");
        }
}
```

3.3 Looping

- C is executed line by line starting from the main function after any
- If statements allow different sections of code to be run, however while loops allow us to repeat sections of code
- `while()` loops can be commonly controlled in three ways:

- Count loops
  - Repetition a set number of times

```c
#include <stdio.h>

int main(void) {
        int i = 0;
        while (i < 3) {
                printf("Yippee!\n");
                i++;                // Increments int i by 1
        }
}
```

- Sentinel loops
  - Repetition until a conditional is met
  - The variable that defines whether or not the loop runs is called the sentinel
  - The "termination condition" can be checked in the while expression

```c
#include <stdio.h>

int main(void) {
        int is_correct = 0;
        while (is_correct == 0) {
                do something
        }
        return 0;
}
```

- Conditional loops
  - Can also use a condition to decide to exit a loop at any time

3.3.1 Example Problem (continued)

```c
#include <stdio.h>

int main(void) {
        int guess;
        printf("Guess how many energy drinks were given out at O-Week: ");
        scanf("%d", &guess);
        int answer = 94;
        int is_correct = 0;
        while (is_correct == 0) {
                if (guess == answer) {
                        printf("Congratulations!\n");
                        is_correct = 1;
                } else if (answer - guess <= 5 && answer - guess >= -5) {
                        printf("So close, but no cigar\n");
                } else if (guess < answer) {
                        printf("Too small!\n");
                } else {
                        printf("That guess is too big!\n");
                }
```

```
            }
        }
```