Unit 6: The CS1010 I/O Library

interacting with the user

Learning Objectives

After this unit, students should: - understand the notion of standard input (`stdin`) and output (`stdout`) channels in Unix; - be able to use the various print and read functions available in `libcs1010` to perform I/O. - be aware that there exist pitfalls in using `printf` and `scanf` functions provided by C and they should only use these functions if they know what they are doing.

Interacting with Users

Our first C program that computes the hypotenuse doesn't do much -- it simply computes $\sqrt{3^2 + 4^2}$. The value to be computed is hard-coded, and the result computed is not displayed.

To make this program more general and useful, we need to do two things. First, we need to compute the hypotenuse for any length of the base and the height of a right-angled triangle. We cannot hard-code the length in the program. We should read these values from the users. Second, we need to output the result of the computation to the users. In other words, to make the program more general and useful, we need to add input and output, or I/O, functions.

Standard Input and Standard Output

Before we talk about how to read the input values and display the output values, we have to first talk about where input comes from and

where the output goes.

In Unix-flavored operating systems, the input is read from an abstract channel called the *standard input* or `stdin` for short, and an output is sent to an abstract channel called the *standard output* or `stdout` for short.

The fact that these channels are abstract is a powerful concept -- when we write our code, we do not have to worry about where the inputs come from and where the outputs go to. It will depend on how the user runs our program. Thus, it allows the users of our program the flexibility to control where the data comes from or goes to.

For instance, the standard input, by default, reads from the keyboard. But the user can choose to read from a file, using the redirection `<` operator *from the command line* or from the output of another process, using the pipe `|` operator *from the command line*. Similarly, the standard output, by default, writes to the terminal. But the user can choose to write to a file using the redirection `>` operator *on the command line* or to the input of another process, using the pipe `|` operator, again, on the command line when invoking the program. You will see how cool these are later. But for C programming, it suffices to know for now that we only need to read from `stdin` and write to `stdout` in our code, and we let the users decide where they come from / go to.

No `printf` and `scanf` (yet)

In almost all articles and textbooks on C that I have seen, the `scanf` and `printf` functions are taught as the standard C library functions to perform the input and output respectively. The function `scanf`, however, is tricky to use correctly and securely. The function `printf` comes with many nuances, such as having to remember the different

conversion specifiers and modifiers. I would rather not teach you `scanf` and `printf` at this stage and focus on how to solve problems instead. As such, CS1010 provides you with a library to perform I/O. The library provides a small set of essential functions to read and write `long` values, `double` values, space-separated words, and lines of text.

> If you insist on using `printf` and `scanf` for your assignments, make sure you know what you are doing. Otherwise, you will run into strange bugs if you are not using it correctly (and it is non-trivial to use them correctly). If you are an experienced programmer with C, you can skip ahead to take a look at Unit 29 on the pitfalls of using `printf` and `scanf`.

You can find the documentation for the CS1010 I/O Library here. We will see how to use the library to improve our hypotenuse computation program here.

Using the CS1010 I/O Library

Let's modify our earlier program to now read the base and height from `stdin`, compute the hypotenuse, and print the results out to `stdout`.

```
#include <math.h>
#include "cs1010.h"

long square(long x)
{
  return x*x;
}

double hypotenuse_of(long base, long height)
{
  return sqrt(square(base) + square(height));
}

int main()
```

```
14  {
15    double hypotenuse;
16    long base = cs1010_read_long();
17    long height = cs1010_read_long();
18    hypotenuse = hypotenuse_of(base, height);
19    cs1010_println_double(hypotenuse);
20  }
21
```

The first change you see (on Line 2) is to include the file `cs1010.h`, which includes the declaration of functions provided by the library. On Lines 17 and 18, we introduce two new `long` variables named `base` and `height`, which we initialized with the returned value from `cs1010_read_long()`. The function `cs1010_read_long` reads a `long` value from `stdin` and returns the value. For now, we assume that the inputs are correctly passed to the program.

> **Remember to include ()**
>
> C requires that function calls be identified with `()`. It is common for beginners to skip `()` and call a function like `long base = cs1010_read_long;`, leading to compilation errors.

Finally, on Line 20, we print the resulting hypotenuse to `stdout` using the library function `cs1010_println_double`. Note that there are two versions of functions to print a `double` value: `cs1010_println_double` and `cs1010_print_double`. The one with `println` prints a newline character as well.

Refer to CS1010 Compilation Guide on how to compile a program that uses the CS1010 I/O library.

Note that the `main` function above can be written as a single statement without any variable and assignment. The resulting code, however, is not necessarily easier to understand.

```
int main()
{
  cs1010_println_double(hypotenuse_of(cs1010_read_long(),
cs1010_read_long()));
}
```