CS50

Exit Codes

Overview

You may have noticed that the **`main`** function definition returns an **`int`**, but in the past we haven't been returning any value at the end of the main function. By default, if no return value is specified in the **`main`** function, the compiler will automatically assume that the **`main`** function returns **`0`**. The value that the **`main`** function returns is referred to as the program's **exit code**. As your programs become longer and more complicated, exit codes can be a valuable tool.

Key Terms

- exit code
- input validation

```
#include <cs50.h>
#include <stdio.h>

int main(int argc, string argv[])
{
    if (argc == 2)
    {
        printf("hello, %s\n", argv[1]);
        return 0;
    }
    else
    {
        return 1;
    }
}
```

Using Exit Codes

By convention, if a program completed successfully without any problems, then it should return with an exit code of **`0`**. That's why the compiler assumes that if no return statement is provided at the end of **`main`**, the program should return **`0`**. You could, however, explicitly specify **`return 0`** at the end of a program.

Any non-zero exit code (commonly **`1`** or **`-1`**) conventionally means that there was some sort of error during the program's executing that prevented the program from completing successfully.

One common use of exit codes is during **input validation**: when the program checks to make sure that the inputs provided by the user are valid. For instance, if a program expects two command line arguments, but only receives one, it might return a non-zero exit code to signal an error.

Take the above program, which takes (in addition to the program's name) a command line argument specifying the user's name. The program then says hello to the user.

Upon starting the **`main`** function, the program checks to see whether **`argc`** is 2. If it is, then the user's input is valid: the program can say hello, and successfully return with exit code **`0`**.

On the other hand, if the user didn't provide the correct input values (say, by not providing enough command line arguments, or by providing too many), then **`argc`** would not be equal to **`2`**.

The program would then execute the **`else`** block, which returns the number **`1`** as an exit code, indicating that there was an error in the program's execution.

Debugging

If you run programs normally from the command line, you won't actually see the return values that the **`main`** function returns. However, many debugging tools, which are programs designed for helping programmers find sources of problems in their code, will allow you to see the exit code with which the **`main`** function exited.

Knowing the exit code can be a valuable tool for determining why a program failed during the debugging process. In larger programs which may include many instances of error checking and input validation, the program may return a different exit code for each error.

If the program fails, knowing which status code the program exited with can help to find out what went wrong.