CS50 Syntax

General Reference Sheet

Overview

C is a programming language with which you can write programs. Programming languages like C require you to write using a very specific **syntax**: a set of rules that describe how to arrange words and symbols (like brackets and parentheses) in order to write working statements that together can form a complete program. C's syntax might seem complicated at first, but with practice, the syntax of the language will start to become second nature to you.

Key Terms

- C
- syntax
- function
- string
- compile

Your First C Program

```
1 | #include <stdio.h>
2 |
3 | int main(void)
4 | {
5 |     printf("hello, world\n");
6 | }
```

The code to the left is an example of a simple program in C which displays "hello, world" in the terminal window when it runs. Line numbers have been added to the left side of each line for reference, but they shouldn't be included in the code itself.

On line 1, `#include <stdio.h>`, tells your program to access a set of pre-written **functions** stored in a file called `stdio.h`, where a function is a collection of programming statements that performs a particular task.

By including `stdio.h`, your program can now take advantage of code that people have already written in the past: in particular, a function called `printf`, which displays text on your screen.

On line 3, `int main(void)` defines a function which acts as the beginning of your program, serving the equivalent of the "When Green Flag Clicked" button in Scratch. When your C program runs, it will look for the `main` function to know where to start. The curly braces on lines 4 and 6 hold the code of the `main` function together. Anything inside of the curly braces is therefore a part of the `main` function.

In this program, the `main` function has just one programming statement: `printf("hello, world\n")`. `printf` is a function (that was written in `stdio.h`) which displays a **string** (which is just a fancy way of saying "text") on the screen. In C, strings are always surrounded by double quotation marks.

Within the parentheses for `printf`, we've provided `printf` with a string as input, so that `printf` knows what string to display on the screen. In this case, our string is `"hello, world\n"`. The `\n` character tells `printf` to display a new line. The result of displaying `"hello, world\n"`, then, is to print out the words `hello, world` on the screen, followed by a new line. Finally, at the end of line 5 is a semicolon (`;`), which is C's way of defining the end of a programming instruction.

Compile and Run Your Program

Now that you've written your program, it should be saved in a file (typically ending with `.c`). In this case, we might call our file `hello.c`. This is your source code file. However, computers can't understand C code directly: remember that computers can only understand sequences of 0s and 1s. First, we need to **compile** our program: converting it from source code to object code, which is just sequences of 0s and 1s. Once the source code is compiled into object code, it can be executed. Any lines of code that begin with `//` or are enclosed with `/*` and `*/` will be ignored by the compiler: these lines are called comments, and will frequently be used by programmers to document their code so that people reading the code later (including themselves) can understand what's happening in the code.

Several compilers exist, including `clang` and `gcc`. We can also compile our program by typing the command `make hello` at the command line, which uses the `clang` compiler. If the program compiles successfully (without errors), then we can run the program by typing `./hello` at the command line. If all goes well, `hello, world` should be printed to the screen.