Variables

General Reference Sheet

Overview

A **variable** is a storage container for data that is capable of holding different values that may change or update as programs execute. Your program can read the contents of a variable, update the contents of a variable, and display the value of a variable on the screen. Computer programs can use variables in order to remember useful information that the programs can then use later in the code.

Key Terms

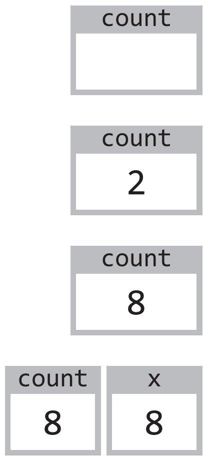
- variables
- declaration
- type
- integer

```
1  int count;

2  count = 2;

3  count = 8;

4  int x = count;
```



Declaring and Setting Variables

The first step to using a variable in C is to let your program know that you want the variable to exist. This step is called the variable's **declaration** (also known as initialization). In C, this is done by first specifying the variable's **type**, which tells the program what kind of information will be stored inside of the variable, and then by specifying the variable's name (followed by a semicolon to end the programming statement).

For instance, in line 1 to the left, we've declared a new variable of type `int` to be named `count`. An `int` is a data type which stores an **integer**, which could be positive whole numbers, negative whole numbers, or zero (but not fractions or decimals). Currently, no value has been assigned to `count`: we've just told the program to create a space within which values can be stored later.

Once a variable has been declared, it can be manipulated in various ways. Line 2 takes the variable `count` and assigns its value to be `2`. Now, the number `2` is stored inside of the variable `count`. Optionally, we could have combined lines 1 and 2 into a single programming statement to declare a variable and set its value at the same time, via a line of code such as: `int count = 2;`.

After a variable has been given a value, its value can be updated. Line 3 updates the value of `count` again, this time to be `8`. Now, `count` forgets the number `2` and remembers the number `8` instead.

The value of a variable can be accessed just by using its name. For instance, line 4 declares a new variable (also of type `int`) this time named `x`, and initially sets its value to be `count`. This tells your program to go to the `count` variable, see what value is inside, and set the value of `x` to be that value. Since the current value of `count` is `8`, the value of `x` is set to also be `8`.

Variables from User Input

In many cases, a program may need to take input from the user and store the input as a variable. CS50 has written several functions (declared in a file called `cs50.h`) that serve this very purpose.

For instance, `GetInt()` prompts the user to input an integer. In the program to the right, line 6 uses `GetInt()` to take in an integer as input from the user, and saves that integer in a variable called `i`.

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int i = GetInt();
7      printf("i is %d", i);
8  }
```

Line 7 then displays the value of the variable on the screen. The `%d` in the string is a special syntax which acts as a placeholder for an integer. We tell `printf` what integer to use in that placeholder by passing it an additional argument, where an argument is just a value inside of the parentheses of a function. Inside of the parentheses next to `printf` we've included two arguments: the string `"i is %d"`, and the integer `i`, which will take the place of `%d`. For example, if the user were to enter the number `28` as input on line 6, then line 7 would replace `%d` with the value of `i` (which is `28`) and display the string `"i is 28"` on the screen.