1.3a — A first look at cout, cin, and endl

BY ALEX ON JANUARY 12TH, 2015 | LAST MODIFIED BY ALEX ON AUGUST 21ST, 2017

std::cout

As noted in previous sections, the std::cout object (in the iostream library) can be used to output text to the console. As a reminder, here's our Hello world program:

```
#include <iostream>

int main()
{
    std::cout << "Hello world!";
    return 0;
}
```

To print more than one thing on the same line, the output operator (<<) can be used multiple times. For example:

```
#include <iostream>

int main()
{
    int x = 4;
    std::cout << "x is equal to: " << x;
    return 0;
}
```

This program prints:

```
x is equal to: 4
```

What would you expect this program to print?

```
#include <iostream>

int main()
{
    std::cout << "Hi!";
    std::cout << "My name is Alex.";
    return 0;
}
```

You might be surprised at the result:

```
Hi!My name is Alex.
```

std::endl

If we want to print things to more than one line, we can do that by using std::endl. When used with std::cout, std::endl inserts a newline character (causing the cursor to go to the start of the next line).

For example:

```
#include <iostream>

int main()
{
    std::cout << "Hi!" << std::endl;
    std::cout << "My name is Alex." << std::endl;
    return 0;
}
```

This prints:

```
Hi!
My name is Alex.
```

std::cin

std::cin is the opposite of std::cout -- whereas std::cout prints data to the console using the output operator (<<), std::cin reads input from the user at the console using the input operator (>>). Now that you have a basic understanding of variables, we can use std::cin to get input from the user and store it in a variable.

```
//#include "stdafx.h" // Uncomment this line if using Visual Studio
#include <iostream>

int main()
{
    std::cout << "Enter a number: "; // ask user for a number
    int x; // no need to initialize x since we're going to overwrite that value on the very next line
    std::cin >> x; // read number from console and store it in x
    std::cout << "You entered " << x << std::endl;
    return 0;
}
```

Try compiling this program and running it for yourself. When you run the program, it will print “Enter a number: ” and then wait for you to enter one. Once you enter a number (and press enter), it will print “You entered ” followed by the number you just entered.

For example (I entered 4):

```
Enter a number: 4
You entered 4
```

This is an easy way to get input from the user, and we will use it in many of our examples going forward.

If your screen closes immediately after entering a number, please see section **0.7 -- a few common cpp problems** for a solution.

(As an interesting side note, if you enter a really big number, the results may surprise you. Try it! This happens because x can only hold numbers up to a certain size. After that, it “overflows”. We’ll discuss overflow in a future section.)

std::cin, std::cout, <<, and >>

New programmers often mix up std::cin, std::cout, << and >>. Here’s an easy way to remember:

- std::cin and cout always go on the left-hand side of the statement.
- std::cout is used to output a value (cout = character output)
- std::cin is used to get an input value (cin = character input)
- << is used with std::cout, and shows the direction that data is moving from the r-value to the console. `std::cout << 4` moves the value of 4 to the console
- >> is used with std::cin, and shows the direction that data is moving from the console into the variable. `std::cin >> x` moves the value from the console into x

(Admin note: Discussion of the std::namespace and using statements has been moved to lesson **1.8a -- Naming conflicts and the std namespace**)


1.4 -- A first look at functions and return values


Index