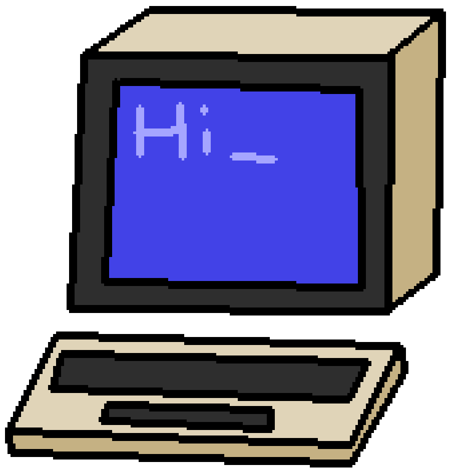
Table of Contents

Introduction

When we're getting started with most programming languages, we begin by writing simple text to the screen, and getting simple text from the user.

Different languages have different ways of doing this. We will cover how C++, C#, and Java output text to the screen.



Output

Output library (C++)

- For C++, you will need to **`#include <iostream>`** at the top of your source file.
- For C#, to use Console.WriteLine instead of System.Console.WriteLine, you need to put **`using System;`** at the top of your source file.

Print on separate lines

To output a line of text to the screen, we must use the appropriate function. If we are hard-coding a string, it must be contained within double-quotes. For this example, let's pretend that **`variableName`** is storing the value “Hello, KC!”

Language	Code	Output

C++	`cout << "Hello, world!" << endl;`
	`cout << variableName << endl;`
		`Hello, world!`
		`Hello, KC!`

C#	`Console.WriteLine( "Hello, world!" );`
	`Console.WriteLine( variableName );`	

Java	`System.out.println( "Hello, world!" );`
	`System.out.println( variableName );`	

Print on the same line

If we don't put a new line at the end of our output, then the text output will continue on the same line, and will only go to the next line once it hits the edge of the screen.

Language	Code	Output

C++	`cout << "Hello, world!";`
	`cout << variableName;`
		`Hello, world!Hello, KC!`

C#	`Console.Write( "Hello, world!" );`
	`Console.Write( variableName );`	

Java	`System.out.print( "Hello, world!" );`
	`System.out.print( variableName );`	

Concatenating strings in our output

In C++, C#, and Java, we can **concatenate** strings. When we do this, we can keep adding variable values and string literals (hard-coded strings within " and ") to one output statement.

In C++, we use **streams** so you put **<<** between each item. With Java and C#, put **+** between each.

Below, let's assume that the value of **`city`** is "Overland Park".

Language	Code	Output
C++	`cout << "City: " << city;`	`City: Overland Park`
C#	`Console.Write( "City: " + city );`	
Java	`System.out.print( "City: " + city );`	

You can keep chaining variables and strings together as well.

Language	Code
C++	`cout << "City: " << city << " State: " << state << endl;`
C#	`Console.Write( "City: " + city + "State: " + state );`
Java	`System.out.print( "City: " + city + "State: " + state );`

Special characters

If we want to put a lot of information in one output statement, we can use some special characters to format our text.

`\t` will add a tab to our output:

```
Name:       Luna
```

`\n` will add a new line to our output:

```
Name:
Luna
```

These values go within your string literal:
`"This is text \n and on a new line", "Label: \t" + value`

Input

A program isn't very interesting to use unless we can get user input. With these console-based applications, we will get input from the user via the keyboard.

Input library (C++, Java)

- For C++, you will need to **`#include <iostream>`** at the top of your source file.
- For Java, you will need to **`import java.util.Scanner;`** at the top of your source file.

Setup (Java)

Before you start getting the user's input in Java, you need to create a **Scanner object**. A scanner contains methods and variables within it that handle getting information from the user and giving it to your program. Declare a scanner object like this:

```
Scanner scanner = new Scanner( System.in );
```

Then we can use **scanner** to get input from the user.

Setup (others)

C++ and C# don't require special setup in order to start getting user input.

Getting user input in C++

To get the user's input in C++, you need to have a variable already declared that you will store the information in. The code will be the same for any basic data-type:

C++

```
cin >> city;
```

This will work for floats, doubles, integers, chars, and strings. However, if you want to get more than one word for a string (basically, a string that contains spaces), you have to use the **getline** method instead.

Instead of using **cin** and the stream operator **(>>)** pointing toward the variable, use **getline**, with **cin** as the first argument, and your storage variable as the second argument.

C++

```
getline( cin, city ); // Get input from keyboard, store in city
```

Getting user input in C#

In C#, we can get the user's input with **Read** or **ReadLine**. If we use Read, we can only type in one word (no spaces). If we use ReadLine, we can enter a line of text, including spaces. By default, keyboard input will be a string.

C#

```
string input = Console.Read();
```

C#

```
string input = Console.ReadLine();
```

If you want to turn the user's input into an integer, float, or other data type, we need to cast it:

C#

```
Console.WriteLine( "How old are you?" );   // ask user
string input = Console.Read();             // get input
int age = Convert.ToInt32( input );        // convert to int
```

Getting user input in Java

In Java, we have different methods we use with the **scanner** object in order to get different data-types:

Integers	`int myInteger = scanner.nextInt();`
String (no whitespace)	`String word = scanner.next();`
String (with whitespace)	`String word = scanner.nextLine();`
Double	`double myDouble = scanner.nextDouble();`

The string with no whitespace is essentially how you can store one word – it cannot contain spaces. With whitespace, you could store a line of text, including spaces.

Example Programs

Let's see some examples of using input and output in each language...

C++

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    cout << "What is your name? ";
    string name;
    cin >> name;

    cout << "Hello, " << name << "!" << endl;

    cout << "How old are you? ";
    int age;
    cin >> age;

    cout << "How much money do you have? ";
    double money;
    cin >> money;

    double moneyPerYear = money / age;

    cout << "You must have earned "
          << moneyPerYear << " per year!" << endl;

    return 0;
}
```

C#

```
using System;

public class SampleProgram
{
    public static void main()
    {
        Console.Write( "What is your name? " );
        string name = Console.ReadLine();

        Console.WriteLine( "Hello, " + name + "!" );

        Console.Write( "How old are you? " );
        int age = Convert.ToInt32( Console.Read() );

        Console.Write( "How much money do you have? " );
        double money = Convert.ToDouble( Console.Read() );

        double moneyPerYear = money / age;

        Console.WriteLine( "You must have earned "
          + moneyPerYear + " per year!" );
    }
}
```

Java

```
public class SampleProgram
{
    public static void main()
    {
        Scanner scanner = new Scanner( System.in );

        System.out.print( "What is your name? " );
        String name = scanner.nextLine();

        System.out.println( "Hello, " + name + "!" );

        System.out.print( "How old are you? " );
        int age = scanner.nextInt();

        System.out.print( "How much money do you have? " );
        double money = scanner.nextDouble();

        double moneyPerYear = money / age;

        System.out.println( "You must have earned "
            + moneyPerYear + " per year!" );
    }
}
```

Example Output

```
What is your name? Bill Gates
Hello, Bill Gates!
How old are you? 60
How much money do you have? 79.4
You must have earned 1.3233333333333335 per year!
```

Note that this is 79.4 billion USD for Bill Gates, but 79,400,000,000 is too large of a number for the program. ;)