# Pre-Lab for Lab 1

## Lab Overview

In this lab, you will learn to write a basic C++ program, compile and run a program, and apply basic C++ syntax to compute values.

For all your pre-labs, your task is to complete and submit the pre-lab and then **bring the completed pre-lab to the lab.**

## Pre-Lab Submission

Complete this Pre-Lab and bring it with you to your lab. Submit it as a PDF by 23:59 the night before your lab section for Lab 1.

* Complete the following sections of this document before submitting:
	+ Lab Prep 1: Declaring Variables
	+ Lab Prep 2: Prompt the user and get input

## Learning Goals

* learn to write a basic c++ program
* learn to compile and run a basic c++ program
* apply basic c++ syntax to compute a value

## 1. Declaring Variables

In today’s lab we’ll write a program to compute some values. In order to do so, we will need to declare variables to hold values. For example, we’ll need to store the values we get from the user to use them later to compute the result.

We’ll also need to declare variables to hold interim results.

Remember, when we declare a variable we have to specify:

* the variable’s type
* the variable’s identifier (its name)

##### Declaring variables example 1

Figure : Identifying appropriate Type and Identifier for a variable based on what is needed

Declare a variable to hold the user’s name:

* Type: an integer makes sense, since we don’t usually say someone is 34.2 years old
* Identifier: *age* would make sense if we only needed one variable for this value

Figure 2: Declaring a variable named “age” that is an integer

int age;

##### Declaring variables example 2

Figure 3: Identifying appropriate Type and Identifier for a variable based on what is needed

Figure 4: Declaring a variable named “stringLength” that is a floating point number. If more precision was needed, we could have used double instead of float

float stringLength;

Declare a variable to hold the length of a piece of string.

* Type: since we might want any length, a floating point number makes sense (for 4.2, or 100.01)
* Identifier: *stringLength* lets us know that it is a length, and that it is for a length of string

Lab Prep 1: Declaring Variables

Our program is going to prompt the user to enter two values (initially):

* the speed they drive (on average)
* the distance they travel.

We’ll also need to store:

* the time (in hours)
* time (in minutes).

Think about the type of variable you need (integer? Floating point?), and remember to give your variables good, meaningful names.

Fill out the following table with the variable declarations you’ll need in your program for these values (you may need others later, but this is a good start).

|  |  |
| --- | --- |
| **Need to store:** | **Variable Declaration** |
| User’s speed |  |
| Distance travelled |  |
| Computed time in hours |  |
| Computed time in minutes |  |

## 2. Prompt the user for input

We need to prompt the user for input, and then store their input in the variables we declared. Since we don’t want (yet, anyway) to write our own code to deal with getting keyboard input, we can use the cstdio library for user input and output. This library has a number of functions that let us get user input, and output things to the user on the console

This means we need to add the following line to the top of our program:

## Including a library for input/output (i/o) example

Figure 5: Including a library in a cpp program

/\*

 \* header comment here

 \*/

#include <cstdio>

## Printf for outputting to the user

printf(“hi there\n”);

Figure 6: Using printf for output

For outputting to the user, we’ll use the printf function. In this case we’re simply outputting a sentence, so we can pass this sentence to printf as a *string*.

A *string* is a series of characters, and is enclosed in quotes (“ and “) in C++ (and many other languages, as well).

Figure 7: cpp code to print out the sentence “hi there”. The \n part adds a newline to the end of the string

## Special Characters in Strings

Sometimes we want to include characters in our strings that aren’t just letters, numbers and punctuation. For example, we can put things like newlines and tabs in our strings to help format them nicely for output. We indicate these special characters as follows:

|  |  |
| --- | --- |
| **Special Character** | **Representation** |
| newline | \n |
| tab | \t |

## Scanf for getting user input

Figure 9: cpp code to take an integer entered by the user and copy it into the variable “age”. %d indicates an integer.

scanf("%d", &age);

Figure 8: Using scanf for input

To get input from the user, we’ll use scanf. It is a bit more complicated than printf because we also have to pass it the variable we want the user input to get saved to. We have to tell scanf what *type* of value we’re expecting, and for reasons that we’ll ignore for now we have to add the *&* symbol before the variable’s identifier. In this case, scanf takes 2 parameters:

* the string that represents what we’re expecting from the user
* the variable the user’s input will go into

scanf is also in the cstdio library, so we *don’t* need to add another include to use it.

### Placeholders

When using scanf, we use *placeholders* to indicate what type of value we’re expecting in a string. For more details, see: <http://www.cplusplus.com/reference/cstdio/scanf/>

|  |  |  |
| --- | --- | --- |
| Type | Placeholder | Example |
| Integer (int) | %d | scanf(“%d”, &count); |
| Floating point (float) | %f | scanf(“%f”, &length); |
| Character (char)  | %c | scanf(“%c”, &initial); |
| Scientific or floating | %g | scanf(“%g”, &radius); |
| String | %s | scanf(“%s”, lastName); |

Lab Prep 2: Prompt the user and get input

Your program is going to prompt the user to enter two numbers: their speed and the distance they travel. Fill in this table with the printf and scanf commands you’ll use to accomplish this.

|  |  |
| --- | --- |
| **Task** | **Code** |
| Prompt user to enter the distance (printf) |  |
| Get distance from the user (scanf) |  |
| Prompt user to enter the speed (printf) |  |
| Get speed from the user (scanf) |  |

## 3. Compute the values

In lab we’ll work together to compute the value, by converting the formula given in the lab to a C++ expression.