Lab 1: Recipe

Information

Topics

- Building a basic program
- Variables
- Data Types
- Output with cout
- Input with cin

Turn in
Once you are finished, turn in each **part** 's source file. These will have the extention .cpp, and in Windows, it will show up as **C++ source file** .

Introduction: Project, main, and cout

Create the project

First, you will need to create a new project in your IDE (Visual Studio, Code::Blocks, etc.). Name the project **LAB1_INTRO_PROJECT** . You will also create one source file, name it **lab1_intro.cpp**

IDE means Integrated Development Environment

For help on how to create a project and a file, see below.

Creating a project in Visual Studio

1. Go to **File - New - Project...** .
2. In the New Project window, select **Visual C++** from the left side.
3. In the center, select **Empty Project** . (It must be Empty Project!)
4. At the bottom, give your project a name in the **Name** textbox.
5. Next to the **Location** textbox, click on **Browse...**
   - If you're working on a school computer, it might be good to navigate to the Desktop and place your project here for quick access.
   - Avoid putting your project in the *temp* directory.
   - If you store your project on a thumbdrive, the compile process will be very slow. Move the project to your thumbdrive after you're done with your work.
6. Click OK

At this point, you will have a new project, but it will be empty. You will need to add a source file.

1. In the **Solution Explorer** , right click your project and select **Add - New Item...** .
2. Select **C++ File (.cpp)** , and at the bottom of the window, enter a name in the **Name** textbox.
3. Click **Add** .

Now when you double-click the source file in the Solution Explorer, it will open up.

Add libraries

To display text to the screen, we will need to include the **iostream** library in our program. The iostream library contains a command called `cout` which allows us to do this.

At the top of your source file, add the following lines:

> A **library** is code that has been written by somebody else, and "packaged" so that the code can be used across multiple programs.

```
1 #include <iostream>
2 using namespace std;
```

To import libraries of code, we use the `#include` command.

The C++ Standard Library contains many types of functionality, including drawing text to the screen, getting input from the keyboard, calculating square roots and trig functions, reading and writing text files, and a lot more. **iostream** allows us to use `cin` (console-input) and `cout` (console-output) in our programs.

The `using namespace std;` line, for the time being, we can take for granted. It is basically stating that we're using the *standard* C++ library.

Add comments

In C++, there are two ways to add comments:

```
1 // single line
```

or

```
1 /*
2 Multi-
3 Line
4 Comment !
5 */
```

At the top of your source file, add a comment with your name.

Program starting point

Every program in C++ begins at the **main** function. For now, we will just memorize how this part of the code looks. Later on, we will write our own functions.

```
1 int main()
2 {
3     return 0;
4 }
```

main is the name of the function. Every function opens and closes with { and }, respectively. Any code between the opening and closing curly braces are *inside* the function.

The `return 0;` command is where our program ends - the 0 signifies, "no errors occurred during the program execution".

While working in main() for now, your program code will go below the { and above the `return 0;` .

Displaying output

To write text to the screen in C++, we use the **cout** command. It will look like the following, and go within the `main()` function, after the opening curly brace { and before the `return 0;` .

```
1 int main()
2 {
3     cout << "Hello, world!";
4     return 0;
5 }
```

In C++, **statements** end with a semicolon `;` . The semicolon is how C++ knows you're done with a command.

Commands can span several lines, as long as the semicolon is there at the end. In the following example, the `cout` statement spans five lines, and ends at `<< endl;`

```
int main()
{
    cout
        << "Hello"
        << endl
        << "World!"
        << endl;

    return 0;
}
```

The `endl` commands stand for "end-line", and are used to start writing text on a new line. The output for this example code would look like this:

```
Hello
World!
```

If we didn't have any `endl` statements, then all text would run together on the same line.

Type out the following code, adn then build and run to practice.

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

int main()
{
    cout << "Hello, world!" << endl;

    cin.get();
    return 0;
}
```

Build your program: In order to test out your program, you must first *build* it, and then *run* it. To build your program in Visual Studio, go to **Build - Build Solution** from the top menu.

Build status: At the bottom of the screen, there is a pane titled **Output** . This will show the status while your program is building.

Build errors: If there is a **syntax error** in your program, such as a typo, then you will get a **build error** , and you will have to fix it and build again. Build errors are listed out in the **Error List** pane.

Run your program: Once the build is completed, you can run the program by going to **Build - Start Debugging** .

Program opens and closes immediately!

If you see a black box pop up and immediately close, that's your program running. Once the program hits the `return 0;` command, it closes the program.

To make sure your program doesn't end immediately, add `cin.get();` before the `return 0;`

Project 1: Displaying a recipe

Create a new project in your IDE. Name the project **LAB1_PART1_PROJECT**. You will also create one source file, name it **lab1_part1.cpp**

Use the following as the starting point for the program:

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

int main()
{

    return 0;
}
```

For this project, you will display a cookie recipe to the screen. Using `cout` statements, write out each of the following ingredients:

1 tsp	baking soda
1/2 tsp	baking powder
1 c	butter
1 1/2 c	white sugar
1	egg

You can have multiple `cout` statements, or put all of these in the same statement. You can also use `"\t"` to add a tab in your output.

When you run the program, your output should look like this:

```
1 tsp baking soda
1/2 tsp baking powder
1 c butter
1 1/2 c white sugar
1 egg
```