

## Project 1 - Click to See Project 1 SPECS

100 可能的分數

1/14/2025

嘗試 1



進行中

下一個：提交作業



添加評論

允許無限制的嘗試

1/15/2025

▼ 詳細資料



## Instructions

## Programming Assignment 1

## Getting Started with C++

**Time due: 9:00 PM Tuesday, January 13th**

The purpose of this assignment is to have you start learning how to use the g31 and either the Visual C++ or Xcode environments, and understand a variety of programming errors.

Here's what you are to do:

1. (optional) Obtain a copy of Visual C++ and install it. You don't need to do this if you prefer to use Visual C++ on the SEASnet Windows Server or if you prefer using Xcode.  
(optional) Obtain a copy of Xcode and install it. You don't need to do this if you prefer using Visual C++.
2. Enter the C++ program found at the end of this assignment instruction into your development environment. Do not change the program yet.
3. Build the executable from the program. (Fix any typos you may have made when entering the program.)
4. Execute the program with a variety of reasonable input integers to see if it runs as one would expect from reading the source code. If you're running Visual C++ and the console window disappears when your program finishes executing before you have a chance to see the output, either you launched your program by double-clicking on the .exe file, or you're running Visual C++ 2017 and you either selected Start Debugging or forgot to do step 4 from the Visual C++ writeup.
5. Using the program as given, without changing it in any way, run it with input integers that cause it to produce incorrect, unusual, or nonsensical output. (Notice we're saying to try input *integers*, not input like `1273.54` or `vote`.)
6. Starting from the program as given, introduce into the source code an error that someone might make that, while not preventing a successful build, causes the program when it runs to produce incorrect results from reasonable input.
7. Again starting from the program as given, introduce two distinct types of mistakes that someone might make, each of which would cause the program to fail to compile correctly.

You should create a separate project for each of steps 2, 6, and 7, since neither Visual C++ nor Xcode allow you to have multiple files in the same project if more than one has a main routine.

In addition to running the programs under Visual C++ or Xcode, run them using g31 as the g++ with Linux writeup tells you. (As the Project Requirements document tells you, "run using g31" is shorthand for "run using g31 on cs31.seas.ucla.edu" — that specific command (g31, not g++) on a SEASnet machine reached via that specific name.)

What you will turn in for this assignment is a compressed file in zip format containing exactly four files:

- A file named **original.cpp** that contains the program as given.

- A file named **logic\_error.cpp** with the program you produced in step 6.
- A file named **compile\_error.cpp** with the program you produced in step 7.
- A file named **report.docx** (in Microsoft Word format) or **report.txt** (an ordinary text file) that specifies
  - the specific input you provided to the program built from original.cpp in step 5 above that produced incorrect, unusual, or nonsensical output. (Notice that we're asking about step 5, not step 4.)
  - the error you introduced into logic\_error.cpp
  - the two errors you introduced into compile\_error.cpp

Briefly discuss any error messages the compiler reported, and incorrect, unusual, or nonsensical results. This report may well end up being much less than a page long.

The zip file itself may be named whatever you like.

Do **not** include anything else in the zip file. (Some Windows users seem not to be aware of Windows filename extensions, so end up putting the wrong files in their zip file.) To create a zip file on a SEASnet machine, you can select the four files you want to turn in, right click, and select "Send To / Compressed (zipped) Folder". Under macOS, copy the files into a new folder, select the folder in Finder, and select File / Compress "*folderName*"; make sure you *copied* the files into the folder instead of creating aliases to the files.

We will be using software tools to help us grade your projects, so there are certain requirements you must meet for the tools to work: **The zip file you turn in for this project must have exactly four files in it, with exactly the names indicated. If you do not follow these requirements, your score on this project will be zero.** "Do you mean that if I do everything right except misspell a file name or include an extra file, I'll get no points whatsoever?" Yes. That seems harsh, but attention to detail is an important skill in this field. A draconian grading policy certainly encourages you to develop this skill.

The only exception to the requirement that the zip file contain exactly four files of the indicated names is that if you create the zip file under macOS, it is acceptable if it contains the additional files that the macOS zip utility sometimes introduces: `__MACOSX`, `.DS_Store`, and names starting with `._` that contain your file names.

Although the Project Requirements document tells you that programs you turn in must run successfully, for this project, there's an exception: `compile_error.cpp` must not build successfully, and `logic_error.cpp` must build successfully, but when executed, must produce incorrect results from reasonable input.

By Sunday, January 12th, there will be links under Assignments or Modules - Week 1 on BruinLearn that will enable you to turn in your zip file electronically. Turn in the file by the due time above. You may turn in your file multiple times; we will consider only the last one you submit. Remember that most computing tasks take longer than expected; this applies especially to steps 1, 2, and 3 above. Start this assignment now!

Use this project as an opportunity to learn what happens when you make mistakes. After you've turned in what's required for this project, play around. Introduce a mistake into the program and see what happens. Fix it. Introduce a different mistake and see what happens then. Fix it. Keep doing this so you can see the kinds of problems that might arise when you develop your own programs and what the compilers say for each particular problem (if they even detect them at all). See what happens if you make more than one mistake in the program. Will they all be detected? Will an earlier mistake interfere with the reporting of a later one?

Here is the C++ program:

```
// Code for Project 1
// Report poll results

#include <iostream>
using namespace std;

int main()
{
    int numberSurveyed;
    int forTrarris;
    int forHump;

    cout << "How many registered voters were surveyed? ";
    cin >> numberSurveyed;
    cout << "How many of them say they will probably vote for Trarris? ";
    cin >> forTrarris;
```

```
cout << "How many of them say they will probably vote for Hump? ";
cin >> forHump;

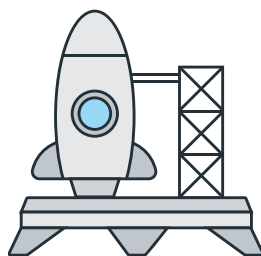
double pctTrarris = 100.0 * forTrarris / numberSurveyed;
double pctHump = 100.0 * forHump / numberSurveyed;

cout.setf(ios::fixed);
cout.precision(1);

cout << endl;
cout << pctTrarris << "% say they will probably vote for Trarris." << endl;
cout << pctHump << "% say they will probably vote for Hump." << endl;



if (forTrarris > forHump)
    cout << "Trarris is predicted to win the election." << endl;
else
    cout << "Hump is predicted to win the election." << endl;
}
```

## 選擇提交項目類型



選擇要上傳的檔案

或

 網路攝影機照片 Canvas 檔案<https://bruinlearn.ucla.edu/courses/198832/modules/items/7172115><https://bruinlearn.ucla.edu/courses/198832/modules/items>