MTE 121 Assignment 1 Fall 2024

Deliverables

This assignment will be completed in groups. You are free to choose your tutorial partners. Please talk to Katherine Idzik (Kat) if you would like her help finding a partner.

In this assignment you will:

- Debug a program
- Receive user input
- Output to the console
- Apply basic math operations

There are three deliverables for this assignment:

- Question 1 – C++
- Question 2 – C++
- Question 3 – C++

DEV C++

This course presumes the use of the DEV C++ integrated development environment. It is not mandatory that you use DEV C++, but the support staff are most familiar with that environment and may be less able to help resolve issues with others.

Naming Conventions

File and folder names should be eight characters or less, and contain only letters, numbers, and underscore character. (For example: *a1_q1.cpp*)
While DEV C++ will allow you to name your files and folders anything you like, there can be issues with spaces and special characters.

Using DEV C++ on the school computers (in person)

DEV C++ should be installed in all engineering computer labs. Due to network security, executables cannot be run from the network drive (N:). The recommended process is to initially **save your work in a folder on your N: drive** and then copy your code (.cpp) over to your desktop to run.

A list of engineering computer lab locations is available at: https://uwaterloo.ca/engineering-computing/computer-labs

Remotely using DEV C++ on the school computers (englab)

DEV C++ should be installed in all engineering computer labs and can be run remotely by logging into one of these computers. Go to https://uwaterloo.ca/engineering-computing/ and follow the instructions for installing a VPN on your computer. Once you are running the VPN, you can login to the engineering computer lab machines by going to https://englab.uwaterloo.ca/

Due to network security, executables cannot be run from the network drive (N:). The recommended process is to initially **save your work in a folder on your N: drive** and then copy your code (.cpp) over to your desktop to run.

Using DEV C++ on your own computer:

If you are working on your own computer, you will need to download Dev-C++. Links and an instruction video are posted on Learn.

Please note that the teaching team has experience supporting the Dev-C++, but it will only work on Windows. If you are using a Mac, or you'd prefer to use a different program, we will try our best to help but it is your responsibility to understand the software you choose. Mac options include Xcode (easier to install) and CodeBlocks (old, need to install gnu tools to run, easier to use once installed).

1.

Question 1- Debugging (in C++)

Part 1: Getting the Code to Run

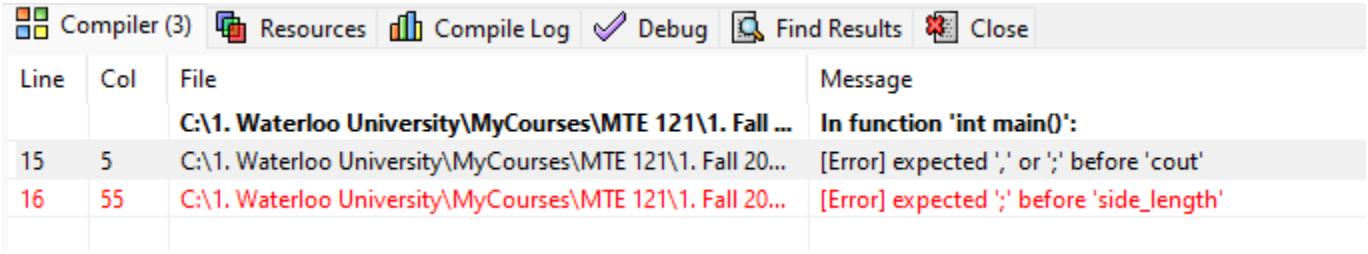
1. Start the compiler by selecting “Search Windows” and typing “Dev C++.” The application should appear in the search results.
2. Download *square.cpp* from learn.
3. Click on File -> Open and browse to find the sample program file *square.cpp*. Click Open.
4. Change the first line so that it includes your name.
5. Press “F9” or click the  icon as a shortcut. This “compiles” your code, meaning that it attempts to turn the code you see on the screen into a program that the computer can run. However, there are errors in the code, placed there on purpose so you get experience debugging code.
6. You will see several error messages listed in the message window.
7. Read the section below about dealing with error messages. Fix all the errors in the code.
8. When all of the errors are gone, the compiler will produce a file with the extension “.exe.” This is the executable file which can be run.
    - If you are using a school computer, find the “.exe” file in the folder your code is in, then drag it on to the desktop. Double click it and a new window will open up. It will show you the output of the code you just made
    - If you are using your own computer, you may just click “Compile & Run” and the executable should run directly

Debugging

Making errors that prevent compilation is very common, especially at first. Dev-C++ and many other programming environments will tell you two important things about your errors:

1. Where the errors in the code are.
2. What the errors might be Note:
    - The compiler can only identify syntax errors
    - The compiler usually is very good at telling you *where* the error is
    - The compiler usually is not very good at telling you *what* the error is

In the image below, we can see a screenshot of the Dev-C++’s error messages for square.cpp:



1. Double-click on the first message. Line 15 in your code should be highlighted.
2. The error is indicating that it expects a ‘**;**’ before the ‘ `cout`.’
3. We know that semicolons go at the end of statements, but the error seems to imply it should go before “`cout`.” This is an example of the compiler not quite getting it right.
4. What it means: the statement before “`cout`” is missing a semicolon. This is actually a previous line of code.
5. Add the missing semicolon at the appropriate location.
6. Fix the remaining errors and run your program.
7. Note that there may be logical errors. If necessary, fix your code to ensure the expected output.

Part 2: Getting Input from the User

1. Modify the program to prompt the user to input the value “side_length” before it is squared and cubed. Compile and run the program again.

2. To capture any console output, highlight all of the text then copy and paste it.

a. Depending on your version of windows and settings, you may need to click the top left corner of the console window (black screen) and then click “Edit/Mark” and use the mouse to select the output. Press the Enter key and your output goes to the clipboard. You can now paste it at the bottom of your program using Ctrl-V

What You Need to Submit into Crowdmark

1. Your modified code from Part 2 above

2. The output of your modified code with a test case of side_length = 8, copied and pasted into a /*block comment*/ at the bottom of your code. An example of the expected format is below.

```
1
2   /*
3    Ryan Consell
4
5   */
6
7   #include <iostream>
8   #include <cmath>
9   #include <cstdlib>
10
11  using namespace std;
12
13  int main()
14  {
15      cout<<"This is just an example for submission formatting purposes."<<endl;
16      cout<<"What is your name? ";
17      string your_name;
18      cin>> your_name;
19      cout<<"Hello "<< your_name;
20
21      return EXIT_SUCCESS;
22  }
23
24  /*
25  This is just an example for submission formatting purposes.
26  What is your name? Ryan
27  Hello Ryan
28  --------------------------------
29  Process exited after 4.292 seconds with return value 0
30  Press any key to continue . . .
31
32
33  */
```

Question 2

Problem Description

The conversion for a temperature given in Fahrenheit to Celsius is given by:

$$Temperature\ ℃ = \frac{5}{9} \times (Temperature\ °\mathrm{F} - 32)$$

Write a program that:

- Prompts the user for a temperature in Fahrenheit
- Outputs to the screen the temperature in Celsius

What You Need to Submit into Crowdmark

- Run the program for temperatures of 212, 100, 32, -40
- Submit your code with the output from the given test cases pasted at the bottom as a /*block comment*/

Question 3- Coin Withdrawal (C++)

Problem Description

Prompt the user to enter the amount of money that they would like to withdraw from the bank (for example $1.17). Note that $1 (dollar)= 100 ¢ (cents). The user only wants to withdraw coins (i.e. quarters (25 ¢), dimes (10 ¢), nickels (5 ¢) and pennies (1 ¢)). Return the minimum number of total coins needed for the amount of money they would like to withdraw. ***Note: Make sure that the amount of money that the user would like to withdraw from the bank is stored as a double.***

Write a Program that:

- Prompts the user for the amount of money that they would like to withdraw from the bank.
- Outputs to the screen the number of quarters, nickels, dimes and pennies.

What You Need to Submit into Crowdmark

- Run the program for withdrawal amounts of $1.17 and $5.24
- Submit your code with the output from the given test case pasted at the bottom as a /*block comment*/