CS 201 Homework 1

Solomon Himelbloom

September 11, 2020

- Repository Link: `https://github.com/techsolomon/cs201`
- Git Commits: `https://github.com/techsolomon/cs201/commits`
- This homework took approximately *5* hours to complete.

1 Design

This second homework assignment once again utilized a monorepo to store four directories under the *hw1* folder – the main program (diamond), two additional programs (dotcross and mileskm), and a practice folder (for extra programs, such as a draft of greatest). Every .cpp file starts with the author's name, the computer science class code, date file was created or last edited, and finally a short description. In addition, every directory has a Makefile.

The diamond shape was constructed using six for loops, each with a distinct purpose of drawing a subsection of the larger image. If you remove or reorder the loops, you will get an output either in the incorrect direction or orientation. Dotcross takes six items of user input and stores them as floats or floating-point decimals. We can use *setprecision(5)* to maintain up to five digits of accuracy. Mileskm, in addition to the previous programs, uses if/else if/else statements to control some of the printed text to avoid grammar issues with plural and singular values (example: mile vs. miles).

2 Post Mortem

The main, diamond project, was the most difficult and time-consuming to create, with dotcross and mileskm each following in a slightly easier fashion. Each project was created initially using a template directory, Makefile, and .cpp header. I committed the files often to my GitHub repository to track changes and stay organized.

In the future, I hope to understand specific types of errors to resolve issues faster and use appropriate comments and descriptive variable names for better readability. I overall enjoyed working through the multiple practice problems and would like to learn more about the optimal time to replace if/else statements with switches and the pros and cons of different types of loops.

3 Answers to Questions

1. The purpose of a compiler is to translate my written source code to another language – in this case, one that is readable by the computer. The stages from source code to executable are as follows: source code to preprocess to source code (preprocessed) to compile to object code to link to executable code (typically one file) to load and finally ending with the executing program.

2. Header files are where the C++ code is stored in preparation for the compiler. A compile-time error relates to errors that humans make that must be fixed before running the program. Linkers convert and connect multiple segments of object code into code that is easily executed. Finally, statements are small sections of a C++ program that must be run in a certain order or sequence.

3. The difference between a source file and an object file is that source files contain C++ code intending to be compiled; file extensions typically include the ".cpp" ending. Object

files, on the other hand, are where the compiled object code comes from. There may be references to other functions, but this code and respective files are not ready to be executed yet.

4. It is necessary to practice rather than having a false sense of "textbook understanding" as repetition helps cement a deeper understanding of computer science fundamentals. If you only learn from textbooks, you will be less prepared to deal with errors and real-world examples of problems throughout your programs.

5. The term prompt is used within an introduction to C++ as it is when you ask or request information from the user. This is directly related to input and output (I/O) as we shift information across multiple files and functions.

6. The shortcut "\n" is called an escape character for a line break and has the main purpose of adding a new line.

7. A variable is something that you want a computer to remember which can be recalled later. An object is a reference in computer science that acts as a classification storage device for sorting and organizing information. A literal represents fixed values in code that cannot be modified, typically standing alone, but still easily referenced later in your C++ code.

8. Different kinds of literals include Booleans, integers, floating points, strings, and characters.

9. Five examples of legal names that you shouldn't use include: cpp, cin, cout, endl, and include, as they are likely to confuse the compiler and future readers of the codebase.

10. A conversion from a double to int or vice versa is considered a bad thing as you lose precision and data with your newly calculated or printed values.

4 Program 1

4.1 Sample Output/Screenshot

Listing 1: Sample Program Output

```
Type in a positive integer: 7

      #
     ###
    #####
   #######
  #########
 ###########
#############
 ###########
  #########
   #######
    #####
     ###
      #
```

4.2 Git Commit Messages

Date	Message
2020-09-11	add: dotcross; move: practice/greatest; update: diamond
2020-09-11	update: diamond if/else logic and for loops
2020-09-10	add: hw1 file structure and project directory

4.3 Source Code

```cpp
// diamond.cpp
// Solomon Himelbloom
// 10 September 2020
// Diamond example for CS 201.

#include <iostream>
using std::cout;
using std::endl;
using std::cin;

int main() {
    int diamond_column, diamond_row, user_input;

    cout << "Type in a positive integer: ";
    cin >> user_input;

    if (user_input <= 0) {
        cout << "You typed in zero, a negative number, or a letter." << endl;
        cout << "Please try again with a positive integer." << endl;
    }

    else {
        for (diamond_row = 0; diamond_row <= user_input; diamond_row++) {
            for (diamond_column = 1; diamond_column <= user_input - diamond_row; diamond_column++) {
                cout << " ";
            }
            for (diamond_column = 1; diamond_column <= 2 * diamond_row - 1; diamond_column++) {
                cout << "#";
            }
            cout << endl;
        }

        for (diamond_row = user_input - 1; diamond_row >= 1; diamond_row--) {
            for (diamond_column = 1; diamond_column <= user_input - diamond_row; diamond_column++) {
                cout << " ";
            }
            for (diamond_column = 1; diamond_column <= 2 * diamond_row - 1; diamond_column++) {
                cout << "#";
            }
            cout << endl;
        }
    }
}
```

5 Program 2

5.1 Sample Output/Screenshot

Listing 2: Sample Program Output

```
Enter Ax: 1.1
Enter Ay: 1.2
Enter Az: 1.3
Enter Bx: 2.1
Enter By: 2.2
Enter Bz: 2.3

Dot product calculation: 7.94

Cross product (x): -0.1
Cross product (y): 0.2
Cross product (z): -0.1

A dot B = 7.94
A cross B = (-0.1, 0.2, -0.1)
```

5.2 Git Commit Messages

Date	Message
2020-09-11	add: user input, dot/cross product, and precision
2020-09-11	add: dotcross; move: practice/greatest; update: diamond
2020-09-10	add: hw1 file structure and project directory

5.3 Source Code

```cpp
// dotcross.cpp
// Solomon Himelbloom
// 11 September 2020
// Dot cross example for CS 201.

#include <iostream>
#include <iomanip>
using std::cout;
using std::endl;
using std::cin;

int main() {
    float Ax, Ay, Az, Bx, By, Bz;
    float dot_product = 0;
    float cross_x, cross_y, cross_z = 0;

    // User Input: 3 floating-point numbers for x, y, z { each for vector A and B.
    cout << "Enter Ax: ";
    cin >> Ax;

    cout << "Enter Ay: ";
    cin >> Ay;

    cout << "Enter Az: ";
    cin >> Az;

    cout << "Enter Bx: ";
    cin >> Bx;

    cout << "Enter By: ";
    cin >> By;

    cout << "Enter Bz: ";
    cin >> Bz;

    dot_product = (Ax * Bx + Ay * By + Az * Bz);

    cout << " " << endl;
    cout << "Dot product calculation: " << std::setprecision(5) << dot_product << endl;

    cross_x = (Ay * Bz - Az * By);
    cross_y = (Az * Bx - Ax * Bz);
    cross_z = (Ax * By - Ay * Bx);

    cout << " " << endl;
    cout << "Cross product (x): " << std::setprecision(5) << cross_x << endl;
    cout << "Cross product (y): " << std::setprecision(5) << cross_y << endl;
    cout << "Cross product (z): " << std::setprecision(5) << cross_z << endl;

    // Additional Printed Results [A dot B = answer; A cross B = (x, y, z)]:
    cout << " " << endl;
    cout << "A dot B = " << std::setprecision(5) << dot_product << endl;
    cout << "A cross B = (" << std::setprecision(5) << cross_x << ", " <<
        std::setprecision(5) << cross_y << ", " << std::setprecision(5) << cross_z << ")" << endl;
}
```

6 Program 3

6.1 Sample Output/Screenshot

Listing 3: Sample Program Output

```
Enter the number of miles: 75
75 miles is equal to 120.675 kilometers.
```

6.2 Git Commit Messages

Date	Message
2020-09-11	refactor: account for negative miles input
2020-09-11	add: miles to km converter program
2020-09-10	add: hw1 file structure and project directory

6.3 Source Code

```cpp
// mileskm.cpp
// Solomon Himelbloom
// 11 September 2020
// Miles to kilometers conversion for CS 201.

#include <iostream>
using std::cout;
using std::endl;
using std::cin;

int main() {
    float miles, kilometers;

    cout << "Enter the number of miles: ";
    cin >> miles;

    kilometers = miles * 1.609;

    if (miles == 1) {
        cout << miles << " mile is equal to " << kilometers << " kilometers." << endl;
    }

    else if (miles < 0) {
        cout << "Please enter a positive value for miles and try again." << endl;
    }

    else {
        cout << miles << " miles is equal to " << kilometers << " kilometers." << endl;
    }
}
```