Matthew Strong

Arcadia Zhang

Computer Science 1300

19 November 2017

Assignment 8

Music today is extremely accessible via the internet. Simply walking around the campus of CU Boulder, numerous people are plugged in, listening to music during the short time before the next 50 minutes (or longer) of classes. Headphones of all sorts are present on campus, some being the normal I-pod headphones, while others and foreign devices that are from another planet. In short, music is dominated by technology. A developing passion of mine has been the integration of music with computer science. The current horizon for such developments remains exceptionally high, but the goal of the assignment is to begin this long journey into a combination of two fields that I find very enjoyable. The algorithms I will use in the assignment are rooted in the math of sounds and music, but all of it can be done with coding in C++. This assignment, which I intend to be a project I develop over the years at this university, will be called CumuSic Program. With that, this version will be the first version, version 1.0. This project will be able to play musical notes (A, B, C#, Eb, etc.) with the implementation of the Beep function in C++. This program will be able to play basic scales, major and minor, in any key, and it will also be able to generate random music in different moods. Lastly, the program will be able to read in a simple file with written in notes and the amount of beats in order to play that melody (I have not been able to fully understand playing two beeps at the same time).

The goal of this project is to gain a more solid understanding of music through having to be able to code it in C++, which requires a solid understanding of C++ as well. I will have furthered my abilities in both C++ and music. This assignment, however, is in a long term goal of mine to be a master at programming music in C++.

Within my .h files, there are 4 classes that I declare. They will be elaborated in depth within this text.

The first class is the Readmusic class. This class is fairly straightforward; the goal here is to read music in from a file. Unfortunately, I will not be reading in notes from file, I will be reading the letters of notes and rests to translate into a certain piece of music. This music will then be played in a variety of ways, played how it is, up an octave, down an octave, and played at a faster pace. Using this part of the program is fairly simple.

I designed the class this way (as seen in the attached .h files) because I wanted to implement file i/o and for my program to see a note, like “A”, and convert that to a note that the user can actually hear. I designed the class with methods that return an integer of tempo or the octave, but the important aspect of this class is for the user to hear a sound produced by this program. I designed my class with several data members to supplement the methods and constructors. The first member is a string array, readpieces, that will store the name of each file (or composition) into an array for the users’ benefit. I also have a data member for tempo, which will be used, specifically, in the Beep function (with some math as well), so that the tempo allows the melody to be played perfectly. The data member, octave, gives the amount of octaves that the music will be played in. This has a mathematical implication that will be important later in the code, or the implementation of the code. The next two members are arrays, one that has a string array for each note played (such as “A”), while the other one is a float array that gives the corresponding frequency (in hertz), so that I can match up a note with its frequency and then play that sound. Also, I have a string note lengths (this might change) that will have the note lengths of each note, this will be reasonably challenging. To do this, I heavily implement arrays to a great extent. Essentially, I created arrays with all of the notes and their frequencies, then I go through each file and look at the note and length, and after searching for the note, I play that note with the length. I also have the ability to play the files up and down an octave with a different tempo; making these changes was fairly easy from the main constructor. One can run this program by declaring a class with type Readmusic, then going through whatever methods of the class that they want in the main. There are nearly 1,000 lines of code within the cpp file.

The second class is the Sadmusic class. The goal of the class is to generate random sad music created based on a specific key, which will be very challenging. I will implement the minor scale to do this and create some random music that will hopefully sound saddened and with despair. The goal of this class is to create music that will, overall, sound sad, and this will be based on math that will have to be done regarding scales. I design the class this way because I want the user to be able to enter in the desired minor scale that they want the whole melody to be based on, after that, then the user will have the option of hearing this music up an octave, faster, and be able to see that random notes that were created. The data members are: first, there are two arrays, one being string type, and the other being float type, which will be the notes that were created, along with their frequency in hertz and actual name. The next two data members give the letters and frequencies of the minor scale that will be a basis for the whole melody. The last two arrays are arrays that give all the notes that can be produced, so that the user can pull from these arrays based on the key they choose. I also have a tempo member. Also, after the proposal was made, I added in an array with the notes in a created motif as well as the frequencies of those notes. I have the madenotes as well which includes all of the randomly created notes except from the playCool method because that could get very, very large. What I do is similar to the Readmusic class so that in my constructor I create a mini library of all the notes. Then I can create a minor scale based on a specific key and then play randomly generated music from that key. I also have a vector of objects that are user-defined objects where the user can add them in. These objects are class Miscmusic. To use this program is easy; all one has to do is declare an instance of the class, and apply whatever methods they want. They can set the tempo, change the key, play random music up an octave, etc. Note that much of this theory is based on arrays and looking through the main arrays, looking to see if notes match; when they do, that’s where my music theory knowledge will come in. I will be able to explain much more during the interview as well.

The third class is Happymusic, which has the goal of creating randomly generated happy music, which is the opposite of the Sadmusic class. I designed this class in a near identical way to the Sadmusic, so it would be inefficient to elaborate on the methods and data members of the class. The only significant difference between the Sadmusic and Happymusic class is that the scale that is a basis for the Happymusic class is a major scale that will be drawn from the large arrays at the end of the .h file (refer to the .h file to understand what is meant). The members and methods are nearly identical, (I also have a vector of user defined objects) but the only major difference is the basis scale through which everything is played – which is the minor scale. Using this class requires the user to declare an instance of the class, and then apply whatever methods they those to this class in order to implement different concepts.

The last and fourth class is Miscmusic, which will perform miscellaneous tasks related to music theory. The goal of this class is to give to users, especially ones that are not as experienced in music theory as others, a basic view of musical concepts. For example, I design the class to be able to play major and minor scales, arpeggios, different note lengths, and a simple note exercise. These methods will be much easier than the methods from my previous classes. My data members will be a starting note that everything will be based on, and two arrays that contain, essentially, all of the notes I need for creating any music (some frequencies would produce sounds so high that it would not be feasible to consider them notes). I also have a tempo that the user will have to provide. This was the easiest class, but I did have some file i/o that reads in a file of the fundamentals that are to be played. These lines are read and the notes are played. Also, the difficulties of each exercise are written to a text file. This was the first class that I created because it, in of itself, was fundamental for understanding the application of music theory in C++. To use this class, one must declare an instance of the class; from there, they can read in some files and hear the fundamentals, or call the fundamental methods themselves. It is important to know what to put in the parameters for each method.

One aspect that all 4 classes have in common that I will mention now is that I have a getter and setter for the user desired tempo. The first and last classes mentioned have file IO as well where I will write to an output file and read in a file with details about music.

I will show that the design meets the requirements by going through the list of requirements and ensuring that all of them are met.

2+ user defined classes: Yes, I have 4, the Readmusic, Sadmusic, Happymusic, and Miscmusic class.

4+ data members per class: Yes, I explain them in the explanation of each object.

Appropriate methods for each class: Yes, I elaborated on this in the classes. I do include getters and setters.

Array of user Defined Objects – Yes, I have them in two of my classes.

Implementation for the 2 user defined classes must have:

2+ if-else statements: I will have that, especially in getting notes from arrays

2+ while loops: I will have that in the file IO but also in asking a question about what they want.

2+ for loops: Yes, to loop through enough times to create a melody.

File I/O: Yes, to read in files with music notes and write out to a new file

Ultimately, this project will be significantly challenging to work exactly as expected. As it stands, these current methods seem feasible and I feel quite comfortable with them considering that we have a lot of practice with for loops, file IO, while loops, and other techniques. This project seems small compared to what is currently being done with concepts such as machine learning with regarding to generating and composing music. However, as a freshman, I am glad that I am starting on this journey that I hope to continue with in my life.