CSC 150 Programming Project 1

Due 11:59 PM September 15

Fall, 2017

Assignment Overview

This assignment involves the coding and testing of a program that uses arithmetic expressions and `if` statements. The assignment must be submitted by 11:59 PM on September 15.

Background

One of the oldest problems of music is how to map the notes of a musical piece to a set of audio frequencies. There are various "tuning approaches" that provide slightly different ways of assigning notes to a particular frequency. This project will require that you write a program that does one particular kind of this mapping.

First, we must define some notation that can be used to specify musical notes. One common representation is to specify notes by their octave and pitch class (octpch notation). This notation represents each note as a number pair, where the first number indicates which octave the note belongs to and the second number specifies the note within the octave. Each octave is divided into 12 semitones (pitch classes) as shown in the following table:

Note	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
Pc	0	1	2	3	4	5	6	7	8	9	10	11

Octpch representations are often written in decimal format. For example, 5.9 is the fifth octave, ninth semitone. This provides a convenient method for specifying exactly which note is to be played.

In order to create the proper tone, we must convert the octpch representation to an audio frequency, which can be produced by the hardware. Each note has an associated frequency, which can be computed from the octpch representation. We start by choosing a reference note. We can arbitrarily choose octpch 4.9 (A in the fourth octave), which has a defined standard frequency of 440 Hz as our reference note. There is a very nice relationship between octaves. Each note in the next higher octave has a frequency that is twice the previous octave. For example, since we know that 4.9 $\rightarrow$ 440 Hz, we can infer that 5.9 $\rightarrow$ 880 Hz and 3.9 $\rightarrow$ 220 Hz. Our tuning system will assume that each of the semitones within an octave is equally spaced, that is the distance from one

semitone to the next is the same within the octave. This is called a Tempered Scale. The formula we use is:

$$f = x \times 2^{\omega + \frac{m}{12}}, \tag{1}$$

where x is the frequency (in Hz) for a reference note, ω is the difference (sign matters) between the octave of the required note and the octave of the reference note, and m is the difference (sign matters) between the reference semitone and semitone of the required note. Consider the value for octpch 0.0, which is C in the first octave. Octpch 0.0 is 4 octaves and 9 semitones below 4.9 (A in the fourth octave), so the frequency is computed as follows:

$$\begin{aligned} f &= 440 \times 2^{-4 + \frac{-9}{12}} \\ &= 440 \times 2^{-4.75} \\ &= 16.35159 \text{ Hz}. \end{aligned}$$

Problem Statement

Your program will prompt for three octpch pairs, convert each of them to Hertz (Hz), print out the values and then play them using the `Beep` function.

Program Specifications

1. Your program will output a brief, descriptive message when it first starts, indicating the purpose of the program, the input required, and the output that will be produced.
2. The program will then prompt for the three octpch pairs, each pair having two integers: the octave and the pitch class. It will prompt for the octave and pitch class separately, so you do not have to convert 5.9 into the appropriate octave and pitch.
3. The program will convert each octpch pair (octave and pitch class), yielding a floating point result (the frequency in Hz) for each of the three notes.
4. The program will print the frequency of each note, along with the original octave and pitch class values.
5. The program will then play the three notes using the `Beep` function as described at the end of this document.

Program notes

1. Note that the way to do exponentiation in C/C++ is via the `pow` (short for power) function. The `pow` function is part of the math library. You must include the proper header at the beginning of your program:
   ```
   #include <cmath>
   ```

2. the `pow` function has the following form:
   `pow(x,y)`
   where `x` is raised to the power `y`. The base (`x`) must be a floating point type, it cannot be an integer. The `pow` function returns the same data type as the base.

3. Once converted, you can print out the oct and pch values as well as the frequency value (in Hertz) for the pair. See the examples below.

4. The octave entered by the user must be between 0 and 10 inclusive, and the pitch class must be between 0 and 11 inclusive.

5. Your program must do error checking to ensure that the octave and pitch are valid values and that the resulting frequency is between 37 Hz and 30,000 Hz. If any value is not valid, your program must issue a message and terminate with a non-zero return code.

Examples

The following shows the input and output for a typical run for this program:

```
Enter the octave of the first note: 4
Enter the pitch class of the first note: 3
Enter the octave of the second note: 6
Enter the pitch class of the second note: 2
Enter the octave of the third note: 8
Enter the pitch class of the third note: 11
Note 1: 4.3 is 311.127 Hz
Note 2: 6.2 is 1174.66 Hz
Note 3: 8.11 is 7902.13 Hz
```

Note: The program must also produce the three tones through its speaker or headphones.

The following shows the input and output for a run where the first frequency is out of range:

```
Enter the octave of the first note: 1
Enter the pitch class of the first note: 2
Enter the octave of the second note: 3
Enter the pitch class of the second note: 4
Enter the octave of the third note: 8
Enter the pitch class of the third note: 5
Note 1: 1.2 is 36.7081 Hz
The frequency is out of range
```

Note: In this case, the program exits without producing any sound.

You may wish to download the file `program1_example.exe` and run it on your computer to obtain more examples to check your program against.

`Beep` Function

Microsoft Windows provides a sound function that allows you to play a frequency for a fixed period of time. It is in the Windows library. To access it, you need to include the proper header at the beginning of your program:

```
#include <windows.h>
```

The `Beep` function is defined as:

```
bool Beep(int Frequency, int Duration);
```

The `Frequency` parameter is the value calculated using the formula above *converted to an integer*. The `Duration` value is given in milliseconds (1000 milliseconds = 1 second). For example, `Beep(440,500)` would play A in the fourth octave for half a second. **Note:** You must pass the frequency parameter as an integer, or the compiler will emit a warning message. Points will be deducted for compiler warnings.

Comments and suggestions

- Do not delay, Start writing the program early. If you wait until the night before the due date, you will have a miserable night, and probably will not complete the assignment.
- Do not try and write the entire program all at one time. Work on the program in small sections.
- Debugging Hint: Make sure your program's interaction with the user matches the example given above. Then test other values that you have calculated by hand.
- Be sure to follow the coding style guidelines that can be found on the class website at `http://www.mcs.sdsmt.edu/csc150/Course`.
- Name your code file `prog1.cpp`. Points will be deducted if this naming convention is not followed.
- Your program must correctly compile in Visual C++ 2013.
- Be sure your code file is readable and neat. Do not allow lines to extend past 80 characters, use appropriate white space and make sure to use a consistent and attractive indentation scheme.

Program Submission

Submit your program code (the `prog1.cpp` file only) at `http://www.mcs.sdsmt.edu/submit` before midnight of the due date. (Your file gets time stamped, so late submissions will be noted and will be given a late penalty!) Be sure to submit to the correct lecture section!

DO YOUR OWN WORK.