```
/*
  Copyright 2015 Fred Martin,
  Y. Rykalova, 2020
  J. Daly 2022

  Edited by Anson Cheang 2022
  essentially allows the user to play
  a unique sound from 37 different keys.
  the function automatically sets up the sounds
*/

#include "CircularBuffer.h"
#include "StringSound.h"

#include <math.h>
#include <limits.h>

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include <vector>

#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>
#include <SFML/Window.hpp>

#define CONCERT_A 220.0
#define SAMPLES_PER_SEC 44100

// using namespace std;

std::vector<sf::Int16> makeSamples(StringSound& gs) {
    std::vector<sf::Int16> samples;

    gs.pluck();
    int duration = 8;  // seconds
    int i;
    for (i= 0; i < SAMPLES_PER_SEC * duration; i++) {
        gs.tic();
        samples.push_back(gs.sample());
    }

    return samples;
}

int main() {
    sf::RenderWindow window(sf::VideoMode(300, 200),
     "SFML Plucked String Sound Lite");
    sf::Event event;
    char c;
    std::vector<std::unique_ptr<sf::Sound> > kSounds;
    std::vector<sf::Int16> samples;
    std::vector<std::vector<sf::Int16>> KSample;
    std::vector<std::unique_ptr<sf::SoundBuffer> > kBuffer;
    sf::Sound sound;
    sf::SoundBuffer buffer;
    std::string keys = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/' ";

    auto func = [=] (int i) {
        std::vector<sf::Int16> samples;
        const double freq = 440.0 * pow(2.0, (i-24.0)/12.0);
        StringSound gs1(freq);
        samples = makeSamples(gs1);
```

```cpp
        return samples;
    };

    for (int i = 0; i < 37; i++) {
        KSample.push_back(func(i));
        // samples = KSample[i];
        // std::cout << samples.size() << std::endl;
    }

    for (size_t i = 0; i < KSample.size(); i++) {
        if (!buffer.loadFromSamples(&(KSample[i].at(i)), KSample[i].size(),
        2, SAMPLES_PER_SEC))
            throw std::runtime_error(
                "sf::SoundBuffer: failed to load from samples.");
        kBuffer.push_back(std::make_unique<sf::SoundBuffer>(buffer));
    }

    for (size_t i = 0; i < kBuffer.size(); i++) {
        sound.setBuffer(*kBuffer[i]);
        kSounds.push_back(std::make_unique<sf::Sound>(sound));
    }

    /* freq = CONCERT_A * pow(2, 3.0/12.0);
    StringSound gs2(freq);
    sf::Sound sound2;
    sf::SoundBuffer buf2;
    samples = makeSamples(gs2);
    std::cout << samples.size() << std::endl;
    if (!buf2.loadFromSamples(&samples[0], samples.size(), 2, SAMPLES_PER_SEC))
        throw std::runtime_error(
            "sf::SoundBuffer: failed to load from samples.");
    sound2.setBuffer(buf2); */
    int j = 0;
    while (window.isOpen()) {
        while (window.pollEvent(event)) {
            switch (event.type) {
            case sf::Event::Closed:
                window.close();
                break;

            case sf::Event::TextEntered:
                /*switch (event.key.code) {
                case sf::Keyboard::A:
                    // sound1.play();
                    break;
                case sf::Keyboard::C:
                    // sound2.play();
                    break;
                default:
                    break;
                }*/
                c = static_cast<char>(event.text.unicode);
                while (j < static_cast<int>(keys.size()) && c != keys[j]) {
                    j++;
                }
                if (j == static_cast<int>(keys.size())) {
                    throw std::runtime_error("wrong keys");
                }
                kSounds[j]->play();
                j = 0;
                break;

            default:
                break;
            }

            window.clear();
            window.display();
        }
    }
    return 0;
}
```