Assignment 12

Hardware-oriented Programming (62557)

Mads Richardt (s224948)

December 2, 2022

Contents

musciBox.ino file

```
/*
Course: Hardware oriented programming
Assignment nr.: 12
Student Name: Mads Richardt
Student id.: s224948
Date: 2022-12-02
*/

#include "musicBox.h"

#define BUZZER_PIN 2

StateMachine myMusicBox;

// Mester Jakob
const char jakob[] = "t10,f4,g4,a4,f4,f4,g4,a4,f4,a4,h-4,c>8,a4,h-4,c>8,"
                     "c>2,d>2,c>2,h-2,a4,f4,c>2,d>2,c>2,h-2,a4,f4,f4,c4,f8,f4,c4,f8,";

// Olsenbanden
const char olsen[] = "t8,d4,e-4,d2,e-4,f10,p8,d4,e-4,f2,g4,a-10,p8,g4,a-4,g4,"
                     "a-2,h-4,h-4,h-2,a4,g4,f4,f4,f-4,e-4,d4,f2,f4,g2,f4,e-16,p4,e-2,e-4,f2,e-4,"
                     "d16,p4,f2,f4,g2,f4,e-16,p4,e-2,e-4,f2,e-4,d32,c>4,c#>4,d>4,e>4,f>4,c>4,p8,"
                     "p4,c>2,c>4,c->2,h-4,a16,c>4,c#>4,d>4,e>4,f>4,c>4,p8,p4,c>2,c>4,d>2,e>4,f>16,"
                     "d4,e-4,d2,e-4,f10,p8,d4,e-4,f2,g4,a-10,p8,g4,a-4,g4,a-2,h-4,h-4,h-2,a4,g4,f4,"
                     "f4,f-4,e-4,d4,f2,f4,g2,f4,e-16,p4,e-2,e-4,f2,e-4,d16,p4,f2,f4,g2,f4,e-16,p4,"
                     "f2,f4,g2,a4,h-4,p4,h-<4,p4,";

// Marseillaisen
```

```
const char marseille[] = "t20,d1,d3,d1,g4,g4,a4,a4,d>6,h2,g2,p1,g1,h3,g1,"
                         "e4,c>8,a3,f#1,g8,p3,g1,g3,a1,h4,h4,h4,c>3,h1,h4,a4,p4,a3,h1,c>4,c>4,c>4,d>3
                         "c>1,h8,p4,d>3,d>1,d>4,h3,g1,d>4,h3,g1,d8,p3,d1,d3,f#1,a8,c>4,a3,f#1,"
                         "a4,g4,f8,e4,g3,g1,g4,f#3,g1,a8,p4,p2,a2,h-6,h-2,h-2,h-2,c>2,d>2,a12,h-2,a2,
                         "g6,g2,g2,h-2,a2,g2,g4,f#2,p8,p1,d>1,d>11,d>1,h3,g1,a12,p3,d>1,d>11,d>1,h3,g
                         "a10,p2,d4,g8,p4,a4,h8,p8,c>8,d>4,e>4,a10,p2,e>4,d>11,h1,c>3,a1,g12,p4,";
// Sound of silence
const char silence[] = "t30,p2,d1,d1,f1,f1,a1,a1,g8,p1,c1,c1,c1,e1,e1,g1,g1,"
                       "f8,p1,f1,f1,f1,a1,a1,c>1,c>1,d>4,c>4,p2,f1,f1,a1,a1,c>1,c>1,d>4,c>4,p2,f1,f1,
                       "d>1,d>5,d>1,e>1,f>1,f>3,e>1,d>3,c>6,d>1,c>1,a8,p1,f1,f1,f1,c>6,p1,e1,f1,d7,";

void setup()
{
    pinMode(BUZZER_PIN, OUTPUT);
    myMusicBox.speaker = BUZZER_PIN;
}

void loop()
{
    myMusicBox.play(marseille);
    myMusicBox.play(silence);
    myMusicBox.play(olsen);
    myMusicBox.play(jakob);
}
```

musicBox.h file

```
/*
Course: Hardware oriented programming
Assignment nr.: 12
Student Name: Mads Richardt
Student id.: s224948
Date: 2022-12-02
*/

#if !defined(MUSIC_BOX_H)
#define MUSIC_BOX_H

#include <Arduino.h>
#include <AceCommon.h>

// table of frequencies
const unsigned int freq[36] = {
    // c  d  e  f  g  a  h
    131, 139, 147, 156, 165, 175, 185, 196, 208, 220, 233, 247, // low octave
    262, 277, 294, 311, 330, 349, 370, 392, 415, 440, 466, 494, // default octave
    523, 554, 587, 622, 659, 698, 740, 784, 831, 880, 932, 988  // high octave
};

// Name states for state machine
enum States
```

```
{
    Start, // start or after comma
    T1,    // after 't': interpret timebase
    T2,    // after T1 and at least one digit: calculate tempo
    T3,    // after T2 and comma: save tempo
    N1,    // after note a-h or #-<>: Interpret note
    N2,    // after N1 and at least one digit: Calculate duration
    N3,    // after N2 and comma: Play note
    P1,    // after 'p'. Interpret pause
    P2,    // after P1 and at least one digit: Calculate duration
    P3,    // after P2 and comma: Plau pause
    Stop,  // finished
    Error  // syntax error
};
// The class StateMachine defines a state machine for interpreting a music string
class StateMachine
{
public:
    int play(const char *tune); // play the tune protected:
    void SStart();              // start state
    void ST1();                 // state T1: interpret tempo
    void ST2();                 // state T2: calculate tempo
    void ST3();                 // state T3: save tempo
    void SP1();                 // state P1: interpret pause
    void SP2();                 // state P2: calculate length of pause
    void SP3();                 // state P3: play pause
    void SN1();                 // state N1: interpret note
    void SN2();                 // state N2: calculate length of note
    void SN3();                 // state N3: play note
    void SError();              // state error
    void SStop();
    int findIndex(int);
    States state;    // state in syntax parsing
    int timebase;    // time unit
    int pitch;       // note to play
    int duration;    // duration of note or pause
    const char *p;   // position in tune string
    char c;          // current character in tune string
    int speaker = 2; // Pin connected to buzzer on arduino
};

#endif // MUSIC_BOX_H
```

musicBox.cpp file

```
/*
Course: Hardware oriented programming
Assignment nr.: 12
Student Name: Mads Richardt
Student id.: s224948
Date: 2022-12-02
```

```
*/

#include "musicBox.h"

// function to play a tune
int StateMachine::play(const char *tune)
{
    p = tune; // pointer to tune string timebase = 20; // default time base
    timebase = 20;
    state = Start; // start in state Start
    c = *p;        // read first character
    p++;           // advance pointer for reading next character
    // run state machine
    while (state != Stop)
    {
        switch (state)
        {
        case Start:
            SStart();
            break; // Read next item SStart(); break;
        case T1:
            ST1();
            break; // interpret timebase ST1(); break;
        case T2:
            ST2();
            break; // state T2: calculate timebase ST2(); break;
        case T3:
            ST3();
            break; // state T3: save timebase ST3(); break;
        case P1:
            SP1();
            break; // state P1: interpret pause SP1(); break;
        case P2:
            SP2();
            break; // state P2: calculate length of pause SP2(); break;
        case P3:
            SP3();
            break; // state P3: play pause SP3(); break;
        case N1:
            SN1();
            break; // state N1: interpret note SN1(); break;
        case N2:
            SN2();
            break; // state N2: calculate length of note SN2(); break;
        case N3:
            SN3();
            break; // state N3: play note SN3(); break;
        case Error:
            SError();
            return 1; // state error SError();
        case Stop:
            return 0;
```

```
        default:
            return 0;
            break;
        } // end switch (state)
    }     // end while
}
// Functions for each state:
void StateMachine::SStart()
{ // start state
    switch (c)
    {
    case 't': // interpret timebase
        state = T1;
        break;
    case 'p': // pause
        state = P1;
        break;
    case 'a': // note a - h
    case 'b':
    case 'c':
    case 'd':
    case 'e':
    case 'f':
    case 'g':
    case 'h':
        state = N1;
        break;
    case ',':  // ignore comma break;
    case '\0': // end of string
        state = Stop;
        break;
    default: // anything else should give an error state = Error;
        state = Error;
        break;
    }
}

void StateMachine::ST1()
{
    c = *p;
    timebase = (int)c - 48;
    p++;
    c = *p;
    if (c == ',')
    {
        state = T3;
    }
    else
    {
        state = T2;
    }
}
```

```
void StateMachine::ST2()
{

    timebase = timebase * 10 + (int)c - 48;
    p++;
    c = *p;
    if (c == ',')
    {
        state = T3;
    }
}

void StateMachine::ST3()
{
    p++;
    c = *p;
    state = Start;
}

void StateMachine::SN1()
{
    size_t index;

    switch (c)
    {
    case 'c':
        pitch = freq[12];
        break;
    case 'd':
        pitch = freq[14];
        break;
    case 'e':
        pitch = freq[16];
        break;
    case 'f':
        pitch = freq[17];
        break;
    case 'g':
        pitch = freq[19];
        break;
    case 'a':
        pitch = freq[21];
        break;
    case 'h':
        pitch = freq[23];
        break;
    case '-':
        index = ace_common::binarySearch(freq, 36, (const unsigned int)pitch);
        pitch = freq[index - 1];
        break;
    case '#':
```

```
            index = ace_common::binarySearch(freq, 36, (const unsigned int)pitch);
            pitch = freq[index + 1];
            break;
        case '<':
            index = ace_common::binarySearch(freq, 36, (const unsigned int)pitch);
            pitch = freq[index - 12];
            break;
        case '>':
            index = ace_common::binarySearch(freq, 36, (const unsigned int)pitch);
            pitch = freq[index + 12];
            break;
        default:
            break;
        }
        p++;
        c = *p;
        if (isDigit(c))
        {
            duration = 0;
            state = N2;
        }
}

void StateMachine::SN2()
{
        if (duration == 0)
        {
            duration = (int)c - 48;
        }
        else
        {
            duration = duration * 10 + (int)c - 48;
        }
        p++;
        c = *p;
        if (c == ',')
        {
            state = N3;
        }
}

void StateMachine::SN3()
{
        tone(speeker, pitch);
        delay(duration * 7 * timebase);
        noTone(speeker);
        delay(duration * 1 * timebase);

        p++;
        c = *p;
        state = Start;
}
```

```
void StateMachine::SP1()
{
    duration = 0;
    state = P2;
    p++;
    c = *p;
    state = P2;
}

void StateMachine::SP2()
{
    if (duration == 0)
    {
        duration = (int)c - 48;
    }
    else
    {
        duration = duration * 10 + (int)c - 48;
    }
    p++;
    c = *p;
    if (!isDigit(c))
    {
        state = P3;
    }
}

void StateMachine::SP3()
{
    delay(duration * 8 * timebase);
    p++;
    c = *p;
    state = Start;
}

void StateMachine::SError()
{
    exit(1);
}
```