게임 프로그래밍 발표

2021863030 박세희

2023.10.11

목차

원본 소스 9_1_1

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <windows.h>

void draw_check02(int c, int r);
void gotoxy(int x, int y);
void display_piano_keyboard(void);
void touch_keyboard(int code);
void display_manual(void);
void practice_piano(void);
int calc_frequency(int octave, int inx);  //음계의 주파수 계산

int main(void)
{
        display_manual();
        practice_piano();
        return 0;
}
```

```c
void practice_piano(void)
{
        int index[]={0, 2, 4, 5, 7, 9, 11, 12};
        int freq[8], code, i;
        for(i=0;i<8;i++)
                freq[i]=calc_frequency(4, index[i]); //주파수계산
        draw_check02(8, 2);
        display_piano_keyboard();  //화면에 건반표시
        do
        {
                code=getch();
                if ('1'<=code && code<='8')
                {
                        code-=49;
                        touch_keyboard(code);  //누른 건반에 ▲표시
                        Beep(freq[code],300);
                        display_piano_keyboard(); //화면에 건반표시
                }
        }while(code!=27);
}
```

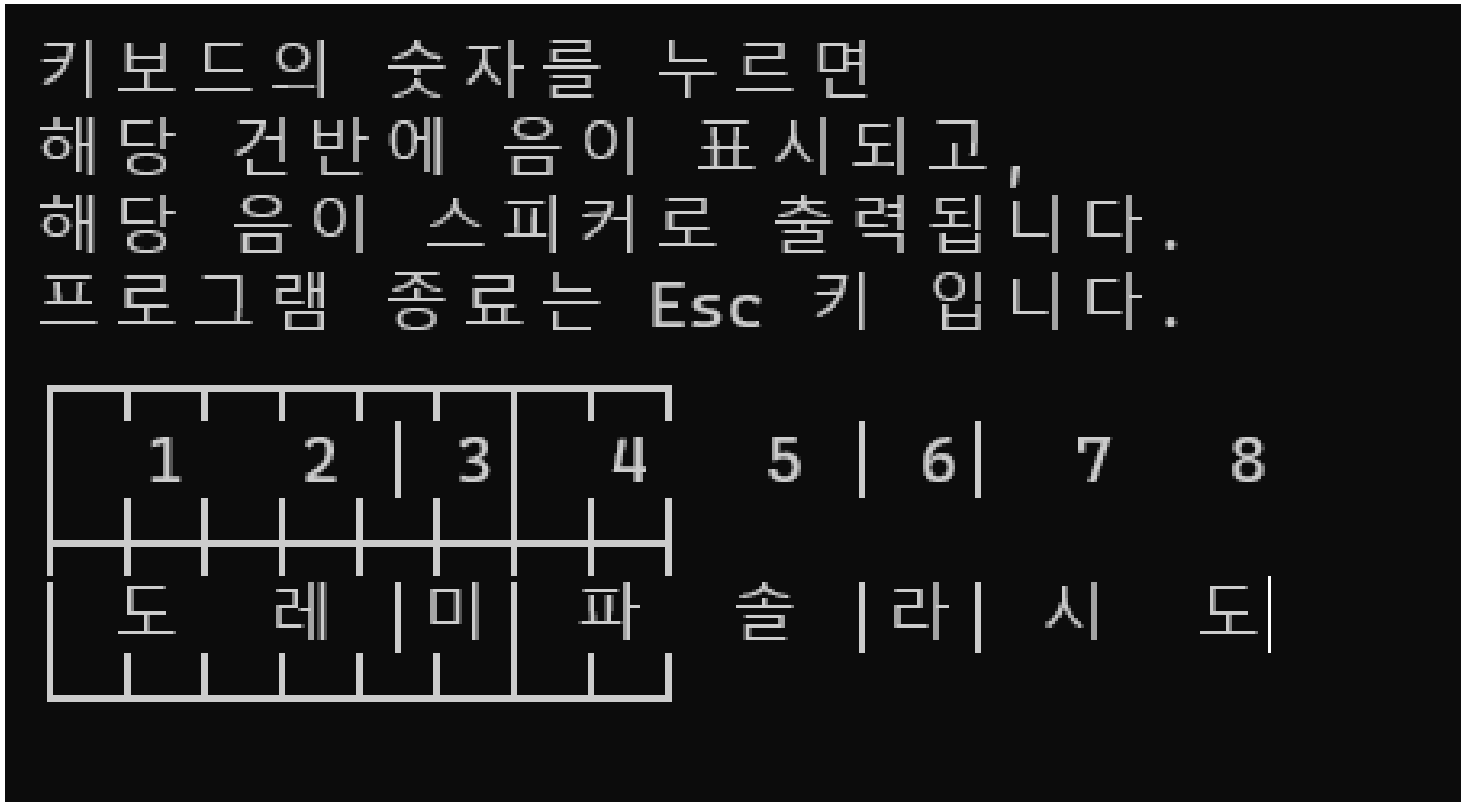
원본 소스 9_1_1

```
void draw_check02(int c, int r)
{
    int i, j;
    unsigned char a=0xa6;
    unsigned char b[12];
    for(i=1;i<12;i++)
        b[i]=0xa0+i;
    printf("%c%c",a, b[3]);
    for(i=0;i<c-1;i++)
    {
        printf("%c%c", a, b[1]);
        printf("%c%c", a, b[8]);
    }
    printf("%c%c", a, b[1]);
    printf("%c%c", a, b[4]);
    printf("\n");
    for(i=0;i<r-1;i++)
    {
        printf("%c%c", a, b[2]);
        for(j=0;j<c;j++)
        {
                printf("  ");
                printf("%c%c",a, b[2]);
        }
```

```
void display_piano_keyboard(void)
{
        int i;
        char code[8][4]={"도","레","미","파","솔","라","시","도"};
        for(i=0;i<8;i++)
        {
                gotoxy(3+i*4,6);
                printf("%2d", i+1);
        }
        for(i=0;i<8;i++)
        {
                gotoxy(3+i*4,8);
                printf("%s", code[i]);
        }
}
void touch_keyboard(int code)
{
        gotoxy(3+code*4,8);
        printf("%c%c", 0xa1, 0xe3);
}
```

```
int calc_frequency(int octave, int inx)
{
        double do_scale=32.7032;
        double ratio=pow(2., 1/12.), temp;
        int i;
        temp=do_scale*pow(2, octave-1);
        for(i=0;i<inx;i++)
        {
                temp=(int)(temp+0.5);
                temp*=ratio;
        }
        return (int) temp;
}
void gotoxy(int x, int y)
{
   COORD Pos = {x - 1, y - 1};
   SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
}
```

실행 화면


키보드의 숫자를 누르면
해당 건반에 음이 표시되고,
해당 음이 스피커로 출력됩니다.
프로그램 종료는 Esc 키 입니다.
1 2 3 4 5 6 7 8
도 레 미 파 솔 라 시 도

MIDI 란?

악기 디지털 인터페이스 (Music Instrument Digital Interface)의 약자로,
전자 악기끼리 디지털 신호를 주고 받기 위해 각 신호를 규칙화한 일종의 규약

악기와 악기끼리 주고받을 수 있는 언어와 통로의 신호 체계 표준

MIDI Library 연결

```c
#include <stdio.h>
#include <windows.h>
#include <mmsystem.h>

#pragma comment(lib, "winmm.lib")
#pragma pack(push, 1)

#define NKEY        29
#define NINSTRUMENT 128
#define NNOTE       128
#define NVOLUME     128

typedef struct {
    BYTE byteState;
    BYTE byteNote;
    BYTE byteVelocity;
    BYTE byteNULL;
} MIDIShortMSG_ST;

typedef union {
    DWORD dwMIDIdata;
    MIDIShortMSG_ST stMIDIData;
} MIDIShortMSG;
```

```c
#pragma pack(pop)

void CALLBACK MidiOutProc(HMIDIOUT hmMidiDevice, UINT umsg, DWORD Instance, DWORD Param1, DWORD Param2) {
}

void MIDIOutputError(MMRESULT mmResult) {
    char szErrMsg[129];
    midiInGetErrorText(mmResult, (char*)szErrMsg, sizeof(szErrMsg));
    MessageBox(0, szErrMsg, "Midi Error!", MB_OK);
}

long MIDIGetDevID(HMIDIOUT hMidiDevice) {
    UINT uDeviceID;
    MMRESULT mmResult = midiOutGetID(hMidiDevice, &uDeviceID);

    if (mmResult != MMSYSERR_NOERROR) {
        MIDIOutputError(mmResult);
        return -1;
    }

    return (long)uDeviceID;
}
```

```c
void MIDIGetDevCaps(HMIDIOUT hMidiDevice, MIDIOUTCAPS* Caps) {
    long lDeviceID;
    MMRESULT mmResult;
    lDeviceID = MIDIGetDevID(hMidiDevice);
    if (lDeviceID < 0)
        return;

    mmResult = midiOutGetDevCaps((UINT)lDeviceID, Caps, sizeof(MIDIOUTCAPS));

    if (mmResult != MMSYSERR_NOERROR) {
        MIDIOutputError(mmResult);
    }
}
```

MIDI Library 연결

```
HMIDIOUT MIDIOpen(WORD wMidiNum) {

    WORD wMidiMax = 0;
    MMRESULT mmResult = 0;
    HMIDIOUT hMidiDevice = NULL;

    wMidiMax = midiInGetNumDevs();

    if (wMidiNum >= wMidiMax) wMidiNum = 0;

    mmResult = midiOutOpen(&hMidiDevice, wMidiNum, (DWORD_PTR)(MidiOutProc),
        (DWORD)NULL, CALLBACK_FUNCTION);

    if (mmResult != MMSYSERR_NOERROR) {
        MIDIOutputError(mmResult);
        return NULL;
    }

    return hMidiDevice;
}
```

```
LRESULT MIDIClose(HMIDIOUT hMidiDevice) {
    MMRESULT mmResult;
    mmResult = midiOutClose(hMidiDevice);

    if (mmResult != MMSYSERR_NOERROR) {
        MIDIOutputError(mmResult);
        return FALSE;
    }

    return TRUE;
}

void MIDISendShortMsg(HMIDIOUT hMidiDevice, BYTE byteState, BYTE byteNote, BYTE byteValo) {
    MIDIShortMSG sMsg;

    sMsg.stMIDIData.byteVelocity = byteValo;
    sMsg.stMIDIData.byteNote = byteNote;
    sMsg.stMIDIData.byteState = byteState;
    sMsg.stMIDIData.byteNULL = 0;

    midiOutShortMsg(hMidiDevice, sMsg.dwMIDIdata);
}

void MIDIALLChannelSoundOff(HMIDIOUT hMidiDevice) {
    BYTE channel;

    for (channel = 0; channel < 16; channel++) {
        MIDISendShortMsg(hMidiDevice, (BYTE)(0xB0 + channel), 0x78, 0);
    }
}
```

gotoxy(int x, int y)

```
void gotoxy(int x, int y) {
    COORD pos;
    pos.X = x;
    pos.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), pos);
}
```

setColor

```
void setColor(unsigned short text, unsigned short back) {
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), text | (back << 4));
}
```

Main()

```
int main() {
    int loop;
    BYTE key = 0;
    BYTE instrument = 0;
    BYTE volume = 127;
    BYTE octave = 48;
    BYTE velocity = 120;

    HMIDIOUT hMidiDevice;

    BYTE pianoKey[NKEY] = {
        0x5A, 0x53, 0x58, 0x44, 0x43,
        0x56, 0x47, 0x42, 0x48, 0x4E, 0x4A, 0x4D,
        0x51, 0x32, 0x57, 0x33, 0x45,
        0x52, 0x35, 0x54, 0x36, 0x59, 0x37, 0x55,
        0x49, 0x39, 0x4F, 0x30, 0x50
    };

    BYTE pianoKeyOnOff[NKEY] = { 0 };
```

```
BYTE pianoKeyOnOff[NKEY] = { 0 };

char instName[][NINSTRUMENT] = {
"Acoustic Grand", "Bright Acoustic", "Electric Grand", "Honky-Tonk", "E
"Music Box", "Vibraphone", "Marimba", "Xylophone", "Tubular Bells", "Du
"Harmonica", "Tango Accordion" "Acoustic Guitar(nylon)", "Acoustic Guit

hMidiDevice = MIDIOpen(0);

if (hMidiDevice == NULL) {
    return 0;
}
```

Main()

```
printf("\n");
setColor(16, 15);
printf("좌우 키를 누르면 악기가 바뀝니다.                                    \n");
printf("상하 키를 누르면 옥타브가 바뀝니다.                                  \n");
printf("+ 키를 누르면 볼륨이 오릅니다.                                       \n");
printf("-키를 누르면 볼륨이 줄어듭니다.                                      \n");
gotoxy(0, 5);
printf("                                                                    ");
gotoxy(1, 5);
printf("INSTRUMENT : %-24s  OCTAVE : %03d   VOLUME : %03d\n", instName[0], octave, volume);
//setColor(16, 15);
puts("┌─┬─┬┬─┬─┬─┬┬─┬┬─┬─┬─┬┬─┬─┬─┬┬─┬┬─┬─┬┬─┬─┬┬─┬┐  ");
puts("| |  ||  | | |  ||  ||  |  | |  ||  | | |  ||  ||  |  | |  ||  | |  ");
puts("| |  ||  | | |  ||  ||  |  | |  ||  | | |  ||  ||  |  | |  ||  | |  ");
puts("| |S ||D | | |G ||H ||J |  | |2 ||3 | | |5 ||6 ||7 |  | |9 ||0 | |  ");
puts("| |  ||  | | |  ||  ||  |  | |  ||  | | |  ||  ||  |  | |  ||  | |  ");
puts("| └─┘└─┘ | └─┘└─┘└─┘  |  └─┘└─┘ | └─┘└─┘└─┘  |  └─┘└─┘ |  ");
puts("| Z | X |C | V | B | N | M | Q | W |E | R | T | Y | U | I | O |P |  ");
puts("└───┴───┴──┴───┴───┴───┴───┴───┴───┴──┴───┴───┴───┴───┴───┴──┴──┘  ");
puts("ESC 버튼을 누르면 종료됩니다.                                      \n");

MIDIALLChannelSoundOff(hMidiDevice);

MIDISendShortMsg(hMidiDevice, 0xB0, 7, volume);
```

Main() – GetKeyState

```
else if (GetKeyState(VK_LEFT) < 0) {
    if (instrument != 0)
        instrument -= 1;

    MIDISendShortMsg(hMidiDevice, 0xC0, instrument, 0);
    gotoxy(1, 5);
    printf("INSTRUMENT : %-25s", instName[instrument]);
    Sleep(200);
}
else if (GetKeyState(VK_UP) < 0) {
    if (octave < (NNOTE - NKEY))
        octave += 12;

    gotoxy(40, 5);
    printf("OCTAVE : %03d", octave);
    Sleep(200);
}
else if (GetKeyState(VK_DOWN) < 0) {
    if (octave != 0)
        octave -= 12;

    gotoxy(40, 5);
    printf("OCTAVE : %03d", octave);
    Sleep(200);
}
```

```
loop = 1;
while (loop) {
    if (GetKeyState(VK_ESCAPE) < 0)
        loop = 0;
    else if (GetKeyState(VK_RIGHT) < 0) {
        if (instrument < (NINSTRUMENT - 1)) {
            instrument += 1;

            MIDISendShortMsg(hMidiDevice, 0xC0, instrument, 0);
            gotoxy(1, 5);
            printf("INSTRUMENT : %-24s", instName[instrument]);
            Sleep(200);
        }
    }
    else if (GetKeyState(VK_LEFT) < 0) {
        if (instrument != 0)
            instrument -= 1;

        MIDISendShortMsg(hMidiDevice, 0xC0, instrument, 0);
        gotoxy(1, 5);
        printf("INSTRUMENT : %-25s", instName[instrument]);
        Sleep(200);
    }
```

Main() – GetKeyState

```
else if (GetKeyState(VK_UP) < 0) {
    if (octave < (NNOTE - NKEY))
        octave += 12;

    gotoxy(40, 5);
    printf("OCTAVE : %03d", octave);
    Sleep(200);
}
else if (GetKeyState(VK_DOWN) < 0) {
    if (octave != 0)
        octave -= 12;

    gotoxy(40, 5);
    printf("OCTAVE : %03d", octave);
    Sleep(200);
}
```

```
else if (GetKeyState(VK_OEM_PLUS) < 0) {
    if (volume < (NVOLUME - 1))
        volume += 5;

    MIDISendShortMsg(hMidiDevice, 0xB0, 7, volume);
    gotoxy(56, 5);
    printf("VOLUME : %03d", volume);
    Sleep(30);
}
else if (GetKeyState(VK_OEM_MINUS) < 0) {
    if (volume != 0)
        volume -= 5;

    MIDISendShortMsg(hMidiDevice, 0xB0, 7, volume);
    gotoxy(56, 5);
    printf("VOLUME : %03d", volume);
    Sleep(30);
}
```

Main() – GetKeyState

```
    else {
        for (key = 0; key < NKEY; key++) {
            if (GetKeyState(pianoKey[key]) < 0) {
                if (pianoKeyOnOff[key] == 0) {
                    if (pianoKeyOnOff[key] == 0) {
                        pianoKeyOnOff[key] = 1;
                        MIDISendShortMsg(hMidiDevice, 0x90, (BYTE)(octave + key), velocity);
                    }
                }
            }
        }
    }
    for (key = 0; key < NKEY; key++) {
        if (!(GetKeyState(pianoKey[key]) < 0)) {
            if (pianoKeyOnOff[key] != 0) {
                pianoKeyOnOff[key] = 0;
                MIDISendShortMsg(hMidiDevice, 0x80, (BYTE)(octave + key), velocity);
            }
        }
    }
}
```

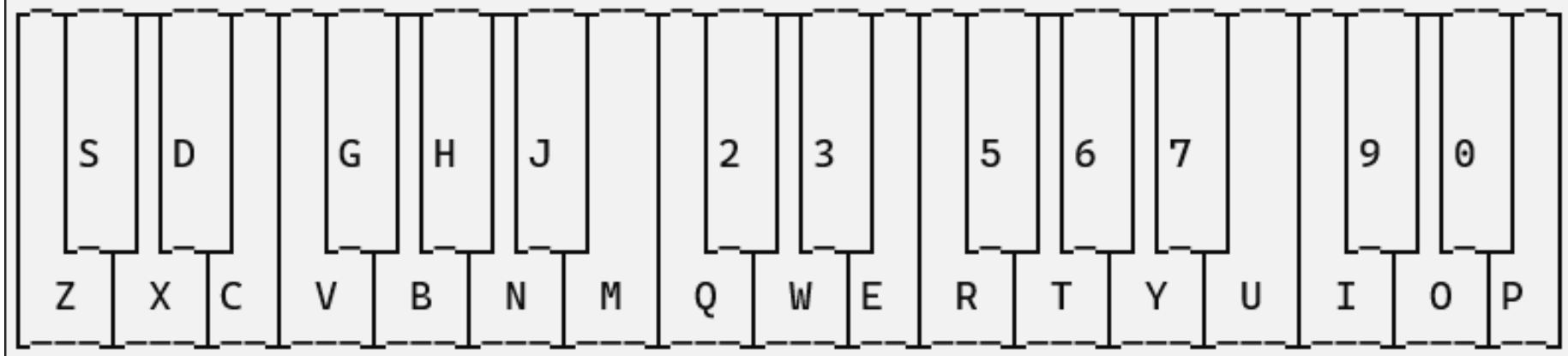
```
    for (key = 0; key < NKEY; key++) {
        if (!(GetKeyState(pianoKey[key]) < 0)) {
            if (pianoKeyOnOff[key] != 0) {
                pianoKeyOnOff[key] = 0;
                MIDISendShortMsg(hMidiDevice, 0x80, (BYTE)(octave + key), velocity);
            }
        }
    }

    MIDIALLChannelSoundOff(hMidiDevice);
    MIDIClose(hMidiDevice);

    return 0;
}
```

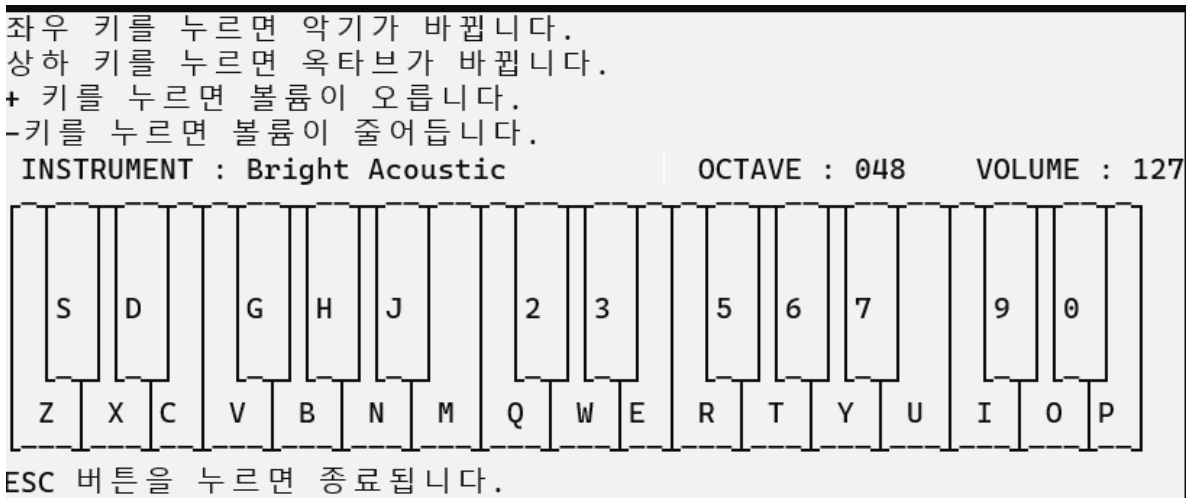
실행 화면

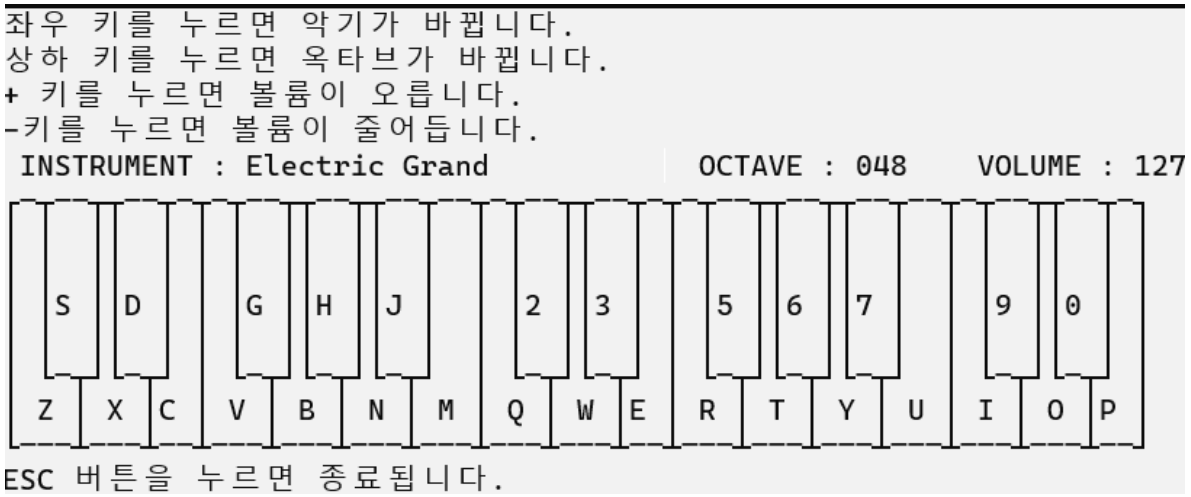
```
좌우 키를 누르면 악기가 바뀝니다.
상하 키를 누르면 옥타브가 바뀝니다.
+ 키를 누르면 볼륨이 오릅니다.
-키를 누르면 볼륨이 줄어듭니다.
 INSTRUMENT : Acoustic Grand              OCTAVE : 048    VOLUME : 127
```



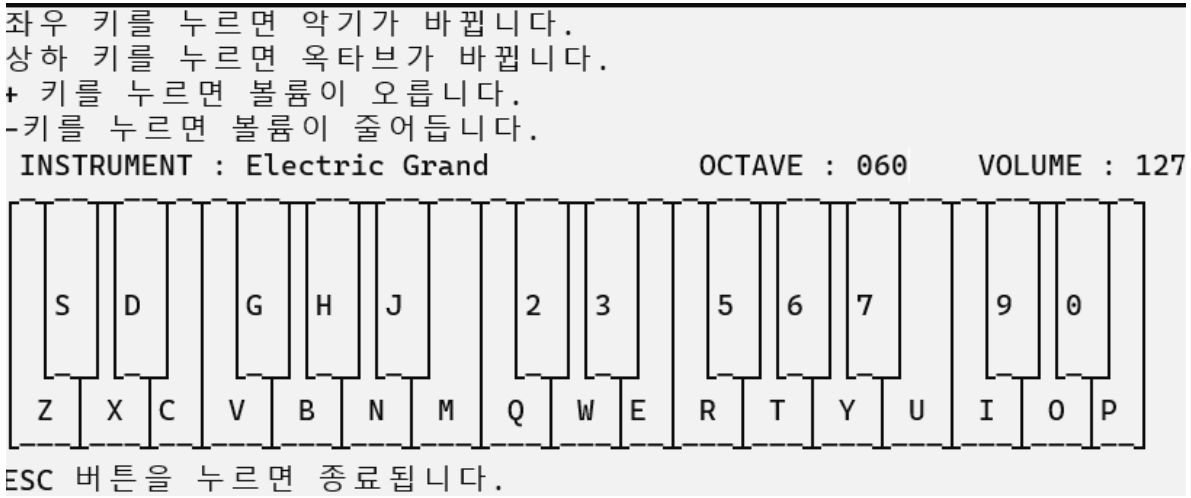

```
ESC 버튼을 누르면 종료됩니다.
```

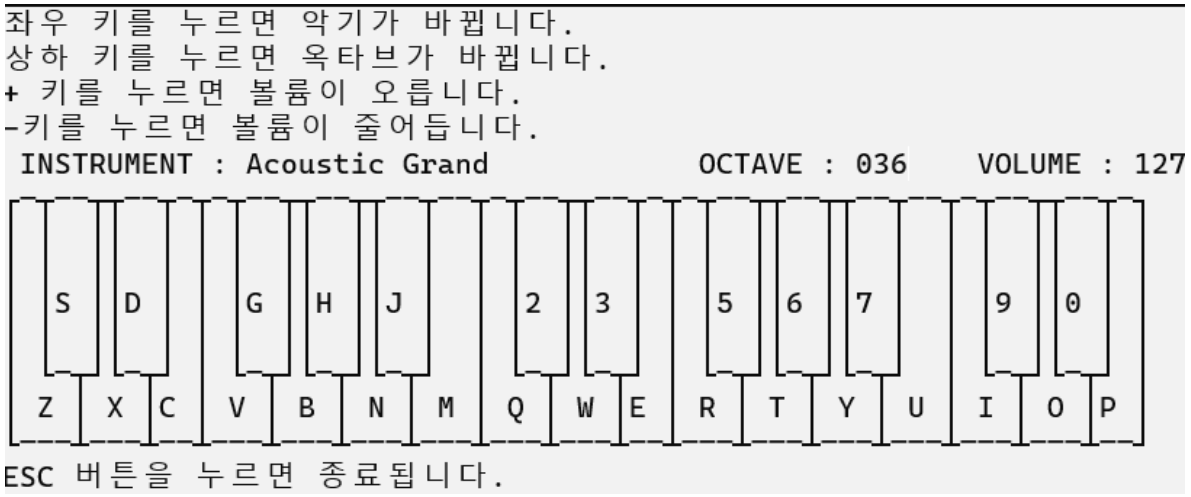
실행 화면 – 악기 변경


좌우 키를 누르면 악기가 바뀝니다.
상하 키를 누르면 옥타브가 바뀝니다.
+ 키를 누르면 볼륨이 오릅니다.
-키를 누르면 볼륨이 줄어듭니다.
INSTRUMENT : Bright Acoustic
OCTAVE : 048
VOLUME : 127
S D G H J 2 3 5 6 7 9 0
Z X C V B N M Q W E R T Y U I O P
ESC 버튼을 누르면 종료됩니다.

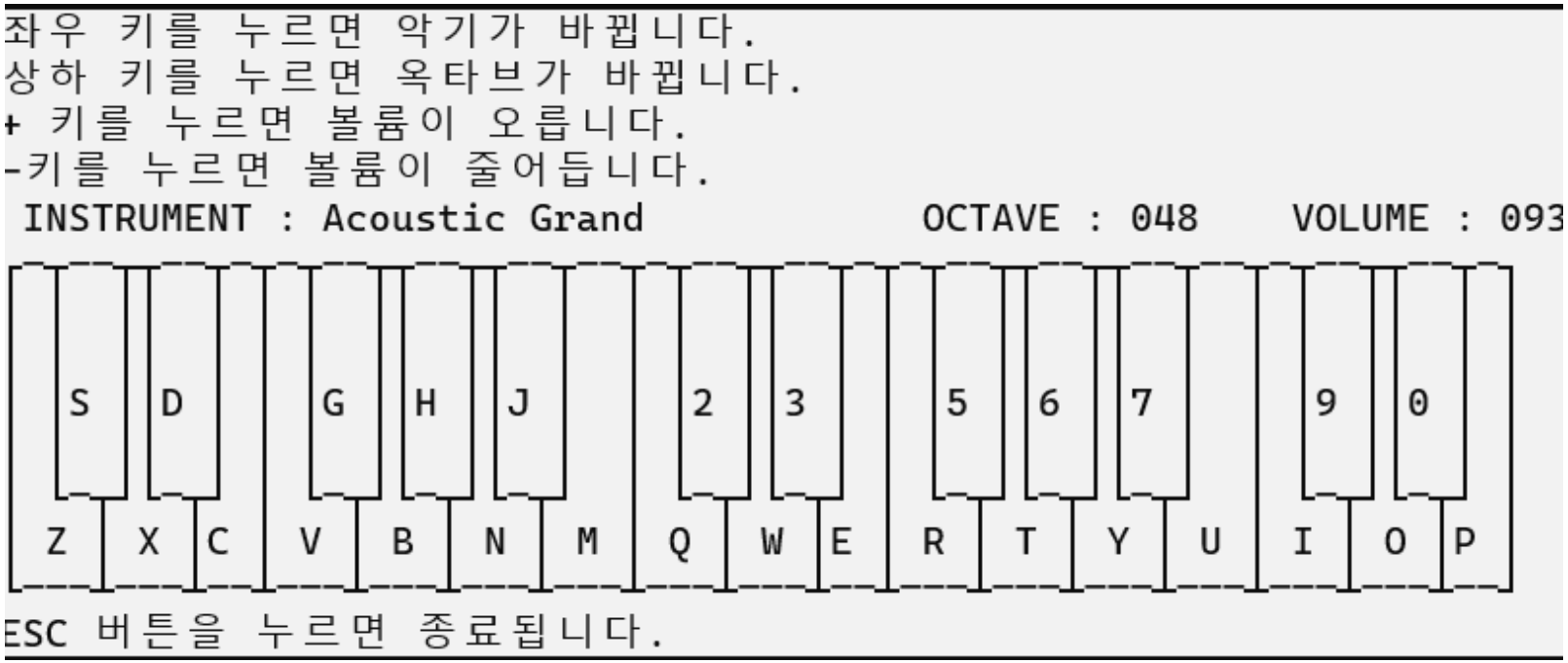

좌우 키를 누르면 악기가 바뀝니다.
상하 키를 누르면 옥타브가 바뀝니다.
+ 키를 누르면 볼륨이 오릅니다.
-키를 누르면 볼륨이 줄어듭니다.
INSTRUMENT : Electric Grand
OCTAVE : 048
VOLUME : 127
S D G H J 2 3 5 6 7 9 0
Z X C V B N M Q W E R T Y U I O P
ESC 버튼을 누르면 종료됩니다.

실행 화면 – 옥타브 변경


좌우 키를 누르면 악기가 바뀝니다.
상하 키를 누르면 옥타브가 바뀝니다.
+ 키를 누르면 볼륨이 오릅니다.
-키를 누르면 볼륨이 줄어듭니다.
INSTRUMENT : Electric Grand
OCTAVE : 060
VOLUME : 127
S D G H J 2 3 5 6 7 9 0
Z X C V B N M Q W E R T Y U I O P
ESC 버튼을 누르면 종료됩니다.


좌우 키를 누르면 악기가 바뀝니다.
상하 키를 누르면 옥타브가 바뀝니다.
+ 키를 누르면 볼륨이 오릅니다.
-키를 누르면 볼륨이 줄어듭니다.
INSTRUMENT : Acoustic Grand
OCTAVE : 036
VOLUME : 127
S D G H J 2 3 5 6 7 9 0
Z X C V B N M Q W E R T Y U I O P
ESC 버튼을 누르면 종료됩니다.

실행 화면 - 볼륨 조절


좌우 키를 누르면 악기가 바뀝니다.
상하 키를 누르면 옥타브가 바뀝니다.
+ 키를 누르면 볼륨이 오릅니다.
-키를 누르면 볼륨이 줄어듭니다.
INSTRUMENT : Acoustic Grand
OCTAVE : 048
VOLUME : 093
S D G H J 2 3 5 6 7 9 0
Z X C V B N M Q W E R T Y U I O P
ESC 버튼을 누르면 종료됩니다.

Reference

- https://ko.wikipedia.org/wiki/MIDI
- https://www.midi.org/index.php
- https://jsjin.tistory.com/entry/C언어에서-Midi-입력을-받는-코드-만들기
- https://www.youtube.com/watch?v=jq08L2TxY4E
- https://blog.naver.com/PostView.naver?blogId=sharonichoya&logNo=220874370397&parentCategoryNo=&categoryNo=22&viewDate=&isShowPopularPosts=false&from=postView
- https://chat.openai.com/