

Unifox 개인 프로젝트 보고서

프로젝트 개요

제가 진행한 개인 프로그래밍 프로젝트는 PlaySound 함수를 이용한 음악 플레이어 제작입니다. 프로그램 제작과정에서 어려움도 있었지만, Win32 API를 활용함으로써 C언어에 대한 이해가 더 깊어지게 되는 계기가 되었습니다.

저는 음악을 컴퓨터 다음으로 좋아하기 때문에 음악에 대한 열정과 관심이 컸고 그런 성격을 살려 음악 플레이어를 만들자고 생각하여 이 프로그램을 제작하게 되었습니다.

개발 환경은 Win32 Api를 활용해야 하므로 현재 사용중인 Mac OS에서는 C언어 특성상 개발하기가 어렵기 때문에 가상머신에 Windows10을 설치하여 가상머신 환경에서 개발을 진행했습니다. 따라서 Windows환경의 컴퓨터에서 정상적으로 작동할 수 있습니다.

이번에 제작한 음악 플레이어는 기본적인 음악 재생 및 정지 기능을 제공할 뿐만 아니라 지정한 디렉토리에 존재하는 재생 가능한 음악 파일의 리스트를 불러와 콘솔에 출력 하며, 사용자가 프로그램 상에서 듣고싶은 음악을 골라 재생하는 기능까지 제공합니다.

음악을 재생시키는 기능을 구현할 때, PlaySound라는 Win32 Api 함수를 사용하게 되었는데, 이 함수는 mp3파일을 재생할 수 없고 wav 파일만 재생할 수 있다는 특성을 가지고 있습니다. 따라서 파일의 크기가 mp3보다 비교적 큰 wav파일을 재생할 때 메모리 공간을 더 차지한다는 단점을 가지고 있다는 것을 새롭게 알게되었고, 추후 mp3 파일을 재생할 수 있는 함수로 교체하는 유지 보수 작업을 진행할 예정입니다.

C언어의 특성상 try ~ catch 구문을 사용할 수 없으므로 예외에 대한 처리가 매우 불편한데, 이에 대해서는 중요한 부분인 파일을 검색하는 기능에 한해서만 파일이 없거나 또는 여러 오류 발생시 발생할 수 있는 문제를 if문으로 미리 예방하였습니다.

이 프로그램은 지정된 디렉토리에 있는 음악 파일을 불러오기 때문에 불필요하게 소스 코드를 수정하여 재생할 음악을 변경시키는 일 없이 지정된 디렉토리에 사용자가 음악 파일을 넣기만 하면 그 음악 파일을 재생할 수 있습니다. 이번 프로젝트에서 이 기능을 가장 자찬하고 싶습니다.

다음은 상세한 설명입니다.

프로그램 기능 상세 설명 및 분석

```
char path[260] = "C:\\Users\\snow\\Desktop\\Music\\*.wav";  
char file[260] = { 0, };
```

본 프로그램의 음악 파일을 검색하는 기본 경로는 위와 같이 바탕화면의 Music 폴더로 지정되어 있습니다. Music 폴더에 .wav 확장자를 가진 음악 파일만 넣어준다면 바로 리스트를 가져와 재생할 수 있습니다.

디렉토리 경로에서 \ 글자가 일반적으로 한번이 아닌 두번 연속으로 쓰여져 있는데, 그 이유는 C 언어는 문자열에서 \를 \n, \t와 같은 개행문자 등의 특수한 기능을 하는 문자를 받을 때 인식하는 것이기 때문에 역슬래시 문자를 한 번 더 써주지 않으면 파일 경로를 불러들일 때 오류가 발생하게 됩니다. 따라서 저렇게 처리를 해 주었습니다.

프로그램에서 문자열을 입력받는 일은 없기 때문에 문자열 오버플로우로 인해 프로그램이 오류를 뿜으며 공중분해 하는 일은 미리 방지가 되었습니다.

mp3 파일을 재생하는 유지 보수와 같이 저 파일의 경로도 사용자가 임의로 지정하여 원하는 디렉토리의 음악 파일을 불러올 수 있게 기획하는 중 입니다.

```
스파게티소스.cpp  main.c  X  굴소스.cpp  
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  #include <Windows.h>  
4  #include <mmsystem.h>    //음악출력을 위함  
5  #include <io.h>  
6  #include <wchar.h>  
7  #include <string.h>  
8  #pragma comment(lib, "winmm")    //음악출력을 위함  
9
```

위 사진은 프로그램 구동에 필요한 각종 라이브러리 파일을 include 한 부분입니다. 음악 재생 함수, 파일 리스트 불러와 함수에 대입, 유니코드 인코딩 변경 등 여러 기능을 수행하기 위해서 총 7개의 라이브러리가 include 되었습니다. mmsystem.h 와 wchar.h는 처음 보는 라이브러리여서 이번 프로젝트를 통해 새롭게 알게 되었습니다. 음악을 재생시키는 PlaySound 함수, 그리고 유니코드 변환 등의 각종 함수가 들어있는 라이브러리 입니다.

```
typedef struct _finddata_t FILE_SEARCH; //파일 탐색 구조체

void printMenu(int num, FILE_SEARCH *file_search); //메뉴 및 파일 출력
void getFileList(int num, FILE_SEARCH *file_search); //특정 파일 이름 반환
void playMusic(int menu_select, FILE_SEARCH *file_search); //음악 재생 및 정지
```

본 프로그램은 총 main() 함수를 제외하고 총 3개의 함수로 이루어져 있으며, 각각 메뉴 및 파일의 리스트를 출력하는 printMenu 함수, 원하는 번호에 대치되는 파일의 이름을 가져오는 getFileList 함수, 음악을 재생하고 정지시키는 부분을 담당하는 playMusic 함수를 정의하였습니다.

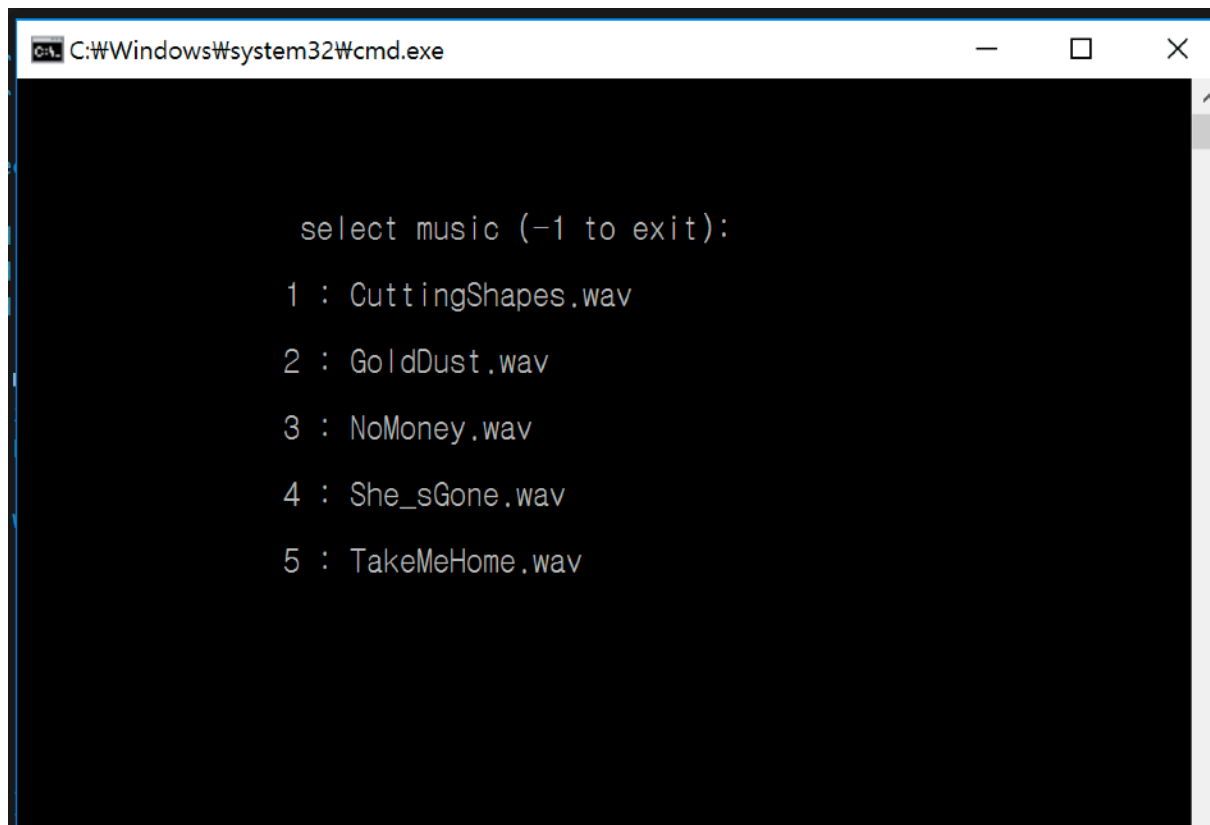
함수의 이름은 Java가 만들어지면서 생긴 일종의 프로그래머의 작명규칙은 낙타 표기법을 따르고 있습니다. Java Beans처럼 private 요소가 있고 getter/setter, toString 메소드 등이 존재하지는 않지만, 유지보수 및 가독성을 향상시키기 위해 이 표기법을 따르고 있습니다.

이러한 함수 및 변수 등의 작명 표기법은 프로그래밍 트렌드에 따라서 많이 바뀌는데, 꼭 따라야 하는 것은 아니지만 실무에 들어가서 회사에서 일을 하게 되면, 꼭 협업을 하게 되기 때문에 규칙을 정하면

```
int main() {
    int menu_select = 0;
    FILE_SEARCH file_search;

    while (1) {
        printMenu(menu_select,&file_search);
        scanf("%d", &menu_select);
        if (menu_select == -1) { // -1 입력하면 프로그램 종료
            exit(0);
        }
        system("cls");
        playMusic(menu_select,&file_search);
    }
}
```

따르는 것이 좋습니다. 다른 사람이 작성한 코드를 조금이라도 더 알기 쉽게 하기 위해서는 매우 중요한 부분을 차지하고 있는 것이 표기법입니다. 요즘 트렌드는 낙타 표기법보다 언더바(_)로 띄어쓰기를 구분하고 전부 소문자로 작성하는 규칙을 더 많이 선호하고 있긴 하지만, 저는 낙타 표기법이 더 편하여 그 표기법을 선택하게 되었습니다.

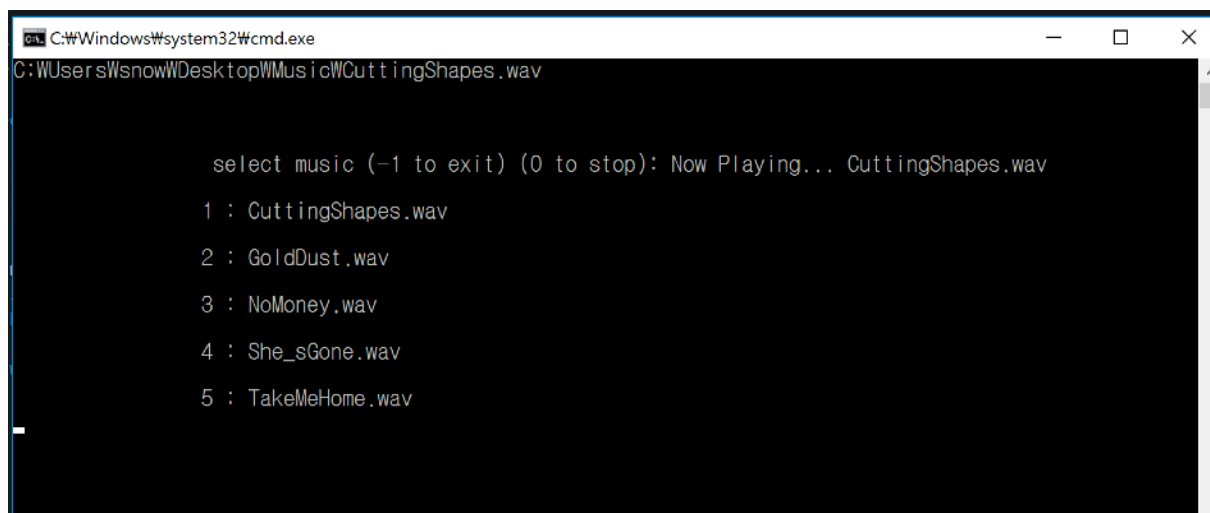


```
C:\Windows\system32\cmd.exe

select music (-1 to exit):

1 : CuttingShapes.wav
2 : GoldDust.wav
3 : NoMoney.wav
4 : She_sGone.wav
5 : TakeMeHome.wav
```

기본적으로 main 함수는 다음과 같이 단순하게 구성이 되어있습니다. 무한루프 내부에서 메뉴와 재생 가능한 음악 파일의 리스트를 출력시킨 다음, 사용자에게 입력을 받아 프로그램의 흐름을 결정합니다. FILE_SEARCH는 상단에서 선언한 파일 검색 구조체의 선언자입니다. 저 구조체를 이용해서 음악 파일의 리스트를 불러오는 등의 기능을 구현하였습니다.



```
C:\Windows\system32\cmd.exe
C:\Users\snow\Desktop\Music\CuttingShapes.wav

select music (-1 to exit) (0 to stop): Now Playing... CuttingShapes.wav

1 : CuttingShapes.wav
2 : GoldDust.wav
3 : NoMoney.wav
4 : She_sGone.wav
5 : TakeMeHome.wav
```

프로그램을 처음 실행하게 되면 위와 같이 초기화면이 나오게 됩니다. 현재 아무 노래도 재생하고 있지 않으므로 Now Playing...과 재생중인 곡 이름이 표시되지 않으며 현재 Music 디렉토리에 존재하는 5개의 노래 리스트를 출력하고 있는 모습입니다. 여기서 -1을 입력하게 되면 프로그램이 종료되며, 콘솔에 출력되는 리스트에 있는 노래에 대치되는 번호를 입력하면 그 곡의 재생이 시작됩니다.

1을 입력했을 때 나오는 화면입니다. 0을 누르면 정지된다고 안내하고 있습니다. Now Playing... [현재 재생중인 노래] 라는 문구가 추가된 것을 확인할 수 있습니다. 또한 이 상태에서 -1을 눌러 프로그램을 바로 종료할 수 있고, 다른 노래의 번호를 입력하여 다른 노래를 들을 수도 있습니다.

함수 설명 및 분석

앞서 설명한 main()함수 외 사용자 정의 함수 3개에 대하여 상세한 설명 및 분석을 하도록 하겠습니다.

```
void printMenu(int num, FILE_SEARCH *file_search) { //메뉴 및 파일 리스트 출력
    int i = 1;
    long h_file;

    if (num == 0) {
        printf("\n\n\n\n\t\t select music (-1 to exit): \n");
    }
    else {
        printf("\n\n\n\n\t\t select music (-1 to exit) (0 to stop): Now Playing... %s\n",file);
    }

    if ((h_file = _findfirst(path, file_search)) == -1L) { //파일 없거나 각종 오류 발생시 함수 종료
        printf("읽기 오류\n");
        return;
    }
    do {
        //char *file = file_search->name;
        strcpy(file, file_search->name);
        printf("\n\t\t%d : %s\n",i, file);
        i++;
    } while (_findnext(h_file, file_search) == 0); //파일 리스트 출력
    _findclose(h_file); //스트림 닫기
}
```

먼저 메뉴와 재생 가능한 음악 파일의 리스트를 출력해주는 함수인 printMenu 함수에 대해서 설명 및 분석하겠습니다.

음악이 재생되고 있지 않을 때 num은 0 값이 들어가 있으므로 num==0이면 Now Playing이 없이 음악을 선택하라고만 출력하게 됩니다. 다음 if문에서는 파일을 검색했을 때 결과가 없으면 읽기 오류를 출력하고 함수를 종료합니다. 검색 결과가 존재한다면 해당 if문은 건너뛰게 됩니다.

밑의 do~while문에서는 파일의 리스트를 출력하는 역할을 하게 됩니다. file이라는 전역변수에 검색한 파일의 이름을 저장한 후, printf를 통해 출력합니다. 그리고 반복문이 돌 때마다 다음 파일을 검색하는 _findnext를 호출하여 파일 리스트를 쭉 출력합니다.

파일을 검색하는 finddata 함수도 데이터의 방향성이 존재하기 때문에 스트림이 존재합니다. 따라서 파일 검색을 마쳤으면 스트림을 닫아주도록 합니다. 굳이 스트림을 안닫아줘도 프로그램이 종료될 때 같이 종료되긴 하지만, 프로젝트, 프로그램의 규모가 커질 경우 효율성이 떨어질 수 있기 때문에 되도록이면 바로바로 닫아주도록 하는게 좋습니다.

```

void getFileList(int num, FILE_SEARCH *file_search) {
    //특정 파일 이름 file에 저장
    long h_file;
    int i = 1;
    if ((h_file = _findfirst(path, file_search)) == -1L) { //파일 없거나 각종 오류 발생시 함수 종료
        printf("읽기 오류\n");
        return;
    }

    do {
        if (i == num) { //사용자가 입력한 번호에 대치하는 음악 반환
            strcpy(file, file_search->name);
            _findclose(h_file);
        }
        i++;
    } while (i <= num && (_findnext(h_file, file_search) == 0));
}

```

다음으로 입력한 번호에 대치되는 파일 이름을 file 변수에 저장시키는 함수인 getFileList입니다. 앞서 설명한 printMenu 함수와 마찬가지로 읽을 수 있는 파일이 없으면 읽기 오류를 출력하고 함수를 종료합니다.

아래 반복문은 printMenu함수의 주 기능인 사용자가 입력한 번호에 대치하는 음악 파일의 이름을 변수에 저장시키는 기능을 합니다. finddata 함수는 리스트의 첫 번째부터 다음 리스트를 불러오는 방식으로 검색을 하기 때문에 원하는 파일 명을 불러오기 위해서는 반복문을 써야 합니다. 반복문이 도는 횟수와 음악에 대치하는 번호가 같다는 특성을 이용해서 위와 같은 로직을 구현했습니다.

저 함수를 처음에는 for문으로 구현했지만, 무조건 한 번은 돌아야 하기 때문에 do while문으로 구현했습니다. file이라는 변수가 전역 변수이기 때문에 딱히 구현의 어려운 문제는 없었습니다.

```

void playMusic(int menu_select, FILE_SEARCH *file_search) {
    if (menu_select == 0) { //0이면 음악 끄기
        PlaySound(NULL, 0, 0);
        return;
    }
    else {
        getFileList(menu_select, file_search);
        char pStr[260] = "C:\\Users\\snow\\Desktop\\Music\\"; //음악파일 들어간 경로
        strcat(pStr, file); //문자열 합치기, 음악파일의 절대경로를 만들어줌
        wchar_t sum[260] = { 0, }; //유니코드 형태의 문자열 저장공간
        mbstowcs(sum, pStr, 260); //유니코드로 복사
        printf("%ls", sum);
        PlaySound(sum, NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
        return;
    }
}

```

이 함수는 main함수와 printMenu함수에서 호출하는 것이 아닌 노래를 재생시키는 함수인 playMusic 함수에서만 호출됩니다. 사용자가 재생할 음악의 번호를 입력했을 때만 작동하면 되기 때문입니다. 그럼 바로 playMusic 함수를 살펴보겠습니다.

위의 playMusic 함수는 음악의 재생 및 종지를 담당하는 함수입니다. 사용자가 입력한 정수 값을 인자로 전달받아 getFileList함수를 호출할 때 사용합니다.

사용자가 수를 입력할 때 0을 입력하는 것은 음악 재생을 중지하라는 의미를 가지고 있으므로 가장 처음 그 조건을 검사해서 참이면 음악 재생을 중지하는 함수를 호출한 뒤 함수를 종료합니다.

만약 입력된 수가 0이 아니면 -1이거나 음악의 번호가 입력이 되는 것인데, -1이 입력되었다면 이미 main에서 필터링 되었을 것이므로 무조건 음악의 번호인 경우에만 해당됩니다.

이 부분에서 해당하는 음악을 재생시키면 되는데, 이 기능을 구현할 때 많은 어려움이 있었습니다. 바로 PlaySound함수의 인자 특성 때문인데요, PlaySound는 맨 처음 파일의 경로를 전달받게 되는데, 이 문자열의 형식이 우리가 일반적으로 알고 있는 char형 문자열이 아닌 유니코드 인코딩을 하고 있는 LPWSTR 형식의 문자열 (wchar_t)을 요구합니다. 처음 이 함수를 구현했을 때 자꾸 프로그램을 실행시키는 도중 작동이 중지되고, 디버깅을 해도 찾을 수 없어서 꽤 애를 먹었습니다. 결국 수많은 구글링 결과 유니코드 인코딩 형식의 문자열을 넣어야 한다는 것을 알아내었고, TEXT() 매크로, L접미사 등 여러 시도를 했지만 그 방법이 통하지 않았고 끝내 찾아내 성공한 방법이 wchar_t형식의 변수를 새로 선언해서 char형식의 문자열을 유니코드로 변환해주는 함수를 이용해 붙여넣는 방법입니다. 위의 pStr을 선언하는 부분부터 mbstowcs 까지가 그 문제를 해결하는 코드입니다.

이만큼 많은 노력이 들어간 만큼 얻는 성취감도 컸는데, 프로그램에서 이 부분을 가장 많이 자랑스럽게 생각하고 있습니다. 원하는 음악에 해당하는 번호를 입력하면 해당 파일 이름을 가져와서 재생까지 할 수 있게 구현한 로직을 보면 제가 봐도 잘짠다는 생각이 들게 합니다. 코드에서 가장 칭찬받고 싶은 부분이 이 부분입니다.

상단의 사진에서 볼 수 있는 else {}는 불필요하기 때문에 현재는 제거를 한 상태입니다.

Review

현재 wav파일을 재생하는 playSound 함수의 특성상 시스템의 자원을 mp3 파일을 재생시킬 때 보다 많이 사용한다는 단점 때문에 mp3파일을 재생할 수 있는 함수로 대체하기 위한 기획을 하고 있는 상태입니다. mp3파일을 재생시키는 함수에서는 일시정지와 같은 더 많은 확장기능도 제공하기 때문에 프로그램 자체의 퀄리티도 더욱 향상하고 시스템 자원 사용량도 현저히 줄일 수 있을 것으로 기대됩니다. 또한 음악을 들을 때 mp3파일을 가장 많이 사용하므로 실용성도 많이 오르게 될 것입니다. 프로젝트가 끝나도 개인적으로 계속 프로그램의 업데이트를 진행할 예정입니다.

이전에는 경험하지 못한 여러가지 api를 이용한 함수들을 사용해 봄으로써 개발 능력을 더욱 향상시켰다는 생각이 들고, 프로그래밍 실력도 한층 향상된 것을 체감하고 있습니다. 개발 중 알 수 없는 오류가 계속 발생해 힘든 순간들도 많았지만, 프로그램을 구동 성공하였을 때 성취감과, 프로그래밍이라는 기술로 더 나은 세상을 만들 수 있다는 가능성을 보았기 때문에 매우 의미있는 프로젝트였다고 생각이 됩니다. 또한 프로그래밍의 기초를 배우고 난 뒤 자신의 첫 공식적 성과물을 만든 것이기 때문에 그 의미는 더욱 부각된다고 생각합니다.

프로그래밍을 더 공부하여 이 프로그램을 GUI까지 지원하게 만든 후, 여러가지 기능들을 추가하여 제 개인 음악 플레이어로 만들어 사용하고 싶습니다. 그만큼 프로그래밍에 대한 열정도 더 심어준 계기가 된 것 같습니다.