**자료 구조 프로젝트2 명세서**

**음원 관리 프로그램**

소프트웨어융합학과 2017103729 서민영

2018.12.03

**1. 프로젝트 목표**

현재 곰플레이어, 네이버 미디어 플레이어 등 많은 음원 플레이어가 있다. 이러한 프로그램처럼 자료구조 수업 시간에 배운 클래스와 기능을 사용하여 음원을 재생하고 관리하는 기능을 제공하는 프로그램을 만들고자 한다. 뿐만 아니라 기존의 시스템과 차별화될 수 있는 자신만의 특색을 갖춘 프로그램을 구현하려 한다.

**2. 프로그램 기능**

**1) 사용자**

- 원하는 곡을 다운받아 추가하거나 삭제할 수 있다.

- 원하는 곡을 곡 리스트 내에서 검색할 수 있다.

- 최근에 추가한 곡 목록을 보여준다.

- 앨범 단위로 검색하고 앨범에 따라 수록된 곡을 보여준다.

. (앨범을 선택하면 수록 곡을 보여준다.)

- 가수 단위로 검색하고 가수에 따라 수록된 곡을 보여준다.

- 장르 단위로 검색하고 장르에 따라 수록된 곡을 보여준다.

- 앨범, 가수, 장르 별로 음악을 화면에 띄운 후 선택한 분야에 속한 곡을 보여준다.

- 나만의 곡 재생 목록을 만들어 그 안에서 곡 추가 및 삭제가 가능하게 한다.

- 년도 별 노래 곡 목록을 보여준다.

- 추가한 음악 중에서 랜덤으로 음악 추천을 해준다

- 곡 정보 내용을 변경할 수 있다.

**2) 관리자**

- 사용자가 할 수 있는 모든 기능을 똑같이 사용할 수 있다.

- 멤버 삭제가 가능하다.

**3. 프로그램 시나리오**

**1) 사용자 주요 기능**

- 곡 추가 : 곡명과 가수를 작성해 기본 음원 리스트에 추가할 수 있다.

- 모든 곡 표시 : 기본 음원 리스트에 있는 곡에 대한 정보들을 표시한다.

- 곡 검색 : 기본 음원 리스트에 있는 곡을 곡명 또는 가수 명으로 검색할 수 있다.

- 곡 삭제 : 기본 음원 리스트에 있는 곡을 삭제할 수 있다.

(다른 리스트에 있는 곡도 삭제할 수 있다.)

- 곡 리스트 생성 : 사용자는 기본 음원 리스트에 있는 곡 중 마음에 드는 곡을 모아 플레이 리스트를 생성해 그 리스트에 추가할 수 있다.(포인터 이용)

- 전체 곡 중에서 랜덤으로 음악 추천을 해준다

- 로그인, 로그아웃, 회원가입

**2) 관리자 주요 기능**

- 사용자 정보 보기 : 현재 사용자의 재생 목록을 볼 수 있고 리스트에 있는 곡들에 대해 삭제가 가능하다.

**4. 클래스 목록**

<Applicaion.h>

class Application

{

public:

Application()

{

this->\_loginedCustomer = NULL;

Customer\* newCustomer = NULL;

}

~Application() { }

void Run();

void Menu(); //메뉴 출력

void Main\_Menu();

//선택지를 보여주는 메뉴

int CustomerMenu();

int AddItem();

void DisplayAll();

\* @brief 사용자에게 Title을 입력받아 SortedList의 Get함수에 넘겨주는 역할을 한다.

bool Search();

void Delete();

SearchListByMemberName함수에 문자열이 설정된 객체를 전달한다.

void RetreiveRecordByMemberName();

\* @brief inData의 이름 부분을 포함하는 문자열을 찾아 출력한다.

void SearchListByMemberName(MusicType &inData);

@brief 사용자에게 Title을 입력받아 Title을 찾아 데이터를 출력

int SearchByTitle();

@brief 사용자에게 가수를 입력받아 가수를 찾아 데이터를 출력

int SearchByArtist();

@brief 사용자에게 앨범을 입력받아 앨범을 찾아 데이터를 출력

int SearchByAlbum();

@brief 사용자에게 장르를 입력받아 장르를 찾아 데이터를 출력

int SearchByGenre();

@brief 최근에 추가된 30곡의 자세한 정보를 화면에 출력

void DisplayNewMusic();

@brief 장르별로 음악 화면출력

int DisplayMusicByGenre();

@brief 앨범별로 음악 화면출력

int DisplayMusicByAlbum();

@brief 가수별로 음악 화면출력.

int DisplayMusicByArtist();

@brief Retrieve item in the list.

int RetrieveByNameNGenre();

@brief m\_List 내에서 랜덤으로 하나의 곡이 화면에 나온다.

int RecommendMusic(); //m\_List 내에서 저장된 곡들 중 랜덤으로 한 곡을 화면에 띄움.

int Login(); //로그인

void Logout(); //로그아웃

void RegisterCustomer(); //회원 등록

void PrintCustomerInfo(); //로그인된 유저정보 출력

bool GetCommand(int& buffer, int minRange = INT\_MIN, int maxRange = INT\_MAX);

bool GetCommand(string& buffer);

//리스트에서 음악 검색

int SearchMusic();

//가수 앨범 장르 별로 음악 검색

int DisplayMusicBy();

<Music Type>

class MusicType

{

public:

\* default constructor.

MusicType()

{

m\_Number = 0;

m\_Title = "";

m\_sArtist = "";

m\_sAlbum = "";

m\_sGenre = "";

m\_sLyrics = "";

}

\* destructor.

~MusicType() {}

\* @brief 곡번호 받기.

int GetNumber()

\* @brief 곡명 받기.

string GetTitle()

\* @brief 가수 받기.

string GetArtist()

\* @brief 앨범 받기.

string GetAlbum()

\* @brief 장르 받기.

string GetGenre()

\* @brief 가사 받기.

string GetLyrics()

\* @brief Set Number.

void SetNumber(int inNumber)

\* @brief Set Title.

void SetTitle(string inTitle)

\* @brief Set Artist.

void SetArtist(string inArtist)

\* @brief Set Album.

void SetAlbum(string inAlbum)

\* @brief Set Genre.

void SetGenre(string inGenre)

\* @brief Set Lyrics. 단, 띄어쓰기까지 인식할 수 있어야 함.

void SetLyrics(string inLyrics)

\* @param inNumber Music Number.

\* @param inTitle Music Title.

\* @param inArtist Music Artist.

\* @param inAblum Music Album.

\* @param inGenre Music Genre.

\* @param inLyrics Music Lyrics.

\* @brief Display Music Number on screen.

void DisplayNumberOnScreen()

\* @brief Display Music Title on screen.

void DisplayTitleOnScreen()

\* @brief Display Music Artist on screen.

void DisplayArtistOnScreen()

\* @brief Display Music Album on screen.

void DisplayAlbumOnScreen()

\* @brief Display Music Genre on screen.

void DisplayGenreOnScreen()

\* @brief Display Music Lyrics on screen.

void DisplayLyricsOnScreen()

\* @brief Display an Music record on screen.

void DisplayRecordOnScreen()

{

DisplayNumberOnScreen();

DisplayTitleOnScreen();

DisplayArtistOnScreen();

DisplayAlbumOnScreen();

DisplayGenreOnScreen();

DisplayLyricsOnScreen();

};

\* @brief Set Music Number from keyboard.

void SetNumberFromKB();

\* @brief Set Music Title from keyboard.

void SetTitleFromKB();

\* @brief Set Music Artist from keyboard.

void SetArtistFromKB();

\* @brief Set Music Album from keyboard.

void SetAlbumFromKB();

\* @brief Set Music Genre from keyboard.

void SetGenreFromKB();

\* @brief Set Music Lyrics from keyboard.

void SetLyricsFromKB();

\* @brief Set Music record from keyboard.

void SetRecordFromKB();

\* @brief Read a record from file.

int ReadDataFromFile(ifstream& fin);

\* @brief Write a record into file.

int WriteDataToFile(ofstream& fout);

RelationType CompareByNumber(const MusicType &data);

protected:

int m\_Number; ///< 곡번호

string m\_Title; ///< 곡명

string m\_sArtist; ///< 가수

string m\_sAlbum; ///< 앨범

string m\_sGenre; ///< 장르

string m\_sLyrics; ///< 가사

};

<SortedList.h>

template <typename T>

class SortedList

{

public:

SortedList()

~SortedList() {}

\* @brief Make list empty.

void MakeEmpty();

\* @brief Get a number of records in current list.

int GetLength();

\* @brief Check capacity of list is full.

bool IsFull();

\* @brief primary key를 기준으로 정렬이 되도록 알맞은 자리에 항목을 추가한다.

int Add(T data);

\* @brief Initialize list iterator.

void ResetList();

\* @brief move list iterator to the next item in list and get that item.

int GetNextItem(T& data);

\* @brief 일부만 채워져있는 data에 알맞은 정보를 저장한다.

int Get(T& data);

\* @brief 사용자가 지정한 MusicType를 배열에서 삭제한다.

int Delete(T data);

\* @brief 사용자가 지정한 MusicType에 해당하는 배열의 자리에 파라미터의 내용을 복사한다.

int Replace(T data);

\* @brief 일부만 채워져있는 data에 알맞은 정보를 저장한다. 대상을 찾을때 Binary Search를 이용한다.

int GetByBinarySearch(T& data);

\* @brief 주어진 Array Size이 가득 찼을 때 Array Size를 늘려줌.

void AddSize()

\* @brief 메모리가 가득 찼을 때 메모리를 증가시킴.

void AddMemory();

private:

int ArraySize = 5;

T \*m\_Array = new T[ArraySize]; ///< list array.

int m\_Length; ///< number of elements in list.

int m\_CurPointer; ///< iterator pointer.

};

<SimpleType.h>

class SimpleType

{

public:

SimpleType()

{

s\_Number = 0;

s\_Title = "";

}

~SimpleType() {}

\* @brief 곡번호 받기.

int GetNumber()

\* @brief 곡명 받기.

string GetTitle()

\* @brief Set Number.

void SetNumber(int inNumber)

\* @brief Set Title.

void SetTitle(string inTitle)

protected:

int s\_Number; ///< 곡번호

string s\_Title; ///< 곡명

};

<GenreList.h>

class GenreList

{

public:

GenreList()

SortedList<SimpleType>\* GetGenreList()

void DisplayRecord()

string genre;

int g\_num;

SortedList<SimpleType> genreList;

};

<CircularQueueType.h>

class FullStack

{

public:

class EmptyStack

{

public:

/\*\*

\* @brief Display a message for full stack on screen.

\*/

void print()

{

cout << "EmtpyStack exception thrown." << endl;

}

};

template <typename T> //generic data을 위해 template 선언

class CircularQueueType

{

public:

\* @brief size에 해당하는 크기의 배열을 동적할당

CircularQueueType(int size);

\* @brief 최대크기로 생성

CircularQueueType(void);

~CircularQueueType(void);

\* @brief queue가 가득차있는지 확인.

bool IsFull() const;

\* @brief queue가 비어있는지 확인

bool IsEmpty() const;

\* @brief queue에 회전하면서 값을 채움.

void EnQueue(T item);

\* @brief queue에 회전하면서 값을 삭제(=front값이 하나 증가)

void DeQueue(T& item);

\* @brief Returns a copy of top item on the stack.

T Top();

\* @brief Print all the items in the stack on screen

void Print();

void MakeEmpty();

\* @brief Get music number

int GetNumber()

\* @brief Get music title

string GetTitle()

\* @brief Set music Number.

void SetNumber(int inNumber)

\* @brief Set music Title.

void SetTitle(string inTitle)

\*@brief move list iterator to the next item in list and get that item.

int GetNextItem(T& data);

\* @brief Initialize list iterator.

void ResetQueue();

\* @brief Compare two item types by item Number.

T CompareByNumber(const T &data);

\* @brief top 받기.

int GetTop();

private:

int m\_iFront;

int m\_iRear;

int m\_nMaxQueue;

int top; ///< Number of elements.

T \*items; ///< Pointer for a stack.

int r\_CurPointer;

};

**5. 개발 환경**

Microsoft Visual Studio 2017 (C++)