자료구조 프로젝트 1

카카오톡 서버와 클라이언트를 위한 자료구조 설계 및 구현.

2013101775 김민규

제출일: 2017.10.15

1. **프로젝트 주제 및 목표**

현재 우리나라에서 가장 대중적으로 사용되고 있는 메신저 카카오톡의 기능들을 분석하고 최적의 자료구조를 이용하여 설계하고 구현한다. 필요한 새로운 기능을 분석해보고 최적의 자료구조를 이용하여 설계 및 구현한다.

1. **프로그램 기능**
   1. 서버 시스템 기능

\* 회원 가입//키보드로 회원정보(이름, 전화번호, ID등)을 입력하여 레코드를 등록한다.

\* 회원 탈퇴 //아이디와 패스워드를 입력하여 레코드를 삭제한다.

\* 로그인 //아이디와 패스워드를 입력하여 로그인 한다.

\* 로그아웃 //로그인 된 아이디를 로그아웃 시킨다.

\* 회원 검색 //ID 혹은 이름으로 검색하여 해당하는 회원의 정보를 출력한다.

\* 회원 정보 수정 //아이디와 패스워드를 입력하여 회원 정보를 수정한다.

* 1. 클라이언트 시스템 기능

\* 친구 검색 //친구들의 번호와 각 번호에 해당하는 친구 목록 출력.

\* 친구 삭제 // 친구 번호를 입력하면 그 번호에 해당하는 친구가 목록에서 삭제됨.

\* 친구 추가 // 아이디를 입력하면 해당하는 회원의 정보가 친구목록에 추가된다.

\* 1:1 채팅 시작 // 채팅대상 친구 번호를 입력하고, 채팅내용을 입력한다. 입력이 완료되면 채팅 목록이 채팅순서에 맞게 저장된다.

\* 채팅 기록에서 기존 채팅항목을 선택하여 메시지 전송 //채팅목록에서 해당하는 채팅항목의 번호를 입력하고 채팅대상을 입력하면, 채팅대상에게 해당하는 채팅내용이 저장된다.

\* 선택된 메시지 삭제 // 채팅목록에서 해당하는 채팅항목의 번호를 입력하면 해당하는 메시지가 목록에서 지워진다.

1. **프로그램 시나리오**

프로그램 실행 시 메인 메뉴가 출력되고 메인 메뉴의 내용은 다음과 같다.

1. 회원가입
2. 회원탈퇴
3. 로그인
4. 로그아웃
5. 회원검색
6. 회원정보 수정
7. 클라이언트 시스템 // 로그인 현황을 확인하고 로그인 상태에만 작동.

(1)1:1 채팅

1) 1:1채팅 시작

2) 채팅 기록에서 기존 채팅항목을 선택하여 메시지 전송

3) 선택된 메시지 삭제

0) 처음으로

(2)친구 관리

1) 친구 검색

2) 친구 삭제

3) 친구 추가

0) 처음으로

1. **ADT**

**유저 클래스**

enum RelationType { LESS, GREATER, EQUAL }; // Relation between two items.

class ItemType

{

public:

ItemType(); //기본 생성자

~ItemType(); // 기본 소멸자

int GetId();

string GetName();

string GetAddress();

int GetBirth();

string GetGender();

string GetStatusMemo();

int GetPw();

string GetBGP();

string GetPFP();

LinkedList<int>\* getFriendList();

LinkedList<int>\* getRecommendFriendList();

// ↑멤버변수에 접근하기 위한 함수들

void SetId(int inId)

void SetName(string inName)

void SetAddress(string inAddress)

void SetBirth(int inBirth)

void SetGender(string inGender)

void SetStatusMemo(string inStatusMemo)

void SetPw(int inPw)

void SetBGP(string inBGP)

void SetPFP(string inPFP)

void SetRecord(int inId, string inName, string inAddress, int inBirth, string inGender, string inStatusMemo, int inPw, string inBGP, string inPFP)

// ↑멤버변수를 수정하기 위한 함수.

void DisplayIdOnScreen()

void DisplayNameOnScreen()

void DisplayAddressOnScreen()

void DisplayBirthOnScreen()

void DisplayGenderOnScreen()

void DisplayStatusMemoOnScreen()

void DisplayPwOnScreen()

void DisplayBGPOnScreen()

void DisplayPFPOnScreen()

void DisplayRecordOnScreen()

// ↑레코드를 출력하기 위한 함수.

void SetIdFromKB();

void SetNameFromKB();

void SetAddressFromKB();

void SetBirthFromKB();

void SetGenderFromKB();

void SetStatusMemoFromKB();

void SetPwFromKB();

void SetBGPFromKB();

void SetPFPFromKB();

void SetRecordFromKB();

// ↑키보드로부터 레코드의 정보를 설정하는 함수.

Int addItemToFriendList(int inID);// 친구 목록에 친구 추가. (추가되면 1반환)

RelationType CompareByID(const ItemType &data); //레코드를 아이디의 크기로 //비교하는 함수.

protected:

string m\_sName; ///< user name.

string m\_sAddress; ///< user PhoneAddress.

int m\_sBirth; ///< user birthday YYYYMMDD.

string m\_sGender; ///< user gender.

string m\_sStatusMemo; ///< user ProfileMessage.

int m\_Id; ///< user ID.

int m\_Pw; /// <user PW.

string m\_BGP; // <user BackGroundPhotography.

String m\_PFP; // <user ProfilePhotography.

LinkedList<int> FriendList; //친구 목록

LinkedList<int> RecommendFriendList; // 추천 친구 리스트.

};

**채팅 클래스**

Class Chatting

{

Public:

Chatting( ); //기본 생성자.

~Chatting( ); // 기본 소멸자.

Void save\_pointer ( ); // 채팅 내용을 문자 배열에 저장하고 그 포인터를 리스트에 저장.

Void ChattingStart ( ); // 채팅 대상의 친구와 채팅을 시작한다.

Protected:

Int ChattingNumber; // 채팅 대상의 친구번호.

String ChatMemo; // 채팅 내용.

} ;

**LinkedList 클래스**

template <typename T>

struct NodeType

{

T data; ///< A data for each node.

NodeType\* next; ///< A node pointer to point succeed node.

};

template <typename T>

class LinkedList

{

public:

LinkedList(); //기본 생성자

~LinkedList(); // 기본 소멸자.

void MakeEmpty( ); // 리스트를 초기화.

int GetLength( ) const; // 리스트의 길이를 반환.

int Add(T item); //리스트에 노드를 추가

int Delete(T item); // 리스트에서 노드를 삭제

int Replace(T item); //리스트에서 노드를 수정.

int Get(T& item); //리스트에서 노드를 가져온다.

void ResetList(); //current pointer를 초기화한다.

void GetNextItem(T& item); //current pointer++한다.

private:

NodeType<T>\* m\_pList; ///< Pointer for pointing a first node.

NodeType<T>\* m\_pCurPointer; ///< Node pointer for pointing current node to use iteration.

int m\_nLength; ///< Number of node in the list.

};

**어플리케이션 클래스**

class Application

{

public:

Application(); //기본 생성자

~Application() {}; // 기본 소멸자

void Run();

int GetCommand(); //커맨드를 받는 함수.

//--------------------------- 서버 시스템 ------------------

int AddInfo(); //회원가입 함수. 성공하면 리턴1 아니면 0

int DeleteInfo(); //회원 탈퇴 함수. 성공하면 리턴1 아니면 0

int DisplayInfo(); // 회원 아이디혹은 이름을 입력받고 회원 정보를 출력. // 성공하면 리턴1 아니면 0

void DisplayAllInfo(); // 모든 회원 정보를 출력.

void MemberLogin(); //아이디와 패스워드를 입력하여 로그인한다.

// 성공시 LogOnStateE 부울 값을 True로 리턴.

void MemberLogout(); //현재 로그인 된 아이디를 로그아웃 한다.

//성공시 LogOnStateE 부울 값을 False로 리턴.

//-------------------------- 서버 시스템 ----------------------

//-------------------------- 클라이언트 시스템 ------------------

void SearchFriend // 친구 번호와 각 번호에 해당하는 친구 목록 출력.

void DeleteFriend // 친구 번호를 입력하여 해당하는 친구를 목록에서 삭제.

Void AddFriend // 아이디를 입력하면 해당하는 회원의 정보가 친구목록에 추가된다.

Void StartChat // 친구 번호를 입력하면 채팅 항목을 생성하여 리스트에 추가

Void CopyChat // 채팅 기록에서 기존 채팅항목을 선택하여 메시지 전송.

Void DeleteMessage // 채팅 목록에서 목록번호를 입력하여 선택된 메시지삭제.

//-------------------------- 클라이언트 시스템 ------------------

private:

LinkedList<User> user\_List; ///< 유저 list.

LinKedList<Chat> Chat\_List; //// <채팅 리스트.

int m\_Command; ///< current command number.

bool LogonStateE; ///< 로그인상태 확인 변수.

};

1. **개발 환경**

Eclipse 3.8 C++ (in Ubuntu)