2020 Spring OOP Assignment Report

과제 번호 : 2
학번 : 20190084
이름 : 권민재
Povis ID : mzg00

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

1. 프로그램 개요

- 이 프로그램은 클래스를 이용하여 Mini SNS를 구현한 프로그램이다.
- 프로그램에서 안내하는 대로 콘솔에 입력하여 프로그램을 이용할 수 있고, 메인 메뉴에서 3을 입력하여 command.txt의 텍스트를 이 프로그램의 입력으로 포워딩 할 수 있다.
- 이 프로그램은 여러 파일로 구성되어 있으며, 별도의 디렉토리 구성은 하지 않았다.

2. 프로그램의 구조 및 알고리즘[1]

A. 입출력

Stream			
입출력 Wrapper 클래스			
private			
멤버 변수		멤버 함수	
ifstream _in	입력 스트림	void log(T &input)	로그 파일에 로그를 남기는 메서드
ofstream _out	출력 스트림	void checkEOF()	입력이 EOF 인지 확인하는 메서드
streambuf *_cin	cin 버퍼 백업	void removeLastLog()	마지막 로그를 삭제하는 메서드
public			
멤버 함수			
bool getInt(int &input)		사용자나 커맨드 파일로부터 int 입력을 수행하는 메서드	
void getLine(string &input)		std::getline 의 Wrapper 메서드	
void loadCommand()		command.txt 의 입력 스트림을 여는 메서드	
bool isLoadingCommand()		command.txt 로부터 입력받는 모드인지 여부를 반환하는 메서드	

[1] 각 클래스의 생성자와 소멸자는 표에서 생략되어 있음.

- 이 프로그램은 복잡한 입출력 구조를 가지기에, 입출력을 관리하기 위한 Stream 클래스를 제작하여 사용하였다.
- **getLine**은 이 클래스에서 중추적인 역할을 하는 기존 std::getline의 wrapper 메서드로, command.txt의 입력 스트림이 열려 있을 경우에는 파일에서, 아닐 경우에는 콘솔에서 입력을 가져오는 구조로 되어있다. 각 getLine 뒤에서는 log를 호출하여 로그를 자동으로 안전하게 기록한다.
- **getInt**는 getLine을 통해 받은 입력을 안전하게 정수형으로 변환하는 역할을 수행한다.

B. 리스트

Node	
List 의 Element 가 되는 노드 클래스	
private	
멤버 변수	
T *node_data	노드에 저장할 데이터의 포인터
Node<T> *node_next, *node_prev	다음 노드의 주소, 이전 노드의 주소
List<T> *list_parent	어떤 리스트의 노드인지 저장
public	
멤버 함수	
void setData(T *data), T *data()	노드에 저장된 데이터의 set / get 메서드
void setNext(Node<T> *next), Node<T> *next()	노드의 다음 노드 주소를 set / get 하는 메서드
void setPrev(Node<T> *prev), Node<T> *prev()	노드의 이전 노드 주소를 set / get 하는 메서드
void setParent(List<T> *parent), List<T>*parent()	노드의 부모 리스트 주소를 set / get 하는 메서드
List	
커서 리스트를 유사하게 구현한 클래스	
private	
멤버 변수	
Node<T> *list	노드 배열이 저장될 포인터
int list_size, max	값이 있는 노드의 개수와 리스트의 최대 용량
Node<T> *first, *last	리스트의 첫번째 노드 주소, 마지막 노드 주소
public	
멤버 함수	
Node<T> *add(T *data)	리스트의 마지막에 노드를 추가하는 메서드
Node<T> *drop(Node<T> *node)	리스트의 특정 노드를 삭제하는 메서드
Node<T> *insert(Node<T> *node, T *data)	리스트의 특정 위치에 새로운 노드를 삽입하는 메서드
int size()	리스트의 사이즈를 반환하는 메서드

Node<T> *begin(),*end()	리스트의 첫 / 마지막 노드의 주소를 반환하는 메서드
bool exist(T *data)	주어진 데이터가 존재하는지 확인하는 메서드
Node<T> *find(T *data)	주어진 데이터의 노드의 주소를 반환하는 메서드

□ 이 프로그램에서 거의 모든 데이터는 이 리스트를 이용해서 관리된다. **List** 클래스는 커서 리스트를 유사하게 구현한 것으로, Node 배열을 효율적으로 관리하는 역할을 수행한다. 리스트에 노드가 삽입될 때 우선 Node 배열의 빈 공간에 데이터를 할당하고, Node의 next와 prev를 연결리스트처럼 연결해서 정렬된 구조를 가지게 해준다. insert, drop을 수행할 때도 next와 prev를 적절히 연결하도록 추상화를 진행했다.

C. 유저

User	
유저를 저장하고 관리하는 클래스	
private	
멤버 변수	
string user_id, name, birthday	유저의 아이디, 이름, 생일
size_t user_password	유저의 비밀번호 해시
Friends *friendList	유저의 친구 리스트
public	
멤버 함수	
string &id()	유저의 아이디를 반환하는 클래스
void printProfile()	프로필을 출력하는 클래스
bool auth(string &id, string &password)	프로필을 출력하는 클래스
Friends &friends()	유저의 친구 리스트를 반환하는 클래스
UserList	
유저 목록을 관리하는 클래스	
private	
멤버 변수	
List<User> *list	유저 노드들을 저장할 리스트 동적할당
public	
멤버 함수	
Node<User> *addUser(Stream &s)	유저 리스트에 유저를 추가하는 메서드
Node<User> *signIn(Stream &s)	로그인 메서드
void removeUser(Node<User> *user, CommentList *commentList, PostList *postList)	유저를 삭제하는 메서드
Node<User> *getUserById(string &id)	ID 로 유저를 찾아서 반환하는 메서드

□ 유저의 일반적인 정보는 string으로 저장되지만, 비밀번호는 안전하게 저장하기 위해 std::hash를 이용하여 해시시켜서 저장하였다.

D. 게시물

Post	
Post 클래스	
private	
멤버 변수	
string content	게시글의 내용을 저장
User *post_user	게시글의 작성자를 저장
List<User> *likeList	좋아요 노드를 저장할 리스트 동적 할당
public	
멤버 함수	
void printPost(), void printPostWrapper(User *user, CommentList *commentList, Stream &s)	게시물을 출력하는 메서드
int num_like()	좋아요 개수를 반환하는 메서드
void add_like(User *target, Stream &s)	좋아요 옵션을 출력하는 메서드
bool isLiked(User *target)	주어진 유저의 좋아요 여부를 반환하는 메서드
User *user()	게시글의 작성자를 반환하는 메서드
PostList	
게시물 목록을 관리하는 클래스	
private	
멤버 변수	
List<Post> *list	게시글 노드를 저장할 배열을 동적 할당
public	
멤버 함수	
Node<Post> *addPost(User *user, Stream &s)	게시글을 추가하는 메서드
void removeUserPost(User *target)	특정 유저의 게시글을 삭제하는 메서드
void printPostList(Stream &s, User *user, CommentList *commentList, List<User> &target)	게시글 목록을 출력하는 메서드
int size()	게시글 개수를 반환하는 메서드

□ 유저가 삭제될 경우 이미 남긴 좋아요는 삭제되지 않으며, 자기 자신의 게시글에 좋아요를 남길 수 있다. 또한, 사용자가 좋아요를 남긴 적이 없다면 좋아요를 무조건 물어보는 구조로 설계되어 있다.

E. 댓글

Comment	
댓글을 저장하고 관리하는 클래스	
private	
멤버 변수	
string comment_content	댓글 내용을 저장
User *comment_user	댓글의 작성자를 저장
Post *comment_post	어떤 게시물의 댓글인지 저장
public	
멤버 함수	
void show()	댓글을 출력하는 메서드
User *user()	작성자를 반환하는 메서드
Post *post()	게시물을 반환하는 메서드
CommentList	
댓글 목록을 관리하는 클래스	
private	
멤버 변수	
List<Comment> *list	댓글들이 저장될 리스트
public	
멤버 함수	
Node<Comment> *addComment(User *user, Post *post, Stream &s)	댓글을 추가하는 메서드
void removeUserComment(User *target)	특정 사용자의 댓글을 삭제하는 메서드
void printComment(Post *target)	특정 게시글의 모든 댓글을 출력하는 메서드
int size()	전체 댓글의 개수를 반환하는 메서드

F. 친구

Friends	
친구 목록을 관리하는 클래스	
private	
멤버 변수	
List<User> *friendsList	친구 목록을 저장하기 위한 리스트 할당
public	
멤버 함수	
void addFriend(Stream &s, User *user, UserList *userList)	친구 목록에 친구를 추가하는 메서드
void removeFriendById(Stream &s, UserList *userList)	아이디를 입력받아서 친구를 삭제하는 메서드
void removeFriendByUser(User *user)	유저 포인터를 입력받아서 친구를 삭제하는 메서드

void printFriends()	친구 목록을 출력하는 메서드
List<User> &list()	친구 리스트를 참조형으로 반환

G. 메뉴

Menu			
메뉴를 관리하는 클래스			
Private			
멤버 변수		멤버 함수	
User *user	현재 유저 저장	int show(string &type, string texts[], bool isProfile, Stream &s)	메뉴를 보여주는 메서드
Node<User> *node_user	현재 유저의 노드 저장		
UserList *userList	유저 목록 주소 저장		
CommentList *commentList	댓글 목록 주소 저장		
PostList *postList	게시글 목록 주소 저장		
public			
멤버 함수			
void main(Stream &s)		메인 메뉴를 출력하는 메서드	
void myPage(Stream &s)		마이페이지 메뉴 출력 메서드	
void friends(Stream &s)		친구 메뉴 출력 메서드	
void feed(Stream &s)		피드 메뉴 출력 메서드	

- 메뉴는 콜 스택을 활용하여 각 메서드에서 다른 메서드를 호출하는 방식으로 구현되어 있다.

3. 토론 및 개선

- 비밀번호를 그냥 저장하지 않고 **해시를 이용**하여 저장함으로서 gdb와 같이 메모리를 직접적으로 볼 수 있는 동적 디버깅 환경에서도 유저의 실제 비밀번호를 노출하지 않을 수 있었다.
- 하지만, std에서 기본적으로 제공하는 해시 함수는 유일성을 완벽하게 보장하지는 않기에, **SHA3 등을 이용하여 개선할 필요**가 있다.
- 이 프로그램은 같은 유저인지 여부를 User 포인터 (주소)를 이용하여 비교하는데, **UAF**[2]로 인해서 탈퇴 후 새로 가입한 회원이 이미 탈퇴한 회원으로 인식되는 것을

[2] UAF. Use-After-Free. 힙 공간에서 사용하던 메모리가 해제된 이후 같은 곳에 재할당 되었을 때

막기 위해 **UUID를 도입할 필요가 있을 것**으로 여겨진다.

- Stream 클래스를 이용하여 입출력을 단일화하여 수행해봄으로써, 추상화의 장점을 알 수 있었다.
- 실행속도를 높이기 위해서 유저에게 **게시글 포인터 캐시, 게시물에 댓글 포인터 캐시 배열**을 달아주면 좋을 것 같다.
- 메뉴는 콜 스택을 활용하여 작동할 수 있도록 설계되어 있는데, 스택 오버플로우가 발생하지 않도록 관리할 필요성이 있다.

4. 참고 문헌

- 파일을 스트링으로 저장하는 방법
  https://stackoverflow.com/questions/9529027/c-delete-last-character-in-a-txt-file

---

일어날 수 있는 버그이다.