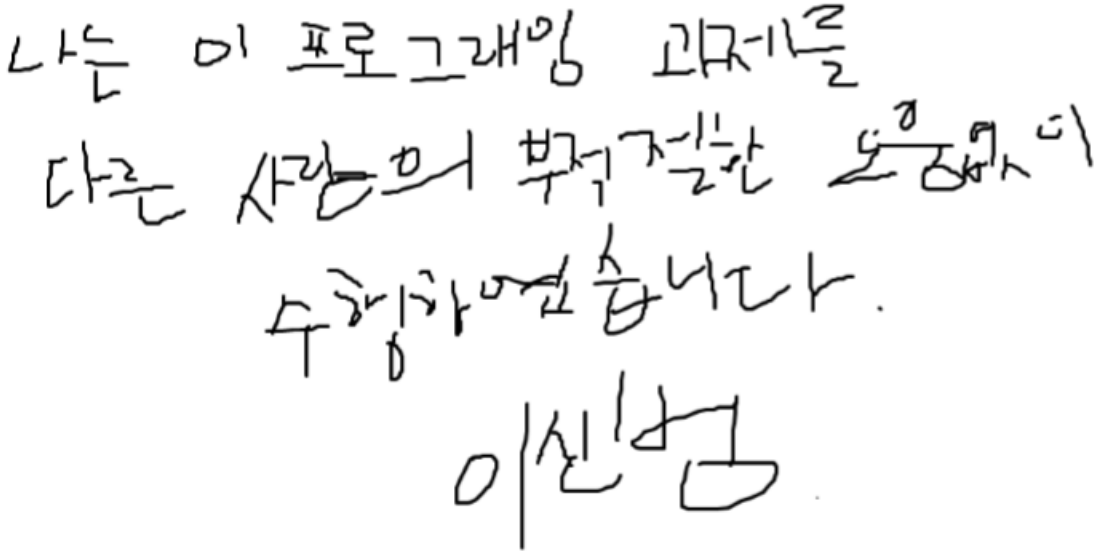
객체 지향 프로그래밍 ASSN3

20180285 이신범

무은재학부

Nm160



** 이번 어싸인에서 1번과 2번 항목까지밖에 구현하지 못했습니다. 죄송합니다.

내용을 따라가지 못한 것은 아닌데 다른 과목 숙제와 학생 활동, 동아리 활동이 시험 기간 동안 밀리면서, 시험이 끝나고 시간 조율을 잘 하지 못했던 것 같습니다.

Late를 해서라도 제출할까 생각했지만, 월요일에는 과제 수행이 시간표 상 어려울 것 같았고, 그렇게 되면 아무리 빠르게 수행한다 해도 이틀이 지난 화요일에 제출할 수 있을 것 같았습니다.

(월요일 시간표와 수요일 시간표가 거의 유사해서 최악의 경우 목요일 이후)

죄송합니다!! Assn4 부터는 다시 assn1 과 assn2와 같이 제대로 수행해서 제출하겠습니다.

Assn3는 개인적으로 코딩해보고, 혹여 화요일에 작성이 완료된다면 메일로 late 해서 제출하겠습니다.

(1) 개요

1. 어싸인 3는 C++의 꽃이라 할 수 있는 '상속'과 '다형성'의 개념을 잘 이해하고, 반복적인 동작을 이를 이용해 효율적으로 처리할 수 있는지를 판단하는 프로그래밍 과제였다. 어싸인 3를 수행하기 위해 알아야 하는 사항은 다음과 같다.

1) C++ 에서의 상속의 개념을 잘 이해하고 사용할 수 있는가?

2) C++ 에서의 다형성, 가상함수의 개념을 잘 이해하고 사용할 수 있는가?

3) 많은 데이터들을 동적 할당을 이용해 효율적으로 관리할 수 있는가?

4) Log 입출력에 관한 내용을 잘 이해했는가? (log 입출력에 대한 부분을 해결하느라 시간이 많이 소요되었습니다 ..ㅠㅜ 특히 read 하는 과정에서 buffer에 엔터가 남아 불필요한 작동이 실행된다던지 하는 문제를 해결하기 어려웠습니다)

2. Class design

총 5개의 class를 사용하였다.

```
class member {

protected:
        string RRN, ID, password;
        int info;


public:
        string back_ID() const {return ID;}
        string back_password() const { return password; }
        string back_RRN() const { return RRN; }
        virtual int back_info() const { return info; }

        member() { info = 0;   }
        member(string put_RRN, string put_ID, string put_password) {
                RRN = put_RRN;
                ID = put_ID;
                password = put_password;
                info = 0;
```

```
	}
	member(const member& put_member) {
		RRN = put_member.RRN;
		ID = put_member.ID;
		password = put_member.password;
		info = 0;

	}
	virtual ~member() {};
	};


class group_member : public member {

	string group_name;

public:
	string back_group_name() const { return group_name; }
	int back_info() const { return info; }

	group_member() : member() { info = 1; }
	group_member(const member& put_member, string put_group_name) :
member(put_member) { info = 1; group_name = put_group_name;}
	~group_member() {}

	// 기타 group_member 함수

};

class group_leader : public member {

	string group_name;

public:
	string back_group_name() const { return group_name; }
	int back_info() const { return info; }

	group_leader() : member() { info = 2; }
	group_leader(const member& put_member, string put_group_name) :
```

```
member(put_member) { info = 2; group_name = put_group_name;}
        ~ group_leader() {}

        // 기타 group_leader 함수

        };
```

위의 3가지 class를 통해 각각 다른 지위를 갖는 member들을 구현했다.

```
class vote_member {
        string vote_topic;
        string vote_sub; // 투표별 항목 개수 이거를 vote_num 배열 개수로 넘겨준다
        string vote_check[50]; // 투표한 회원의 주민번호 등록해놓기

        int vote_num[MAX_NUM_VOTEITEM] = { 0, }; // 투표 항목별 득표 현황 ++ --로 득표 현황 게시
        int vote_check_num = 0; // 처음에 0으로 되어있다가 항목에 vote 하는 member가 생기면++

public:
        vote_member(string vote_topic, string vote_sub);

        string back_vote_topic() const { return vote_topic; } // 투표 주제 반환
        string back_vote_sub() const { return vote_sub; } // 투표별 항목 개수 반환
        string back_RRN(int i) const { return vote_check[i]; } // 중복투표인지 확인하기 위해 RRN 탐색

        int back_vote_check_num() const { return vote_check_num; } // 투표 주제에 투표한 회원 수 반환
        int back_vote_num(int i) const { return vote_num[i]; } // 투표 항목 마다 투표받은 개수 반환

        void plus_vote_check_num(string RRN, int vote); // vote 한 곳에 vote 추가
};

vote_member::vote_member(string put_vote_topic, string put_vote_sub) // 받은 Token[0] [1] [2].. 넣기
```

```
{
	vote_topic = put_vote_topic;
	vote_sub = put_vote_sub;
}

void vote_member::plus_vote_check_num(string RRN, int vote) // vote -> 투표한 항목 번호
{
	vote_check[vote_check_num] = RRN;
	vote_check_num++;
	vote_num[vote]++;
}
```

위의 vote_member class를 통해 투표에 대한 내용을 나타내었다.

```
class group {
	member* leader;
	string group_name, group_RRN[MAX_GROUP_MEMBER],
group_id[MAX_GROUP_MEMBER];
	vote_member* vote_members[MAX_GROUP_VOTE]; // 해당 그룹 투표 배열
	int num_vote; // 해당 group에 존재하는 투표의 종류 개수
	int group_check_num; // 해당 group에 존재하는 인원 수

public:
	group(member* put_leader, string put_group_name)
	{
		leader = put_leader;
		group_name = put_group_name;
		num_vote = 0;
		group_check_num = 0;
	}

	void group_put(string put_RRN, string put_id,int i)
	{
		group_RRN[i] = put_RRN;
		group_id[i] = put_id;
		group_check_num++;
	}
```

```
        string back_group_name () const { return group_name; }
        string back_group_RRN(int i) const { return group_RRN[i]; }
        string back_group_id(int i) const { return group_id[i]; }
        int back_group_check_num() const { return group_check_num;   group_check_num; }
};
```

마지막으로, group class를 통해 main 함수에 group 함수를 선언하고, group 멤버 관리를 가능하게 하였다.

3. 마지막으로, 어싸인 3를 수행하기 위해 Log 입출력을 사용했다.

   1) 커맨드 로그 기록은 최대 6개 까지 존재할 수 있었기에, token의 최대개수를 6이라 두었다.

   2) 로그 입출력 함수는 다음을 사용했다.

   Ifstream inLogRead // commanLog.txt 파일로부터 커맨드 정보를 읽는 객체

   Ofstream inLogWrite // commandLog.txt에 수행된 커맨드를 기록하는 객체

   3) 커맨드 파일 열기

      커맨드를 stdin으로부터 받는 경우

      InLogWrite.open(파일이름, ios::out)을 사용했다.

      커맨드를 파일로부터 읽는 경우

      InLogRead.open(파일이름, ios::in)

      InLogWrite.open(파일이름, ios::in|ios::app)를 이용했다. 이 때 app는 append의 약자로, 파일을 이어서 작성한다는 뜻이다.

   4) Log read는 string tokens를 이용했다.

(2) 알고리즘 및 설명

<structure chart>

Main

I

Logread

I

Dotask

I

Logwrite

I

Dotask

I

Exit

<flow chart> 각각의 menu 1.1 ~1.3 , menu 2.1 ~ 2.9에 대한 구조도는 다음과 같다.

1-1) Join

Input : RRN, ID, password

I

Valid_test

I

Let

I

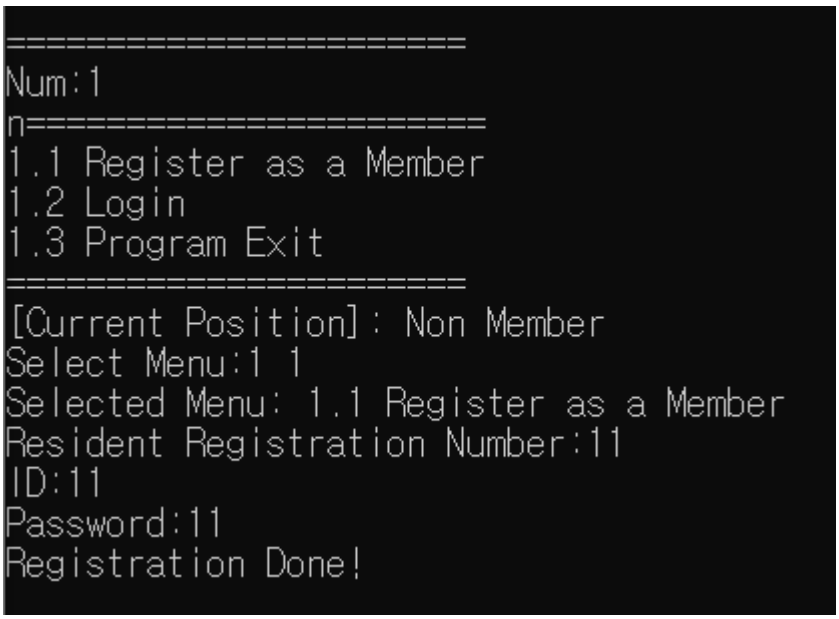
If vaild case

Members 추가

I

If not valid case

Return;



1-2) Login

Input : ID, password

I

Valid_test

I

Let

I

If vaild case

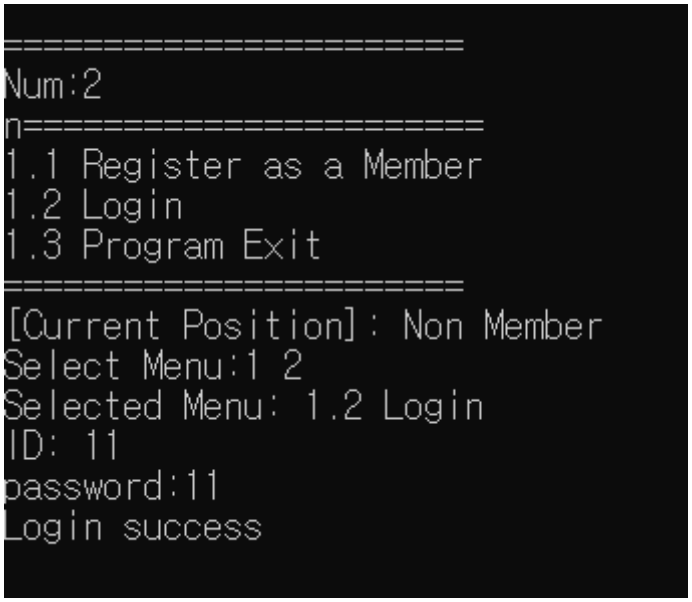
Login

알맞은 temp에 주솟값 추가

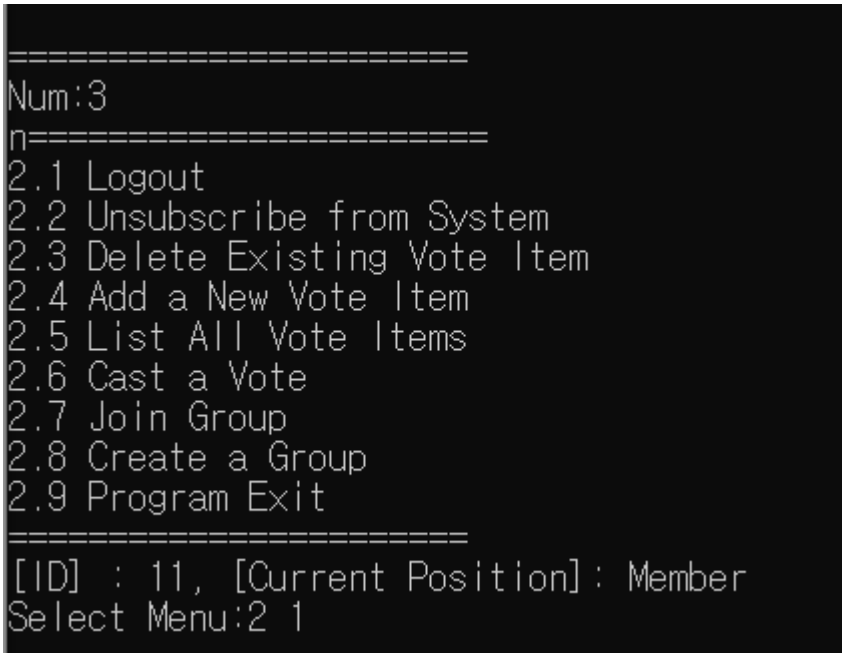
I

If not valid case

Return;



로그인 이후에는 member 모드이므로, 다음과 같이 menu가 변경된다



1-3) Program exit

Return;

2-1) logout

Input : RRN, ID, password

I

Valid_test

I

Let

I

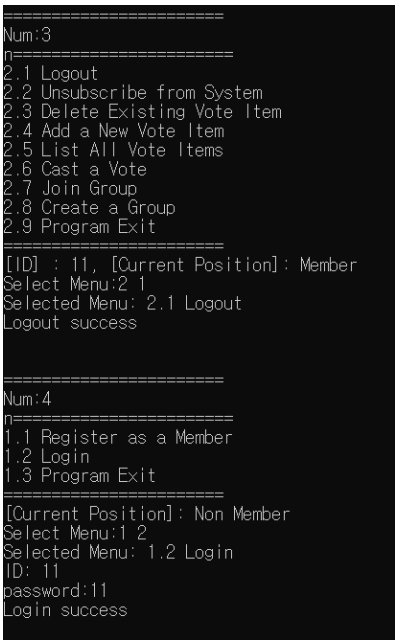
If vaild case

Members 추가

I

If not valid case

Return;



2-2) unjoin

Input : ID, password

I

Valid_test

I

Let

I

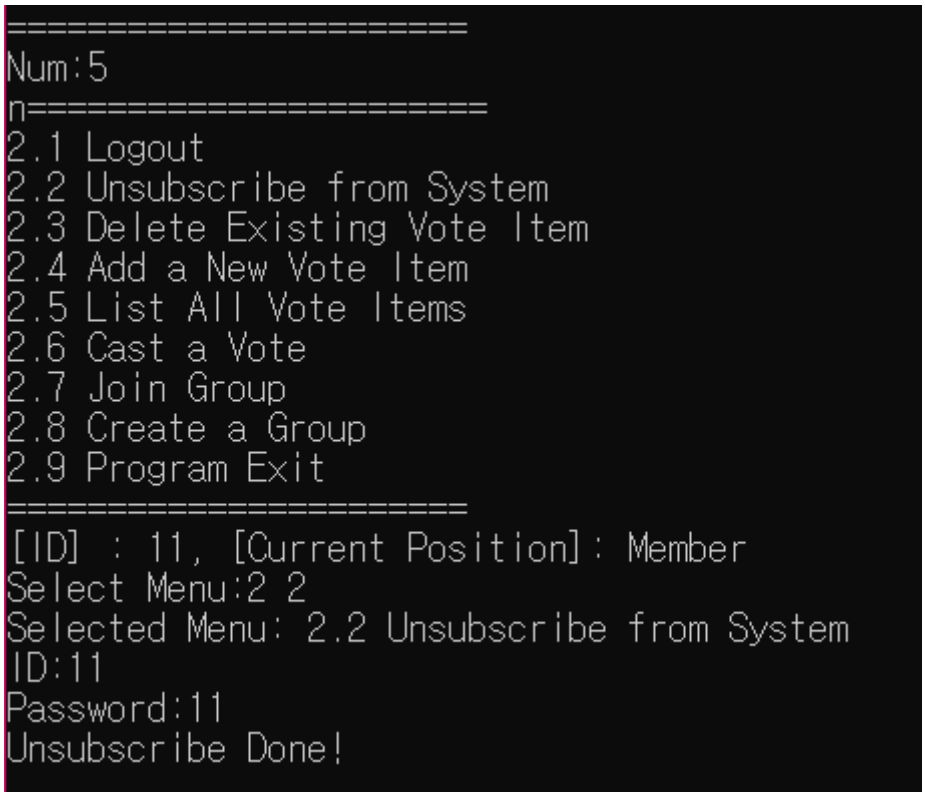
If vaild case

Members 제거하기 (member**배열 이용)

I

If not valid case

Return;



2-3) delete_vote

Input : vote_topic, num_item

I

Valid_test

해당 투표 주제가 존재하는지

삭제를 요청한 회원이 로그인 되어있는지

I

Let

I

If vaild case

Vote_member 제거하기 (vote_member** 배열 이용)

I

If not valid case

Return;

2-4) add_vote

Input : vote_topic, num_item;

I

Valid_test

투표명이 중복되지 않는지 확인

I

Let

투표를 요청한 회원이 로그인 되어있는지 확인

I

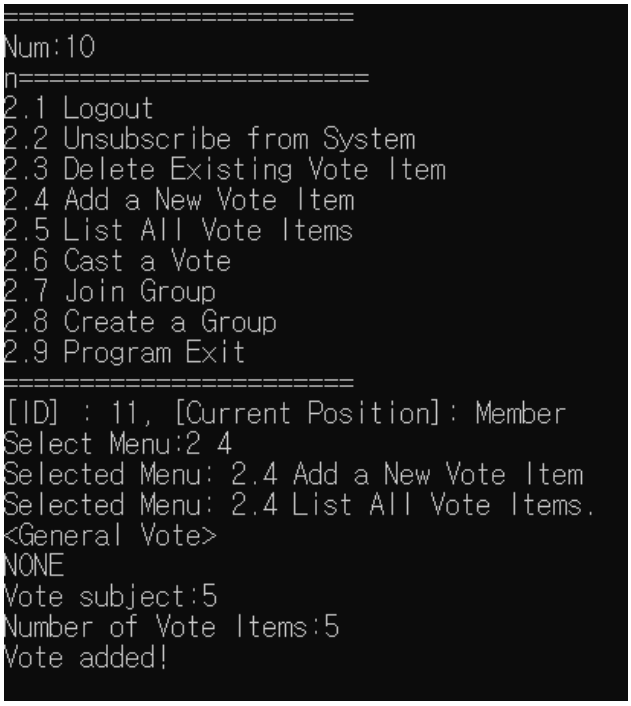
If vaild case

투표 추가하기 (vote_member**배열 이용)

I

If not valid case

Return;



2-5) view_vote

Input : 없음

I

Valid_test

로그인된 사용자에 의해 신청되었는지 확인한다

I

If vaild case

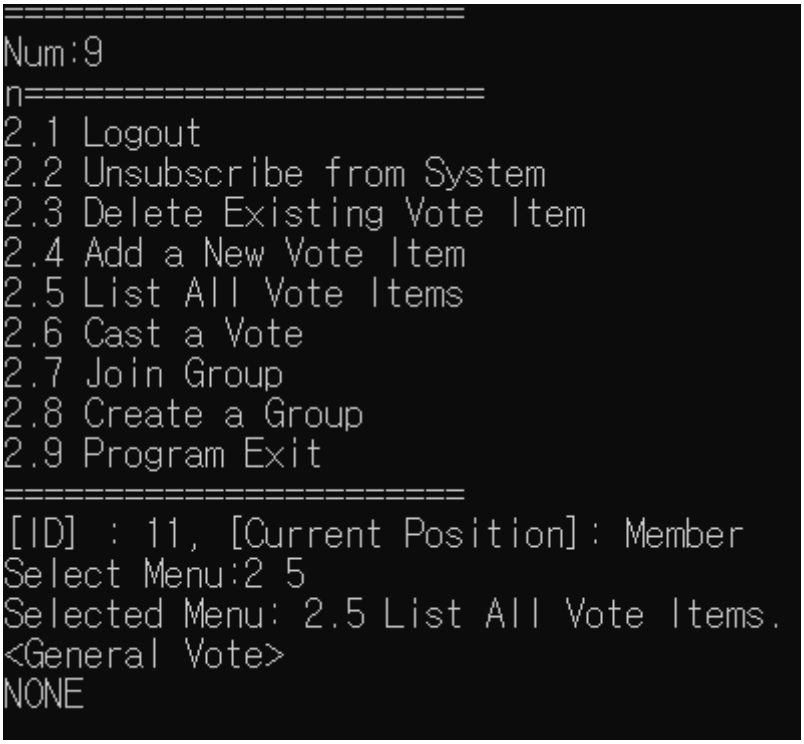
Menu1 이 2면 general vote만 출력하고

Menu1 이 3이나 4면 general vote와 group vote를 출력한다

I

If not valid case

Return;



2-6) cast_vote

Input : vote_topic, num_item

I

Valid_test

회원이 로그인 되어 있는지

입력받은 투표가 존재하는지

I

Let

이미 투표한 회원인지 체크

I

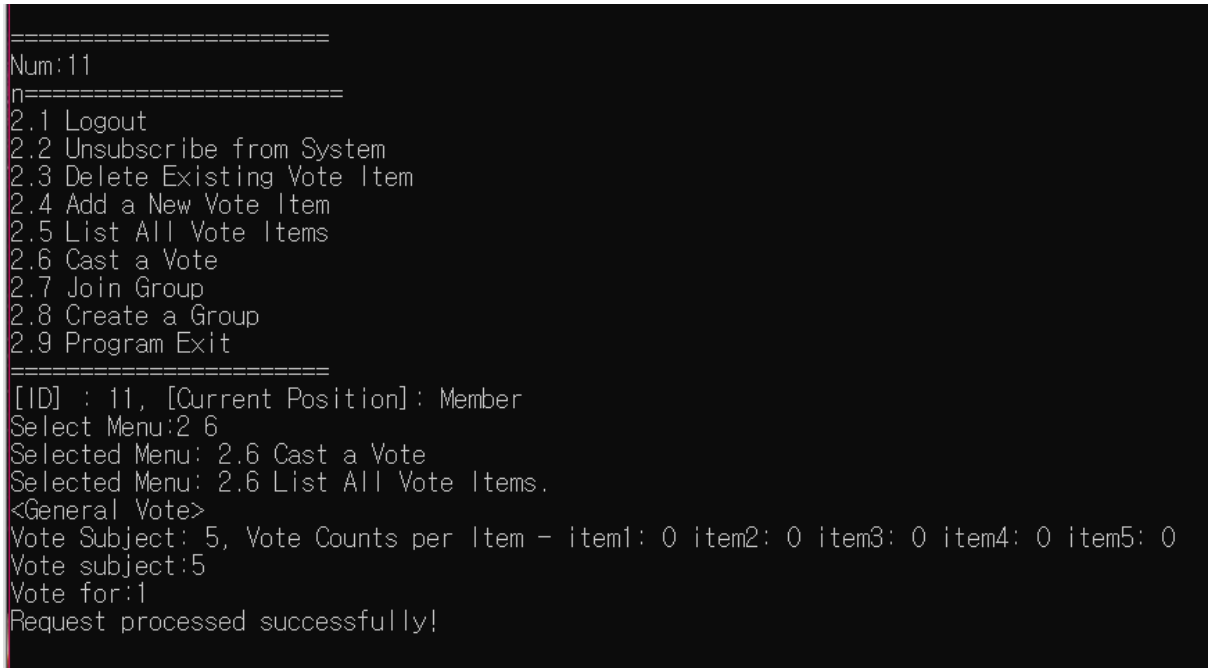
If vaild case

투표하기 (만약, menu1이 3이나 4일 경우 2번 투표가능하도록 설정)

I

If not valid case

Return;



2-7) join_group

Input : select_group

I

그룹 목록 출력 함수 넣기

i

Valid_test

존재하는 group 인지 확인

I

Let

I

If vaild case

Group class 생성하기

Member 를 group_member로 바꿔주기

I

If not valid case

Return;

2-8) create_group

Input : group_name

I

Valid_test

그룹에 가입된 적 없는 사용자인지 확인

I

Let

I

If vaild case

Group 생성하기

Member를 group_leader로 바꿔주기

I

If not valid case

Return;

2-9) program exit

Return;

(3) 토론

이번어싸인에서 3-.. , 4-.. 항목을 작성하지 못해 아쉬웠다.

각 멤버별 수행 가능한 메뉴			
1. Non Member	**2. Member**	**3. Group Member**	**4. Group Leader**
1.1 Register as a Member	2.1 Logout	3.1 Logout	4.1 Logout
1.2 Login	2.2 Unsubscribe from System	3.2 Unsubscribe from System	4.2 Delete Existing Vote
1.3 Program Exit	2.3 Delete Existing Vote Item	3.3 Delete Existing Vote	4.3 Add a New Vote
	2.4 Add a New Vote	3.4 Add a New Vote	4.4 List All Vote Items
	2.5 List All Vote Items	3.5 List All Vote Items	4.5 Cast a Vote
	2.6 Cast a Vote	3.6 Cast a Vote	4.6 List All Group Members
	2.7 Join a Group	3.7 Unregister from Group	4.7 Ban a Group Member
	2.8 Create a Group	3.8 Program Exit	4.8 Program Exit
	2.9 Program Exit		

사실 나머지 부분은 코딩을 하기 전에 전체적인 psedocode를 작성해보면서

이미 비슷한 역할을 할 수 있는 (상속의 여지가 보이는) 함수들을 묶어 놓았기 때문에, 이를 이용해 코딩을 해주면 되었지만,, 앞서 작성드린 내용대로 시간 조율을 잘 하지 못해 기간 내에 작성을 하지 못해 아쉬웠다.

이번 어싸인에는 log 입출력에 대한 개요가 없어 처음부터 시간이 많이 소요되었다. 물론 이전 어싸인에서 했던 방식을 응용하면 됐지만, 사실 이전 어싸인에서 사용되었던 atoi 함수나 .c_str() 함수 같은 것들을 자세히 알지 못했기 때문에, 내가 직접 상황에 맞추어 적으려니 하나부터 열까지 직접 이해해가며 작성하는데 시간이 많이 소요됐다.

특히 menu 입력을 1 이나 12와 같이 하나로 받는 것이 아니라, 1 1 이나 2 1과 같이 두 개의 입력으로 받았기 때문에, 이 부분을 어떻게 log 입출력으로 변환할지 생각하는게 어려웠다.

나는 이 답을

```
menu1 = atoi(line.substr(0).c_str());
menu2 = atoi(line.substr(2).c_str());
```

와 같이 구현함으로써 해결했다. (결국 switch 문을 이중으로 두어 해결했다는 뜻이다.)

(4) 결론

상속이라는 것은 기존의 C 언어를 사용하며 폭풍 ctrl c + ctrl v 했던 것을 효율적으로 해결해주는 구조라고 할 수 있다.

Is a 관계와 has a 관계를 잘 이해할 수 있다면, 상속으로 효율적으로 구현할 수 있으

니, 무턱대고 코딩을 시작하는 것보다 이전에 먼저 전체적인 개요와 함수 프로토타입을 작성하면서 psedocode를 써보는 것이 왜 더 효율적인 프로그래밍을 완성시킬 수 있는지 다시 한 번 뼈저리게 알 수 있을 것 같았다.