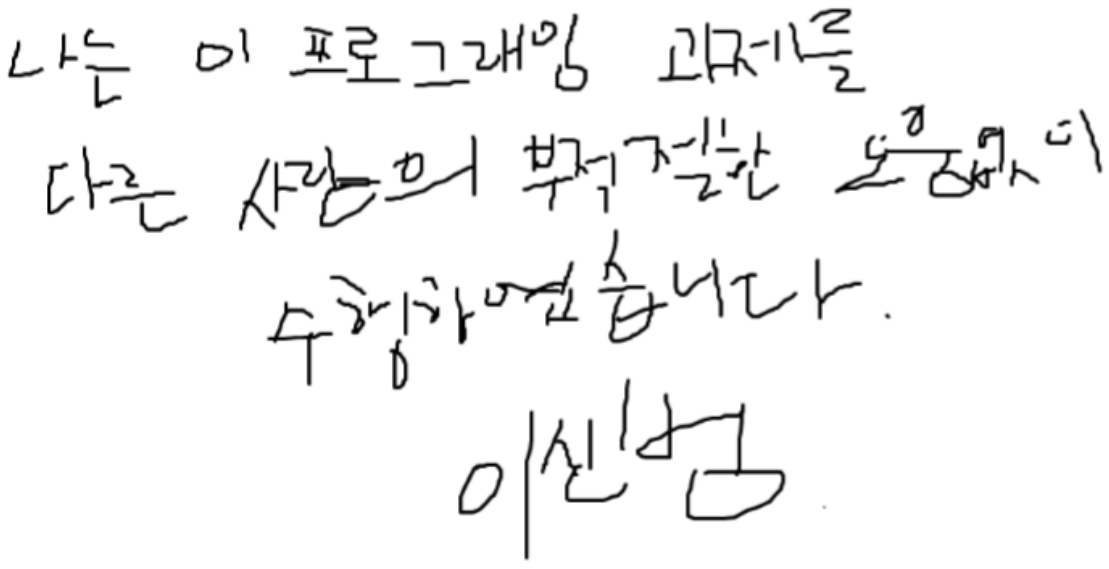
객체 지향 프로그래밍 ASSN2

20180285 이신범

무은재학부

nm160

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.



(1) 개요

1. 어싸인 2는 C++의 기본 사항들을 잘 숙지하고 이를 활용할 수 있는지를 판단하는 프로그래밍 과제였다. 어싸인 2를 수행하기 위해 알아야 했던 사항은 다음과 같다.

1) C++ 에서의 Class 객체를 잘 이해하고 이를 사용할 수 있는가?

2) C++ 에서의 Class 배열을 사용하고 new 와 delete 를 통해 동적할당 할 수 있는가?

3) Log 입출력에 관한 내용을 잘 이해했는가? (조교님께서 짜주신 코드를 통해 Log 입출력을 비롯한 Token의 사용, string 객체의 사용 등을 잘 이해할 수 있었습니다!)

2. 어싸인 2를 수행하기 위해서는 Class에 대한 이해가 선행되어야 했다.

Class design을 통해 member class와 vote_member class 총 2개의 클래스를 사용하였다.

Class member

Private 인자 : stirng RRN, ID, password

Public 인자 : member(RRN, ID, password) // 기본 생성자로 사용했다.

Stirng back_RRN() const;

Stirng back_ID() const;

String back_password() const;

다음의 인자들은 회원의 RRN, ID, password 값이 필요할 때 반환값을 이용할 목적으로 선언해놓고 사용했다.

Class vote_member

Private 인자 : string : vote_topic, vote_sub, vote_check[50]

Int vote_num [10] , vote_check_num;

Vote_topic과 vote_sub은 투표 주제와 투표 항목 수를 저장하기 위해 사용했다.

Vote_check는 회원의 중복투표를 막기위해 투표한 회원의 RRN을 저장한다.

Vote_num[10]은 투표주제에 존재하는 항목의 수를 저장하는 배열이며

Vote_check_num은 투표주제에 투표한 회원의 수를 저장하는 변수이다.

Public 인자 :

vote_member(string vote_topic, string vote_sub); // 기본생성자로 사용했다.

string back_vote_topic() const;
string back_vote_sub() const;
투표 주제 및 투표 항목 수가 필요할 때 반환값을 이용할 목적으로 선언했다.

string back_RRN(int i) const;
회원의 중복투표를 막기위해 비교할 때 vote_check에 존재하는 RRN을 사용했다.

int back_vote_check_num() const;
int back_vote_num(int i) const;

void plus_vote_check_num(string RRN, int vote);

투표한 회원의 RRN 저장, vote_check++의 기능을 구현했다.

3. 마지막으로, 어싸인 2를 수행하기 위해서는 Log 입출력에 대한 이해가 필요했다.

1) 커맨드 로그 기록은 모두 다음의 형식을 따랐다.

[menu] [입력값] [입력값] [입력값]

[ ] 로 구분되는 string 들은 " " 을 기준으로 token에 저장함으로써 구분했다.

2) 로그 입출력 함수는 다음과 같이 사용했다.

Ifstream inLogRead // commandLog.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) 의 형태를 사용했다.

커맨드를 파일로부터 읽는 경우

InLogRead.open(파일이름. Ios::in)

InLogWrite.open(파일이름, ios::in|ios::app)를 이용했다. 이 때 app 는 append 의 약자로 파일을 이어서 작성한다는 이야기이다.

4) Log read 는 string tokens를 이용함으로써 해결했다.

(2) 알고리즘 및 설명

이전에 보고서에서 가독성에 1점 감점을 받아, flow chart와 structure chart를 아이패드로 그리는 것이 아니라, 깔끔해 보일 수 있도록 타이핑 했습니다

<structure chart>

Main

I

LogRead

I

DoTask

I

LogWrite

I

DoTask

I

Exit

<flow chart> 각각의 menu 1 ~ 9 에 대한 구조도를 그려보겠다.

1) Register

Input : RRN, ID, password

I

중복된 RRN이 있는지 CHECK

I

로그인 되어있는지 CHECK

I

IF VAILD CASE

Member[*mem_num] = new member로 클래스 동적할당

*mem_num++

I

Return;

I

IF UNVALID CASE

RETURN;

<알고리즘>

MenuParser 함수로부터 member** members , int* mem_num, member* temp 를 전달받는다.

Members 는 class를 인자로 같는 클래스 배열이고

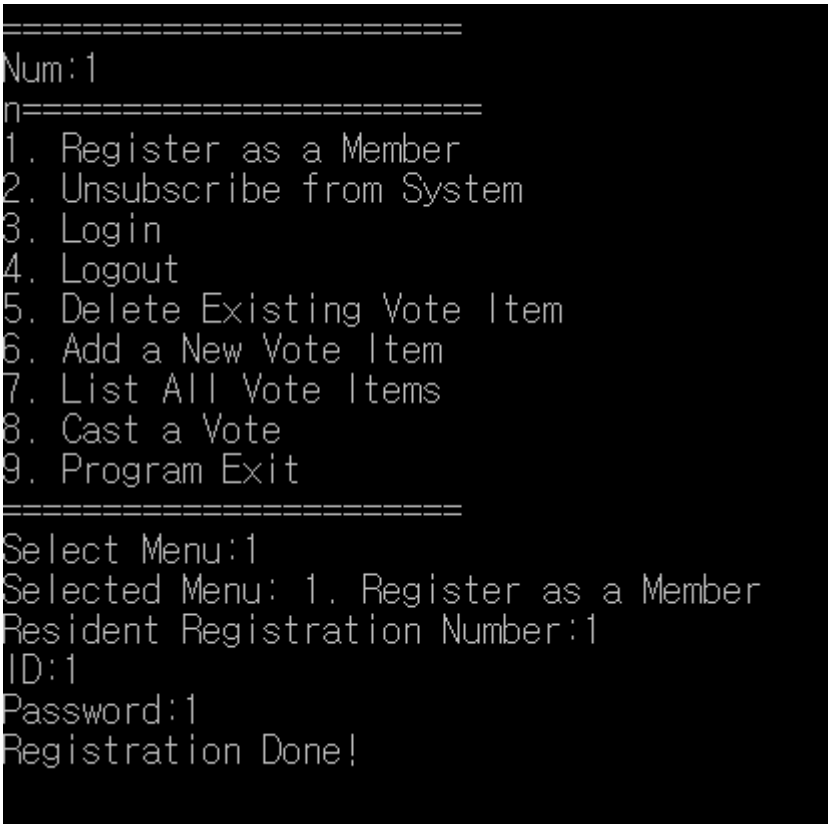
Mem_num 은 현재 등록되어 있는 회원의 수이다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

If valid case

Members 배열에 new member(RRN, ID, password)로 클래스 하나를 추가 동적할당.

회원이 추가되었으므로 (*mem_num)**을 해주면 된다.



회원 등록 화면

2) Unsubscribe

INPUT: ID, PASSWORD

I

해당하는 ID와 PASSWORD를 가진 CLASS가 등록되었는지 확인

I

로그인 된 사용자에 의해 실행되었는지 확인

I

IF VALID CASE

DELETE MEMBERS[SELECTED];

클래스 배열의 원소를 삭제된 INDEX 뒤에서부터 한 칸씩 앞으로 당겨준다

*MEM_NUM—

RETURN;

I

IF UNVALID CASE

RETURN;

<알고리즘>

MenuParser 함수로부터 member** members , int* mem_num, member* temp 를 전달받는다.

Members 는 class를 인자로 같는 클래스 배열이고

Mem_num 은 현재 등록되어 있는 회원의 수이다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

만약 *temp != NULL 이라면 그것은 로그아웃된 상태를 의미하므로 동작할 수 없다.

If valid case

(*temp) = NULL / 삭제된 회원은 로그아웃 될 것이다.

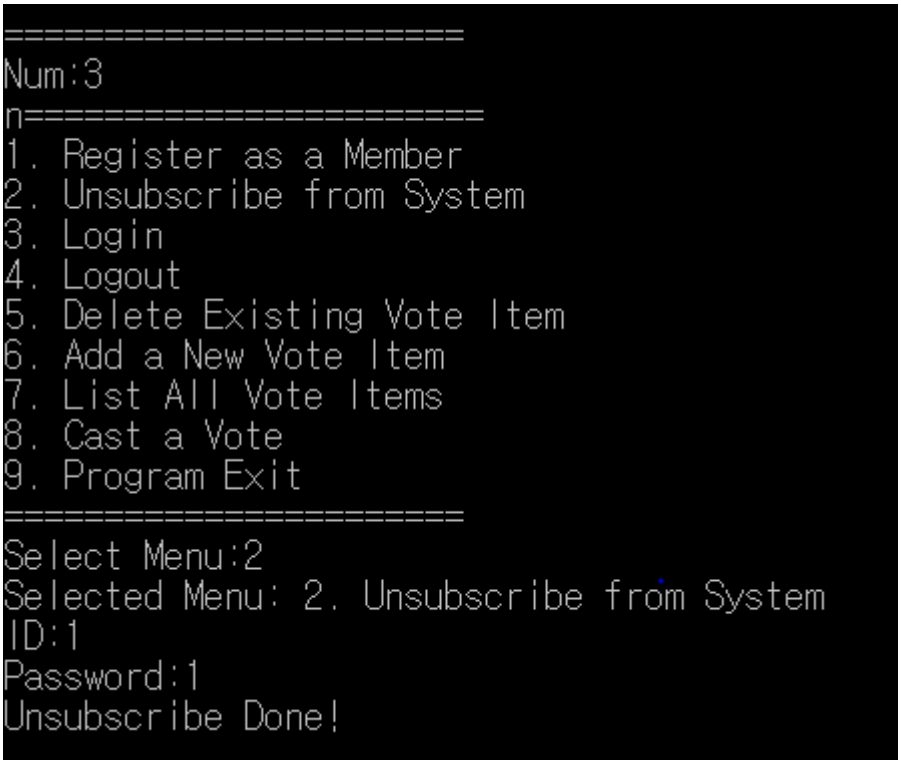
Delete members[select]를 통해 동적할당 했던 것을 free 해주고

For (int I = select_member_num; i< *mem_num -1 ; i++)

Members[i] = members[i+1];

을 통해 배열의 원소들을 한 칸씩 앞으로 당겨준다

멤버가 삭제되었으므로 (*mem_num)--;



회원 탈퇴

3) Login

INPUT : ID, PASSWORD

I

입력된 ID와 PASSWORD를 가진 회원이 등록되었는지 확인

I

로그인 되지 않은 사용자에 의해 요청되었는지 확인

I

IF VALID CASE

*TEMP = MEMBERS[SELECTED] // temp는 class의 주소를 저장하는 포인터

Return;

I

IF UNVALID CASE

RETURN;

<알고리즘>

MenuParser 함수로부터 member** members , int* mem_num, member** temp 를 전달받는다.

Members 는 class를 인자로 같는 클래스 배열이고

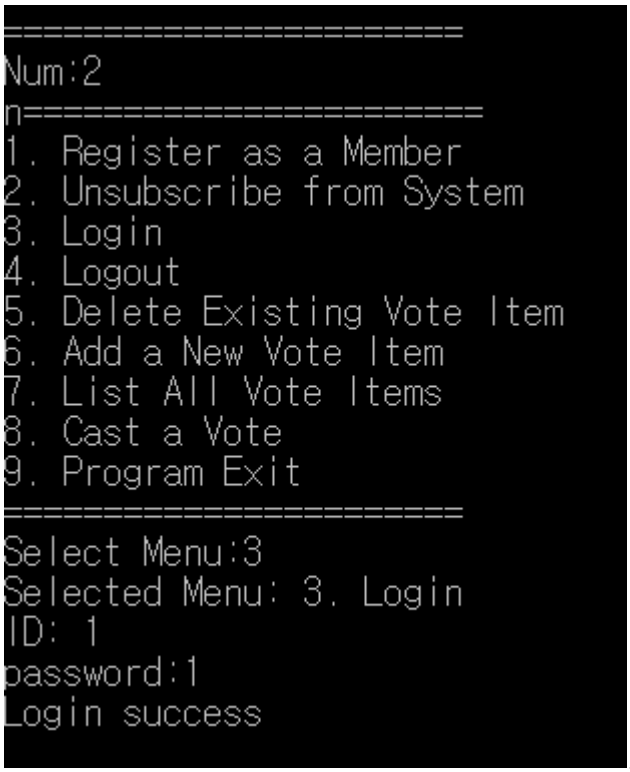
Mem_num 은 현재 등록되어 있는 회원의 수이다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

Logout된 상태에서 요청되었는지 , login 요청한 회원이 등록되어 있는지 확인해준다.

If valid case

*temp = members[select] 를 통해 temp에 class의 주소를 저장해준다.



로그인 화면

4) Logout

LOGIN 된 사용자에 의해 요청되었는지 확인

I

IF VALID CASE

*TEMP = NULL / TEMP가 NULL 이면 로그아웃 된 상태를 의미한다

RETURN;

I

IF UNVALID CASE

RETURN;

<알고리즘>

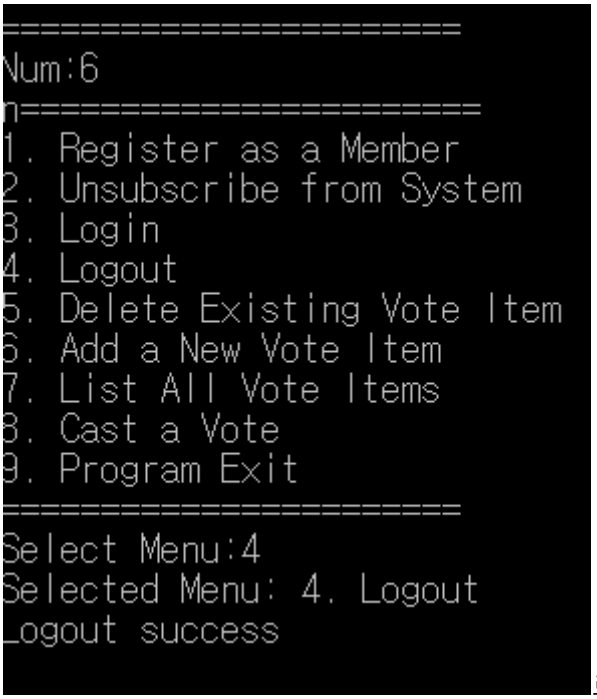
MenuParser 함수로부터 member** temp 를 전달받는다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

Temp에 주솟값이 저장되어 있는지 확인한다.

If valid case

*temp = NULL



로그아웃

5) Delete vote

INPUT : string VOTE_TOPIC, NUM_ITEM

I

해당 투표가 존재하는지 체크

I

삭제를 요청한 회원이 로그인 되어 있는지 확인

I

IF VALID CASE

Delete vote_members[selected]

삭제된 인덱스의 뒤에서부터 클래스 배열을 한 칸씩 당겨준다

*sub_num—

Return;

I

IF UNVALID CASE

RETURN;

<알고리즘>

Menuparser 함수로부터 vote_member** members, int* sub_num, member* temp를 전달 받는다.

Members 는 class를 인자로 같는 클래스 배열이고

sub_num 은 현재 등록되어 있는 투표의 수이다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

5, 6, 7, 8 번의 모든 동작은 '로그인 된 상태'의 회원으로부터 요청되어야 한다.

해당 투표가 존재하는지 체크

삭제를 요청한 회원이 로그인 되어있는지 체크

If valid case

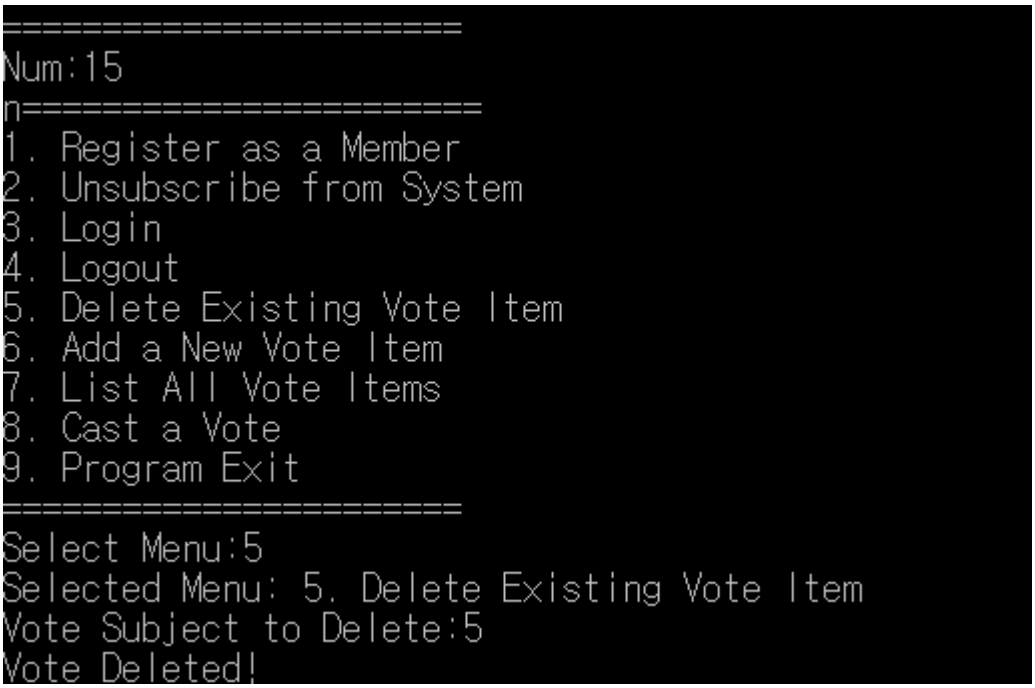
Delete members[select]를 통해 동적할당 하였던 투표 class를 free 해준다

For (int I =select; I < *sub_num; i++)

Members[i] = members[i+1];

을 통해 삭제된 배열의 빈자리를 채우기 위에 배열을 한 칸식 땅겨온다.

투표가 하나 삭제되었으므로, (*sub_num)--;



투표 삭제

6) Add vote

INPUT: VOTE_TOPIC, NUM_ITEM

I

로그인 된 사용자에 의해 요청되었는지 확인

I

ADD VOTE를 신청한 투표 주제가 중복되지 않는지 확인

I

IF VALID CASE

Vote_members[*sub_num] = new .. 동적할당

(*sub_num++);

Return;

I

IF UNVALID CASE

Return;

<알고리즘>

Menuparser 함수로부터 vote_member** members, int* sub_num, member* temp를 전달받는다.

Members 는 class를 인자로 같는 클래스 배열이고

sub_num 은 현재 등록되어 있는 투표의 수이다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

5, 6, 7, 8 번의 모든 동작은 '로그인 된 상태'의 회원으로부터 요청되어야 한다.

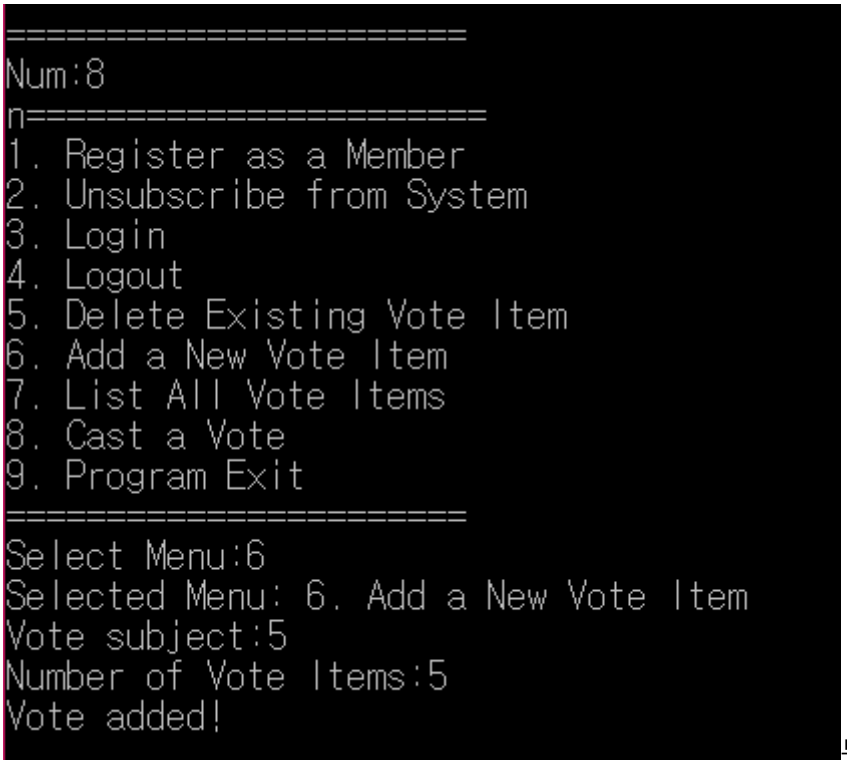
로그인 된 회원으로부터 요청받았는지 체크

투표 주제가 중복되지 않는지 체크

If valid case

Members[*sub_num] = new .. 를 통해 동적할당!

투표 주제가 하나 추가되었으므로, (*sub_num)++ 을 해준다.



투표 추가

7) List vote

로그인 된 사용자에 의해 요청되었는지 확인

I

If valid case

투표 주제 출력

Return;

I

If unvalid case

Return;

Menuparser 함수로부터 vote_member** members, int* sub_num, member* temp를 전달 받는다.

Members 는 class를 인자로 같는 클래스 배열이고

sub_num 은 현재 등록되어 있는 투표의 수이다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

5, 6, 7, 8 번의 모든 동작은 '로그인 된 상태'의 회원으로부터 요청되어야 한다.

로그인 된 회원으로부터 요청받았는지 확인

If valid case

```
for (int i = 0; i < sub_num; i++) {

                    cout << "Vote Subject: ";
                    istringstream ss(members[i]->back_vote_sub());
                    ss >> num;

                    cout << members[i]->back_vote_topic() << ", Vote Counts per Item -";
fflush(stdout);

                    for (int j = 0; j < num; j++) {

                              cout << " item" << j + 1 << ": " << members[i]-
>back_vote_num(j); fflush(stdout);
                    }

                    cout << endl; fflush(stdout);
          }
```

투표 주제 출력 -> 투표 항목 개수만큼 for 문 -> 투표 항목의 int 출력의 알고리즘이다.

```
======================
Num:13
n======================
1. Register as a Member
2. Unsubscribe from System
3. Login
4. Logout
5. Delete Existing Vote Item
6. Add a New Vote Item
7. List All Vote Items
8. Cast a Vote
9. Program Exit
======================
Select Menu:7
Selected Menu: 7. List All Vote Items.
Vote Subject: 5, Vote Counts per Item - item1: 1 item2: 0 item3: 0 item4: 0 item5: 0
```

투표 출력

8) Cast vote

Input : string vote_topic, num_item;

I

로그인 되어있는지 확인

I

입력 받은 투표 주제가 등록되어 있는지 확인

I

이미 주제에 대한 투표를 한 회원인지 RRN을 체크

I

If valid case

투표 주제 class에 회원의 RRN 저장

투표 주제의 항목을 ++ 해준다.

Return;

I

If unvalid case

Return;

<알고리즘>

Menuparser 함수로부터 vote_member** members, int* sub_num, member* temp를 전달 받는다.

Members 는 class를 인자로 같는 클래스 배열이고

sub_num 은 현재 등록되어 있는 투표의 수이다.

Temp 는 class의 포인터로, 만약 로그인 된 회원이 있다면 로그인 된 회원의 class 주소를 갖고 있을 것이며, 만약 temp가 비어있다면 그것은 로그아웃된 상태를 의미한다.

5, 6, 7, 8 번의 모든 동작은 '로그인 된 상태'의 회원으로부터 요청되어야 한다.

로그인 된 회원으로부터 요청받았는지 확인

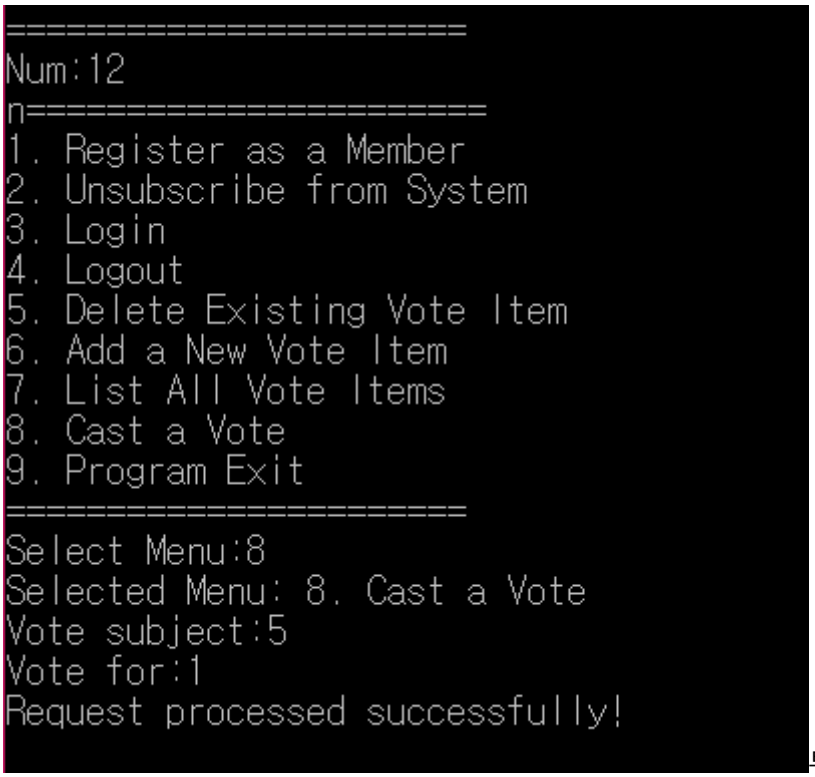
입력 받은 투표 주제가 존재하는지 체크

투표 주제에 한 번 투표했던 RRN은 아닌지 체크

If valid case

해당 투표 항목의 투표 수 ++

해당 투표 주제에다가 투표한 회원의 RRN 저장하기!



투표

9) Exit system

Return;

(3) 토론

이번 어싸인을 하며 가장 어려웠던 부분은 딱 한가지 였다.

우리 눈에 단순한 int 형 변수로 보이는 숫자 하나까지도 tokens 배열을 통해 "string" 으로 받는데, 이것을 어떻게 컴퓨터가 int 로 인식하게 만들까?

나는 그것에 대한 답을 istringstream 을 이용함으로써 해결했다.

String 변수를 int 변수로 바꾸는 간단한 코드를 보이자면, 다음과 같았다.

```
#include<sstream>
Int main(void)
{
    String s = "7";
    Istringstream ss(s);
    Int k;
    Ss >> k;
}
```

이렇게 하면 s에 string 으로 저장되었던 7이라는 변수가 k 에는 int 7로 저장되게 된다. 이를 통해서 가장 어려웠던 문제를 해결할 수 있었다.

(4) 결론

C++ 을 통해 객체지향프로그래밍을 한다는 것은 참 놀라운 일인 것 같다. 나는 이번 어싸인을 통해 내 인생에서 처음으로 C++을 활용한 어싸인을 제대로 해본 것 같다.

Class를 이용함으로써 각각의 객체가 자신만의 변수, 자신만의 함수를 갖게 되니까 함수 오버로딩의 문제도 해결되서 함수와 변수 이름도 중복의 위험 없이 보다 직관적으로 활용할 수 있어서 좋았다.

다만 불편했던 점은, class에 저장된 private 변수의 '값'만을 활용하고 싶다고 하더라도 그 값을 직접 사용하지 못하고 퍼블릭 함수를 통해 반환받아야 했다는 점이었다.