c++프로그래밍 및 실습

스쿼드 메이커

최종 보고서

제출일자:2023.12.24
제출자명:김성준
제출자학번:182659

1. 프로젝트 목표

1) 배경 및 필요성

코로나 19 이후 , 야외 활동이 늘어나면서 축구나, 야구 등 스포츠 관람 뿐만 아니라 직접 동호회나 동아리 등 모임에 가입해서 직접 팀 스포츠에 참여 하는 비중이 늘어나고 있습니다. 프로 스포츠 클럽들과 다르게 동호회나 동아리 등 취미를 목적으로 하는 모임은 참여도가 비교적 자유로워서 경기에 참가하는 인원을 정하고 분배하는데 어려움이 있습니다. 따라서 이 과정을 자동화 하는 프로그램을 개발하는데 목표를 두었습니다.

2) 프로젝트 목표

우선 순위 큐 라는 자료구조를 이용하여 포지션 선호도나 개인 일정 등을 고려하여 경기 참여 선수 분배를 자동화 하는 '스쿼드 메이커' 라는 프로그램을 개발한다.

3) 차별점

선수 선발인원을 정하는 기존 방식은, 각 경기 일정마다 참여할 인원을 모집하고, 참여 인원이 모이면 그 인원에 맞게 사람이 직접 해당 경기의 쿼터수를 고려하여 사람들의 개인일정, 피로도, 선호도 등을 고려해야하는 수고스러운 방식이였습니다. 따라서 기존의 방식은 시간도 오래 걸릴 뿐만 아니라 개인 사정이나 선수들의 부상과 같은 돌발 상황에 취약합니다. 또한 , 각자 선호 하는 포지션에서 플레이 하지 못하거나 공평하게 분배 되지 않을 우려가 있습니다. 따라서 이러한 과정을 자동화 하여 돌발 상황에 대처 할수 있고, 더 빠르게 사람의 수고스러움을 줄여줄수 있는 데에 차별점이 있습니다.

2. 기능 계획

1) 선택 메뉴 출력 기능

- 사용자의 요구에 따라 원하는 메뉴를 출력하여 보여주거나 입력을 저장합니다.

(1) 선택 메뉴 구성

1. 선수 정보 입력/수정/삭제 항목

2. 포메이션 정보 입력/수정/삭제 항목

3. '스쿼드메이커' 항목

2) 선수 정보 입력/수정/삭제 항목 기능

- 선수의 이름, 선호 포지션/비선호 포지션, 등번호를 저장하는 항목입니다.

3) 포메이션 정보 입력/수정/삭제 항목 기능

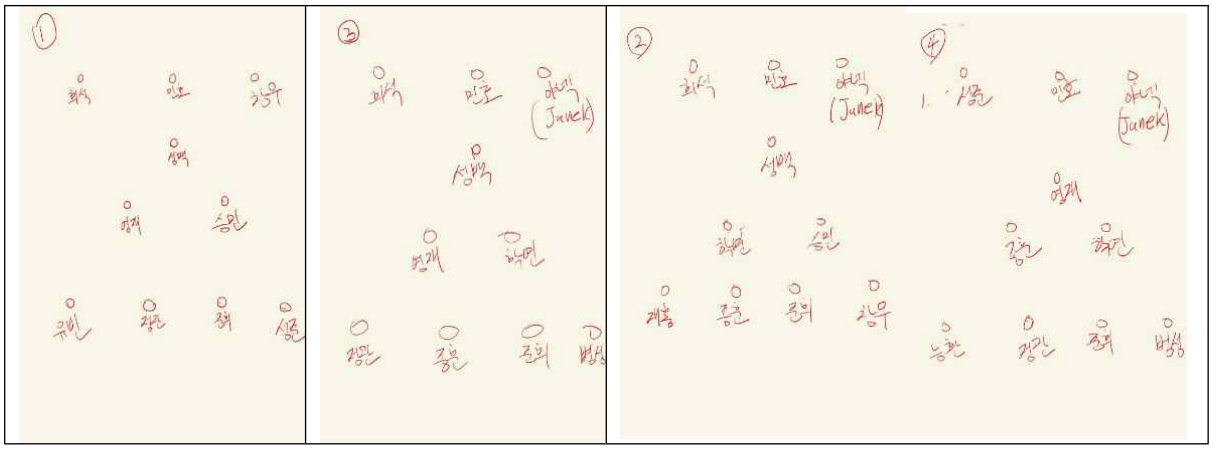
- 경기에서 사용할 포메이션을 저장하는 항목입니다. 주로 팀스포츠는 인원에 따라, 전술에 따라 각 자리의 역할이 달라집니다. 축구를 예로 들자면, 포메이션에 따라 수비수나 미드필더 공격수의 숫자가 달라집니다. 5-2-3 포메이션의 경우 공격-중앙-수비에 각각 5명·2명·3명을, 4-3-3 포메이션의 경우에는 4명·3명·3명을 배치합니다.

4) 스쿼드 메이커 기능

- 입력된 선수정보, 포메이션 정보를 바탕으로 대전방식/참가자수/포메이션/쿼터수/개인사정 등의 사용자의 입력을 받아 우선순위 큐를 바탕으로 각 쿼터에 인원을 배치합니다.

예시) 다음은 실제 경기에서 사용하기 위해 기존의 방식으로 작성한

14명의 참여 인원/ 11명 선발인원/ 4쿼터를 고려하여 작성한 결과물으로, 이번 프로젝트의 예상 되는 결과물 입니다.



세부기능 1) 선택 메뉴를 통해 사용자의 입력 받기

ex) 참여인원/ 선발인원/ 쿼터수/대전방식(내전,대전) / 개인 사정

세부기능 2) 기존의 정보와 사용자 입력을 바탕으로 우선순위를 사용하여 결과물 출력하기

5) FunctionManager.cpp &헤더파일을 통한 함수 분리

- 포메이션 배열, 플레이어 배열을 다루는 함수를 분리하여 별도의 파일에 저장합니다.

최종보고서

3. 기능 구현

1_1)선택메뉴 출력 기능 구현

```
#include <windows.h>  // 콘솔창 clear 및 시간 지연을 주기위한 헤더파일

#include <fstream>
#include <iostream>
#include <limits>
#include <vector>

#include "Formation.h"
#include "Player.h"
#include "FunctionManager.h"

using namespace std;

// system("cls");        //콘솔창 clear하는 명령어

// 함수 선언
void DisplayMenu();  //사용자의 요구에 따라 원하는 메뉴를 출력
void loadPlayersInfo(vector<Player> &players);  // 기존 사용자 정보 불러오기
void loadPlayersInfo(string filename);
void loadFormationsInfo(vector<Formation> &formations);
void PlayerMenu();
void FormationMenu();

vector<Player> players;        //플레이어의 정보가 저장된 배열
vector<Formation> formations;  // 포메이션의 정보가 저장된 배열

int main() {
  cout << "----스쿼드 메이커 프로그램을 시작합니다.------ \n\n";
  cout << "------------기존 정보를 불러옵니다-------------\n\n";

  loadPlayersInfo(players);  // 기존 정보 불러오기
  loadPlayersInfo("Blueveins.txt");
  loadFormationsInfo(formations);  // 기존 포메이션 정보 불러오기

  DisplayMenu();  //메인 메뉴 출력

  return 0;
}
```

- 입출력
  - user_input : 사용자 입력
  - vector<Player> players : 플레이어 정보를 담은 객체 배열

- vector<Formation> formations :포메이션의 정보가 저장된 배열

- 사용된 함수
  - Displaymenu()

```
void DisplayMenu() {
  int user_input;
  while (true) {
    //메인메뉴 출력
    cout << "메인 메뉴입니다.\n\n";
    cout << "\n-------------------------------------------------\n";
    cout << "항목을 선택해주세요.\n\n";
    cout << "0.종료\n";
    cout << "1.선수 정보 입력/수정/삭제\n";
    cout << "2.포메이션 정보 입력/수정/삭제\n";
    cout << "3.스쿼드메이커\n";
    cout << "-------------------------------------------------\n";

    try {
      cout << "사용자 입력:";
      cin >> user_input;
      if (cin.fail()) {
        cin.clear();  // 오류 플래그 초기화
        cin.ignore(200, '\n');
        throw runtime_error("잘못된 입력입니다.다시 입력해주세요.\n");
      } else {
        break;
      }
    } catch (const exception &e) {
      system("cls");
      cout << e.what();
    }
  }

  system("cls");  //콘솔창 clear

  switch (user_input) {
    case 0:
      cout << "프로그램을 종료합니다.\n";
      exit(0);  //프로그램 종료
      break;
    case 1:
      PlayerMenu();
      break;
    case 2:
      FormationMenu();
      break;
    case 3:
      cout << "스쿼드메이커 화면 입니다.\n";
      // 구현 예정
      cout << "아직 기능이 구현되지 않았습니다. 메인메뉴로 돌아갑니다.\n";
      DisplayMenu();
      break;
```

```
    default:
      cout << "잘못된 입력입니다.다시 입력해주세요.\n";
      DisplayMenu();
  }
}
```

- 함수 입출력:
  - user_input : 사용자 입력 저장
- 함수 설명 :
  - 메뉴 항목을 출력하고 입력을 switch 조건문을 이용해 구분한다.
    - 0은 종료
    - 1~3의 입력의 경우 각 항목을 호출한다.
    - 입력(cin)이 잘못된경우 오류플래그 초기화, 입력버퍼 초기화 후, 에러메시지를 출력하고 다시 입력 받는다.
- 적용된 배운 내용 : 예외 처리

- loadPlayersInfo(vector<Player> &players) 함수

```
void loadPlayersInfo(vector<Player> &players) {
  // 저장된 플레이어 정보
  players.push_back(Player("김건휘", "공격", "골키퍼"));
  players.push_back(Player("박민호", "공격", "수비"));
  players.push_back(Player("야넥", "공격", "수비"));
  players.push_back(Player("다윗", "중앙", "수비"));
  players.push_back(Player("승민", "중앙", "수비"));
  players.push_back(Player("종훈", "수비", "없음"));
  players.push_back(Player("희석", "수비", "공격"));
  players.push_back(Player("준희", "수비", "공격"));
  players.push_back(Player("영훈", "골키퍼", "없음"));

  Sleep(1500);
  cout << "플레이어 정보 로드 완료\n\n";
  cout << "---------------------------------------\n";
  Sleep(1000);
  system("cls");
}
```

- 함수 입출력 :
  - vector<Player> &players : player 객체 배열
- 함수 설명 :
  - push_back()함수를 통해 객체배열에 저장된 정보를 입력한다.
  - Sleep()함수를 통해 시각적인 효과를 추가하여 각각 1.5초,1초의 지연을 추가한다.
  - 플레이어 정보 로드 1초 이후 system("cls") 를 통해 콘솔창을 초기화 한다.

- void loadPlayersInfo(string filename) 중복 함수

```
void loadPlayersInfo(string filename) {
  // 플레이어 정보가 저장된 파일 읽기
  ifstream is{filename};
```

```
  if (!is) {
    cerr << "파일 오픈에 실패하였습니다." << endl;
    cout << filename << "의 내용을 다시 확인하세요\n";
    cout << "프로그램을 종료합니다\n";
    exit(1);
  }
  string txt_name, txt_pre,
      txt_non_pre;  //파일의 1행에 담긴 선수 정보(이름,선호,비선호)
  while (is >> txt_name >> txt_pre >> txt_non_pre) {
    players.push_back(Player(txt_name, txt_pre, txt_non_pre));
  }

  Sleep(1500);
  cout << filename << "파일의 플레이어 정보 로드 완료\n\n";
  cout << "-----------------------------------------------\n";
  Sleep(1000);
  system("cls");  // 콘솔창 초기화
}
```

- 함수 입출력 :
  - string file name : player정보가 저장된 txt파일 이름
- 함수 설명 :
  - 입력 스트림의 값을 txt_name, txt_pre , txt_non_pre로 나누어 입력받는다.
  - 각각 (이름,선호포지션,비선호 포지션을 나타낸다.)
  - push_back()함수를 통해 객체배열에 저장된 정보를 입력한다.
  - Sleep()함수를 통해 시각적인 효과를 추가하여 각각 1.5초,1초의 지연을 추가한다.
  - 플레이어 정보 로드 1초 이후 system(”cls”) 를 통해 콘솔창을 초기화 한다.
- 적용된 배운 내용: 파일 입출력.예외 처리

- void loadFormationsInfo(vector<Formation> &formations) 함수

```
void loadFormationsInfo(vector<Formation> &formations) {
  // formationName(""), defenders(0), midfielders(0), forwards(0)
  formations.push_back(Formation("343", 3, 4, 3));
  formations.push_back(Formation("352", 3, 5, 2));
  formations.push_back(Formation("433", 4, 3, 3));
  formations.push_back(Formation("42(31)", 4, 2, 4));
  formations.push_back(Formation("442", 4, 4, 2));
  formations.push_back(Formation("541", 5, 4, 1));
  formations.push_back(Formation("523", 5, 2, 3));

  Sleep(1500);
  cout << "포메이션 정보 로드 완료\n\n";
  cout << "-----------------------------------------------\n";
  Sleep(1000);
  system("cls");  // 콘솔창 초기화
}
```

- 함수 입출력 :
  - vector<Formation> &formations: formation정보가 저장된 객체 배열

        - 함수 설명 :
            - push_back()함수를 통해 객체배열에 저장된 정보를 입력한다.
            - Sleep()함수를 통해 시각적인 효과를 추가하여 각각 1.5초,1초의 지연을 추가한다.
            - 포메이션 정보 로드 1초 이후 system("cls") 를 통해 콘솔창을 초기화 한다.
        - 적용된 배운 내용 : push_back 사용을 통한 객체 벡터 추가
- 기능 설명
    - 함수 시작과 함께 각 함수를 호출하여 기존 정보를 업로드하고,선택메뉴를 출력한다.
        - loadPlayersInfo(players); // 기존 정보 불러오기
          loadPlayersInfo("Blueveins.txt");
          loadFormationsInfo(formations); // 기존 포메이션 정보 불러오기
          DisplayMenu(); //메인 메뉴 출력
    - 메인 메뉴의 선택입력이 0 인경우 프로그램이 종료된다.
    - 선택입력이 1~3인경우 각 메뉴의 화면을 출력하도록 함수를 호출하고, 각 작업이 끝나면 사용자의 입력을 받아 메인 메뉴를 호출하거나 작업을 계속한다.
    - 사용자의 입력이 정상입력이 아닌 경우 다시 입력을 받도록 함수를 다시 호출한다.
- 적용된 배운 내용: 객체 배열 , 함수, switch,

2_1). 선수정보 입력/수정/삭제 기능 구현

```
void PlayerMenu() {
  cout << "\n\n선수 정보 입력/수정/삭제 화면 입니다. \n\n";
  // Player player;
  GetInformation(players);  //현재 플레이어 정보 출력
  cout << "메뉴를 선택해주세요.\n\n";
  cout << "0.메인으로 돌아가기 \n";
  cout << "1.선수 정보 입력\n";
  cout << "2.선수 정보 수정\n";
  cout << "3.선수 정보 삭제\n";
  cout << "\n-------------------------------------------------\n";
  cout << "사용자 입력:";
  int user_input;
  cin >> user_input;
  switch (user_input) {
    case 0:
      DisplayMenu();  //메인 메뉴로 돌아갑니다
      break;
    case 1:
      InsertPlayer(players);
      PlayerMenu();
      break;
    case 2:
      EditPlayer(players);
      PlayerMenu();
      break;
    case 3:
      DeletePlayer(players);
      PlayerMenu();
```

```
      break;
    default:
      cout << "잘못된 입력입니다.다시 선택 해주세요 .\n\n";
      PlayerMenu();
  }
}
```

- 입출력 :
  - players: 플레이어 정보가 저장된 객체 배열
  - user_input: 사용자 입력 저장
- 기능 설명:
  - 메인 메뉴를 통해 PlayerMenu() 함수를 호출하여 기능한다.
  - 기본적인 플레이어 메뉴 항목을 출력하고, 사용자 입력을 switch 조건문을 이용해 구분한다.
  - FunctionManager (추후 설명)의 함수를 이용한다.
    - 1~3은 각각 멤버함수를 이용하여 입력/수정/삭제를 시행하고,0 입력시 메인메뉴로 돌아간다. 그 외의 입력이 들어온 경우 오류 메시지를 출력하고 다시 입력받는다.

2_2) 클래스 및 헤더파일 구현

- Player 클래스& 헤더파일
  - 헤더파일

```
#pragma once

#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Player {
 public:
  string name;          //플레이어 이름
  string pre_pos;       //선호 포지션
  string non_pre_pos;   //비선호 포지션
  Player(); //기본 생성자
  Player(string name, string pre_pos, string non_pre_pos);
```

  - 클래스

```
#include "Player.h"

#include <iomanip>  //setw()사용하기 위함
#include <iostream>
#include <string>
#include <vector>
using namespace std;
// 생성자 정의와 초기화 목록 사용
Player::Player() {}
```

```
Player::Player(string name, string pre_pos, string non_pre_pos) {
  if (name.size() < 3) {
    name = " " + name;
  } else {
    this->name = name;
  }

  this->pre_pos = pre_pos;
  this->non_pre_pos = non_pre_pos;
}
```

- 클래스 설명
  - **멤버변수** :
    - string name; //플레이어 이름
      string pre_pos; //선호 포지션
      string non_pre_pos; //비선호 포지션
  - **멤버함수설명 :**
    - Player::Player(string name, string pre_pos, string non_pre_pos) - 생성자 함수
      - Player클래스의 생성자로 각 멤벼변수를 초기화 하며, name의 경우 3글자 미만 인경우 공백을 삽입하여 길이가 3글자가 되도록 한다.

3_1)포메이션 입력/수정/삭제 기능 구현

```
void FormationMenu() {
  cout << "\n\n포메이션 정보 입력/수정/삭제 화면 입니다. \n\n";
  // Formation formation;
  DisplayFormation(formations);  //현재 플레이어 정보 출력
  cout << "메뉴를 선택해주세요.\n\n";
  cout << "0.메인으로 돌아가기 \n";
  cout << "1.포메이션 정보 입력\n";
  cout << "2.포메이션 정보 수정\n";
  cout << "3.포메이션 정보 삭제\n";
  cout << "-----------------------------------------------\n";
  cout << "사용자 입력:";
  int user_input;
  cin >> user_input;
  switch (user_input) {
    case 0:
      DisplayMenu();  //메인 메뉴로 돌아갑니다
      break;
    case 1:
      InsertFormation(formations);
      FormationMenu();
      break;
    case 2:
      EditFormation(formations);
      FormationMenu();
    case 3:
      DeleteFormation(formations);
      FormationMenu();
      break;
```

```
    default:
      cout << "잘못된 입력입니다.다시 선택 해주세요 .\n\n";
      FormationMenu();
  }
}
```

- 입출력 :
  - formations: 플레이어 정보가 저장된 객체 배열
  - user_input: 사용자 입력 저장
- 기능 설명:
  - 메인 메뉴를 통해 Formation Menu() 함수를 호출하여 기능한다.
  - 기본적인 플레이어 메뉴 항목을 출력하고, 사용자 입력을 switch 조건문을 이용해 구분한다.
  - FunctionManager (추후 설명)의 함수를 이용한다.
    - 1~3은 각각 멤버함수를 이용하여 입력/수정/삭제를 시행하고,0 입력시 메인메뉴로 돌아간다. 그 외의 입력이 들어온 경우 오류 메시지를 출력하고 다시 입력받는다.

3_2) 클래스 및 헤더파일 구현

- Formation 클래스& 헤더파일
  - 헤더파일

```
#pragma once
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Formation {
 public:
  Formation();  // 생성자
  Formation(string formationName, int defenders, int midfielders,
            int forwards);
  string formation_name;          // 포메이션 이름
  int defenders;                       // 수비수 수
  int midfielders;                     // 미드필더 수
  int forwards;                        // 공격수 수
  vector<string> positions;  // 선수 위치
};
```

  - 클래스

```
#include "Formation.h"

#include <iomanip>
#include <iostream>
#include <string>
#include <vector>
using namespace std;
// 생성자 구현
```

```
Formation::Formation()
    : formation_name(""), defenders(0), midfielders(0), forwards(0) {}
Formation::Formation(std ::string formationName, int defenders, int midfielde
                     int forwards) {
  this->formation_name = formationName;
  this->defenders = defenders;
  this->midfielders = midfielders;
  this->forwards = forwards;
}
```

- 클래스 설명
  - **멤버변수** :
    - string name; //플레이어 이름
      string pre_pos; //선호 포지션
      string non_pre_pos; //비선호 포지션
  - **멤버함수설명 :**
    - Formation::Formation(std ::string formationName, int defenders, int midfielders, int forwards)- 생성자 함수
      - 포메이션 클래스의 생성자 함수로 각 멤버변수를 초기화 한다.

4_1) 스쿼드 메이커 기능구현

- 사용된 함수 정의
  - 위치: main.cpp

```
#include <windows.h>  // 콘솔창 clear 및 시간 지연을 주기위한 헤더파일
#include <algorithm>
#include <fstream>
#include <iostream>
#include <limits>
#include <vector>

#include "Formation.h"
#include "Player.h"
#include "FunctionManager.h"

using namespace std;

// system("cls");        //콘솔창 clear하는 명령어

// 함수 선언
----- 다른 함수들 선언  ---

//SquadMaker 함수들 선언
void SquadMakerMenu();
bool ComparePlayers(const Player& a, const Player& b);
void SortPlayersForFormation(vector<Player>& players);
void ExaminePositionPreference(vector<Player>& players);
template <typename T>
void DisplayArray(const vector<T>& arr);
```

```
vector<Player> players;         //플레이어의 정보가 저장된 배열
vector<Formation> formations;  // 포메이션의 정보가 저장된 배열
```

- void SquadMakerMenu()

```
void SquadMakerMenu() {
    system("cls"); // 보드 초기화
    cout << "스쿼드메이커 화면 입니다.\n";
    // 사용자 입력 받기(쿼터수, 참여 인원,게임 참여 여부)
    int num_quarters, player_num;
    cout << "스쿼드 메이커 설정을 시작합니다.\n";
    cout << "쿼터 수를 입력해 주세요 \n";
    cout << "사용자 입력:";
    cin >> num_quarters;

    system("cls");
    while (true) {
        player_num = 0;
        cout << "\n현재 1군에 등록된 선수의 명단입니다.\n";
        GetInformation(players);
        cout << "경기에 참여하지 못하는 선수의 번호를 입력해주세요 \n";
        cout << "빠지는 선수가 없거나 수정을 마쳤다면 0을 입력해주세요\n";
        cout << "사용자 입력:";
        cin >> player_num;
        if (player_num == 0) {
            break;
        }
        else {
            players[player_num - 1].is_participating = false;
        }
    }
    int formationindex;
    cout << "경기에사용할 포메이션을 선택해 주세요\n";
    DisplayFormation(formations);
    cout << "사용자 입력:";
    cin >> formationindex;
    int num_def = formations[formationindex - 1].defenders;
    int num_mid = formations[formationindex - 1].midfielders;
    int num_for = formations[formationindex - 1].forwards;
    int num_keeper = 1;

    //참가 선수별로 각 포지션별 선호도 조사
    ExaminePositionPreference(players);

    //스쿼드 메이킹 및 출력
    for (int quarter = 1; quarter <= num_quarters; quarter++) {
        // 선수를 참여 여부, 포지션 선호도, 포메이션 가중치에 따라 정렬
        SortPlayersForFormation(players);

        // 각 포지션별로 선택된 선수를 저장할 배열
        vector<string> forward, mid, def, keeper;

        // 선수를 순회하며 스쿼드 구성
```

```
        for (auto& player : players) {
            if (!player.is_participating) {
                continue; // 참가하지 않는 선수 건너뜀
            }

            // 스쿼드에 선택되었으므로 참여 가중치 감소
            player.part_weight -= 7;

            // 선호도와 가중치에 따라 포지션에 선수 배정
            if (player.pre_pos == "공격" && forward.size() < num_for) {
                forward.push_back(player.name);
            }
            else if (player.pre_pos == "중앙" && mid.size() < num_mid) {
                mid.push_back(player.name);
            }
            else if (player.pre_pos == "수비" && def.size() < num_def) {
                def.push_back(player.name);
            }
            else if (player.pre_pos == "골키퍼" && keeper.empty()) {
                keeper.push_back(player.name);
            }
            //
            else {
                if (forward.size() + mid.size() + def.size() + keeper.size()
                    break; //11명의 명단 작성완료
                }
                else {
                    while (true) {
                        int randomPosition = rand() % 4;
                        // 랜덤하게 선택된 포지션에 따라 사용할 벡터 설정
                        vector<string>* posVector;
                        int pos_num = 0;
                        switch (randomPosition) {
                        case 0:
                            posVector = &forward;
                            pos_num = num_for;
                            break;
                        case 1:
                            posVector = &mid;
                            pos_num = num_mid;
                            break;
                        case 2:
                            posVector = &def;
                            pos_num = num_def;
                            break;
                        case 3:
                            posVector = &keeper;
                            pos_num = 1;
                            break;
                        default:
                            cerr << "에러: 잘못된 포지션 번호입니다.\n";
                            continue; // 유효하지 않은 번호가 생성되면 다시 시도
                        }
```

```
                        // 해당 포지션에 공간이 있으면 선수를 할당하고 루프를 종료
                        if (posVector->size() < pos_num) {
                            posVector->push_back(player.name);
                            break;
                        }

                    }


                }
            }
        }

        // 현재 쿼터의 스쿼드 표시
        cout << "\n\n========= " << quarter << "쿼터의 스쿼드 ========\n";
        cout << "Forward: ";
        DisplayArray(forward);
        cout << "Mid: ";
        DisplayArray(mid);
        cout << "Defense: ";
        DisplayArray(def);
        cout << "Keeper: ";
        DisplayArray(keeper);
    }


    int user_input;
    cout << "\n0을 입력하면 메인 메뉴로 돌아갑니다.\n";
    cout << "사용자 입력:";
    cin >> user_input;
    if (user_input == 0) {
        DisplayMenu();
    }

}
```

- **입출력:**
  - 입력 :
    - **num_quarters**: 경기의 쿼터 수(정수)
    - **player_num**: 참여하지 않는 것으로 표시하기 위한 플레이어 번호.

      루프를 중단하려면 0(정수)
    - **formationindex**: 경기에 대해 선택된 포메이션의 인덱스(정수)
    - **user_input**: 메인 메뉴로 돌아가기 위한 사용자 입력(정수)
- **설명:**
  - 콘솔창 을 지우고 설정(쿼터 수, 참가자 등)에 대한 사용자 입력을 받습니다.
  - 사용자 입력에 따라 스쿼드를 정보를 저장합니다.
  - 경기에 대한 포메이션을 선택합니다.

- 플레이어 포지션별 선호도를 검사하고 각 쿼터의 스쿼드 구성을 위해 플레이어를 정렬하는 함수를 호출합니다.
- 쿼터별로 구성된 스쿼드를 표시합니다.
- 사용자에게 메인 메뉴로 돌아가라는 메시지를 표시합니다.

▪ **적용 개념:**

- 루프(`while`, `for`), 조건문(`if`, `else`, `switch`).
- 함수 호출 (`GetInformation`, `DisplayFormation`, `ExaminePositionPreference`, `SortPlayersForFormation`, `DisplayArray`).
- 벡터 연산(`push_back`, `size`).
- 난수 생성(`rand()`).

◦ template <typename T>

void DisplayArray(const vector<T>& arr);

```
// 배열의 요소를 표시하는 함수
template <typename T>
void DisplayArray(const vector<T>& arr) {
    for (const auto& elem : arr) {
        cout << elem << " ";
    }
    cout << endl;
}
```

▪ 입출력:

- 입력 : 벡터 템플릿(forward,defense,mid,keeper)

▪ 설명 :

- 각 포지션 에 대한 정보가 담긴 벡터의 내용을 공백으로 구분하여 출력한다.
- 각 포지션에 대한 구분은 템플릿 이름과 줄바꿈(endl)을 통해 구분한다.

▪ 적용된 배운 내용 : 템플릿

◦ bool ComparePlayers(const Player& a, const Player& b)

```
bool ComparePlayers(const Player& a, const Player& b) {
    // 참가 여부에 따라 정렬 (참가 중이면 우선, 참가 중이지 않으면 가장 마지막에)
    if (a.is_participating != b.is_participating) {
        return a.is_participating > b.is_participating;
    }

    // 가중치에 따라 정렬 (가중치 높을수록)
    return a.part_weight > b.part_weight;
}
```

▪ 입출력:

- 입력**:** 두 개의 `Player` 객체(`a` 및 `b`)

▪ 설명:

- 객체의 멤버변수 'is_participating' 이 true인 경우 먼저 정렬될수 있도록 한다.

- 그 다음으로 멤버변수 'part_weight' 을 비교하여 가중치가 높은순으로 정렬될수 있도록 한다.

- void SortPlayersForFormation(vector<Player>& players);

```
// 포메이션 참여 및 선호도에 따라 선수들을 정렬하는 함수
void SortPlayersForFormation(vector<Player>& players) {
    sort(players.begin(), players.end(), ComparePlayers);
}
```

  - 입출력:
    - 입력: `players` 벡터
  - 설명:
    - `sort` 함수를 사용하여 `ComparePlayers` 함수에 정의된 기준에 따라 `Player` 객체 의 벡터를 정렬합니다.

- void ExaminePositionPreference(vector<Player>& players)

```
void ExaminePositionPreference(vector<Player>& players) {
    int pre_attack = 0, pre_center = 0, pre_defense = 0, pre_goalkeeper = 0;


    for (const auto& player : players) {
        if (!player.is_participating) {
            continue; // 참가하지 않는 선수는 건너뜁니다.
        }

        // 선호 포지션에 따라 선호도를 증가시킵니다.
        if (player.pre_pos == "공격") {
            pre_attack++;
        }
        else if (player.pre_pos == "중앙") {
            pre_center++;
        }
        else if (player.pre_pos == "수비") {
            pre_defense++;
        }
        else if (player.pre_pos == "골키퍼") {
            pre_goalkeeper++;
        }

        // 비선호 포지션에 따라 선호도를 감소시킵니다.
        if (player.non_pre_pos == "공격") {
            pre_attack--;
        }
        else if (player.non_pre_pos == "중앙") {
            pre_center--;
        }
        else if (player.non_pre_pos == "수비") {
            pre_defense--;
        }
        else if (player.non_pre_pos == "골키퍼") {
            pre_goalkeeper--;
        }
    }
```

```
    // 포지션 선호도 요약을 출력합니다.
    cout << "참가 중인 선수들의 포지션 선호도 요약:\n";
    cout << "공격: " << pre_attack << "\n";
    cout << "중앙: " << pre_center << "\n";
    cout << "수비: " << pre_defense << "\n";
    cout << "골키퍼: " << pre_goalkeeper << "\n";

    cout << " 선호도 를 반영합니다.\n";

    //선호도를 반영합니다.
    for (auto& player : players) {
        if (!player.is_participating) {
            continue; // 참가하지 않는 선수는 건너뜁니다.
        }

        // 선호 포지션의 경우 가중치를 낮춰 우선순위를 낮춥니다.
        if (player.pre_pos == "공격") {
            player.part_weight -= pre_attack;
        }
        else if (player.pre_pos == "중앙") {
            player.part_weight -= pre_center;
        }
        else if (player.pre_pos == "수비") {
            player.part_weight -= pre_defense;
        }
        else if (player.pre_pos == "골키퍼") {
            player.part_weight -= pre_goalkeeper;
        }

        // 비선호 포지션의 경우 가중치를 높여 우선순위를 높입니다.
        if (player.non_pre_pos == "공격") {
            player.part_weight += pre_attack;
        }
        else if (player.non_pre_pos == "중앙") {
            player.part_weight += pre_center;
        }
        else if (player.non_pre_pos == "수비") {
            player.part_weight += pre_defense;
        }
        else if (player.non_pre_pos == "골키퍼") {
            player.part_weight += pre_goalkeeper;
        }
    }


}
```

- 입출력:
  - 입력: `players` 벡터
- 설명:
  - 각 포지션에 대한 선호도를 조사한다.

- pre_position(attack,center,defense,goalkeeper)의 값을 각각 0으로 초기화 한다.
- 선호 포지션의 경우 각각의 선호도 변수pre_position(attack,center,defense,goalkeeper)+1 을 해준다.
- 비선호 포지션의 경우 각각의 선호도 변수pre_position(attack,center,defense,goalkeeper)-1 을 해준다.
- 각 포지션의 선호도 작성이 완료 되었다면 players 벡터를 탐색하여 각각 플레이어 객체의 참여 가중치를 조정한다.
  - 플레이어의 참여 가중치는 player객체 생성시 0으로 초기화 되어있으며, 이 값에 pre_pos의 선호도에 해당하는 값을 더하고, non_pre_pos의 선호도에 해당하는 값을 빼준다.
  - 만약 포지션 가중치가 (공격/중앙/수비/골키퍼) :(1/5/3/6)이고 선호 포지션이 공격,비선호 포지션이 수비라면 가중치 업데이트 결과는 다음과 같다.
    - player.part_weight=0(객체 생성시 0으로 초기화)+1-3=-2

5) FunctionManager.cpp 및 헤더파일 구현

- **FunctionManager.h**

```
#pragma once
void InsertPlayer(vector<Player> &players);  // 플레이어 추가
void DeletePlayer(vector<Player> &players);  //플레이어 삭제
void EditPlayer(vector<Player> &players);    // 플레이어 수정
void GetInformation(
    const vector<Player> &players);  // 플레이어 정보 보여주는 함수
string GetValidPosition(const string &prompt);  // 입력된 포지션 유효성 검사 및
                                                 // 올바른 입력 반환하는 함수



  // 포메이션을 추가하기 위한 함수
void InsertFormation(vector<Formation> &formations);  // 포메이션 추가
void DeleteFormation(vector<Formation> &formations);  //포메이션 삭제
void EditFormation(vector<Formation> &formations);    //포메이션 수정
bool IsValidFormationInput(int def, int mid, int forward);

// 현재 포메이션 정보를 표시하는 함수
void DisplayFormation(const vector<Formation> &formations);
```

- **FunctionManager.cpp**

```
#include "Player.h"
#include "Formation.h"
#include "FunctionManager.h"
#include <iomanip>  //setw()사용하기 위함
#include <iostream>
#include <string>
#include <vector>
using namespace std;

//자세한 설명은 각 코드 블럭을 통해 설명
void InsertPlayer(vector<Player>& players) {
----생략------
}
```

```
void DeletePlayer(vector<Player>& players) {
----생략------
}

void EditPlayer(vector<Player> &players) {
  ----생략------
}

void GetInformation(const vector<Player> &players) {
----생략------
}

string GetValidPosition(const string &prompt) {
----생략------
}

// 현재 포메이션 정보를 표시하는 함수 구현
void DisplayFormation(const vector<Formation> &formations) {
----생략------
}

void InsertFormation(vector<Formation> &formations) {
----생략------
}
void EditFormation(vector<Formation> &formations) {
 ----생략------
}
void DeleteFormation(vector<Formation> &formations) {
  ----생략------
```

- void InsertPlayer(vector<Player> &players)

```
void InsertPlayer(vector<Player>& players) {
    cout << "-------------------------------------------------------\n";
    cout << "플레이어를 추가합니다.\n\n";
    cout << "플레이어의 정보를 이름/선호 포지션/비선호 포지션 순으로 "
        "입력하세요.\n\n";
    cout << "선호/비선호 포지션은 공격/중앙/수비/골키퍼/없음 중에서 "
        "입력해주세요\n\n";
    string name,pre_pos,non_pre_pos;
    // 이름 입력
    cout << "이름: ";
    cin >> name;

    // 선호 포지션 입력 및 유효성 검사
    pre_pos = GetValidPosition("선호 포지션: ");
    non_pre_pos = GetValidPosition("비선호 포지션: ");
    players.push_back(Player(name,pre_pos,non_pre_pos));
    cout << "-------------------------------------------------------\n";
    cout << "선수가 추가되었습니다.\n\n";
    cout << "-------------------------------------------------------\n";
}
```

- 입출력
  - 입력: 선수 정보(이름, 선호 포지션, 비선호 포지션)
  - 출력: 새 플레이어를 추가하여 `players` 벡터를 수정
- 설명
  - 입력 조건을 출력한다.
  - 사용자 입력 받기 : 이름 과 달리 선호/비선호 포지션은 사용자의 입력이 정상 입력인지 유효성 검사를 하는 함수를 호출한다.(유효할때 까지 입력 받는다.)
  - push_back()를 통해 players 배열에 추가하고 알림문구를 출력한다.
- 적용 개념
  - 함수 호출(`GetValidPosition`).
  - 벡터 연산(`push_back`).

◦ void DeletePlayer(vector<Player> &players); //플레이어 삭제

```
void DeletePlayer(vector<Player>& players) {
    cout << "-------------------------------------------------------\n";
    cout << "플레이어를 삭제합니다.\n\n";
    cout << "삭제할 플레이어의 번호를 입력해주세요 \n\n";

    int delete_index;  // 삭제할 플레이어 번호

    // 삭제할 번호 입력
    cout << "번호: ";
    cin >> delete_index;
    players.erase(players.begin() + delete_index - 1);

    cout << "선수가 삭제 되었습니다.\n";
}
```

- 삭제할 플레이어의 번호를 입력받고, erase()함수를 이용해 해당 인덱스의 객체를 제거한다.
- **입출력:**
  - 입력 : 삭제할 플레이어 번호
  - 출력: 기존 플레이어를 삭제하여 `players` 벡터를 수정
- **설명:**
  - 플레이어 삭제 여부를 물어보는 메시지를 표시합니다.
  - 삭제할 플레이어 번호를 입력하라는 메시지가 표시됩니다.
  - `players` 벡터에서 지정된 플레이어를 삭제합니다.
  - 삭제 후 확인 메시지를 출력합니다.

◦ void EditPlayer(vector<Player> &players); // 플레이어 수정

```
cout << "-------------------------------------------------------\n";
  cout << "플레이어를 수정합니다.\n\n";
  cout << "수정할 플레이어의 번호를 입력해주세요 \n\n";

  int edit_index;  //수정할 플레이어 번호
```

```
    // 수정할 플레이어 번호 입력
    cout << "수정할 플레이어 번호: ";
    cin >> edit_index;
    cout << "플레이어의 정보를 이름/선호 포지션/비선호 포지션 순으로 "
            "입력하세요.\n\n";
    cout << "선호/비선호 포지션은 공격/중앙/수비/골키퍼/없음 중에서 "
            "입력해주세요\n\n";
    cout << "이름: ";
    cin >> players[edit_index - 1].name;
    players[edit_index - 1].pre_pos = GetValidPosition("선호 포지션:");
    players[edit_index - 1].pre_pos = GetValidPosition("비선호 포지션:");

    cout << "선수정보가 수정 되었습니다.\n";
```

- **입출력:**
  - 입력 : 수정을 위한 플레이어 정보입니다.
  - 출력: 기존 플레이어를 편집하여 `players` 벡터를 수정합니다.
- **설명:**
  - 수정을 위해 플레이어 번호를 입력하라는 메시지가 표시됩니다.
  - 수정할 플레이어의 번호를 입력받고,해당 인덱스의 객체 정보를 덮어쓰기 한다. 덮어쓰기 하는 과정은 선수 입력 과정과 동일하게 유효성 검사를 한다.
  - 위치 확인을 위해 `GetValidPosition` 기능을 활용합니다.
  - `players` 벡터의 플레이어 정보를 편집합니다.
- **적용 개념:**
  - 함수 호출(`GetValidPosition`).
  - 벡터 인덱싱

- void GetInformation(const vector<Player> &players); // 플레이어 정보 보여주는 함수

```
void GetInformation(const vector<Player>& players) {
    cout << "번호 |  이름                | 선호 포지션      | 비선호 포지션\n";
    cout << "-------------------------------------------------------\n";
    int players_count=0;
    for (int i = 0; i < players.size(); i++) {
        if (players[i].is_participating == true) {
            cout << setw(4) << i + 1;
            cout << " | " << setw(19) << players[i].name;
            cout << " | " << setw(16) << players[i].pre_pos;
            cout << " | " << players[i].non_pre_pos << endl;
            players_count++;
        }
    }
    cout << endl;
    cout << "-------------------------------------------------------\n";
    cout << "1군 선수의 수 :" << players_count<<endl;
}
```

- **입출력:**
  - 입력: `players` 벡터

- 출력: 참가 플레이어에 대한 정보를 출력합니다.

- **설명:**
  - 참가하는 플레이어에 대한 정보를 출력합니다(is_participating이 true인 경우만)
  - 참가 플레이어 수를 계산합니다.
  - 1군 선수의 총 인원수를 출력합니다.
  - setw()함수를 이용하여 출력화면에 고르게 정렬되도록 한다.

- string GetValidPosition(const string &prompt); // 입력된 포지션 유효성 검사 및 올바른 입력 반환하는 함수

```
string GetValidPosition(const string& prompt) {
    string position;  // 입력을 통해 들어온 포지션
    while (true) {
        cout << prompt;
        cin >> position;

        if (position == "공격" || position == "중앙" || position == "수비" ||
            position == "골키퍼" || position == "없음") {
            break;
        }
        else {
            cout << "잘못된 포지션입니다. 공격/중앙/수비/골키퍼/없음 중 하나를 "
                "입력하세요.\n";
        }
    }
    return position;
}
```

  - 입출력
    - 입력: 포지션에 대한 사용자 입력
    - 출력: 유효한 포지션 나타내는 문자열
  - 설명
    - prompt : 값을 달리하여 선호/비선호 포지션 입력시 재사용 할수 있도록 사용자 입력 칸에 출력되는 값을 저장한다.
    - 공격/중앙/수비/없음 중의 입력이 아닌경우 오류 메시지를 출력하고 다시 입력 받는다.
    - 유효성 검사를 통과한 포지션 값을 반환한다.

- void DisplayFormation(const vector<Formation> &formations)

```
void DisplayFormation(const vector<Formation>& formations) {
    // 현재 포메이션 정보를 출력하는 함수
    cout << "번호 |포메이션 이름 | 수비수 | 미드필더 | 공격수\n";
    cout << "-----------------------------------------------\n";

    for (int i = 0; i < formations.size(); i++) {
        cout << setw(4) << i + 1 << " |";
        cout << setw(13) << formations[i].formation_name;
        cout << " | " << setw(6) << formations[i].defenders;
        cout << " | " << setw(8) << formations[i].midfielders;
        cout << " | " << setw(6) << formations[i].forwards << endl;
    }
```

```
    cout << endl;
    cout << "-----------------------------------------------\n";
}
```

- 입출력
  - 입력: `formations` 벡터
  - 출력: 포메이션 정보
- 설명
  - 현재 포메이션 정보를 출력하는 함수
  - 반복문을 통해 객체 배열을 돌면서 배열에 저장된(포메이션이름,수비수 숫자,미드필더 숫자,공격수 숫자)를 출력한다.

- void insertFormation(vector<Formation> &formations)

```
void InsertFormation(vector<Formation>& formations) {
    string formationName;
    cout << "추가할 포메이션의 이름:\n";
    cin >> formationName;
    int def, mid, forward;
    while (true) {
        cout << "수비수 숫자:\n";
        cin >> def;
        cout << "미드필더 숫자\n";
        cin >> mid;
        cout << "공격수 숫자\n";
        cin >> forward;
        if (def > 0 && mid > 0 && forward > 0) {
            if (def + mid + forward == 10) {
                formations.push_back(Formation(formationName, def, mid, forwa
                cout << "새로운 포메이션이 추가되었습니다.\n";
                break;
            }
            else {
                cout << "필드 플레이어는 10명 입니다.다시 입력해주세요\n";
                cout << "현재 필드 플레이어 숫자 :" << def + mid + forward << end
            }
        }
        else {
            cout << "각 포지션의 플레이어 숫자는 최소 한명이여야 합니다. 다시 "
                "입력해주세요 \n";
        }
    }
}
```

입출력:

- 입력: `formations` 벡터(포메이션 정보가 저장 된 벡터)

- 설명 :
  - 포메이션 정보 추가 하는 함수
  - 사용자의 입력을 통해 들어온 포메이션 이름을 formationName 변수에 임시로 저장한다.
  - 반복문을 통해 수비수/미드필더/공격수 숫자를 입력 받으며, 각각의 숫자가 0이상인지 확인한다.

- 모든 숫자가 0보다 크지 않다면, 다시 입력 받는다.
- 모든 숫자가 0 보다 크다면 각 숫자의 합이 10이 되는지 확인한다.(골키퍼를 제외한 필드플레이어 숫자의 합은 항상 10이다)
  - 필드플레이어의 숫자가 10이라면 push_back을 통해 formations 배열에 각 정보를 통해 객체를 추가한다.
  - 필드 플레이어의 숫자 합이 10이 아니라면 다시 입력받는다.

- void EditFormation(vector<Formation> &formations)

```
void EditFormation(vector<Formation>& formations) {
    cout << "-------------------------------------------------------\n";
    cout << "포메이션 정보를 수정합니다.\n\n";
    cout << "수정할 포메이션의 번호를 입력해주세요 \n\n";

    int edit_index;  //수정할 플레이어 번호

    // 수정할 플레이어 번호 입력
    cout << "수정할 포메이션 번호: ";
    cin >> edit_index;
    cout << "이름: ";
    cin >> formations[edit_index - 1].formation_name;
    int def, mid, forward;
    while (true) {
        cout << "수비수 숫자:\n";
        cin >> def;
        cout << "미드필더 숫자\n";
        cin >> mid;
        cout << "공격수 숫자\n";
        cin >> forward;
        if (def > 0 && mid > 0 && forward > 0) {
            if (def + mid + forward == 10) {
                formations[edit_index - 1].defenders = def;
                formations[edit_index - 1].midfielders = mid;
                formations[edit_index - 1].forwards = forward;
                cout << "포메이션 정보가 수정되었습니다.\n";
                break;
            }
            else {
                cout << "필드 플레이어는 10명 입니다.다시 입력해주세요\n";
                cout << "현재 필드 플레이어 숫자 :" << def + mid + forward << end
            }
        }
        else {
            cout << "각 포지션의 플레이어 숫자는 최소 한명이여야 합니다. 다시 "
                "입력해주세요 \n";
        }
    }
}
```

  - 입출력:
    - 입력: `formations` 벡터(포메이션 정보가 저장 된 벡터)
  - 설명:

- 현재 포메이션 정보를 수정하는 함수
- 사용자 입력을 통해 수정할 포메이션의 번호를 입력 받고 해당 인덱스에 있는 객체의 인덱스를 계산한다. (edit_index-1)
- formations 객체배열에서 해당 인덱스를 조회 하고, InsertFormation 과 동일한 과정을 통해 해당 인덱스의 객체 정보에 덮어 씌운다.

- void DeleteFormation(vector<Formation> &formations)

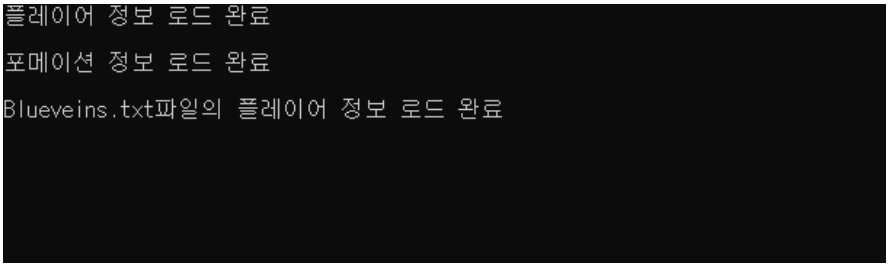
```
void DeleteFormation(vector<Formation>& formations) {
    cout << "-------------------------------------------------------\n";
    cout << "포메이션을 삭제합니다.\n\n";
    cout << "삭제할 포메이션의 번호를 입력해주세요 \n\n";

    int delete_index;  // 삭제할 플레이어 번호

    // 삭제할 번호 입력
    cout << "번호: ";
    cin >> delete_index;
    formations.erase(formations.begin() + delete_index - 1);

    cout << "포메이션이 삭제 되었습니다.\n";
}
```

  - 입출력:
    - 입력: `formations` 벡터(포메이션 정보가 저장 된 벡터)
  - 설명
    - 사용자로부터 삭제할 포메이션의 번호를 입력받고,그 입력을 delete_index에 저장한다.
    - erase()함수를 이용하여 해당 인덱스의 정보를 삭제한다.

4.테스트 결과

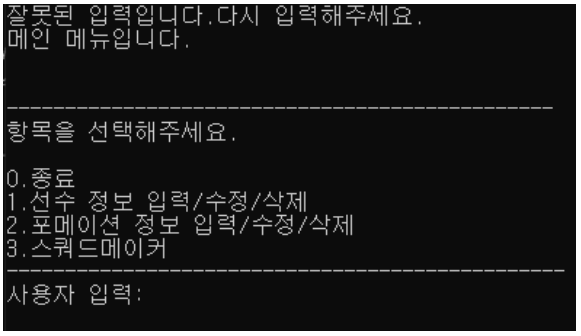
(1) 선택메뉴 출력

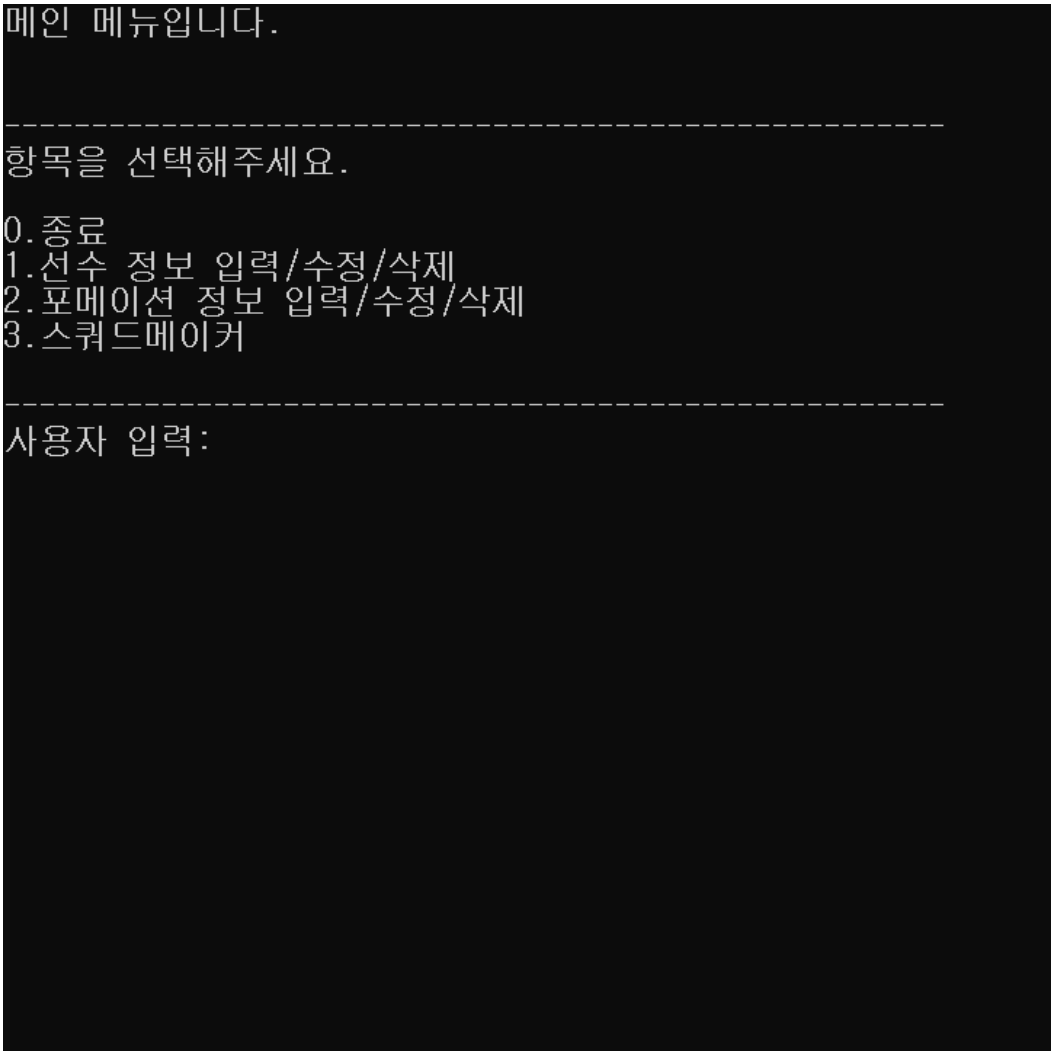
세부기능1: 저장된 정보 불러오기



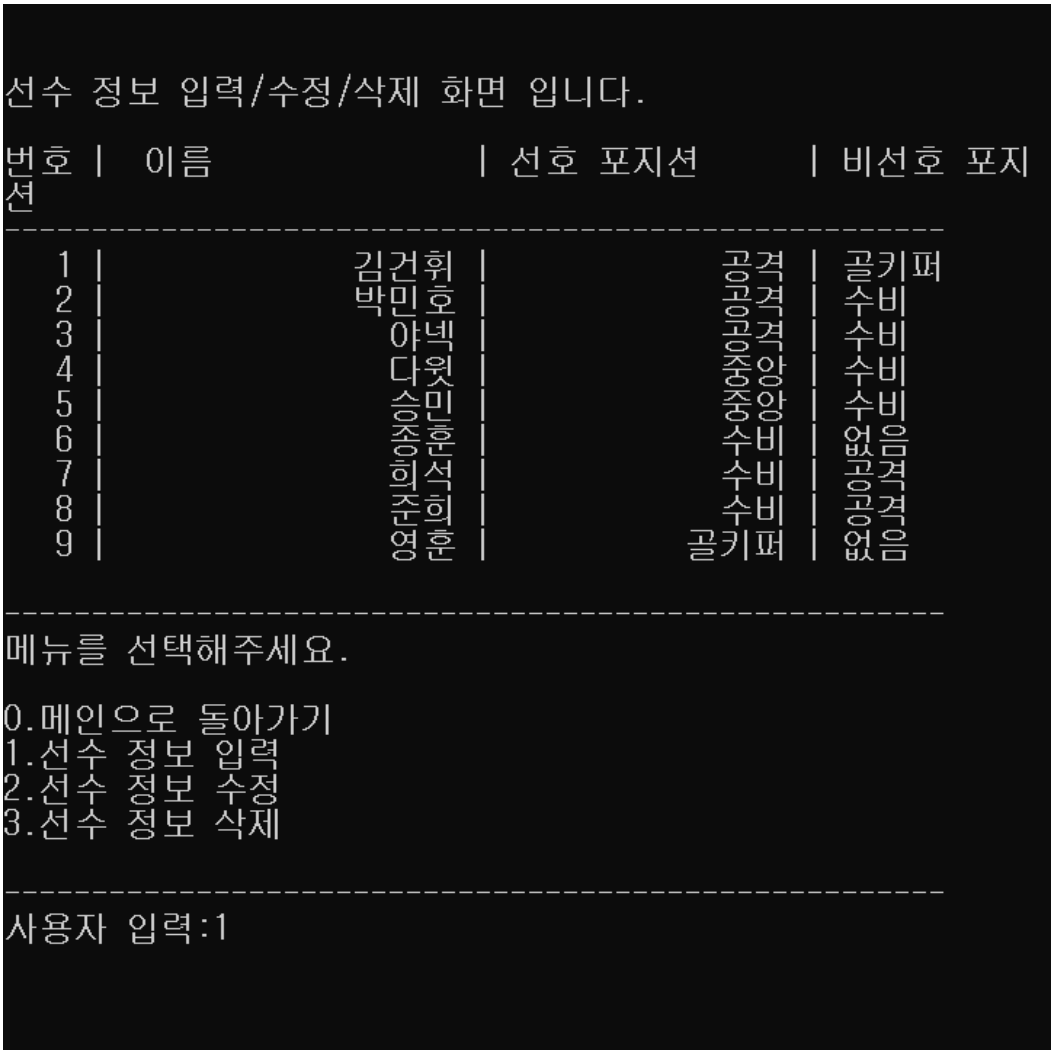
세부기능2: 메인메뉴 출력

-잘못된 입력의 경우(숫자가 아닌 텍스트 입력시)

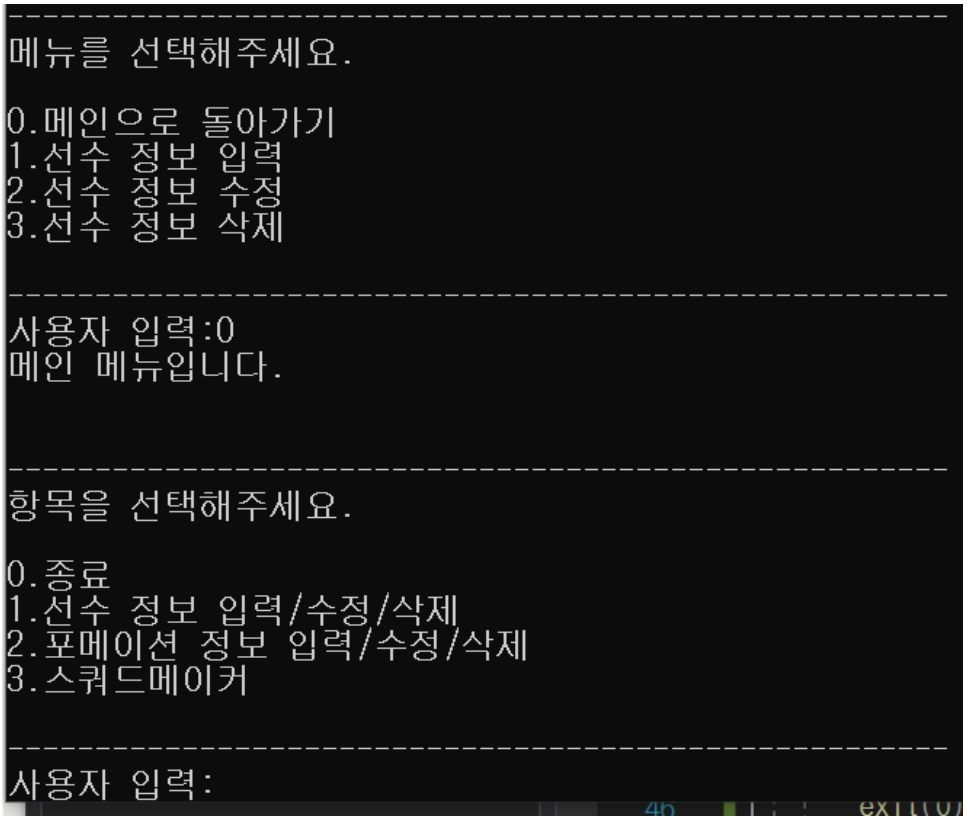




선수정보 입력/수정/삭제 화면


선수 정보 입력/수정/삭제 화면 입니다.
번호 | 이름 | 선호 포지션 | 비선호 포지션
1 | 김건휘 | 공격 | 골키퍼
2 | 박민호 | 공공격 | 수비
3 | 야넥 | 공공격 | 수비
4 | 다윗 | 중앙 | 수비
5 | 승민 | 중앙 | 수비
6 | 종훈 | 수비 | 없음
7 | 희석 | 수비 | 공격
8 | 준희 | 수비 | 공격
9 | 영훈 | 골키퍼 | 없음
메뉴를 선택해주세요.
0.메인으로 돌아가기
1.선수 정보 입력
2.선수 정보 수정
3.선수 정보 삭제
사용자 입력:1

메인메뉴 되돌아가기



(2) 선수 정보 입력/수정/삭제 항목 기능

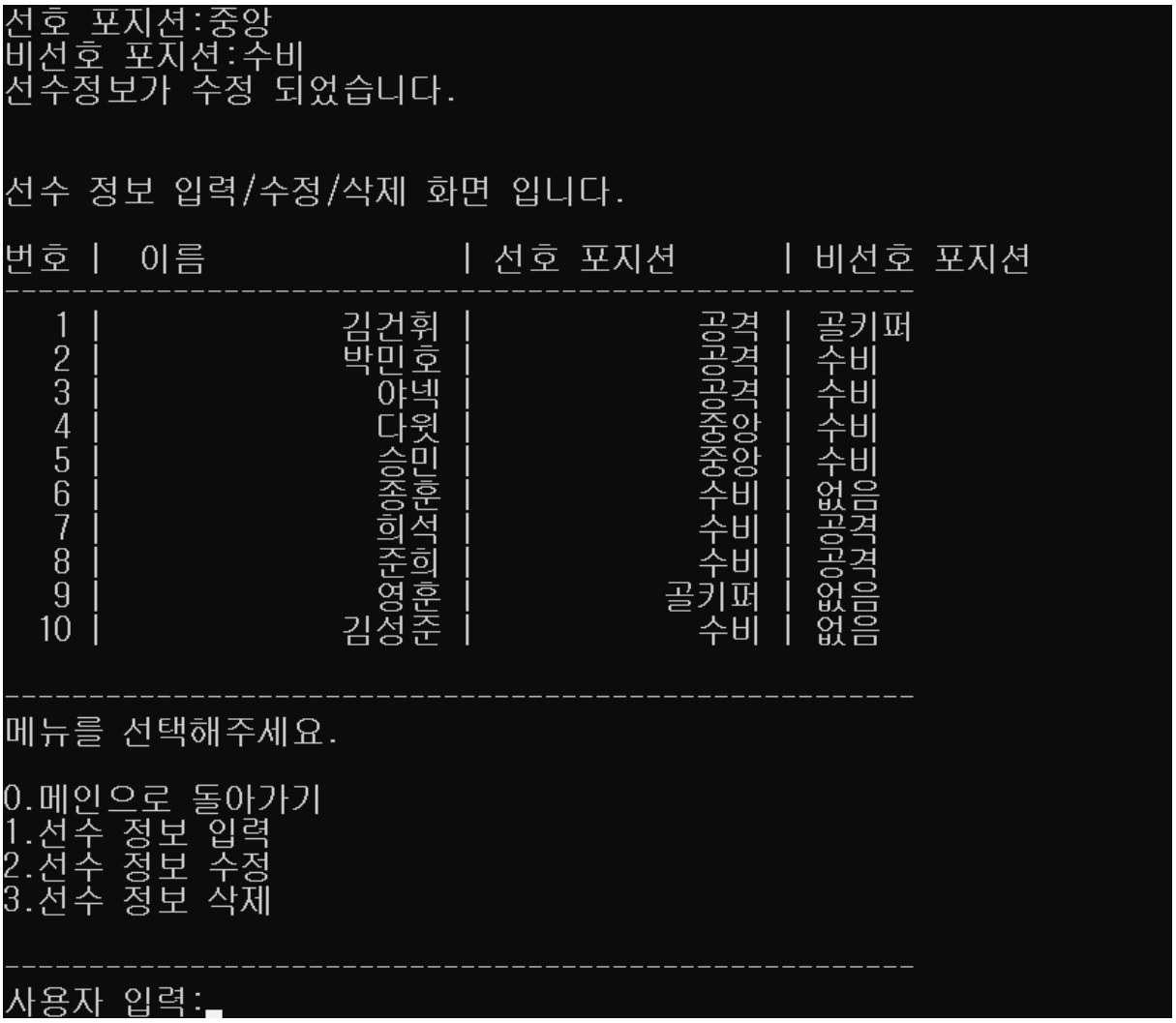
- 메인화면

```
선수 정보 입력/수정/삭제 화면 입니다.

번호 |  이름                | 선호 포지션       | 비선호 포지션
-------------------------------------------------------
   1 |              김건휘 |             공격 | 골키퍼
   2 |              박민호 |             공격 | 수비
   3 |                야넥 |             공격 | 수비
   4 |                다윗 |             중앙 | 수비
   5 |                승민 |             중앙 | 수비
   6 |                종훈 |             수비 | 없음
   7 |                희석 |             수비 | 공격
   8 |                준희 |             수비 | 공격
   9 |                영훈 |           골키퍼 | 없음

-------------------------------------------------------
메뉴를 선택해주세요.

0.메인으로 돌아가기
1.선수 정보 입력
2.선수 정보 수정
3.선수 정보 삭제

-------------------------------------------------------
사용자 입력:1
```

- 플레이어 추가

```
-------------------------------------------------------
사용자 입력:1
-------------------------------------------------------
플레이어를 추가합니다.

플레이어의 정보를 이름/선호 포지션/비선호 포지션 순으로 입력하세요.

선호/비선호 포지션은 공격/중앙/수비/골키퍼/없음 중에서 입력해주세요

이름: 김성준
선호 포지션: 죽악
잘못된 포지션입니다. 공격/중앙/수비/골키퍼/없음 중 하나를 입력하세요.
선호 포지션: 중앙
비선호 포지션: 있음
잘못된 포지션입니다. 공격/중앙/수비/골키퍼/없음 중 하나를 입력하세요.
비선호 포지션: 없음
-------------------------------------------------------
선수가 추가되었습니다.

-------------------------------------------------------
```

- 플레이어 수정



- 플레이어 삭제

```
삭제할 플레이어의 번호를 입력해주세요

번호: 10
선수가 삭제 되었습니다.


선수 정보 입력/수정/삭제 화면 입니다.

번호 |  이름                | 선호 포지션       | 비선호 포지션
-----------------------------------------------------
   1 |              김건휘 |              공격 | 골키퍼
   2 |              박민호 |              공격 | 수비
   3 |                야넥 |              공격 | 수비
   4 |                다윗 |              중앙 | 수비
   5 |                승민 |              중앙 | 수비
   6 |                종훈 |              수비 | 없음
   7 |                희석 |              수비 | 공격
   8 |                준희 |              수비 | 공격
   9 |                영훈 |            골키퍼 | 없음

-----------------------------------------------------
메뉴를 선택해주세요.

0.메인으로 돌아가기
1.선수 정보 입력
2.선수 정보 수정
3.선수 정보 삭제

-----------------------------------------------------
사용자 입력:
```

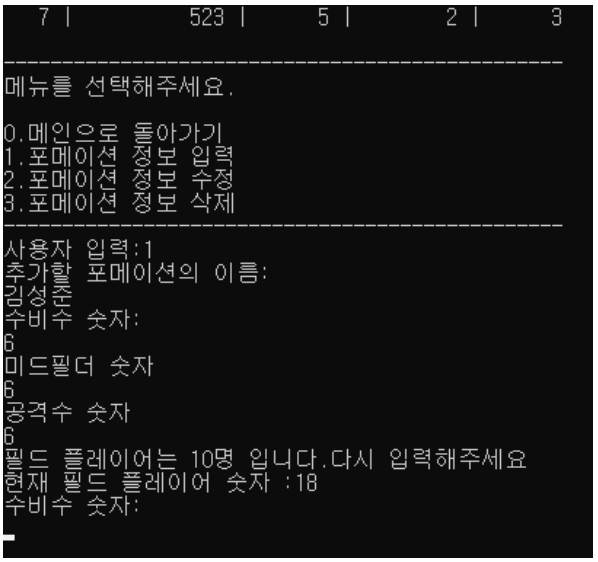
(3) 포메이션 정보 입력/수정/삭제 기능

- 메인화면

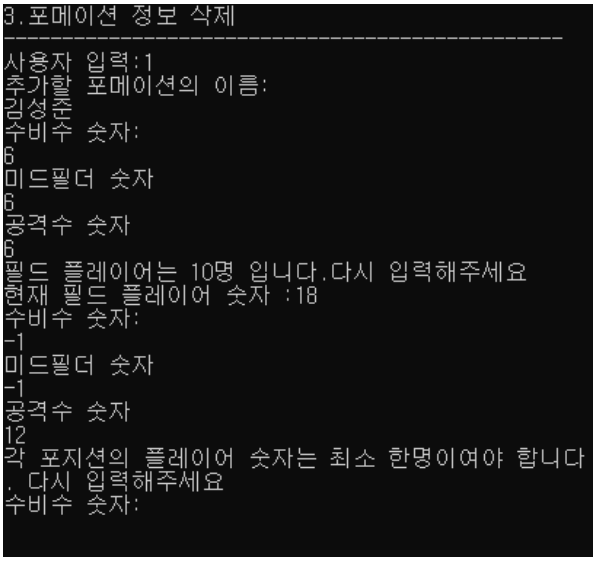
```
포메이션 정보 입력/수정/삭제 화면 입니다.

번호 |포메이션 이름 | 수비수 | 미드필더 | 공격수
-----------------------------------------------
   1 |          343 |      3 |        4 |      3
   2 |          352 |      3 |        5 |      2
   3 |          433 |      4 |        3 |      3
   4 |       42(31) |      4 |        2 |      4
   5 |          442 |      4 |        4 |      2
   6 |          541 |      5 |        4 |      1
   7 |          523 |      5 |        2 |      3

-----------------------------------------------
메뉴를 선택해주세요.

0.메인으로 돌아가기
1.포메이션 정보 입력
2.포메이션 정보 수정
3.포메이션 정보 삭제
-----------------------------------------------
사용자 입력:
```

- 입력 기능

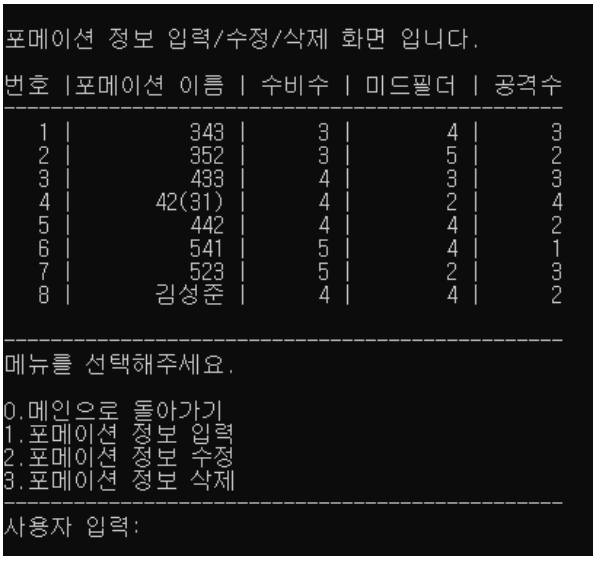
- 잘못된 입력1: 필드 플레이어 숫자가 10이 아닌경우



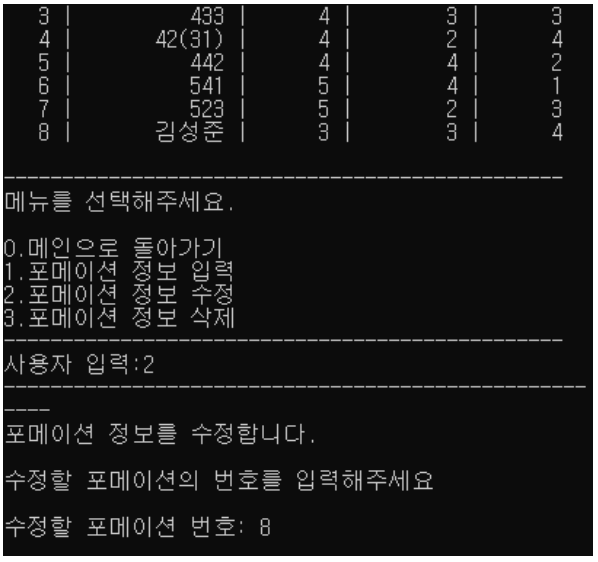
- 잘못된 입력 2: 필드 플레이어 합이 10이지만 유요하지 않은경우



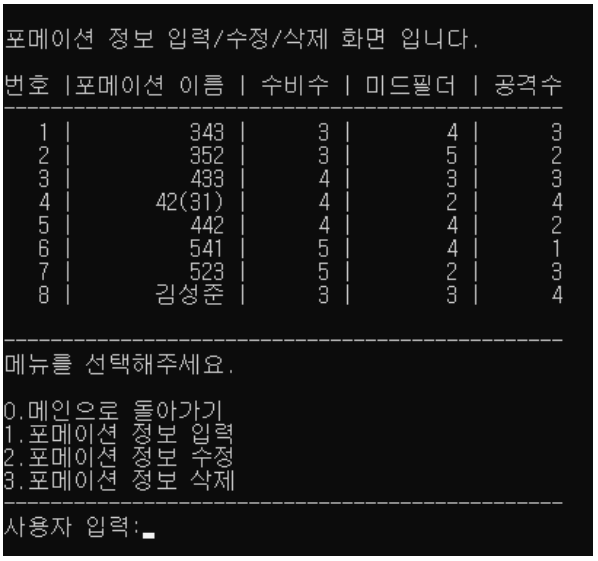
- 정상적인 입력(8번 포메이션 “김성준” 추가)



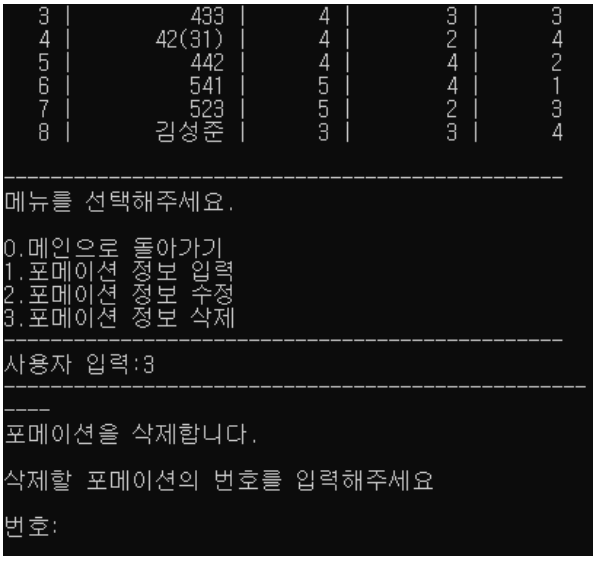
- 수정 기능
  - 입력 화면



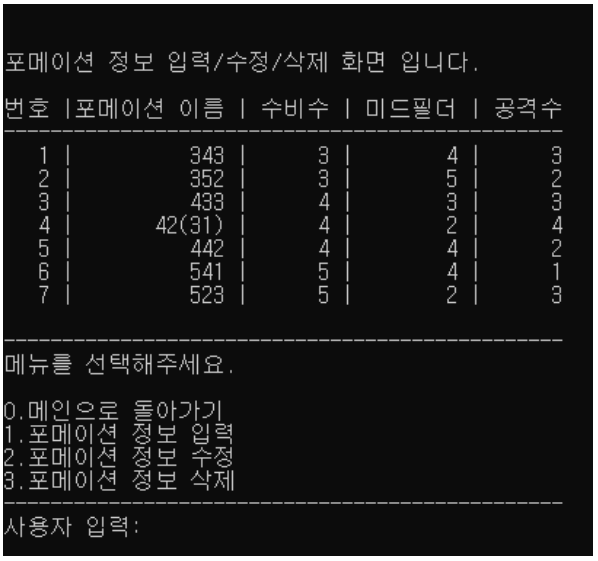
  - 수정 완료



- 삭제 기능
  - 삭제할 포메이션 인덱스 입력



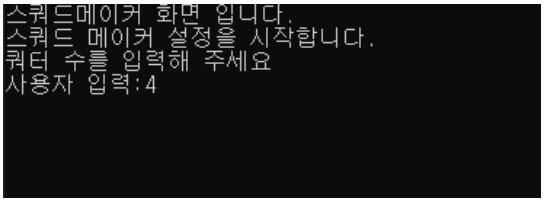
  - 삭제 완료 (8번 포메이션 삭제)



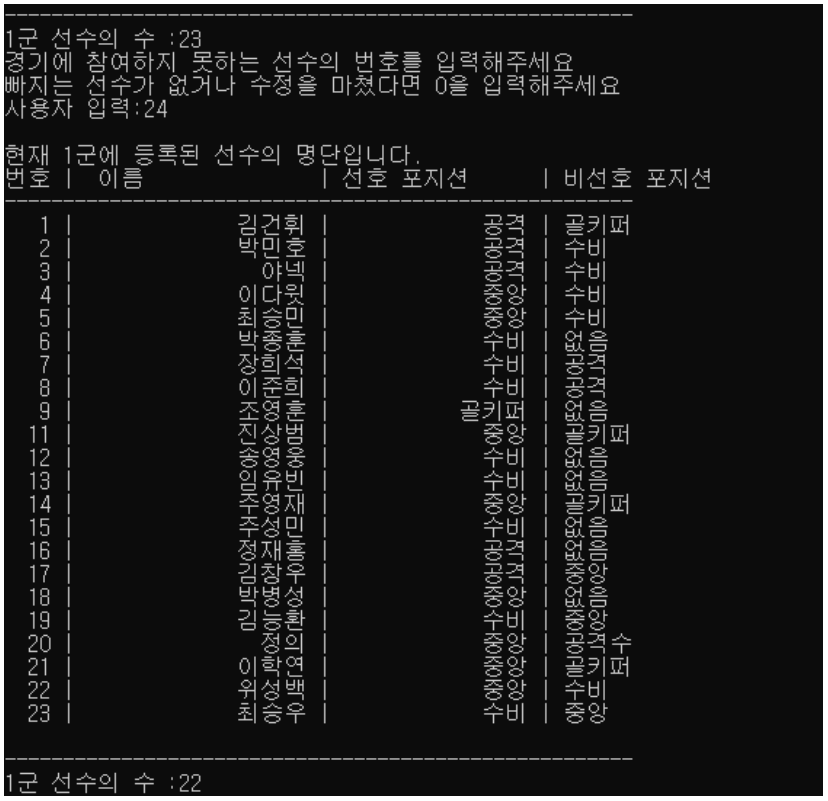
(4) 스쿼드 메이커 기능

- 쿼터수 및 지정

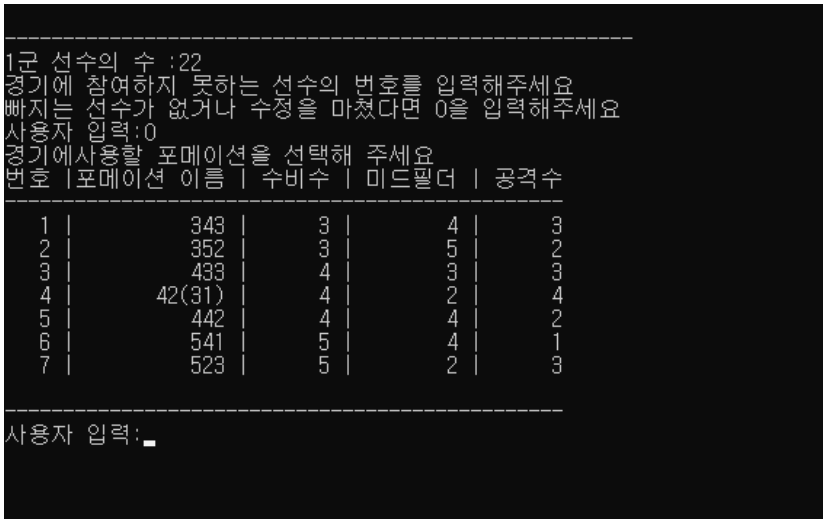
  - 쿼터수 4



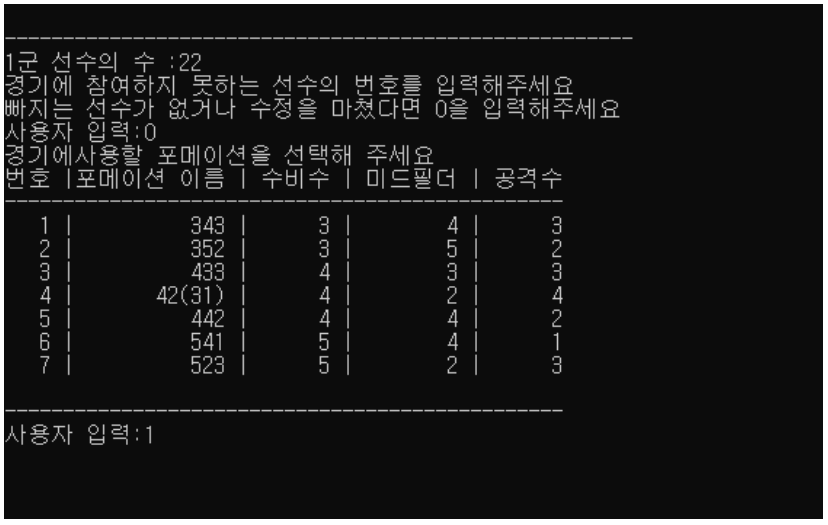
- 불참선수 제외 (10, 24번 플레이어 제외)
  - 1군에 등록된 선수의 명단을 출력하고 참여하지 않는 플레이어에 대한 입력을 마치면, 업데이트된 선수 명단을 출력한다.



- 0 입력을 통한 편집 종료(포메이션 선택으로 넘어간다.)



- 포메이션 선택(1번 343포메이션 선택)



- 스쿼드메이커 기능 최종 결과
  - 이 프로그램의 주요 기능으로 포메이션 입력이 완료되면 각 포지션의 선호도를 출력하고, 쿼터수에 맞게 스쿼드를 구성해서 출력한다.

5. 계획 대비 변경 사항

1) 선택메뉴 출력 기능 (내용 추가)

- 추가내용1: 사용자 입력중 비정상 입력을 처리하는 예외처리 코드를 삽입 하였습니다.

- 사유
  - 숫자가 아닌 다른 값(ex) 글자입력) 입력시 입력으로 0이 처리되어 프로그램이 종료되는 에러가 있었습니다. 예외 처리를 추가하여 프로그램이 정상 실행되도록 수정했습니다.
- 추가내용 2: 사용자 정보를불러오는 loadPlayersInfo()의중복함수loadPlayersInfo("Blueveins.txt")를 추가하였습니다.
- 사유
  - 기존의 함수는 어느정도 내장된 정보가 필요하다 생각해서 삽입한 함수이지만, 실제 사용하는 입장에서 생각해봤을때 txt파일을 읽어오는 것이 사용하기 편하겠다고 판단되어 중복함수를 추가하였습니다.

2) FunctionManager.cpp &헤더 추가

- 사유 : 기존에는 formations,players 배열의 객체에 대한 정보를 관리하기 위해 각 클래스의 멤버함수로 선언되어 있었고, 사용하려면 빈 객체를 추가해야하는 번거로움이 있었습니다. 또한 객체의 사용이 직관적이지 못한것 같아서 두 클래스의 객체 배열을 관리하는 FunctionManager 클래스&헤더를 추가하였습니다.

3) SquadMaker 기능 축소

- 사유 : 기존에는 각 쿼터 별로 참여여부를 조사하고, 그 결과를 반영하고 싶었으나, 구현 방법이 너무 복잡해서 기능을 축소하였습니다.

6. 느낀점

- 다른 수업들을 통해서 테스팅이나 프로젝트 설계에 대해서 배웠지만 제대로 사용해 보지 못한것 같아서 아쉬웠다. 처음 프로젝트를 설계한대로 구현 하고 싶었으나, c++ 수업 때 배운 내용을 하나라도 적용 해보고 싶어서 계속 수정을 거치다 보니 구현 부분이 자꾸 바뀌어서 혼란스러웠다. 처음 설계 할 때 개인 능력을 너무 과대 평가한 탓인지 실제 간트차트에 따라 구현을 하는 과정에서 설계가 너무 단순하고 일정을 너무 빡빡하게 잡은것 같다고 생각했다. 코드를 짜는것 말고도 일정관리나 테스팅 부분에 대해 복습이 필요한것 같다. 또한 소프트웨어 공학론 에서 배웠던 디자인 패턴이나, 디자인 원리, 모델링 방법등을 적용했다면 설계를 더 쉽게 하지 않았을까 후회한다. 이렇게 긴 호흡으로 프로젝트를 해본것이 거의 처음이라 이번 경험을 토대로 이 프로젝트의 기능을 확장하거나 다른 프로젝트를 수행해보면 도움이 될 것 같다.