진척 보고서 2차

3. 진척사항

1) 기능 구현

기능 1.선택메뉴 출력

```
#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,

기능2. 선수정보 입력/수정/삭제 기능

```
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) 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.

기능3. 포메이션 입력/수정/삭제 기능

```
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.

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 midfielders,
                     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.

3) FunctionManager 클래스 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;
// 다른 멤버 함수들의 구현은 여기에 추가
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";
  Player input;
  // 이름 입력
  cout << "이름: ";
  cin >> input.name;

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

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";
}

void EditPlayer(vector<Player> &players) {
  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";
}

void GetInformation(const vector<Player> &players) {
  cout << "번호 |  이름               | 선호 포지션      | 비선호 포지션\n";
  cout << "-------------------------------------------------------\n";

  for (int i = 0; i < players.size(); i++) {
    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;
  }
  cout << endl;
  cout << "-------------------------------------------------------\n";
}

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;
}

// 현재 포메이션 정보를 표시하는 함수 구현
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";
}
```

```
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, forward));
        cout << "새로운 포메이션이 추가되었습니다.\n";
        break;
      } else {
        cout << "필드 플레이어는 10명 입니다.다시 입력해주세요\n";
        cout << "현재 필드 플레이어 숫자 :" << def + mid + forward << endl;
      }
    } else {
      cout << "각 포지션의 플레이어 숫자는 최소 한명이여야 합니다. 다시 "
              "입력해주세요 \n";
    }
  }
}
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 << endl;
      }
    } else {
      cout << "각 포지션의 플레이어 숫자는 최소 한명이여야 합니다. 다시 "
              "입력해주세요 \n";
    }
  }
}
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";
}
```

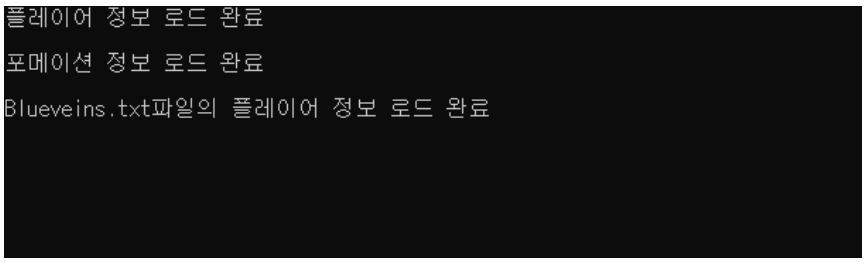
- void InsertPlayer(vector<Player> &players)

- 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 객체 배열에 추가하고 알림문구를 출력한다.
- void DeletePlayer(vector<Player> &players); //플레이어 삭제
  - 삭제할 플레이어의 번호를 입력받고, erase()함수를 이용해 해당 인덱스의 객체를 제거한다.
- void EditPlayer(vector<Player> &players); // 플레이어 수정
  - 수정할 플레이어의 번호를 입력받고,해당 인덱스의 객체 정보를 덮어쓰기 한다. 덮어쓰기 하는 과정은 선수 입력 과정과 동일하게 유효성 검사를 한다.
- void GetInformation(const vector<Player> &players); // 플레이어 정보 보여주는 함수
  - 객체배열을 파라미터로 입력 받고, 저장된 값을 출력한다.
  - setw()함수를 이용하여 출력화면에 고르게 정렬되도록 한다.
- string GetValidPosition(const string &prompt); // 입력된 포지션 유효성 검사 및 올바른 입력 반환하는 함수
  - 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)
  - 현재 포메이션 정보를 출력하는 함수
  - formations객체 배열을 입력으로 받는다.
  - 반복문을 통해 객체 배열을 돌면서 배열에 저장된(포메이션이름,수비수 숫자,미드필더 숫자,공격수 숫자)를 입력받는다.
- void insertFormation(vector<Formation> &formations)
  - 포메이션 정보 추가 하는 함수
  - formations 객체 배열을 입력으로 받는다.
  - 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.
- void EditFormation(vector<Formation> &formations)
  - 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)
  - 사용자로부터 삭제할 포메이션의 번호를 입력받고,그 입력을 delete_index에 저장한다.
  - erase()함수를 이용하여 해당 인덱스의 정보를 삭제한다.

2) 테스트 결과

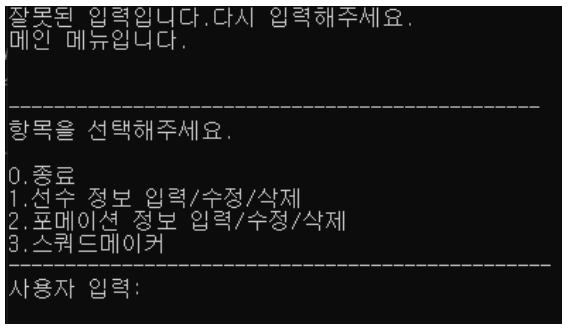
(1) 선택메뉴 출력

세부기능1: 저장된 정보 불러오기



세부기능2: 메인메뉴 출력

-잘못된 입력의 경우(숫자가 아닌 텍스트 입력시)



```
메인 메뉴입니다.


------------------------------------------------------------
항목을 선택해주세요.

0.종료
1.선수 정보 입력/수정/삭제
2.포메이션 정보 입력/수정/삭제
3.스쿼드메이커

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

선수정보 입력/수정/삭제 화면

```
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
```

메인메뉴 되돌아가기

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

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

------------------------------------------------------
사용자 입력:0
메인 메뉴입니다.


------------------------------------------------------
항목을 선택해주세요.

0.종료
1.선수 정보 입력/수정/삭제
2.포메이션 정보 입력/수정/삭제
3.스쿼드메이커

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

(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

```
선호 포지션:중앙
비선호 포지션:수비
선수정보가 수정 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
  10 |              김성준 |             수비 | 없음

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

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

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

- 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

```
   7 |          523 |      5 |        2 |      3

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

0.메인으로 돌아가기
1.포메이션 정보 입력
2.포메이션 정보 수정
3.포메이션 정보 삭제
-------------------------------------------------
사용자 입력:1
추가할 포메이션의 이름:
김성준
수비수 숫자:
6
미드필더 숫자
6
공격수 숫자
6
필드 플레이어는 10명 입니다.다시 입력해주세요
현재 필드 플레이어 숫자 :18
수비수 숫자:
```

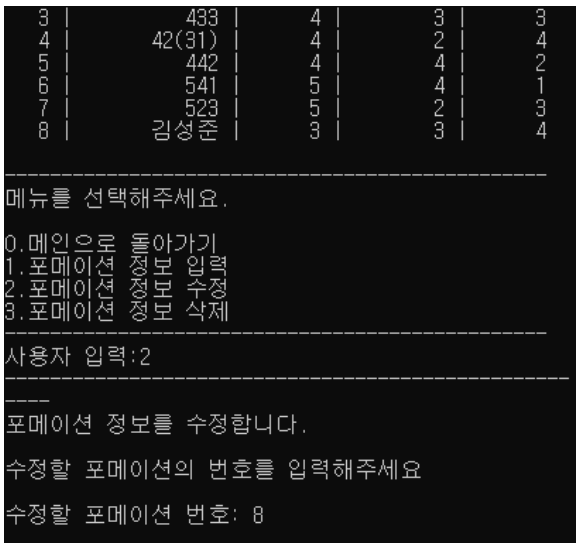
  - 잘못된 입력 2: 필드 플레이어 합이 10이지만 유요하지 않은경우

```
3.포메이션 정보 삭제
-------------------------------------------------
사용자 입력:1
추가할 포메이션의 이름:
김성준
수비수 숫자:
6
미드필더 숫자
6
공격수 숫자
6
필드 플레이어는 10명 입니다.다시 입력해주세요
현재 필드 플레이어 숫자 :18
수비수 숫자:
-1
미드필더 숫자
-1
공격수 숫자
12
각 포지션의 플레이어 숫자는 최소 한명이여야 합니다
. 다시 입력해주세요
수비수 숫자:
```

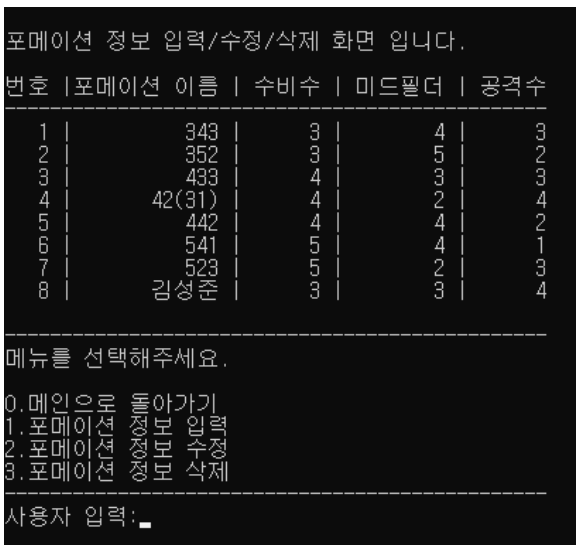
  - 정상적인 입력(8번 포메이션 “김성준” 추가)

```
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
   8 |       김성준 |      4 |        4 |      2

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

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

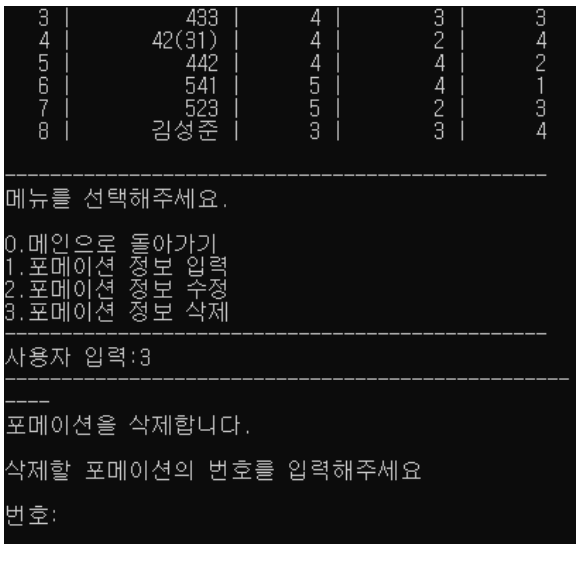
- 수정 기능
  - 입력 화면



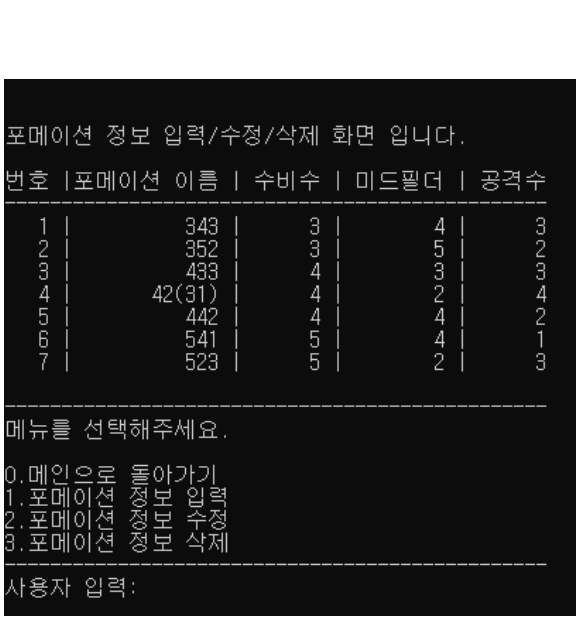
- 수정 완료



- 삭제 기능
  - 삭제할 포메이션 인덱스 입력



  - 삭제 완료 (8번 포메이션 삭제)



4. 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**) FunctionManager클래스 &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.

5. 프로젝트 일정 (간트차트)

일정	소요일
제안서 작성	3일
기능1(메뉴 출력)	2일
기능2(선수 정보)	2일
기능3(포메이션 정보)	3일
기능4_1(사용자 입력 받기)	2일(기능 1·2·3 선행)
기능4_2(최종 결과 출력)	5일 (기능 1~4_1 선행)
기능5-1Player 클래스	3일
기능5-2Formation클래스	3일
함수 분리 (FunctionManager	3일
예외 처리 및 오류 수정	3일

업무	**11/3**	**11/10**	**11/17**	**11/26**	12/03	12/10	12/12	12/15	12/17	12/24
제안서 작성	**------>**									
기능1			**---->**							
기능2				**---->**						
기능3					**--**	**--->**				
기능4	**세부기능 1**							**----->**		
	세부기능 2							**---**	**--->**	
기능5	**세부기능 1**			**---->**						
	세부기능 2				**---->**					
함수 분리						**--->**				
예외&오류 처리							**--->**			

- 수정 사항으로 인해 간트차트가 수정되었습니다.
  - 예외&오류 처리 하는 부분이 생겨서 간트차트를 수정 했습니다.(3일)
  - 함수 분리 (FunctionManager) 코드를 추가하였습니다. (3일)
- 수정 사항으로 인해 일정이 변경 되어 기능 4에대한 구현이 1주일 연기되었습니다.

제목 없음