C++ 프로그래밍 및 실습

성적 관리 시스템

진척 보고서 #2

제출일자: 2023.12.17
제출자명: 임중훈
제출자학번: 191111

1. 프로젝트 목표

1) 배경 및 필요성

대학 생활을 하는 중 성적에 대한 관심과 욕심이 있어 이 프로그램을 만들고자 했고, 저뿐만 아니라 대한민국 모든 학생들이라면 성적에 관심이 많아서 항상 자 신의 성적을 받아서 평균을 내고 그에 따른 석차를 많이 궁금해하기 때문에 이 프로그램을 만들고자 했습니다.

성적 관리를 컴퓨터로 하기 때문에 효율성이 향상할 수 있고, 오류 또한 발생 가 능성이 감소합니다. 수기로 작성하여 성적을 계산하는 것은 오류가 발생할 확률 이 있지만, 컴퓨터 프로그램을 통해 성적을 입력만 하면 컴퓨터가 계산을 해주기 때문에 많은 오류와 시간을 아낄 수 있습니다.

또한, 한 시스템에서 학생들의 정보를 입력하고 교수님, 선생님들은 그 해당 학생 들의 성적을 입력해서 해당 과목의 평균도 같이 계산해주는 시스템이 존재하면 학생과 교수님들 모두 성적에 관해서 시간을 투자하는 시간이 줄어들 것이다라 고 생각했습니다.

마지막으로 각 과목 별 점수만 알려주는 것이 아닌 성적 통계를 통해 자신의 성 적 위치를 알 수 있을 뿐만 아니라 학점 계산 및 석차까지 알 수 있기 때문에 많 은 효율이 있을 것이라고 생각합니다. 물론, 모든 학생들이 다른 학생의 석차를 알면 보안의 위험이 있기 때문에 성적 조회는 이름뿐만 아니라 학번과 주민등록 번호를 통해 성적 조회를 할 수 있습니다.

2) 프로젝트 목표

학생들이 온전히 학업에만 집중을 하고 부가적인 서비스는 이 프로그램을할 수 있게 하는 것이 목표입니다.

① 학생들의 인적 사항을 입력함으로써 해당 과목을 듣는 학생들을 관리할 수 있다.

② 학생들의 편의를 위한 것이기 때문에 학생이 자신의 성적을 조회하면 자신의 점수와 해당 과목 평균을 알려준다.

③ 자신의 석차 조회를 통해 자신의 석차를 알 수 있게 해준다.

④ 성적 통계를 통해 해당 과목의 평균을 구해줌으로써 자신이 얼마나 더공부를 해야하는지도 알 수 있게 해준다.

3) 차별점

성적이 완전히 기입이 안된 상황에서(Ex: 중간고사) 자신의 성적이 어느 정도 위 치인지 궁금해 할 수 있을 것이라고 생각하여 이전에는 자신이 맞은 점수가 어느 위치인지 알 수 있게 해준다는 점.

또한, 자신의 석차가 궁금하면 석차 조회를 위해 대학 본부로 가는 상황들을 줄 여서 학생들의 편의성을 증대 시켜준다는 점.

각 학점 별 점수를 알려줘서 자신들이 원하는 학점을 맞으려면 몇 점의 점수가 더 필요한지도 알려준다는 점.

2. 기능 계획

1) 정보 관리

- 학생의 수를 입력받고 그 학생 수 만큼 학생의 정보 학번, 이름, 전화번호를 입력

- 올바르지 않는 정보를 입력했을 경우 예외처리.

(1) 학생 관리

- 입력받은 학생의 정보를 저장하고 학생 정보 조회를 눌렀을 경우, 이 저장한 값들을 출력합니다.

- 각 학생의 성적을 기입할 수 있는 성적표를 이 입력받은 값을 통해서 생성합니다.

2) 성적 입력

- 학생 학번을 입력하고 성적을 입력하면 그 데이터를 저장하고 관리

- 성적 입력란에 실수가 아니라면 예외처리

3) 성적 통계

- 입력 받은 성적을 이용해 평균을 구하고, 학생의 학점 또한 계산

- 각 학점별 경계선을 알려주도록 계산

    + 이것은 앞으로 성적 조회할 때 사용

4) 성적 조회 (학점 계산, 석차 등)

- 학생 이름, 학번을 통해 성적을 조회할 수 있는 기능.

- 자신의 성적을 알려주고 석차 조회를 원하면 석차를 알려주는 기능

    1) 점수 -> 학생의 원점수를 알려줌

    2) 등수 -> 현재 학생의 등수를 알려줌

    3) 학점 -> 현재 학생의 점수를 통해 학점을 계산하여 알려줌

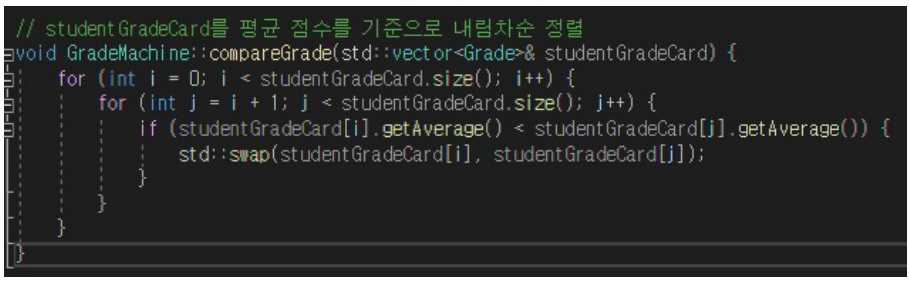
5) 목표 성적 (자신이 원하는 학점을 달성하기 위해 필요한 점수)

- 학번을 입력하고 자신이 받고 싶은 성적을 입력 받은 후 자신이 받고 싶은 학점을 위해 얼마나 더 점수를 받아야하는지 알려줌

3. 진척사항

(1) 성적 통계

- GradeMachine.cpp

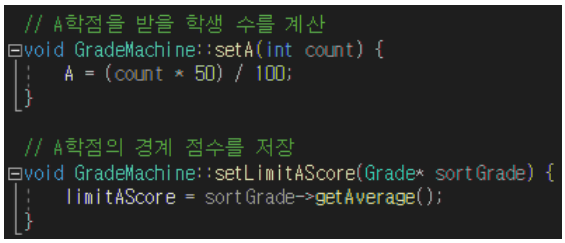


```
// studentGradeCard를 평균 점수를 기준으로 내림차순 정렬
void GradeMachine::compareGrade(std::vector<Grade>& studentGradeCard) {
    for (int i = 0; i < studentGradeCard.size(); i++) {
        for (int j = i + 1; j < studentGradeCard.size(); j++) {
            if (studentGradeCard[i].getAverage() < studentGradeCard[j].getAverage()) {
                std::swap(studentGradeCard[i], studentGradeCard[j]);
            }
        }
    }
}
```

- 성적 정보가 저장되어 있는 studentsGradeCard의 값을 '&'참조자를 이용해서 값을 받아오고, 그 값을 내림차순으로 정렬한다.

- 참조자, 2중 for문을 이용

- 처음엔 algorithm 라이브러리를 이용해서 sort하려고 하였지만, 되질 않아서 sort를 직접 함수로 구현을 하였습니다.



```
// A학점을 받을 학생 수를 계산
void GradeMachine::setA(int count) {
    A = (count * 50) / 100;
}

// A학점의 경계 점수를 저장
void GradeMachine::setLimitAScore(Grade* sortGrade) {
    limitAScore = sortGrade->getAverage();
}
```

- 학점을 계산하기 위해 A학점을 받을 상위 50% 학생들을 구하기 위해 전체 학생 수에 0.5를 곱한 값을 A에 저장한다

- 또한, 위에서 계산한 A학점의 학생 수에 해당하는 학생의 성적표를 갖고와서 limitAScore값에 저장한다.

- main에서 할 수 있는 계산들을 해당 데이터가 존재하는 cpp파일에서 계산하도록 코드 구현 및 포인터 변수를 활용하여 값을 받아오도록 작성

- GradeMachine.cpp

```
// GradeMachine의 기본 적인 설정을 하기 위한 생성자
GradeMachine::GradeMachine(std::vector<Grade>& studentGradeCard, int count) {
    compareGrade(studentGradeCard);
    setA(count);
}

// A학점의 경계 점수를 찾아서 set함수를 호출할 함수
void GradeMachine::searchLimitALineScore(GradeMachine& gradeMachine, std::vector<Grade>& studentGradeCard) {
    if (gradeMachine.A != 0) {
        Grade grade = studentGradeCard[A - 1];
        gradeMachine.setLimitAScore(&grade);
    }
}
```

- Grade.cpp

```
// 성적 산출하는 함수
void Grade::getGradeCard(Grade* grade, std::string examName) {
    if (grade->midtermScore != 0 && grade->lasttermScore != 0) {
        grade->calculateExamAverage(grade);
    }
    else if (grade->lasttermScore == 0) {
        grade->average = grade->midtermScore;
    }
    else if (grade->midtermScore == 0) {
        grade->average = grade->lasttermScore;
    }
}
```

```
// 성적의 평균을 낼 함수
void Grade::calculateExamAverage(Grade* grade) {
    grade->average = (grade->midtermScore + grade->lasttermScore) / 2;
}
```

- GradeMachine의 생성자에 성적표 참조값을 매개변수로 받아와서 위에서 작성한 compareGarde함수를 통해 성적표를 내림차순으로 정렬 하고 count : 학생 값을 setA로 넘겨주어 A를 받을 학생 수를 계산하도록 작성

- 학생들의 성적 조회를 위해서 searchLimitALineScore 함수를 생성하여 A의 학점 경계에 있는 학생의 성적을 저장할 수 있도록 setLimitAScore함수 호출할 함수

- 생성자, 참조자를 이용하여 함수를 구현

(2) 성적 조회

- GradeMachine.cpp

```cpp
// 성적 조회를 위한 함수
void GradeMachine::searchStudentGrade(std::vector<Grade>& studentGradeCard, int studentID, std::string format) {
    // 해당 학생이 존재하는지 안하는지 확인할 bool 변수
    bool exist = false;
    // 원하는 성적 출력이 학점일 경우
    if (format == "학점") {
        for (auto it : studentGradeCard) {
            if (it.getStudentID() == studentID) {
                exist = true;
                if (it.getAverage() < 0) {
                    std::cout << studentID << "학생의 성적이 입력되어 있지 않습니다." << std::endl;
                }
                else if (it.getAverage() >= limitAScore) {
                    std::cout << "A학점 입니다." << std::endl;
                }
                else if (it.getAverage() < limitAScore) {
                    std::cout << "B학점 입니다." << std::endl;
                }
            }
        }
    }
    // 원하는 성적 출력이 점수일 경우
    if (format == "점수") {
        for (auto it : studentGradeCard) {
            if (it.getStudentID() == studentID) {
                exist = true;
                if (it.getLasttermScore() >= 0 && it.getMidtermScore() >= 0) {
                    std::cout << studentID << "학생의 점수 평균은 = " << it.getAverage() << "입니다." << std::endl;
                }
                else if (it.getLasttermScore() < 0 && it.getMidtermScore() >= 0){
                    std::cout << studentID << "학생의 중간고사 점수은 = " << it.getMidtermScore() << "입니다." << std::endl;
                }
                else {
                    std::cout << studentID << "학생의 성적은 아직 입력되어 있지 않습니다." << std::endl;
                }
            }
        }
    }
```

```cpp
    // 원하는 성적 출력이 등수일 경우
    if (format == "등수") {
        int index = 0;
        for (auto it : studentGradeCard) {
            index++;
            if (it.getStudentID() == studentID) {
                exist = true;
                if (it.getAverage() < 0) {
                    std::cout << studentID << "학생의 성적이 입력되어 있지 않습니다." << std::endl;
                }
                else {
                    std::cout << studentID << "학생의 등수는 = " << index << "입니다." << std::endl;
                }
            }
        }
    }

    // 해당 학번이 존재하지 않을 경우
    if (!exist) {
        std::cout << "해당 " << studentID << "학번은 존재하지 않습니다." << std::endl;
    }
}
```

- Application.cpp

```
void searchStudentGrade( ) {
    int studentID;
    string gradeFormat;

    cout << "학번을 입력하세요 : ";
    cin >> studentID;

    cout << "점수가 궁금하면 '점수'" << endl
        << "등수가 궁금하면 '등수'" << endl
        << "학점이 궁금하면 '학점'" << endl
        << "입력해주세요 : ";
    cin >> gradeFormat;

    gradeMachine.searchStudentGrade(studentGrade, studentID, gradeFormat);
}
```

- 학생들의 성적 조회를 위한 함수

- 학생들이 출력을 원하고자 하는 것 '학점', '등수', '점수'인지를 main에서 입력 받아 매개변수로 넘어온 format을 통해 조건문으로 작성

- 학생들의 성적표를 입력받아서 범위 for문을 활용하여 성적표를 순회하여 그 해당 학생의 학번을 찾는다.

- 찾는 학번이 없을 경우 exist변수는 처음 생성한 그대로 false 이므로 맨 아래에 있는 if문이 실행.

- 학번이 존재할 경우 사용자의 원하는 형태에 따라 값을 출력하도록 작성 exist 변수는 True로 초기화.

- 많은 조건분기, 함수, 범위 for문을 활용

(3) 목표 성적

- GradeMachin.cpp

```cpp
// 학생의 목표 성적을 입력하여 정보를 얻기 위한 함수
void GradeMachine::searchStudentGoalGrade(std::vector<Grade>& studentsGradeCard, int studentID, std::string goalGrade) {
    int index = 0;
    bool exist = false;
    for (auto it : studentsGradeCard) {
        index++;
        if (it.checkEveryExamScore()) {
            if (it.getStudentID() == studentID) {
                exist = true;
                if (goalGrade._Equal("A")) {
                    if (index > A) {
                        std::cout << "현재 최하 A 학점의 점수는 = " << limitAScore << std::endl;
                        std::cout << "현재 " << studentID << "학생의 점수 = " << it.getAverage() << std::endl;
                        std::cout << "따라서 " << studentID << "학번은 B 학점 입니다." << std::endl;
                    }
                    else {
                        std::cout << "당신은 이미 A 학점 입니다." << std::endl;
                    }
                }
                else {
                    if (index < A) {
                        std::cout << "당신의 현재 학점은 A 입니다." << std::endl;
                    }
                    else {
                        std::cout << "당신은 이미 목표 학점인 " << goalGrade << " 학점 입니다." << std::endl;
                    }
                }
            }
        }
```

```cpp
        else {
            if (it.getStudentID() == studentID) {
                exist = true;
                if (goalGrade._Equal("A")) {
                    if (index > A) {
                        std::cout << "현재 최하 A 학점의 점수는 = " << limitAScore << std::endl;
                        std::cout << "현재 " << studentID << "학생의 중간 고사 점수 = " << it.getMidtermScore() << std::endl;
                        std::cout << "따라서 " << goalGrade << "학점은 받기 위해서 " << studentID << "학번은 " << it.getMidtermScore() - limitAScore << "만큼의 점수가 더 낮습니다." << std::endl;
                    }
                    else {
                        std::cout << "당신은 이미 A 학점 입니다." << std::endl;
                    }
                }
                else {
                    if (index < A) {
                        std::cout << "당신의 현재 학점은 A 입니다." << std::endl;
                    }
                    else {
                        std::cout << "당신은 이미 목표 학점인 " << goalGrade << " 학점 입니다." << std::endl;
                    }
                }
            }
        }
    }

    if (!exist) {
        std::cout << "해당 " << studentID << "학번은 존재하지 않습니다." << std::endl;
    }
}
```

- Application.cpp

```
void searchWantGoalScore() {
    int studentID;
    string goalGrade;

    cout << "학번을 입력하세요 : ";
    cin >> studentID;

    cout << "원하는 학점을 적으세요. (A or B) : ";
    cin >> goalGrade;

    gradeMachine.searchStudentGoalGrade(studentGrade, studentID, goalGrade);
}
```

- 학번을 입력받아서 해당 학생의 성적을 조회하고 그 학생이 받고 싶은 점수를 입력받는다. 그것에 따른 학생의 위치를 알려주고, 그 해당 학점을 받지 못한 상태라면 A학점의 학생과 몇 점의 차이가 존재하는지 알려주는 기능을 한다.

- _Equal를 이용해서 문자열 비교, 범위 for문, 많은 조건분기

(4) 성적 입력에 있어서 편의성 향상

- 학생 점수를 입력하기 전 입력할 학생의 수를 입력 받아서 그 수 만큼 점수를 입력하도록 작성

```
void inputStudentScore() {
    int count;
    cout << "성적 입력을 하고 싶은 학생의 수를 입력하세요 : ";
    cin >> count;

    for (int i = 0; i < count; i++) {
        int tmp = 0;
        int studentID;
        float score;
        string examName;

        cout << "학생의 학번을 입력하세요 : ";
        cin >> studentID;

        cout << "중간고사면 '중간' 기말고사면 '기말'을 입력해주세요 : ";
        cin >> examName;

        cout << "학생의 점수를 입력하세요 : ";
        cin >> score;

        for (auto& it : studentGrade) {
            if (it.getStudentID() == studentID) {
                it.inputStudentScore(&it, score, examName);
                it.getGradeCard(&it, examName);
                tmp++;
            }
        }

        if (tmp == 0) {
            cout << "해당 " << studentID << "학번은 존재하지 않습니다." << endl;
        }
    }

    gradeMachine = GradeMachine(studentGrade, studentCount);
    gradeMachine.searchLimitALineScore(gradeMachine, studentGrade);
}
```

- for문을 활용하여 학생들의 성적 입력 받는 for문 반복

(5) 입력 값에 대한 확인 기능

```
// 해당 점수가 맞는지 확인하는 함수
bool Grade::checkScore(float score) {
    std::cout << score << "점이 맞습니까?(y/n): ";
    std::string answer;
    std::cin >> answer;
    if (answer == "y") {
        return true;
    }
    else if (answer == "n") {
        return false;
    }
    else {
        throw answer;
    }
}
```

```
// 중간고사 점수를 저장할 함수
bool Grade::setMidtermScore(float score) {
    try {
        if (checkScore(score)) {
            midtermScore = score;
        }
        else {
            std::cout << "점수를 다시 입력해주세요 : ";
            std::cin >> score;
            midtermScore = score;
        }
        return true;
    }
    catch (std::string input) {
        std::cout << "답은 y or n으로만 답하셔야합니다." << std::endl;
        return false;
    }
}
```

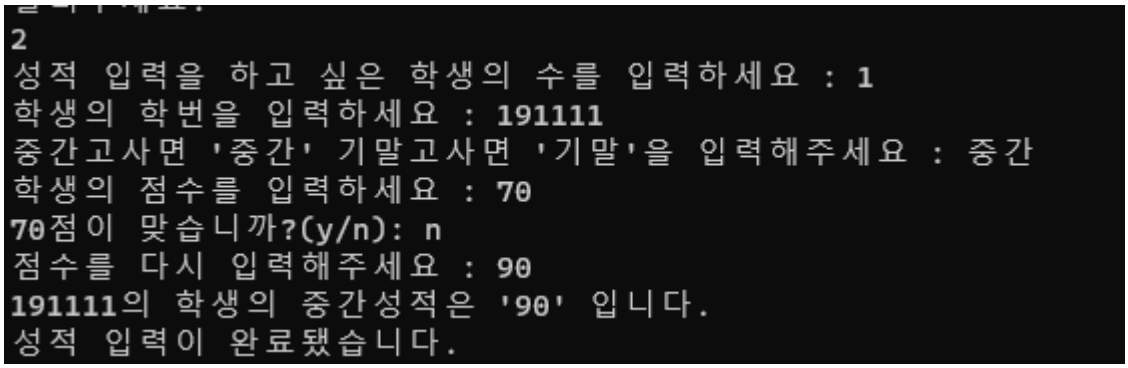
- 해당 점수를 저장하는 setMidtermScore 함수에서 점수 여부를 확인하는 함수 checkScore함수를 호출하여 점수 여부를 확인하도록 작성

- checkScore함수를 따로 뺀 이유 : 중간고사 성적을 입력하는 함수와 기말고사 점수를 입력하는 함수의 중복 코드 발생 -> 해당 check하는 함수를 함수로 추출

- 함수의 재사용성 활용 및 코드 중복 제거를 위한 노력, 예외 처리

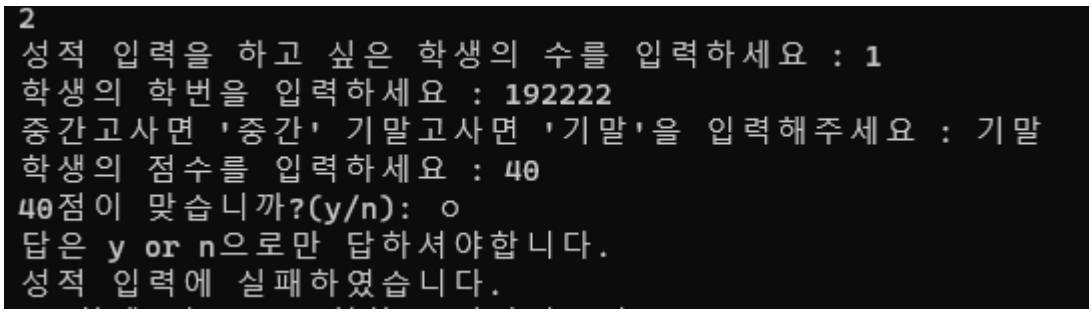
2) 테스트 결과

(1) 성적 입력 확인 여부에서 'n'을 입력했을 경우



- 점수를 입력했는데 잘못 입력했을 경우 'n'을 입력하여 재입력 받도록 구현.

(2) 점수 확인 여부 검사에서 y/n을 입력안했을 경우 (예외)



- 예외 처리를 통해 성적 입력 실패 메시지 출력.

(3) 학생 성적 조회 기능

```
3
학번을 입력하세요 : 191111
점수가 궁금하면 '점수'
등수가 궁금하면 '등수'
학점이 궁금하면 '학점'
입력해주세요 : 점수
191111학생의 점수 평균은 = 70입니다.
1. 학생 정보를 조회하고 싶다면 1번
2. 학생 성적를 입력하고 싶다면 2번
3. 성적 조회를 하고 싶다면 3번
4. 원하는 성적을 받기 위해 필요한 점수를 알고 싶다면 4번
0. 종료하고 싶다면 0번
눌러주세요.
3
학번을 입력하세요 : 191111
점수가 궁금하면 '점수'
등수가 궁금하면 '등수'
학점이 궁금하면 '학점'
입력해주세요 : 등수
191111학생의 등수는 = 1입니다.
```

```
3
학번을 입력하세요 : 191111
점수가 궁금하면 '점수'
등수가 궁금하면 '등수'
학점이 궁금하면 '학점'
입력해주세요 : 학점
A학점 입니다.
```

- 자신이 확인하고자 하는 성적 format을 입력하고 그 형태로 출력

(4) 목표 성적 출력

```
4
학번을 입력하세요 : 192222
원하는 학점을 적으세요. (A or B) : A
현재 최하 A 학점의 점수는 = 100
현재 192222학생의 중간 고사 점수 = 40
따라서 A학점은 받기 위해서 192222학번은 60만큼의 점수가 더 낮습니다.
```

```
4
학번을 입력하세요 : 191111
원하는 학점을 적으세요. (A or B) : A
당신은 이미 A 학점 입니다.
```

```
4
학번을 입력하세요 : 192222
원하는 학점을 적으세요. (A or B) : A
현재 최하 A 학점의 점수는 = 100
현재 192222학생의 점수 = 50
따라서 192222학번은 B 학점 입니다.
```

- 학생의 목표 성적을 입력하고 해당 학점이 아닐 경우 필요한 점수를 출력하고 해당 학점일 경우는 이미 해당 학점이라고 출력.

- 중간고사 기말고사 성적이 다 입력되어 있을 경우는 점수와 학점만 출력.

4. 계획 대비 변경 사항

1) 변경 내역 제목

- 이전 : 1. 성적 입력할 때마다 성적 조회 메뉴 번호를 눌러서 성적을 기입.

2. 성적 입력이 잘못됐을 시 그냥 입력이 됨

3. 성적 입력 사항을 for문 안에서 입력 받음.

- 이후 : 1. 성적 입력메뉴를 누르면 성적 입력할 학생 수를 입력받음.

2. 성적 입력 여부 확인 조사, 추후 'y'가 나올때 까지 체크함수 호출하도록 수정

3. 추후에 for문을 돌리기 전에 입력 받을 조건을 입력하도록 작성하기

-> ex) '중간' or '기말'

- 사유 : 1. 편의성 향상을 위해서.

2. 테스트를 해보면서 의도치 않게 성적이 입력이 됐는데 그것을 n을 하고 다시 입력을 했을 때도 잘못 입력했다면 고칠 방도가 없기 때문에. 이것 또한 편의성 향상.

3. 테스트를 하다보면서 불편함 및 성적은 '중간'이면 중간만 '기말'이면 기말만 입력한다고 생각하기 때문에

앞으로 남은 프로젝트는 코드를 보고, 테스트 해보면서 편의성 향상을 위해 계속 코드를 수정해 가는 방향으로 설정.

5. 프로젝트 일정

업무	11/3	11/10	11/26	12/1	12/15
제안서 작성	완료				
정보 관리 / 학생 관리		완료			
성적 입력			완료		
성적 통계				완료	
성적 조회				완료	
목표 성적				완료	