C++ 프로그래밍 및 실습

성적 관리 시스템

최종 보고서

제출일자: 2023.12.23
제출자명: 임중훈
제출자학번: 191111

1. 프로젝트 목표

1) 배경 및 필요성

대학 생활을 하는 중 성적에 대한 관심과 욕심이 있어 이 프로그램을만들고자 했고, 저뿐만 아니라 대한민국 모든 학생들이라면 성적에 관심이많아서 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 발생가능성이 감소합니다. 수기로 작성하여 성적을 계산하는 것은 오류가발생할 확률이 있지만, 컴퓨터 프로그램을 통해 성적을 입력만 하면 컴퓨터가계산을 해주기 때문에 많은 오류와 시간을 아낄 수 있습니다.

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

마지막으로 각 과목 별 점수만 알려주는 것이 아닌 성적 통계를 통해 자신의 성적 위치를 알 수 있을 뿐만 아니라 학점 계산 및 석차까지 알 수있기 때문에 많은 효율이 있을 것이라고 생각합니다. 물론, 모든 학생들이다른 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 조회하면자신의 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 정도위치인지 궁금해 할 수 있을 것이라고 생각하여 이전에는 자신이 맞은 점수가어느 위치인지 알 수 있게 해준다는 점.

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

각 학점 별 점수를 알려줘서 자신들이 원하는 학점을 맞으려면 몇 점의점수가 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 싶은학점을 위해 얼마나 더 점수를 받아야하는지 알려줌

3. 기능 구현

(1) 학생의 수와 학생들의 정보 입력

- 입출력

```
CPP 성적 관리 시스템입니다.
학생 수를 입력해주세요 : 2
학생의 이름을 입력해주세요 : 임중훈
학생의 학번을 입력해주세요 : 191111
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-1111-1111
학생의 이름을 입력해주세요 : 김지민
학생의 학번을 입력해주세요 : 192222
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-2222-2222
```

- 학생의 수를 입력 받고, 그 학생들의 이름과 학번, 전화번호를 입력받아서 vector에 저장한다. 여기에 입력한 정보들은 성적을 입력할 때, 성적을 조회할 때 등 학생이 존재하는지 여부를 체크할 때 사용한다.

- vector, 사용자 정의 데이터 타입(Student), for문, while문, validation 체크(조건 분기)

- 코드 스크린샷

```
void inputStudentCount() {
    cout << "학생 수를 입력해주세요 : ";
    cin >> studentCount;
}
```

```
Student inputStudentInformation() {
    Student student;
    string studentName = "";
    string phoneNumber = "";
    int studentID = 0;

    while (student.getStudentName() == "") {
        cout << "학생의 이름을 입력해주세요 : ";
        cin >> studentName;

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

        cout << "학생의 번호를 입력해주세요('-'로 구분해주세요) : ";
        cin >> phoneNumber;

        student = Student(studentName, studentID, phoneNumber);
    }

    return student;
}
```

```cpp
// 학생의 학번이 올바른 학번인지 확인할 bool 함수
// 6자리가 아니라면 return false
bool Student::checkStudentID(int studentID) {
    std::string strID = std::to_string(studentID);

    if (strID.length() != 6) {
        return false;
    }
    return true;
}

// 학생의 번호가 올바른 번호인지 확인할 bool 함수
bool Student::checkStudentPhoneNumber(std::string phoneNumber) {
    // 번호를 확인하기 위해 '-'를 기준으로 쪼개기 위한 과정
    int firstDash = phoneNumber.find('-');
    int secondDash = phoneNumber.find('-', firstDash + 1);

    std::string areaCode = phoneNumber.substr(0, firstDash);
    std::string firstPart = phoneNumber.substr(firstDash + 1, secondDash - firstDash - 1);
    std::string secondPart = phoneNumber.substr(secondDash + 1);

    // 올바른 번호가 아니라면 false 반환
    if (areaCode != "010") {
        return false;
    }
    if (firstPart.length() != 4) {
        return false;
    }
    if (secondPart.length() != 4) {
        return false;
    }
    return true;
}
```

```cpp
// 학번과 전화번호가 올바른 정보인지 확인할 함수를 호출할 함수
bool Student::checkStudentInformationValidation(int studentID, std::string phoneNumber) {
    if (checkStudentID(studentID)) {
        if (checkStudentPhoneNumber(phoneNumber)) {
            return true;
        }
    }
    std::cout << "다시 입력해주세요." << std::endl;
    return false;
}

// 기본 생성자
Student::Student() {
    studentName = "";
    phoneNumber = "";
    studentID = 0;
}

// 입력한 값을 각 학생 객체 인스턴스 변수에 저장
Student::Student(std::string studentName, int studentID, std::string phoneNumber) {
    if (checkStudentInformationValidation(studentID, phoneNumber)) {
        this->studentName = studentName;
        this->studentID = studentID;
        this->phoneNumber = phoneNumber;
    }

    if (studentName == "" || studentID == 0 || phoneNumber == "") {
        std::cout << "학생의 정보가 입력이 되지 않았습니다." << std::endl;
    }
}
```

(2) 학생 정보 조회

- 입출력

```
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.
1
학생의 이름 : 임중훈
학생의 학번 : 191111
학생의 전화번호 : 010-1111-1111
학생의 이름 : 김지민
학생의 학번 : 192222
학생의 전화번호 : 010-2222-2222
```

- 입력 받은 학생의 정보를 출력해준다.

- switch-case문, for문, vector, 함수 분리

```
while (true) {
    bool finish = false;
    result = printMenu();

    switch (result) {
    case 1:
        searchStudentInformation();
        break;
    case 2:
        inputStudentScore();
        break;
    case 3:
        searchStudentGrade();
        break;
    case 4:
        searchWantGoalScore();
        break;
    case 0:
        finish = true;
        break;
    }
```

```
void searchStudentInformation() {
    for (int i = 0; i < studentCount; i++) {
        cout << "학생의 이름 : " << students[i].getStudentName() << endl
            << "학생의 학번 : " << students[i].getStudentID() << endl
            << "학생의 전화번호 : " << students[i].getPhoneNumber() << endl;
    }
}
```

(3) 학생 성적 입력

- 입출력

```
성적 입력을 하고 싶은 학생의 수를 입력하세요 : 2
중간고사면 '중간' 기말고사면 '기말'을 입력해주세요(둘 다 입력시 '중간기말'을 입력해주세요) : 중간기말
학생의 학번을 입력하세요 : 191111
학생의 점수를 입력하세요('중간기말'이라면 ','를 이용해서 구분해주세요) : 80,90
80점이 맞습니까?(y/n): y
191111의 학생의 중간 성적은 '80' 입니다.
성적 입력이 완료됐습니다.
90점이 맞습니까?(y/n): y
191111의 학생의 기말 성적은 '90' 입니다.
성적 입력이 완료됐습니다.
학생의 학번을 입력하세요 : 192222
학생의 점수를 입력하세요('중간기말'이라면 ','를 이용해서 구분해주세요) : 70,60
70점이 맞습니까?(y/n): y
192222의 학생의 중간 성적은 '70' 입니다.
성적 입력이 완료됐습니다.
60점이 맞습니까?(y/n): n
점수를 다시 입력해주세요 : 80
80점이 맞습니까?(y/n): y
192222의 학생의 기말 성적은 '80' 입니다.
성적 입력이 완료됐습니다.
```

- 학생의 성적을 입력하고 점수 입력하고 싶은 시험 유형을 적고 해당 점수가 맞는지 확인 후 'y'가 입력이 되었으면 해당 점수를 기입하는 형식으로 구현. 잘못된 문자열을 입력하면 예외를 던져서 예외 처리를 하도록 구현.

- while문, 사용자 정의 데이터 타입(Grade), for문, auto, 참조자'&', 포인터, 조건 분기, try-catch문, 함수 재사용성을 향상 시키기 위한 함수 분리, switch-case문

```
void inputStudentScore() {
    int count;
    string examName;

    cout << "성적 입력을 하고 싶은 학생의 수를 입력하세요 : ";
    cin >> count;

    cout << "중간고사면 '중간' 기말고사면 '기말'을 입력해주세요(둘 다 입력시 '중간기말'을 입력해주세요) : ";
    cin >> examName;

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

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

        cout << "학생의 점수를 입력하세요('중간기말'이라면 ','를 이용해서 구분해주세요) : ";
        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);
}
```

```cpp
// 기본적인 성적 입력의 호출에 따른 함수들을 호출할 함수
void Grade::inputStudentScore(Grade* grade, std::string scores, std::string examName) {
    if (examName == "중간") {
        try {
            if (grade->setMidtermScore(isUsedOtherText(scores))) {
                std::cout << grade->studentID << "의 학생의 " << examName << "성적은 '" << grade->midtermScore << "' 입니다." << std::endl;
                std::cout << "성적 입력이 완료됐습니다." << std::endl;
            }
        }
        catch (...) {
            std::cout << "잘못된 문자를 넣었습니다." << std::endl;
            std::cout << "성적 입력에 실패하였습니다." << std::endl;
        }
    }
    if (examName == "기말") {
        try {
            if (grade->setLasttermScore(isUsedOtherText(scores))) {
                std::cout << grade->studentID << "의 학생의 " << examName << "성적은 '" << grade->lasttermScore << "' 입니다." << std::endl;
                std::cout << "성적 입력이 완료됐습니다." << std::endl;
            }
        }
        catch (...) {
            std::cout << "잘못된 문자를 넣었습니다." << std::endl;
            std::cout << "성적 입력에 실패하였습니다." << std::endl;
        }
    }
```

```cpp
    if (examName == "중간기말") {
        std::istringstream ss(scores);
        std::string token;
        std::vector<float> score;

        while (std::getline(ss, token, ',')) {
            try {
                score.push_back(isUsedOtherText(token));
            }
            catch (...) {
                std::cout << "잘못된 문자를 넣었습니다." << std::endl;
                std::cout << "성적 입력에 실패하였습니다." << std::endl;
            }
        }

        if (score.size() == 2) {
            if (grade->setMidtermScore(score[0])) {
                std::cout << grade->studentID << "의 학생의 중간 성적은 '" << grade->midtermScore << "' 입니다." << std::endl;
                std::cout << "성적 입력이 완료됐습니다." << std::endl;
            }
            if (grade->setLasttermScore(score[1])) {
                std::cout << grade->studentID << "의 학생의 기말 성적은 '" << grade->lasttermScore << "' 입니다." << std::endl;
                std::cout << "성적 입력이 완료됐습니다." << std::endl;
            }
        }
        else if (score.size() > 2) {
            std::cout << "점수 입력이 많습니다." << std::endl;
            std::cout << "성적 입력에 실패하였습니다." << std::endl;
        }
    }
}
```

```cpp
/*
float 정규 표현식과 일치하는지 검사
[-+]? : 부호가 있거나 없는 숫자
[0-9]* : 0개 이상의 숫자
₩₩.? : 소수점이 있거나 없는 숫자
[0-9]+ : 1개 이상의 숫자
([eE][-+]?[0-9]+)? : 선택적으로 'e' 또는 'E'를 포함하고, 그 뒤에 부호가 있거나 없는 숫자가 오는 지수 표현
이 중에서 하나라도 맞는게 있다면 true;
없다면 false를 반환
*/
bool Grade::isValidFloat(const std::string& s) {
    std::regex e("^[-+]?[0-9]*₩₩.?[0-9]+([eE][-+]?[0-9]+)?$");
    if (std::regex_match(s, e)) {
        return true;
    }
    else {
        return false;
    }
}
```

```cpp
// 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;
    }
}
```

```cpp
// 중간고사 점수를 저장할 함수
bool Grade::setMidtermScore(float score) {
    try {
        while (!checkScore(score)) {
            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;
    }
}

// 기말고사 점수를 저장할 함수
bool Grade::setLasttermScore(float score) {
    try {
        while (!checkScore(score)) {
            std::cout << "점수를 다시 입력해주세요 : ";
            std::cin >> score;
        }
        lasttermScore = score;
        return true;
    }
    catch (std::string input) {
        std::cout << "답은 y or n으로만 답하셔야합니다." << std::endl;
        return false;
    }
}
```

```cpp
// "중간기말" 점수를 입력할 때 다른 문자열이 들어갔는지 확인
float Grade::isUsedOtherText(std::string score) {
    // 정규 표현식과 일치하는지 검사 틀리다면 예외 발생.
    if (!isValidFloat(score)) {
        throw new std::exception;
    }

    // string -> float 변환 중 발생할 수 있는 예외 처리 try-catch
    try {
        float number = std::stof(score);
        return number;
    }
    catch (const std::invalid_argument& e1) {
        throw e1;
    }
    catch (const std::out_of_range& e2) {
        throw e2;
    }
}
```

(4) 학생 성적 조회

- 입출력

```
3
학번을 입력하세요 : 191111
점수가 궁금하면 '점수'
등수가 궁금하면 '등수'
학점이 궁금하면 '학점'
입력해주세요 : 점수
191111학생의 점수 평균은 = 85입니다.
```

```
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.
```

- 학생이 알고싶은 정보의 형태로 출력해주기 위해서 먼저 원하는 출력 형태를 입력받고 해당 형태로 출력하도록 구현.

- 함수 분리, 2중 for문, 많은 조건 분기, 참조자'&', 사용자 정의 데이터 타입(GradeMachine, Grade), switch-case문

```
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);
}
```

```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]);
            }
        }
    }
}
```

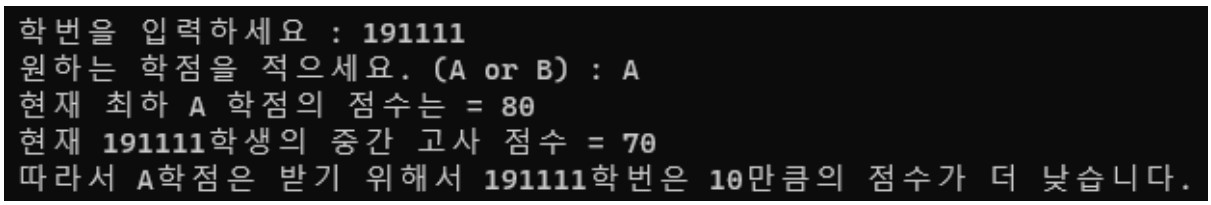
```cpp
// 성적 조회를 위한 함수
void GradeMachine::searchStudentGrade(std::vector<Grade>& studentGradeCard, int studentID, std::string format) {
    // 해당 학생이 존재하는지 안하는지 확인할 bool 변수
    bool exist = false;
    // 원하는 성적 출력이 학점일 경우
    if (format == "학점") {
        for (auto it : studentGradeCard) {
         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

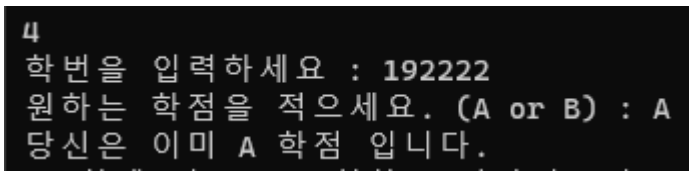
```cpp
    // 원하는 성적 출력이 점수일 경우
    if (format == "점수") {
        for (auto it : studentGradeCard) {
            if (it.getStudentID() == studentID) {
                exist = true;
                if (it.getLasttermScore() >= 0 && it.getMidtermScore() >= 0) {
                    std::cout << studentID << "학생의 점수 평균은 = " << it.getAverage() << "입니다." << std::endl;
                }
                else if (it.getLasttermScore() < 0 && it.getMidtermScore() >= 0){
                    std::cout << studentID << "학생의 중간고사 점수는 = " << it.getMidtermScore() << "입니다." << std::endl;
                }
                else {
                    std::cout << studentID << "학생의 성적은 아직 입력되어 있지 않습니다." << std::endl;
                }
            }
        }
    }
    // 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;
    }
}
```

(5) 학생 목표 성적

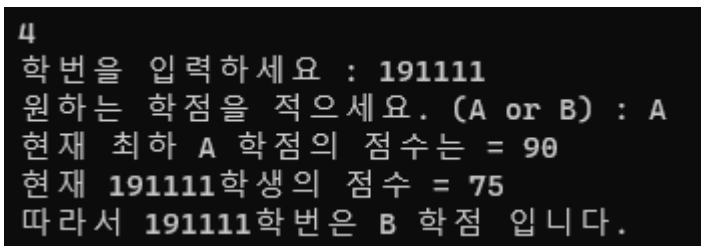
- 입출력

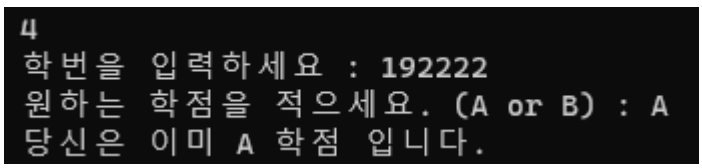
ⅰ) 중간 고사만 입력이 되어 있을 경우





ⅱ) 중간, 기말 모두 입력이 되어 있을 경우





- 학생이 받고 싶어하는 학점을 입력을 하고, 그 해당 학점에 이미 들어가 있을 경우

"당신은 이미 '원하는 학점' 학점 입니다,"를 출력하고,

해당 학점에 못들어가 있는데, 중간 고사 점수만 입력이 되어 있을 경우

그 해당 학점의 최하 점수를 알려주고, 그 점수와의 차이를 계산해서 알려준다

하지만, 기말고사까지 모두 입력이 된 상황이라면,

평균을 출력하여, 학점만 알려주도록 구현.

- 조건 분기, 참조자'&', 사용자 정의 데이터 타입(Grade, GradeMachine), 함수 분리, switch-case문

for 문, vector

```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);
}
```

```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++;
        // 현재 성적이 다 입력되어 있을 경우
        if (it.checkEveryExamScore()) {
            if (it.getStudentID() == studentID) {
                exist = true;
                // 목표 학점이 'A'일 경우
                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;
                    }
                }
                // 'B'일 경우
                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
        // 성적 입력이 다 안되어 있을 경우
        else {
            if (it.getStudentID() == studentID) {
                exist = true;
                // 목표 점수가 'A'일 경우
                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 << "학점을 받기 위해서 " << studentID << "학번은 " << limitAScore - it.getMidtermScore() << "만큼의 점수가 더 낮습니다." << std::endl;
                    }
                    else {
                        std::cout << "당신은 이미 A 학점 입니다." << std::endl;
                    }
                }
                // 'B'일 경우
                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;
    }
}
```

'A학점 계산'

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

```
// 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);
    }
}
```

4. 테스트 결과

(1) 올바른 학생 정보 값이 아닐 경우

```
CPP 성적 관리 시스템입니다.
학생 수를 입력해주세요 : 2
학생의 이름을 입력해주세요 : 임중훈
학생의 학번을 입력해주세요 : 1911111
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-1111-1111
다시 입력해주세요.
학생의 이름을 입력해주세요 : 임중훈
학생의 학번을 입력해주세요 : 191111
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-11111-1111
다시 입력해주세요.
학생의 이름을 입력해주세요 : 임중훈
학생의 학번을 입력해주세요 : 191111
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-1111-1111
학생의 이름을 입력해주세요 : 김지민
학생의 학번을 입력해주세요 : 192222
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-2222-2222
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.
1
학생의 이름 : 임중훈
학생의 학번 : 191111
학생의 전화번호 : 010-1111-1111
학생의 이름 : 김지민
학생의 학번 : 192222
학생의 전화번호 : 010-2222-2222
```

- 학번은 6자리, 번호는 010-4자리-4자리 확인 후 아닐 경우엔 학생 정보를 등록하지 않고 다시 입력받을 수 있도록

(2) 올바른 점수 값이 아닐 경우 예외 처리

```
2
성적 입력을 하고 싶은 학생의 수를 입력하세요 : 1
중간고사면 '중간' 기말고사면 '기말'을 입력해주세요(둘 다 입력시 '중간기말'을 입력해주세요) : 중간기말
학생의 학번을 입력하세요 : 191111
학생의 점수를 입력하세요('중간기말'이라면 ','를 이용해서 구분해주세요) : 80/90
잘못된 문자를 넣었습니다.
성적 입력에 실패하였습니다.
```

- 성적 구분은 ","로 하기로 하였으므로, ","이외의 분리선이 들어가면 예외 처리

(3) 학번이 존재하지 않는 학번일 경우

```
2
성적 입력을 하고 싶은 학생의 수를 입력하세요 : 1
중간고사면 '중간' 기말고사면 '기말'을 입력해주세요(둘 다 입력시 '중간기말'을 입력해주세요) : 중간
학생의 학번을 입력하세요 : 193333
학생의 점수를 입력하세요('중간기말'이라면 ','를 이용해서 구분해주세요) : 70
해당 193333학번은 존재하지 않습니다.
```

- 학생의 성적 데이터를 저장한 studentGrade에서 해당 학번이 존재하지 않을 경우,

"해당 '입력값' 학번은 존재하지 않습니다" 출력

-> 올바른 값 입력 테스트는 위에 입출력 상황에서 보인 것 같아서 따로 작성하진 않겠습니다.

5. 계획 대비 변경 사항

2차 진척 보고서 다음으로 계획에 변경사항은 따로 없습니다.

6. 느낀점

이번 처음 C++ 코드로 프로젝트를 하면서 기존에 하던 Java와 많은 형태가 달라서 데이터 저장하는 것에서 처음에 많은 애를 먹었습니다. 자바에서는 기본형이 아닌 참조형으로 변수를 만들면 해당 컨테이너 안에는 데이터가 아닌 주소 값을 저장하면서, 포인터와 참조자를 따로 표현하지 않아도 주소 값이 넘어가면서 데이터가 잘 정리가 되었다. 하지만, C++에서는 포인터와 참조자를 이용해서 주소 값을 넘겨주면서 해당 컨테이너에 데이터를 저장해야 했던 상황들에서 C++과 Java가 많이 다른 점을 알았습니다.

또한, 코드를 구현하다 보면서 시험 종류 별로 점수를 입력 받으므로 시험이란 클래스 파일을 따로 만들어서 다형성을 이용해서 코드를 구현하고자 생각을 하였으나, 이 생각을 했을 당시에 프로젝트 기간이 많이 남지 않아서 구현을 하지 못한 점이 아쉽습니다. 클린 코드를 위해서 코드 중복을 없애기 위해서 메소드 분리를 하려고 하였지만, 세부 구현이 조금씩 틀려서 따로 함수를 분리해서 작성을 하지 못한 것도 아쉬웠습니다. 이 점이 하드 코딩을 하지 않으려고 했으나 String으로 상수화도 하지 못한 이유이기도 했습니다.

이번 프로젝트를 하면서 해야할 프로젝트가 많아서 '객체 지향'을 많이 사용 못한 것이 아쉽지만, 최대한 노력을 했습니다. C++이 Java와 많이 다르지 않다고 생각했지만, 세부적으로 들어가면 많이 다르다는 것을 느꼈습니다. 프로젝트의 완성도를 높히기 위해서 많은 Validation(유효성) 검사를 많이 추가하여 예외 처리를 최대한 해주려고 노력했고, 입력값에 대해서 check하는 함수를 넣어서 올바르지 않는 값을 넣었을 때, 다시 입력을 받도록 작성하였습니다.