C++ 프로그래밍 및 실습

성적 관리 시스템

진척 보고서 #1

제출 일자	2023.11.02
제출자	임중훈
제출자 학번	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 가능성이 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 학생들의 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 프로그램을 할 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 더 공부를 해야하는지도 알 수 있게 해준다.

3) 차별점

성적이 완전히 기입이 안된 상황에서(Ex: 중간고사) 자신의 성적이 어느 정도 위치인지 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 정보를 기입하고 과목 이름, 해당 과목의 학점, 교수 등의 정보들을 입력받아 관리
- 올바르지 않는 학생의 정보를 기입했을 경우에는 예외처리.

(1) 학생 관리

- 학생의 이름, 학번, 전화번호 등의 정보를 입력받아 저장을 하고 이 관리한 정보는 학생들의 전교 석차를 출력하거나 학생의 성적 조회를 할 때 사용합니다.

2) 성적 입력

- 학생 학번을 입력하고 성적을 입력하면 그 데이터를 저장하고 관리
- 성적 입력란에 실수가 아니라면 예외 처리.

3) 성적 통계

- 입력 받은 성적을 이용해 각 과목별 평균을 구하고, 학생의 학점 또한 계산
- 각 학점별 경계선을 알려주도록 계산

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.

5) 목표 성적 (자신이 원하는 학점을 달성하기 위해 필요한 점수)

- 학번을 입력하고 자신이 받고 싶은 성적을 입력 받은 후 얼만큼의 성적이 더 필요한지 알려주는 기능.

3. 진척사항

1) 학생의 수와 학생들의 정보를 입력

- 입출력

```
CPP 성적 관리 시스템입니다.
학생 수를 입력해주세요 : 4
학생의 이름을 입력해주세요 : 임중훈
학생의 학번을 입력해주세요 : 191111
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-4043-2531
학생의 이름을 입력해주세요 : 김지민
학생의 학번을 입력해주세요 : 192222
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-1111-1111
학생의 이름을 입력해주세요 : 김리민
학생의 학번을 입력해주세요 : 193333
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-4444-4444
학생의 이름을 입력해주세요 : 어렵다
학생의 학번을 입력해주세요 : 194444
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-5555-5555
1. 학생 정보를 조회하고 싶다면 1번
2. 학생 성적를 입력하고 싶다면 2번
3. 성적 조회를 하고 싶다면 3번
종료하고 싶다면 0번
눌러주세요.
```

- 먼저 학생의 수를 입력 받고, 그 학생들의 이름과 학번, 전화번호를 입력받아서 vector에 저장해서 이 정보들은 성적을 입력할 때 존재하는 학번에 성적을 기입했는지 판단할 때 사용 합니다.

- Student라는 클래스를 만들고 이 Student는 studentName, studentID, phoneNumber라는 인스턴스 변수를 갖고 있고 입력받은 값들로 student 객체를 생성해서 vector<Student> 배열에 저장합니다.

- student.h

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

class Student {
private :
    std::string studentName;
    std::string phoneNumber;
    int studentID;
    bool checkStudentInformationValidation(int studentID, std::string phoneNumber);
    bool checkStudentID(int studentID);
    bool checkStudentPhoneNumber(std::string phoneNumber);

public :
    Student();
    Student(std::string studentName, int studentID, std::string phoneNumber);
    std::string getStudentName();
    std::string getPhoneNumber();
    int getStudentID();
};
```

- student.cpp

```cpp
#include "Student.h"

// 학생의 학번이 올바른 학번인지 확인할 bool 함수
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;
    }
}

// private 변수 값을 데이터 값을 알려주기 위한 get함수 들
std::string Student::getStudentName() {
    return studentName;
}

std::string Student::getPhoneNumber() {
    return phoneNumber;
}

int Student::getStudentID() {
    return studentID;
}
```

- Application.cpp

```
int main() {
    cout << "CPP 성적 관리 시스템입니다." << endl;
    // 학생의 수를 입력받는 함수
    inputStudentCount();

    // 학생의 수 만큼 학생 객체를 만들어 students라는 vector에 push하기 위한 for문
    for (int i = 0; i < studentCount; i++) {
        students.push_back(inputStudentInformation());
    }

    // 학생들의 정보를 바탕으로 각 학생의 성적표를 Grade 객체로 생성해서 studentGrade vector에 push하기 위한 for문
    for (int i = 0; i < studentCount; i++) {
        studentGrade.push_back(makeStudentGradeCard(students[i].getStudentID()));
    }
```

2) 학생의 정보 조회

```
1. 학생 정보를 조회하고 싶다면 1번
2. 학생 성적를 입력하고 싶다면 2번
3. 성적 조회를 하고 싶다면 3번
종료하고 싶다면 0번
눌러주세요.
1
학생의 이름 : 임중훈
학생의 학번 : 191111
학생의 전화번호 : 010-4043-2531
학생의 이름 : 김지민
학생의 학번 : 192222
학생의 전화번호 : 010-1111-1111
학생의 이름 : 김리민
학생의 학번 : 193333
학생의 전화번호 : 010-4444-4444
학생의 이름 : 어렵다
학생의 학번 : 194444
학생의 전화번호 : 010-5555-5555
```

- 단순 반복문을 통해 Student의 객체의 get함수들을 호출해서 데이터 값 출력

```
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) 학생의 성적표 만들기(학생 별 Grade 객체 생성)

```
for (int i = 0; i < studentCount; i++) {
    studentGrade.push_back(makeStudentGradeCard(students[i].getStudentID()));
}

Grade makeStudentGradeCard(int studentID) {
    Grade grade(studentID);
    return grade;
}
```

4) 학생의 성적 입력

- 범위 반복문을 사용해서 해당 학번이 존재한다면 그 해당 학번의 성적을 입력하고 없다면 조건문을 통해(tmp == 0) 해당 학번은 존재하지 않습니다. 출력

```
1. 학생 정보를 조회하고 싶다면 1번
2. 학생 성적를 입력하고 싶다면 2번
3. 성적 조회를 하고 싶다면 3번
종료하고 싶다면 0번
눌러주세요.
2
학생의 학번을 입력하세요 : 191111
중간고사면 '중간' 기말고사면 '기말'을 입력해주세요 : 중간
학생의 점수를 입력하세요 : 90
191111의 학생의 중간성적은 '90' 입니다.
성적 입력이 완료됐습니다.
1. 학생 정보를 조회하고 싶다면 1번
2. 학생 성적를 입력하고 싶다면 2번
3. 성적 조회를 하고 싶다면 3번
종료하고 싶다면 0번
눌러주세요.
2
학생의 학번을 입력하세요 : 192323
중간고사면 '중간' 기말고사면 '기말'을 입력해주세요 : 중간
학생의 점수를 입력하세요 : 100
해당 192323학번은 존재하지 않습니다.
1. 학생 정보를 조회하고 싶다면 1번
2. 학생 성적를 입력하고 싶다면 2번
3. 성적 조회를 하고 싶다면 3번
종료하고 싶다면 0번
눌러주세요.
```

- Application.cpp

```
void inputStudentScore() {
    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;
    }
}
```

- Grade.h

```
#include "Grade.h"

// 성적표의 각 정보를 set함수를 통해 설정.
void Grade::setStudentID(int studentID) {
    this->studentID = studentID;
}

void Grade::setMidtermScore(float score) {
    midtermScore = score;
}

void Grade::setLasttermScore(float score) {
    lasttermScore = score;
}

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

```cpp
#pragma once
#include <iostream>

class Grade {
private:
    int studentID;
    float midtermScore;
    float lasttermScore;
    float average;
    std::string grade;
    void setStudenID(int studentID);
    void setMidtermScore(float score);
    void setLasttermScore(float score);
    void calculateExamAverage(Grade* grade);

public :
    Grade(int studentID);
    void inputStudentScore(Grade* grade, float score, std::string examName);
    void getGradeCard(Grade* grade, std::string examName);
    int getStudentID();
    float getMidtermScore();
    float getLasttermScore();
    float getAverage();
    std::string getGrade();
};
```

- Grade.cpp

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


Grade::Grade(int studentID) {
    setStudenID(studentID);
    midtermScore = 0;
    lasttermScore = 0;
    average = 0;
    grade = "";
}

// 기본적인 성적 입력의 호출에 따른 함수들을 호출할 함수
void Grade::inputStudentScore(Grade* grade,float score, std::string examName) {
    if (examName == "중간") {
        grade->setMidtermScore(score);
        std::cout << grade->studentID << "의 학생의 " << examName << "성적은 '" << grade->midtermScore << "' 입니다." << std::endl;
        std::cout << "성적 입력이 완료됐습니다." << std::endl;
    }
    if (examName == "기말") {
        grade->setLasttermScore(score);
        std::cout << grade->studentID << "의 학생의 " << examName << "성적은 '" << grade->midtermScore << "' 입니다." << std::endl;
        std::cout << "성적 입력이 완료됐습니다." << std::endl;
    }
}
```

```cpp
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;
    }
}

// 각 데이터 값 반환하기 위한 get함수들
int Grade::getStudentID() {
    return studentID;
}

float Grade::getMidtermScore() {
    return midtermScore;
}

float Grade::getLasttermScore() {
    return lasttermScore;
}

float Grade::getAverage() {
    return average;
}

std::string Grade::getGrade() {
    return grade;
}
```

5) 예외 테스트

- 학번이 올바르지 않을 경우

```
CPP 성적 관리 시스템입니다.
학생 수를 입력해주세요 : 3
학생의 이름을 입력해주세요 : 임중훈
학생의 학번을 입력해주세요 : 1911111
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-1111-1111
다시 입력해주세요.
```

- 전화번호가 올바르지 않을 경우

```
학생의 이름을 입력해주세요 : 임중훈
학생의 학번을 입력해주세요 : 191111
학생의 번호를 입력해주세요('-'로 구분해주세요) : 010-11111-1111
다시 입력해주세요.
```

- 학번이 없을 때 성적 입력할 경우

```
학생의 학번을 입력하세요 : 192222
중간고사면 '중간' 기말고사면 '기말'을 입력해주세요 : 중간
학생의 점수를 입력하세요 : 40
해당 192222학번은 존재하지 않습니다.
```

- 학번이 없을 때 성적 조회를 할 경우

```
3
학번을 입력하세요 : 193333
점수가 궁금하면 '점수'
등수가 궁금하면 '등수'
학점이 궁금하면 '학점'
입력해주세요 : 점수
해당193333학번은 존재하지 않습니다.
```

4. 계획 대비 변경 사항

- 이전 : 각 과목의 이름을 받아서 각 과목별 Subject 객체를 만들어서 vector에 저장해 각각의 과목 별로 성적을 통계를 내고자 했었다. 또한, 학생 정보 입력 메뉴를 만들어서 각 학생의 정보를 입력 받도록 하고자 하였다. 학번 외에 자신의 주민등록 번호 입력을 통해 성적 조회를 하도록 하기로 할려고 했다.

- 이후 : CPP 과목에만 성적을 입력받고 성적을 통계 내기로 수정. 학생 정보 입력 메뉴를 삭제하고, 처음 프로그램을 시작하자마자 학생의 수를 입력받고 각 학생의 정보를 입력받은 것으로 Student 객체를 생성하였고, 그것을 토대로 각 학생별 성적표를 Grade 객체를 생성해 초기화 하였다. 학번을 통해서만 수정하려고 한다.

- 사유 : 처음에 하고자 하던 대로 코드를 작성해서 테스트를 하는 도중 너무 많은 입력값들이 필요했고, 시간상 너무나도 큰 규모였다는 생각 및 'unordered_map'을 통한 성적 통계 과정의 고난(map을 vector로 변환하기에는 너무나도 큰 데이터 값들).

5. 프로젝트 일정 (참고: 간트차트)

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