HW1: Programming Project #1 **(Report Start From Page 3)**

Due: April 20th, 23:59

This is the first part of an object-oriented programming project that continues throughout the book. Each part extends the project with new file structures. We begin by introducing two classes of data objects. These objects apply the concepts of the book to produce an information system that maintains and processes information about students and course.

1. Design a class **Student**. Each object represents information about a single student. Members should be included for identifier, name, address, date of first enrollment, and number of credit hours completed. Methods should be included for initialization (constructors), assignment (overloaded "=" operator), and modifying field vales, including a method to increment the number of credit hours.

2. Add methods to class **Student** to read student field values form an input stream and to write the fields of an object to an output stream, nicely formatted. You may also want to be able to prompt a user to enter the field values. Use the C++stream operations to implement these methods. Write a driver program to verify that the class is correctly implemented.

3. Design a class **CourseRegistration**. Each object represents the enrollment of a student in a course. Members should be included for a course identifier, student identifier, number of credit hours, and course grade. Methods should be included as appropriate.

4. Add methods to a class **CourseRegistration** to read course registration field values from an input stream and to write the fields of an object to an output stream, nicely formatted. You may also want to be able to prompt a user to enter the fields values. Use the C++ stream operations to implement these methods. Write a driver program to verify the class is correctly implemented.

5. Create a list of student and course registration information. This information will be used in subsequent exercises to test and evaluate the capabilities of the programming project.

Requirements:

- The program should be compiled and executed in the Linux environments.
- Students compile their programs using g++ and make utility
- Please use separate compile (multiple compile)

What to submit

- Please upload your source codes, Makefile, the report file (soft copy) **to your github repository** with a push command.
- Use an attached file as a report template.

Report Of HW1 File_Structure

학부 :	정보컴퓨터공학부
학번 :	201624548
이름 :	이재윤

■ 목표

Linux (Ubuntu) 시스템에서 C++ 프로그래밍을 할 수 있다.

C++의 클래스 설계와 캡슐화에 익숙하다.

C++의 기본 프로그래밍에 익숙하다.

C++ 객체 지향 프로그래밍 및 오버로딩에 익숙하다.

■ 내용

HW1_Student.h (Student Class 선언 및 함수 선언)

HW1_Student.cpp (Operator Overloading 및 함수 Body 설계)

HW1_CoureRegistration.h (CoureRegistration. 선언 및 함수 선언)

HW1_CoureRegistration.cpp (Operator Overloading 및 함수 Body 설계)

HW1_Driver.cpp (Student, CourseRegistration Class에 대한 테스트 드라이버)

■ 내용 및 설계

1) HW1_Student.h

Student(int Identifier, char * Name, char * Address, time_t First, int Hours)

캡슐화를 위해 private 내에 변수들을 선언하였다.

int identifier; // 학번

char name[max_namelength]; // 이름

char address[max_addresslength]; // 주소

time_t first_enrollment; // 첫 등록

int credit_hours; // 학점

기존 변수에 대한 set, get add 함수 및 <<, >>, = 에 대한 오버로딩을 완료.

2) HW1_Student.cpp

HW1_Student.h 에 선언된 함수에 대한 body를 구현하였다.

cin, cout을 통하여 input output이 가능하게 설계하였다.

첫 등록 변수(first_enrollment)는 생성될 때 받아오거나, 자동으로 등록된다.

3) HW1_CourseRegistration.h

CourseRegistration(int course_identifier,   int class_credit_hours)

첫 생성시에 학생은 추가될 수 없다. (학생은 addGrade(int std_identifier, int addgrade) 함수로 추가 될 수 있다.)

int course_identifier; // 수업에 대한 식별자

int class_credit_hours; // 수업의 학점

STD * course_grades; // 학생 학번, 학생 학번에 따른 점수에 대한 구조체

```
typedef struct Linked_Student_Course_Grade {
        int std_identifier; // 학생 학번
        int grade_sum; // 학생 점수 (소수점 없이 다 더하고 필요할 때 float으로 사용)
        Linked_Student_Course_Grade * nxt_std = NULL; // 링크드 리스트
} STD;
```

4) HW1_CourseRegistration.cpp

HW1_CourseRegistration.h 에 선언된 함수에 대한 body를 구현하였다.

학생은 addstudent (int), addstudent (int, int) 함수로 추가 될 수 있다.

학생의 추가 점수는 addGrade(int , int)로 추가할 수 있다.

이미 존재하는 학생이라면, STD 구조체의 grade_sum에 addgrade만을 더한다.

cin, cout을 통하여 input output이 가능하게 설계하였다.

5) HW1_Driver.cpp

10명의 임의의 학생, 10명의 임의의 수업 (random을 사용 – 랜덤 id에 대한 점검 X)

1명의 학생을 입력받고, 1개의 수업을 입력받는다.

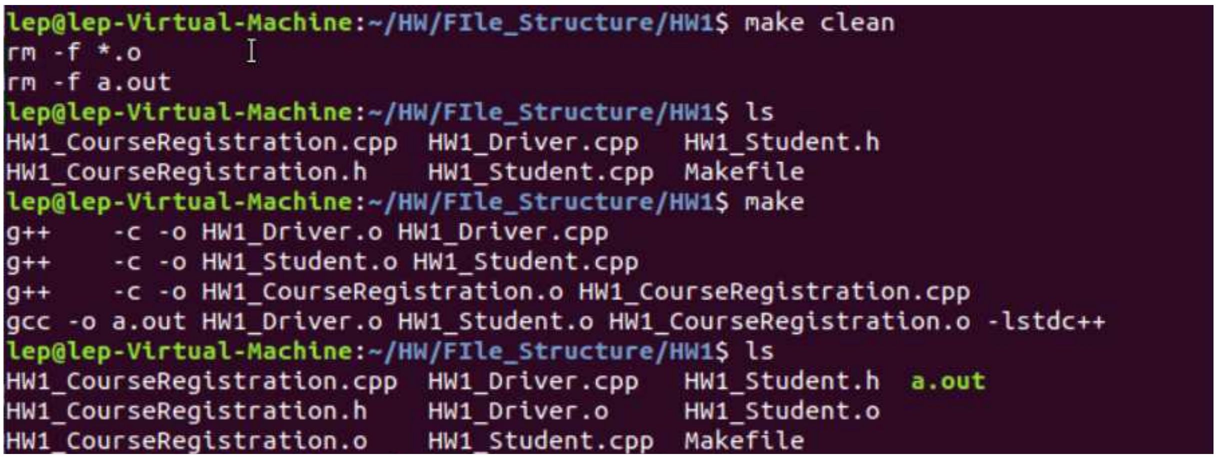
11개의 수업에 임의로 학생들을 넣는다. (1명)

총 11명의 학생과, 11개의 수업을 출력한다.

6) Makefile

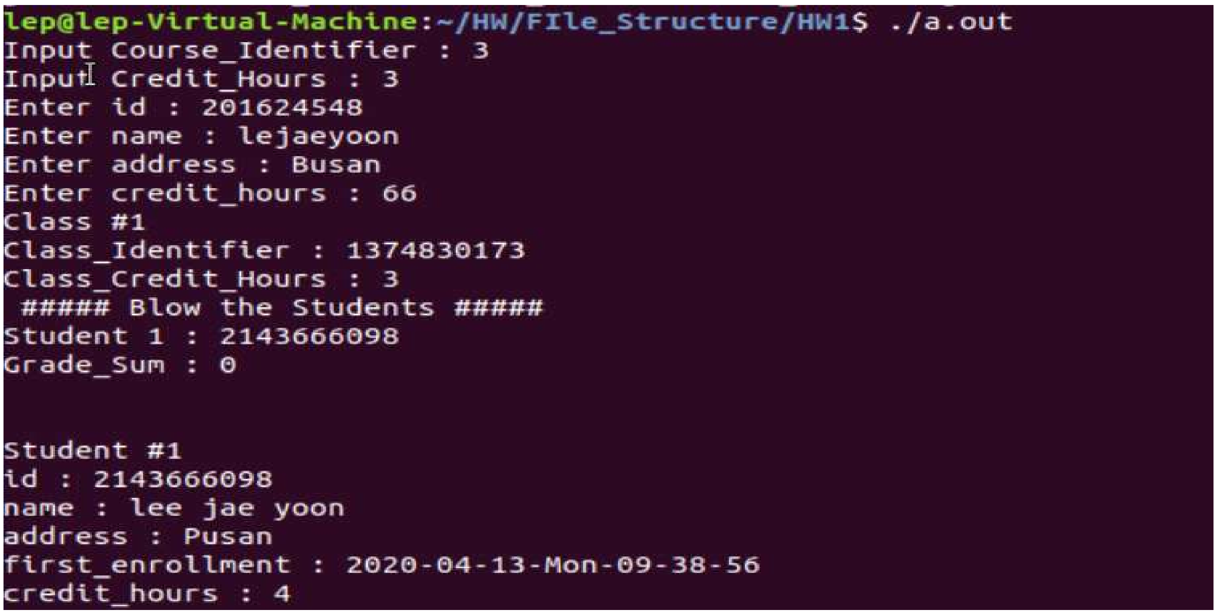
■ 결과

1. Makefile



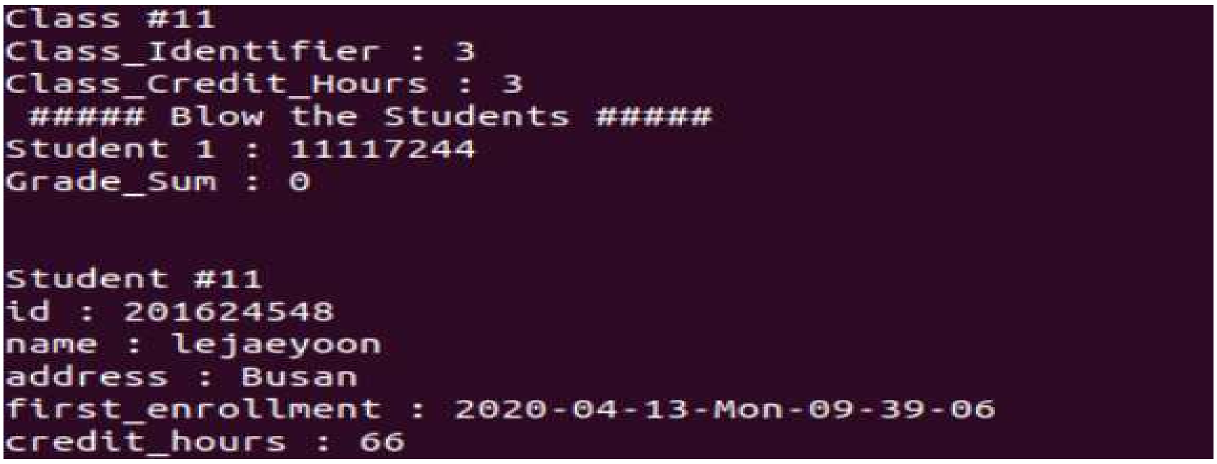
```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW1$ make clean
rm -f *.o
rm -f a.out
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW1$ ls
HW1_CourseRegistration.cpp  HW1_Driver.cpp   HW1_Student.h
HW1_CourseRegistration.h    HW1_Student.cpp  Makefile
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW1$ make
g++     -c -o HW1_Driver.o HW1_Driver.cpp
g++     -c -o HW1_Student.o HW1_Student.cpp
g++     -c -o HW1_CourseRegistration.o HW1_CourseRegistration.cpp
gcc -o a.out HW1_Driver.o HW1_Student.o HW1_CourseRegistration.o -lstdc++
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW1$ ls
HW1_CourseRegistration.cpp  HW1_Driver.cpp   HW1_Student.h  a.out
HW1_CourseRegistration.h    HW1_Driver.o     HW1_Student.o
HW1_CourseRegistration.o    HW1_Student.cpp  Makefile
```

2. Driver



```
lep@lep-Virtual-Machine:~/HW/FIle_Structure/HW1$ ./a.out
Input Course_Identifier : 3
Input Credit_Hours : 3
Enter id : 201624548
Enter name : lejaeyoon
Enter address : Busan
Enter credit_hours : 66
Class #1
Class_Identifier : 1374830173
Class_Credit_Hours : 3
 ##### Blow the Students #####
Student 1 : 2143666098
Grade_Sum : 0


Student #1
id : 2143666098
name : lee jae yoon
address : Pusan
first_enrollment : 2020-04-13-Mon-09-38-56
credit_hours : 4
```



```
Class #11
Class_Identifier : 3
Class_Credit_Hours : 3
 ##### Blow the Students #####
Student 1 : 11117244
Grade_Sum : 0


Student #11
id : 201624548
name : lejaeyoon
address : Busan
first_enrollment : 2020-04-13-Mon-09-39-06
credit_hours : 66
```