

# 2023 Spring OOP Assignment Report

과제 번호 : 2  
학번 : 20220100  
이름 : 박기현  
Povis ID : kihyun

## 명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

I completed this programming task without the improper help of others.

## 1. 프로그램 개요

- 본 프로그램은 학생의 데이터를 입력 받은 후 피벗 변환을 수행하고, 수행 결과를 테이블 및 차트로 시각화하여 출력하는 프로그램이다.
- 피벗 변환이란 주어진 데이터를 특정 열의 카테고리 기준으로 묶고, 다른 열의 수치에 대해 평균, 최댓값, 최솟값 등을 구하는 변환이다.
- 학생 데이터는 총 4가지로, 학과(Dept), 성별(Gender), 이름(Name), 나이(Age)로 구성된다.
- 학과는 string형을 사용한다. 입력 시 공백을 포함하지 않으며, 대문자만 입력 가능하다. 학과 종류는 9개 이하이며, 각 학과에 소속된 학생 수는 10000명 이내이다.
- 성별은 string형을 사용한다. 입력 시 공백을 포함하지 않으며, M과 F만 입력 가능하다.
- 이름은 string형을 사용한다. 입력 시 공백을 포함하지 않으며, 대소문자 모두 입력 가능하다.
- 나이는 int형을 사용한다. 나이의 범위는 18세 이상 30세 미만이다.
- 프로그램을 실행하면 메뉴가 출력되고, 사용자로부터 메뉴 번호를 입력 받는다.
- 메뉴에 따른 기능 수행이 끝나면 메뉴를 다시 출력하고 다시 입력 받는다.
- 1. 학생 추가
  - ◆ 위에서 설명한 학생의 데이터를 사용자로부터 입력 받는다. 이때 입력된 학생의 데이터의 4가지 모두 일치하는 데이터가 이미 존재하면 중복으로 간주하여 추가하지 않고 "The student already exists"를 출력한다. 만약 중복이 아니라면 입력

받은 데이터를 추가한 뒤 "A student is added in table!"을 출력한다.

□ 2. 학생 삭제

- ◆ 사용자로부터 학생의 데이터를 입력 받은 후 주어진 입력이 이미 추가된 리스트 내에 존재하면 데이터를 삭제한다. 이때 삭제가 성공적으로 이뤄지면 "Deleted!"를 출력한다. 만약 존재하지 않거나, 리스트가 비어 있다면 "Can't Delete it"을 출력한다.

□ 3. 학생 리스트 출력

- ◆ 입력 받은 학생의 데이터를 테이블 형태로 출력한다. 첫 번째 행은 Dept, Gender, Name, Age를 출력한다. 두 번째 행부터는 학생의 데이터를 출력하되 문자열은 사전 순서대로, 숫자는 오름차순으로 정렬한다. 각 열은 'wt'로 구분되며, 각 줄의 맨 뒤에는 탭이나 공백이 없다.

□ 4. 피벗 테이블 출력

- ◆ Dept, Gender, Dept and Gender의 세 가지 카테고리를 선택한다. 이후 Average, Max, Min의 세 가지 함수를 선택한다. 함수의 대상은 Age이다.
- ◆ 피벗 테이블 출력 시 첫 번째 행은 입력 받은 카테고리와 함수를 출력하고, 두 번째 행부터는 카테고리별 Age에 함수를 적용하여 테이블 형태로 출력하되 카테고리를 사전 순서대로 정렬한다. 각 열은 'wt'로 구분되며, 각 줄의 맨 뒤에는 탭이나 공백이 없다. 또한, 함수로 적용된 값이 소수일 경우, 소수점 첫째자리까지 반올림하여 나타낸다.
- ◆ 카테고리에 속하는 학생이 없을 경우 해당 출력을 생략한다.

□ 5. 프로그램 종료

- ◆ "Exit!"을 출력하여 프로그램을 종료한다.

## 2. 프로그램의 구조 및 알고리즘

□ Assign2\_20220100.cpp

- ◆ <iostream>, <string>, <cmath> 라이브러리와 "student.hpp", "node.hpp", "list.hpp", "function.hpp" 헤더 파일을 포함한다.
- ◆ string형 selection 변수는 사용자의 메뉴 선택을 저장하는 변수이다.
- ◆ student\_list를 생성한다.

- ◆ 5번 메뉴를 선택하기 전까지 반복한다.
- ◆ 1번 메뉴 선택 시 학생 리스트에 새 노드를 저장한다. (save\_node)
- ◆ 2번 메뉴 선택 시 학생 리스트에서 노드를 삭제한다. (delete\_node)
- ◆ 3번 메뉴 선택 시 학생 리스트를 출력한다. (print\_node)
- ◆ 4번 메뉴 선택 시 카테고리 와 함수를 입력 받는다.
  - string형 category 변수는 사용자의 카테고리 선택을 저장하는 변수이다.
  - string형 function 변수는 사용자의 함수 선택을 저장하는 변수이다.
  - 제대로 입력 받을 때까지 반복하며, 제대로 입력 받으면, 피벗 테이블을 출력한다. (print\_pivot\_table)
- ◆ 5번 메뉴 선택 시 프로그램을 종료한다.

#### □ student.hpp

- ◆ student 클래스
- ◆ <iostream>, <string> 라이브러리를 포함한다.
- ◆ private 멤버로, string형 dept, gender, name, int형 age 변수를 선언한다. 순서대로 학과, 성별, 이름, 나이를 저장하는 변수이다.
- ◆ public 멤버로, 데이터를 입력 받는 함수 (input\_info), 같은 데이터를 가진 학생인지 확인하는 함수 (check\_equal), 학생의 학과를 반환하는 함수 (return\_dept), 학생의 성별을 변환하는 함수 (return\_gender), 학생의 나이를 반환하는 함수 (return\_age), 출력을 위해 학생의 데이터를 반환하는 함수 (print\_data), 두 학생의 데이터를 비교하는 함수 (compare\_data)를 선언한다.

#### □ student.cpp

- ◆ <iostream>, <string> 라이브러리와 "student.hpp", "function.hpp" 헤더 파일을 포함한다.
- ◆ int student::input\_info(student data)
  - int형 error\_value 변수는 에러 발생 시 이후 행동을 저장하는 변수이다.
  - 각 데이터를 입력 받을 때 모두 오류가 없을 때까지 반복한다.
  - 각 상황에서 오류 발생시, 오류 메시지를 출력하고, print\_error 함수를 통해 다시 입력 받을지, 메뉴로 돌아갈지 결정한다. error\_value가 1이면, 다시 입

력 받고, 0이면 -1을 반환한다.

- 공백 존재 여부 확인을 위해 getline으로 dept를 입력 받는다.
- 반복 전 error\_value를 -1로 초기화한다.
- 아무것도 입력하지 않으면, 오류 메시지를 출력하고 다시 입력 받을 때까지 반복한다.
- 학과 문자열 처음부터 끝까지 반복하여 오류를 확인한다. 공백이 있는 경우와 학과 문자열이 대문자가 아닌 경우, 오류 발생 시 이후 행동을 수행한다.
- 반복이 종료되면(오류가 없으면) 학생 데이터 중 학과 정보를 저장한다.
- 공백 존재 여부 확인을 위해 getline으로 gender를 입력 받는다.
- 반복 전 error\_value를 -1로 초기화한다.
- M과 F가 아닌 경우, 오류 발생 시 이후 행동을 수행한다. 오직 M과 F인 경우에만 오류가 아니므로, 옳은 입력만을 확인하여, 나머지 입력은 같은 오류 처리를 통해 예외처리한다.
- 반복이 종료되면(오류가 없으면) 학생 데이터 중 학과 정보를 저장한다.
- 공백 존재 여부 확인을 위해 getline으로 name을 입력 받는다.
- 반복 전 error\_value를 -1로 초기화한다.
- 아무것도 입력하지 않으면, 오류 메시지를 출력하고 다시 입력 받을 때까지 반복한다.
- 학과 문자열 처음부터 끝까지 반복하여 오류를 확인한다. 공백이 있는 경우, 오류 발생 시 이후 행동을 수행한다.
- 반복이 종료되면(오류가 없으면) 학생 데이터 주 이름 정보를 저장한다.
- 반복 전 error\_value를 -1로 초기화한다.
- 입력에서 에러 발생 시(cin.fail이 참을 반환할 시), cin.clear로 cin.fail을 초기화하고, cin.ignore로 입력 버퍼를 정리한다. 오류 발생 시 이후 행동을 수행한다.
- 18세 이상 30세 이하의 입력이 아닌 경우, 오류 발생 시 이후 행동을 수행한다.
- 반복이 종료되면(오류가 없으면) 학생 데이터 중 이름 정보를 저장한다.

- 성공적으로 학생 데이터가 입력되면 1을 반환한다.
- ◆ `bool student::check_equal(student data1, student data)`
  - 두 학생의 데이터가 모두 같으면 `true`를 반환하고, 다르면 `false`를 반환한다.
- ◆ `string student::return_dept(student data)`
  - 학생의 학과를 반환한다.
- ◆ `int student::return_gender(student data)`
  - 학생의 성별을 반환한다.
- ◆ `int student::return_age(student data)`
  - 학생의 나이를 반환한다.
- ◆ `void student::print_data(student& data, string* dept, string* gender, string* name, int* age)`
  - 출력을 위해 `string`형 `dept`, `gender`, `name`, `age` 변수의 주소를 매개변수로 전달받아 학생의 데이터를 저장한다.
- ◆ `int student::compare_data(student data1, student data2, string metric)`
  - `int`형 `a`, `b` 변수는 두 학생의 학과 문자열 길이 변수이다.
  - `int`형 `length` 변수는 문자열 길이 변수이다.
  - `string`형 `metric` 매개변수로 `"dept"`를 입력 받으면, 두 학생의 학과 문자열 길이를 비교하여, 더 짧은 길이만큼 반복한다. 첫 번째 학생의 문자열이 작으면 1을, 크면 0을 반환하고, 문자열이 같은 경우에는 첫 번째 학생의 문자열의 길이가 짧으면 1을, 길면 0을, 같으면 2를 반환한다.
  - `string`형 `metric` 매개변수로 `"gender"`를 입력 받으면, 첫 번째 학생이 여자, 두 번째 학생이 남자인 경우에 1을, 반대인 경우에 0을, 성별이 같은 경우에 2를 반환한다.
  - `string`형 `metric` 매개변수로 `"name"`을 입력 받으면, 두 학생의 나이 문자열 길이를 비교하여, 더 짧은 길이만큼 반복한다. 첫 번째 학생의 문자열이 작으면 1을, 크면 0을 반환하고, 문자열이 같은 경우에는 첫 번째 학생의 문자열의 길이가 짧으면 1을, 길면 0을, 같으면 2를 반환한다.
  - `string`형 `metric` 매개변수로 `"age"`를 입력 받으면, 학생 1의 나이가 적은 경우에 1을, 많은 경우에 0을, 같은 경우에 2를 반환한다.

#### □ node.hpp

- ◆ node 클래스
- ◆ <iostream>, <string> 라이브러리와 "student.hpp" 헤더 파일을 포함한다.
- ◆ public 멤버로, student 클래스 data와, node 클래스 next 포인터를 선언한다.

#### □ list.hpp

- ◆ list 클래스
- ◆ <iostream>, <string> 라이브러리와 "node.hpp" 헤더 파일을 포함한다.
- ◆ private 멤버로, int형 count, dept\_cnt 변수를 선언한다. 순서대로 노드의 개수와 학과의 개수를 저장하는 변수이다. string형 dept[9] 배열을 선언한다. 학과는 최대 9개까지 가능하므로, 배열의 크기는 9이다. int형 dept\_student[9] 배열을 선언한다. 각 학과 학생 수를 저장하는 배열이며, 마찬가지로 배열의 크기는 9이다. node 클래스 head 포인터와 tail 포인터를 선언한다.
- ◆ public 멤버로, list 클래스 생성자, 소멸자, 빈 리스트 확인 함수 (empty\_list), 노드 삽입 함수 (insert\_node), 노드 삭제 함수 (delete\_node), 노드 출력 함수 (print\_node), 피벗 테이블에 사용될 노드 출력 함수 (print\_pivot\_node), 노드 정렬 함수 (sort)를 선언한다.

#### □ list.cpp

- ◆ list::list()
  - list를 생성하고, 초기화한다.
- ◆ list::~list()
  - list를 소멸시키고, 초기화와 동적 할당 해제한다.
- ◆ bool list::empty\_list() const
  - 빈 리스트이면 true를 반환한다.
- ◆ void list::insert\_node()
  - 새 노드를 동적 할당한다. 노드의 next를 초기화한다.
  - 새 노드에 student 클래스의 input\_info 함수로 학생 데이터를 입력한다. 이때 반환값이 -1이면, 데이터 입력이 올바르게 수행되지 않은 것이므로 새 노드를 동적 할당 해제하고, 함수를 종료한다.

- student 클래스의 check\_equal 함수로 입력 받은 학생 데이터가 이미 존재하는지 확인한다. 이때 반환값이 true이면, 이미 존재하는 것이므로 새 노드를 동적 할당 해제하고, 함수를 종료한다. 리스트의 head부터 tail까지 반복하여 모든 학생과 비교한다.
- string형 new\_node\_dept 변수에 student 클래스의 return\_data 함수로 새로 입력 받은 학생의 학과 문자열을 저장하여 이미 존재하는 학과인지 확인한다. 존재하는 학과인 경우 int형 dept\_num 변수에 학과 배열에 몇 번째 index에 저장되어 있는지 저장하고, 아닌 경우 -1을 저장한다.
- error\_value를 -1로 초기화하고, 입력 받은 학과의 예외 처리 여부를 확인한다.
- 존재하지 않는 학과일 때, 기존에 존재하는 학과가 최대 개수인 9개라면, 오류 발생 시 이후 활동을 수행한다. 다시 입력 받는다면 동적 할당 해제 후 다시 insert\_node를 진행하고, 메뉴로 돌아가는 경우 동적 할당 해제 후 함수를 종료한다. 기존에 존재하는 학과 개수가 9개 미만이라면, 학과 배열에 새 학과를 저장하고, 해당하는 학과 학생 수에 1을 저장한다. 이때 사전 순서대로 정렬하기 위해 학과 문자열을 비교하여 알맞은 배열 위치에 저장한다.
- 이미 존재하는 학과일 때, 학과 내 학생 수가 10000명이 넘으면, 오류 발생 시 이후 활동을 수행한다. 다시 입력 받는다면 동적 할당 해제 후 다시 insert\_node를 진행하고, 메뉴로 돌아가는 경우 동적 할당 해제 후 함수를 종료한다.
- 빈 리스트인 경우, 새 노드를 헤드와 테일로 지정한 후, 학생 저장 메시지를 출력하고 함수를 종료한다.
- int형 move\_count는 새 노드가 알맞은 위치에 삽입되기 위해 이동할 횟수를 저장하는 변수이다.
- list 클래스의 sort 함수로, 알맞은 위치에 들어갈 수 있도록 움직일 횟수를 저장하고 정렬한다.
- move\_count에 맞게 노드를 이동시키고, 리스트에 연결한 후, 학생 저장 메시지를 출력하고 함수를 종료한다.
- ◆ void list::delete\_node()
  - 삭제할 학생의 데이터를 입력 받을 노드를 동적 할당하고, student 클래스의 input\_info 함수로 학생의 데이터를 저장한다. 이때 반환값이 -1이면, 데이터

입력이 올바르게 수행되지 않은 것이므로 새 노드를 동적 할당 해제하고, 함수를 종료한다.

- 빈 리스트이면, 삭제 불가 메시지를 출력하고 함수를 종료한다.
- student 클래스의 check\_equal 함수로 헤드부터 테일까지 비교하여 동일한 데이터를 가진 노드가 있으면, int형 delete\_check 변수에 1을 저장하고, 반복을 종료한다.
- delete\_check 변수가 1이 아니면, 동적 할당 해제 후 함수를 종료한다. 반대로 1이면, student 클래스의 return\_dept 함수로 삭제할 학과 문자열을 저장하고, 학과 학생 수를 변경한다. 이때 학과 학생 수가 1이면, 학과 배열에서 학과를 제거하고, 배열을 다시 정렬한다.
- 함수 수행이 끝나면, 동적 할당 해제 후 함수를 종료한다.

◆ void list::print\_node()

- student 클래스의 print\_data로 학생 데이터를 전달 받아 순서대로 출력한다. 헤드부터 테일까지 반복하고 함수를 종료한다.
- 빈 리스트이면, 빈 리스트 메시지를 출력하고 함수를 종료한다.

◆ void list::print\_pivot\_node(string category, string function)

- int형 sum 변수는 나이 총합을 저장하는 변수, int max\_or\_min은 나이의 최댓값 또는 최솟값을 저장하는 변수, float형 value 변수는 나이 평균을 저장하는 변수, int형 woman\_value 변수와 man\_value 변수는 성별별 나이 평균 또는 최댓값, 최솟값 등을 저장하는 변수, int형 woman\_count 변수와 man\_count 변수는 성별별 명수를 저장하는 변수이다.
- 카테고리에 따라 출력을 다르게 한다. 1번이면 학과, 2번이면 성별, 3번이면 학과와 성별을 기준으로 나이를 정리하여 출력한다.
- 함수에 따라 출력을 다르게 한다. 1번이면 평균, 2번이면 최댓값, 3번이면 최솟값을 출력한다.
- 평균을 구하기 위해서 student 클래스의 return\_age 함수로 나이를 전달받아 sum 변수 또는 woman\_value, man\_value 변수에 더하고, 학생 수만큼 나누어 평균을 구한다. 이때 카테고리에 맞춰 1번이면 학과별로 반복을 진행하고, 2번이면 성별별로 반복을 진행하고, 3번이면 학과별 내에서 성별별로 반복을 진행하여 평균을 따로 구할 수 있다. 평균이 소수이면 소수점 첫째 자리까지 반올림하여 출력한다.



- 최댓값을 구하기 위해서 student 클래스의 return\_age 함수로 나이를 전달 받아 최댓값이면 woman\_value, man\_value 변수에 저장한다. 이 역시 카테고리 고리에 맞춰 반복을 진행한다.
  - 최솟값을 구하기 위해서 student 클래스의 return\_age 함수로 나이를 전달 받아 최솟값이면 woman\_value, man\_value 변수에 저장한다. 이 역시 카테고리 고리에 맞춰 반복을 진행한다.
  - 출력 시 카테고리에 해당하는 값이 없다면 출력을 생략한다.
- ◆ int list::sort(int\* move\_count, node\* one\_node, string metric)
- "dept"를 기준으로 정렬하면, student 클래스의 compare\_data 함수로 두 학생의 학과를 비교하여 다음 기준으로의 정렬 진행 여부를 결정한다. 다음 기준으로 정렬할 필요가 없으면 0을 반환하고, 다음 기준으로 정렬을 진행하고자 하면 -1을 반환한다.
  - "gender"를 기준으로 정렬하면, "dept"를 기준으로 정렬한 이후를 확인하기 위해 이동하고, 마찬가지로 student 클래스의 compare\_data 함수로 두 학생의 성별을 비교하여 다음 기준으로의 정렬 진행 여부를 결정한다.
  - "name"을 기준으로 정렬하면, "dept"와 "gender"를 기준으로 정렬한 이후를 확인하기 위해 이동하고, 마찬가지로 student 클래스의 compare\_data 함수로 두 학생의 이름을 비교하여 다음 기준으로의 정렬 진행 여부를 결정한다.
  - "age"를 기준으로 정렬하면, "dept"와 "gender", "name"을 기준으로 정렬한 이후를 확인하기 위해 이동하고, 마찬가지로 student 클래스의 compare\_data 함수로 두 학생의 나이를 비교하여 다음 기준으로의 정렬 진행 여부를 결정한다.
  - 반복을 통해 노드가 올바른 위치까지 이동할 횟수를 나타내는 move\_count 변수를 조절하여 반환한다.

#### □ function.hpp

- ◆ <iostream>, <string> 라이브러리와 "list.hpp" 헤더 파일을 포함한다.
- ◆ 메뉴 출력 함수 (print\_menu), 카테고리 메뉴 출력 함수 (print\_pivot\_menu\_1), 함수 메뉴 출력 함수 (print\_pivot\_menu\_2), 노드 저장 함수 (save\_node), 노드 삭제 함수 (delete\_node), 노드 출력 함수 (print\_node), 피벗 테이블 출력 함수 (print\_pivot\_table), 에러 출력 함수 (print\_error)를 선언한다.

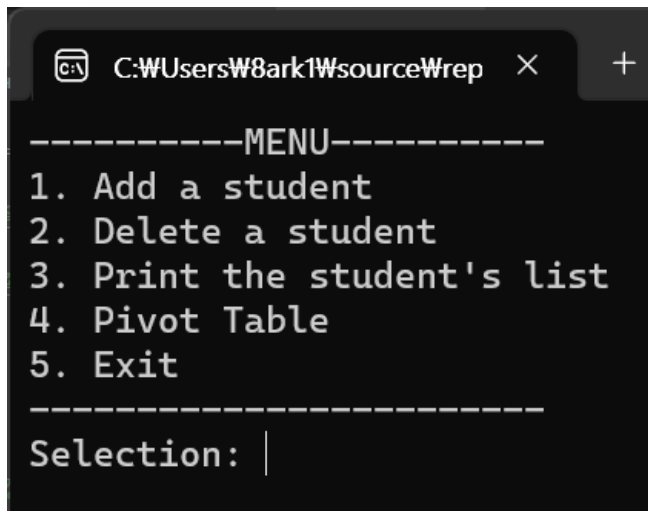
## □ **function.cpp**

- ◆ <iostream>, <string> 라이브러리와 "function.hpp" 헤더 파일을 포함한다.
- ◆ void print\_menu()
  - 메뉴를 출력한다.
- ◆ void print\_pivot\_menu\_1()
  - 카테고리 메뉴를 출력한다.
- ◆ void print\_pivot\_menu\_2()
  - 함수 메뉴를 출력한다.
- ◆ void save\_node(list& list)
  - 새 노드를 저장한다.
  - main 함수에서 사용자의 메뉴를 입력 받을 때 사용한 cin에서의 입력 버퍼를 정리한다.
  - list 클래스의 insert\_node 함수에 접근하여 리스트에 새 노드를 삽입한다.
- ◆ void delete\_node(list& list)
  - 기존 노드를 삭제한다.
  - main 함수에서 사용자의 메뉴를 입력 받을 때 사용한 cin에서의 입력 버퍼를 정리한다.
  - list 클래스의 delete\_node 함수에 접근하여 리스트에서 노드를 삭제한다.
- ◆ void print\_node(list& list)
  - 노드를 출력한다.
  - main 함수에서 사용자의 메뉴를 입력 받을 때 사용한 cin에서의 입력 버퍼를 정리한다.
  - list 클래스의 print\_node 함수에 접근하여 리스트의 노드를 출력한다.
- ◆ void print\_pivot\_table(list& list, string category, string function)
  - main 함수에서 사용자의 메뉴를 입력 받을 때 사용한 cin에서의 입력 버퍼를 정리한다.
  - list 클래스의 print\_pivot\_table 함수에 접근하여 피벗 테이블을 출력한다.

◆ int print\_error()

- string형 option 변수는 오류 발생 시 사용자의 선택을 입력 받는 변수이다.
- 제대로 입력 받을 때까지 오류 메시지를 출력하고, 1을 입력하면 1을 반환, 0을 입력하면 0을 반환한다. 반환하기 전에는 option을 입력 받을 때 사용한 cin에서의 입력 버퍼를 정리한다.

□ 프로그램 실행 예제는 다음과 같다.



```
C:\Users\W8ark1\source\Wrep  ×  +  
-----MENU-----  
1. Add a student  
2. Delete a student  
3. Print the student's list  
4. Pivot Table  
5. Exit  
-----  
Selection: |
```

```
-----MENU-----
1. Add a student
2. Delete a student
3. Print the student's list
4. Pivot Table
5. Exit
-----
Selection: 1
Dept: CSE
Gender: M
Name: PKH
Age: 21
A student is added in table!

-----MENU-----
1. Add a student
2. Delete a student
3. Print the student's list
4. Pivot Table
5. Exit
-----
Selection: |
```

```
-----MENU-----
1. Add a student
2. Delete a student
3. Print the student's list
4. Pivot Table
5. Exit
-----
Selection: 1
Dept: CSE
Gender: M
Name: PKH
Age: 21
The student already exists
```

```
-----MENU-----  
1. Add a student  
2. Delete a student  
3. Print the student's list  
4. Pivot Table  
5. Exit  
-----  
Selection: 2  
Dept: CSE  
Gender: M  
Name: PKH  
Age: 21  
Deleted!
```

```
-----MENU-----  
1. Add a student  
2. Delete a student  
3. Print the student's list  
4. Pivot Table  
5. Exit  
-----  
Selection: 2  
Dept: CSE  
Gender: M  
Name: PKH  
Age: 21  
Can't Delete it
```

```
-----MENU-----
1. Add a student
2. Delete a student
3. Print the student's list
4. Pivot Table
5. Exit
```

```
-----
Selection: 3
```

Dept	Gender	Name	Age
CSE	F	Apkh	19
CSE	F	Bpkh	18
CSE	M	Apkh	20
CSE	M	Cpkh	21

```
-----MENU-----
1. Add a student
2. Delete a student
3. Print the student's list
4. Pivot Table
5. Exit
```

```
-----
Selection: 4
```

```
-----Category-----
```

```
1. Dept
2. Gender
3. Dept and Gender
```

```
-----
Selection: 2
```

```
-----Function-----
```

```
1. Average
2. Max
3. Min
```

```
-----
Selection: 1
```

Gender	Average
F	18.5
M	20.5

```

-----MENU-----
1. Add a student
2. Delete a student
3. Print the student's list
4. Pivot Table
5. Exit
-----
Selection: 4
-----Category-----
1. Dept
2. Gender
3. Dept and Gender
-----
Selection: 3
-----Function-----
1. Average
2. Max
3. Min
-----
Selection: 1
Dept      Gender  Average
CSE       F       18.5
CSE       M       20.5
EE        M       27.5
LIFE      F       26
LIFE      M       23.5

```

```

-----MENU-----
1. Add a student
2. Delete a student
3. Print the student's list
4. Pivot Table
5. Exit
-----
Selection: 5
Exit!

```

### 3. 토론 및 개선

- 클래스의 private 멤버와 public 멤버를 활용하여 객체 개념에 대한 이해도를 확인할 수 있었던 과제였다. 과제 안내 pdf에서 객체 개념에 대해 설명되어있지 않았지만, 스스로 어떻게 하면 객체 개념을 적용할 수 있을지 오히려 더 고민해보는 계기가 되었다. 하지만 그럼에도 객체지향프로그래밍 관점에서 올바르게 코드를 구현했는가를 스스로 질문을 던진다면, 자신있게 답하지 못할 것 같다. 이번 과제에서 구현한 코드는 충분히 비효율적이라는 생각이 들었고, 그 이유는 overloading, 또는 template을 활용한다면 더욱 recursive하게 코드를 간편화할 수 있을 것이라 생각하였기 때문이다. 아직 프로그래밍과 문제해결 과목에서 배운 C언어의 문법에 익숙해져 있어 객체 개념, encapsulation과 data hiding 등을 적절히 활용하지 못하는 것 같아 조금 더 공부와 이해의 필요성을 느꼈다.
- 가장 어려운 점을 느꼈던 것은 다른 클래스에서 또 다른 클래스의 private 멤버에 접근해야 하는 상황에서 어떻게 구현해야 객체 개념을 살리면서 효율적으로 코드를 짤 수 있을까 고민하는 것이었다. 이 부분에 대한 완벽한 해답을 찾지 못해 조금 아쉬운 코드 구성이 되었다고 생각했다. 내가 떠올린 해결책은 필요한 정보를 public 멤버의 함수를 통해 넘겨주는 방법이었는데, 과연 이것이 data hiding의 목적에 부합하는 것인지 아직까지 잘 모르겠다. 앞으로 수업과 과제를 통해 c++ 문법에 조금 더 익숙해지고, 객체 개념에 대해 이해하고자 노력해야겠다.

#### 4. 참고 문헌

- 조성현 교수님의 <객체지향프로그래밍> 수업 자료
- <https://cplusplus.com/reference/istream/istream/ignore/>
- <https://cplusplus.com/reference/ios/ios/fail/>
- <https://cplusplus.com/reference/ios/ios/clear/>