2023 Spring OOP Assignment Report

과제 번호 : 2
학번 : 20220778
이름 : 표승현
Povis ID : hyeony312

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

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

프로그램을 하다 보면 결정해야 할 세부 사항이 많은데, 이러한 세부 사항을 처리한 방법과 이유를 보고서에 쓰십시오.
독창적인 아이디어와 추가 기능은 보너스 점수를 받을 수 있으므로, 보고서에 명확히 기재하십시오.
문제가 여러 개인 경우, 각 문제별로 정리해서 작성합니다.

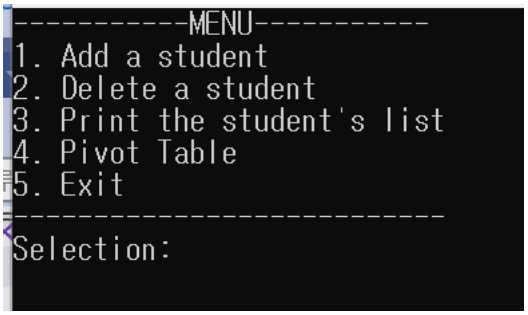
아래 문항별 설명은 편의를 위한 것으로, 삭제하고 제출한다.

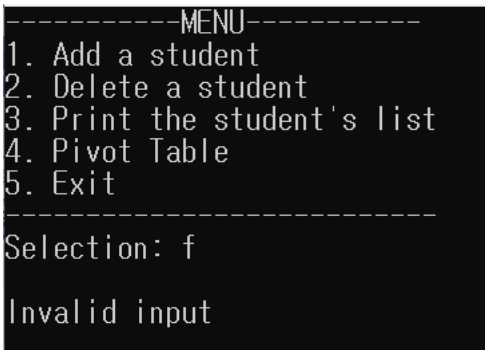
1. 프로그램 개요

학생의 데이터를 피벗 변환 하고 이를 출력하는 프로그램이다. 피벗 변환이란 일련의 데이터를 주어진 카테고리를 기준으로 그룹화하고 각 카테고리의 수치 값을 합, 평균, 최대 또는 최소값을 구하는 작업이다.

이를 구현하기 위해 학생의 데이터를 저장할 student class를 선언한다. 학생들의 데이터를 저장, 삭제 및 분류하기 위해 linked list를 활용한다. 이를 위해 student를 데이터로 저장하고 next 노드의 주소를 저장하는 node class와 list class를 선언한다.

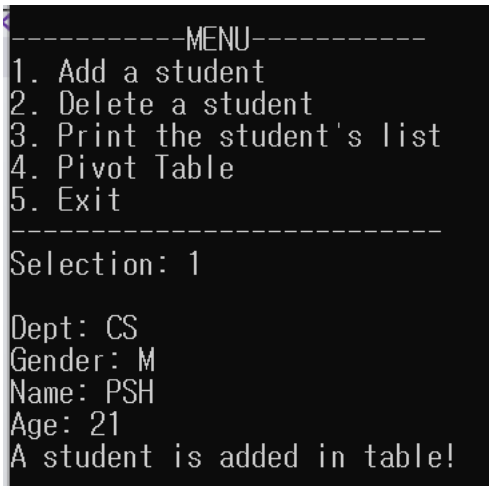
본 프로그램은 처음 실행했을 때 5가지 메뉴를 고를 수 있다. 이때 메뉴는 1부터 5까지의 숫자를 선택함으로서 실행할 수 있는데, 잘못된 타입이나 범위에 해당하지 않는 숫자가 입력되면 "Invalid input"을 출력하고 메뉴 화면을 다시 출력한다.





1) Add a student

1을 입력하면 학생의 정보(학과, 성별, 이름, 나이)를 차례대로 입력 받고 리스트에 추가한다.

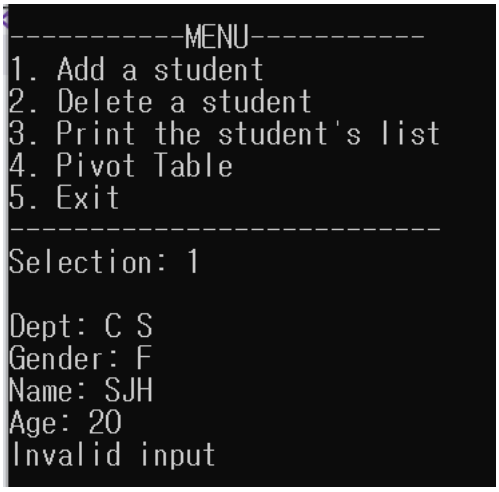


** 예외처리 **

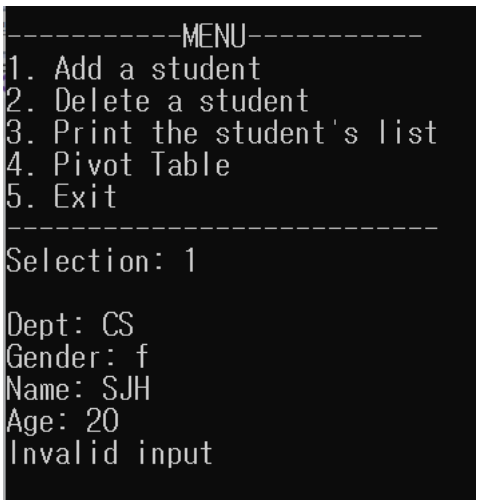
ㄱ. Dept의 종류는 최대 9이며 이를 초과할 시 "Too many Dept in list"를 출력하고 메뉴 화면으로 돌아간다.

ㄴ. 각 Dept에 해당하는 학생의 수가 10000을 넘으면 "Too many students in the Dept"를 출력하고 메뉴 화면으로 돌아간다.

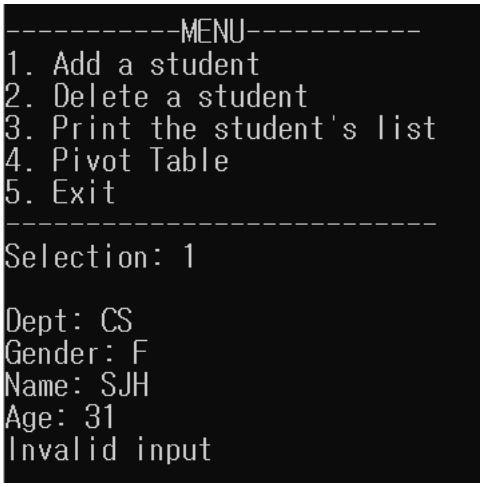
ㄷ. Dept를 입력하는 과정에서 공백은 허용되지 않는다. 또한 알파벳 대문자만을 입력받으며 이외의 입력이 들어오면 "Invalid input"을 출력하고 메뉴 화면으로 돌아간다.



ㄹ. Gender는 M 또는 F의 입력만을 취급한다. 이외의 입력이 들어오면 "Invalid input"을 출력하고 메뉴 화면으로 돌아간다.

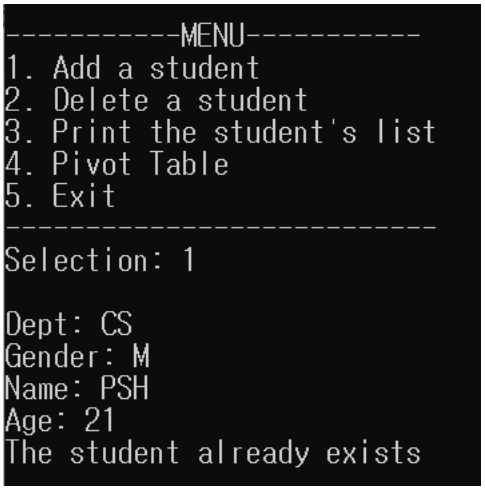


ㅁ. Age의 범위는 18 이상 30 이하이다. 잘못된 타입을 입력받거나 범위를 초과하는 숫자가 입력되면 "Invalid input"을 출력하고 메뉴 화면으로 돌아간다.



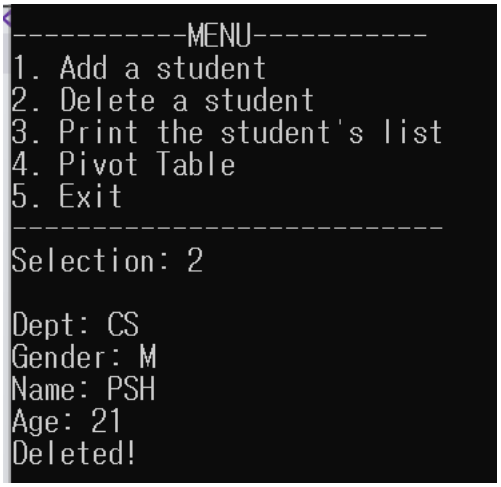
ㅂ. 정보를 입력 받았을 때 이미 리스트에 해당하는 정보를 가진 학생이 존재한다면 중복

으로 간주하고 "The student already exists"를 출력하고 추가하지 않는다.



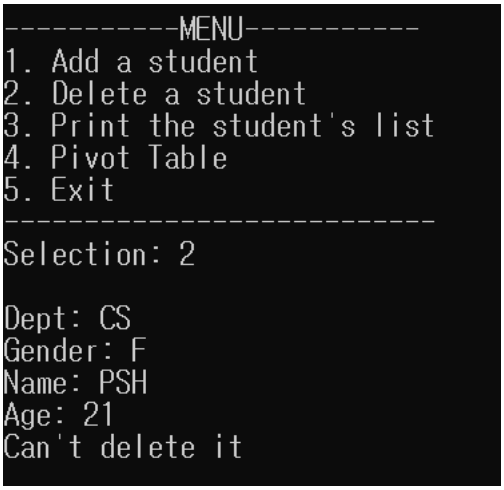
2) Delete a student

2를 입력하면 학생의 정보를 입력 받고 해당하는 학생이 리스트에 존재하면 이를 삭제한다.



** 예외처리 **

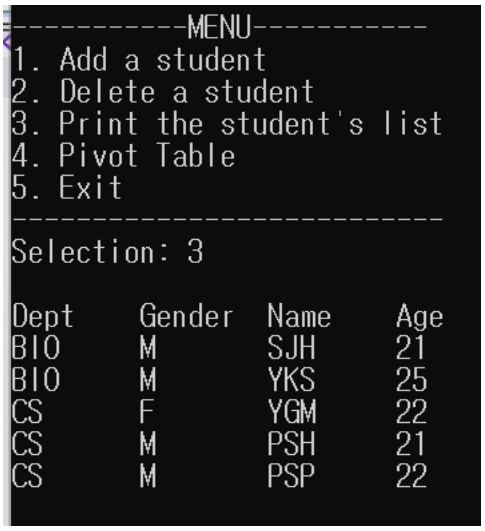
ㄱ. 입력 받은 정보에 해당하는 학생이 없다면 "Can't delete it"을 출력하고 메뉴 화면을 출력한다.



ㄴ. 리스트가 비어있다면 "Can't delete it"을 출력하고 메뉴 화면을 출력한다.

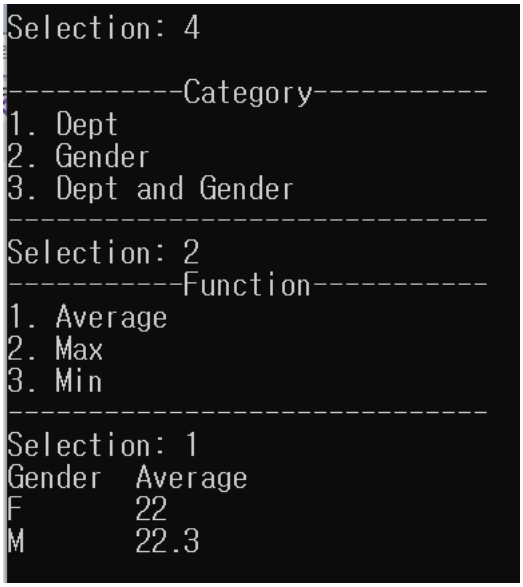
3) Print the student's list

3을 입력하면 리스트의 학생 정보를 확과, 성별, 이름, 나이를 사전순으로 정렬하여 출력한다.



4) Pivot Table

는 피벗 변환을 하는 선택지로 카테고리는 학과, 성별, 학과와 성별 세 가지를 기준으로 진행할 수 있고 수치는 평균, 최대, 최소를 구하여 출력할 수 있다. 평균이 소수로 표현될 경우 소수점 첫째자리까지 반올림하여 나타낸다.



5) Exit

5를 입력하면 프로그램을 종료한다.

2. 프로그램의 구조 및 알고리즘

```
class student {
private:
        string dept, name, gender;
        int age;
public:
        void input_info();
        void print_info();
        int findnode(string dept, string gender, string name, int age); // Find a node that
has same data
        int datacmp(student data); // Compare data. Return 1 when next data is smaller.
        string access_dept() {
                return dept;
        }
        string access_gender() {
                return gender;
        }
        int access_age() {
                return age;
        }
};
```

학생들의 정보를 저장하고 데이터를 관리하는 함수들이 정의되어있다.
string dept, name, gender;
int age; 는 학생의 데이터에 해당한다.
void input_info() 를 통해 학생의 정보를 사용자에게서 입력받는다.
void print_info() 를 통해 학생의 정보를 차례대로 출력한다.
int findnode(string dept, string gender, string name, int age) 데이터를 비교하여 같은 데이터의 수만큼의 정수를 반환한다. 모두 같으면 4를 반환한다.
int datacmp(student data) 데이터를 비교하여 사전순으로 data가 더 앞일 경우 1을, 뒤일 경우에는 -1을, 같을 경우에는 0을 출력한다. 정렬에 이용한다.

이외의 access함수는 private 멤버에 접근하기 위한 함수이다.

```
class node {
public:
	student data;
	node* next;
};
```

List에 저장되는 node 클래스이다. 위에서 정의한 student 클래스를 데이터로 삼고 node* next를 정의해 다음 노드의 주소를 저장한다.

```
class list {
public:
	int count, dept_cnt;
	string dept[9];
	int stu_dept[9] = { 0 };
	node* head;

	void save_node(node* fnode);
	void delete_node(string fdept, string fgender, string fname, int fage);
	void sort();
	int update_data(); // update data of dept_cnt, dept[]
	void pivot(int num1, int num2);
	void print_func(int* Age, int group, int num);
};
```

다음은 리스트를 정의하는 클래스이다.
int count, dept_cnt 리스트에 저장된 노드의 개수와 학과의 종류를 의미한다.
string dept[9] 최대 9 종류의 학과를 저장할 수 있도록 크기 9의 array를 선언한다.
int stu_dept[9] = { 0 } 각 학과의 학생 수를 기록한다. 학과 순서는 dept[]와 일치하며 10000명이 넘어가면 오류 문구를 출력하는데 활용한다.
node* head 리스트의 head를 정의한다.
void save_node(node* fnode) 새로운 노드를 받아 리스트에 저장한다.
void delete_node(string fdept, string fgender, string fname, int fage) 전달 받은 정보에 해당하는 노드를 찾아 리스트에서 삭제한다.
void sort() 주어진 기준에 따라 학생들을 정렬한다.
int update_data() dept의 종류, 학과 학생 수 등을 메뉴를 실행할 때마다 업데이트한다.
void pivot(int num1, int num2) 카테고리와 function를 선택하는 커맨드를 각각 입력받아 그에 따른 피벗 변환을 수행한다.
void print_func(int* Age, int group, int num) pivot 함수에서 활용하는 함수로 function 커맨드에 따른 평균, 최대, 최소를 구하여 출력한다.

```cpp
int main()
{
    string command; // Command for select menu
    list stu_list;
    stu_list.count = stu_list.dept_cnt = 0; // 0으로 초기화

    while (1) {
        cout << "----------MENU----------" << endl
            << "1. Add a student" << endl
            << "2. Delete a student" << endl
            << "3. Print the student's list" << endl
            << "4. Pivot Table" << endl
            << "5. Exit" << endl
            << "--------------------------" << endl;
        cout << "Selection: ";
        cin >> command;
        cout << endl;

        if (command.compare("1")==0) { // Add new student
            node* temp = new node;
            int test = 1;
            temp->data.input_info();

            // Check if input is valid
            int _length = temp->data.access_dept().length();
            for (int i = 0; i < _length; i++) {
                if (temp->data.access_dept()[i] < 'A' || temp->data.access_dept()[i] > 'Z') {
                    cout << "Invalid input" << endl;
                    test = 0;
                    break;
                }
            }
            //Redundancy check
            if (test) {
                for (node* curr = stu_list.head; curr != NULL; curr = curr->next) {
                    if (curr->data.datacmp(temp->data) == 0) {
                        test = 0;
                        cout << "The student already exists" << endl;
                        break;
                    }
                }
            }
            //Check if Dept is full
            if (test&&stu_list.dept_cnt == 9) {
                test = 0;
                for (int i = 0; i < 9; i++) {
                    if (temp->data.access_dept().compare(stu_list.dept[i]) == 0) test = 1;
                }
                if (test == 0) cout << "Too many Dept in list" << endl;
            }
            //Check if number of student of dept over 10000
            if (test) {
                for (int i = 0; i < stu_list.dept_cnt; i++) {
                    if (temp->data.access_dept().compare(stu_list.dept[i]) == 0) {
                        if (stu_list.stu_dept[i] >= 9999) {
                            test = 0;
                            cout << "Too many students in the Dept" << endl;
                            break;
                        }
                    }
                }
            }
            //Check if age is valid
            if (test) {
                if (temp->data.access_age() < 18 || temp->data.access_age() > 30) {
                    cout << "Invalid input" << endl;
                    test = 0;
                }
            }
            //Check if gender is valid
            if (test) {
```

```
            if (temp->data.access_gender()!="M" && temp->data.access_gender()!= "F") {
                cout << "Invalid input" <<endl;
                test = 0;
            }
        }
        // if test is 0, do not add node
        if (test) {
            stu_list.save_node(temp); // add new node
            stu_list.sort();
            cout << "A student is added in table!" << endl;
        }
    }
    else if (command.compare("2") == 0) {
        string dept, gender, name;
        int age;

        cout << "Dept: "; cin >> dept;
        cout << "Gender: "; cin >> gender;
        cout << "Name: "; cin >> name;
        cout << "Age: "; cin >> age;

        stu_list.delete_node(dept, gender, name, age); // delete node
        stu_list.sort();
    }
    else if (command.compare("3") == 0) {
        cout << "DeptWtGenderWtNameWtAge" << endl;
        for (node* curr = stu_list.head; curr != NULL; curr = curr->next) {
            curr->data.print_info();
        }
    }
    else if (command.compare("4") == 0) {
        int num1, num2;
        stu_list.update_data();
        cout << "----------Category----------" << endl
            << "1. Dept" << endl
            << "2. Gender" << endl
            << "3. Dept and Gender" << endl
            << "----------" << endl
            << "Selection: ";
```

```
        }
    }
    else if (command.compare("4") == 0) {
        int num1, num2;
        stu_list.update_data();
        cout << "----------Category----------" << endl
            << "1. Dept" << endl
            << "2. Gender" << endl
            << "3. Dept and Gender" << endl
            << "----------" << endl
            << "Selection: ";
        cin >> num1;
        cout << "----------Function----------" << endl
            << "1. Average" << endl
            << "2. Max" << endl
            << "3. Min" << endl
            << "----------" << endl
            << "Selection: ";
        cin >> num2;

        if (num1 == 1) cout << "DeptWt";
        else if (num1 == 2) cout << "GenderWt";
        else if (num1 == 3) cout << "DeptWtGenderWt";

        if (num2 == 1) cout << "Average" << endl;
        else if (num2 == 2) cout << "Max" << endl;
        else if (num2 == 3) cout << "Min" << endl;

        stu_list.pivot(num1, num2);
    }
    else if (command.compare("5") == 0) {
        cout << "Exit!";
        break;
    }
    else cout << "Invalid input" << endl;
    stu_list.update_data();
    cout << endl;
}
}
```

다음은 main 함수이다. Command로 입력을 받고 1~5사이의 정수일 경우에만 메뉴를 실행하고

이외에는 "Invalid input"을 출력한 후 다시 입력받는다. 앞서 선언한 class와 멤버 함수를 활용하여 각 메뉴를 구현한다. 입력 데이터 타입이나 범위와 같은 예외는 대부분 main에서 처리한다.

3. 토론 및 개선

List의 멤버 함수를 이용하여 student 의 private 멤버 변수에 접근하기 위해 access 함수를 작성하였는데 비효율적인 것 같다. 각 클래스의 변수를 활용하는 함수는 각 클래스에 선언하여 별도의 접근 함수 없이 멤버 변수에 접근할 수 있도록 개선하면 좋을 것 같다. 또한 입력값의 예외 처리를 main에서 해결한 부분이 많았는데 이는 main 함수의 길이가 길어지는 문제를 발생시켰다. Input 함수나 다른 새로운 함수를 통해 입력값을 일괄적으로 받고 예외 사항을 처리할 수 있도록 하면 이를 개선할 수 있을 것이라 생각한다.

4. 참고 문헌