2023 Spring OOP Assignment Report

과제 번호 : Assign2
학번 : 20190445
이름 : 허수범
Povis ID : sbh408

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

본 프로그램은 학생 데이터를 입력받아 이를 목록에 추가하거나 목록에서 제거할 수 있는 기능을 제공하며, 학생 데이터 목록을 정렬하여 출력하거나, 피벗 변환을 수행하여 출력할 수 있도록 해주는 프로그램이다.

본 프로그램을 실행하면, 수행할 수 있는 동작들의 목록이 뜨며, 특정 동작에 해당하는 번호를 입력하면, 동작을 수행한다. "1"을 입력하면, 학생 데이터를 입력할 수 있고, valid한 학생 데이터를 입력하면, 학생 데이터 객체가 연결 리스트에 추가된다. "2"를 입력하면, 마찬가지로 학생 데이터를 입력할 수 있는데, 입력한 데이터에 매칭되는 학생이 있다면 해당 학생을 연결 리스트에서 찾아 삭제한다. "3"을 입력하면, 목록의 모든 학생들의 데이터가 정렬되어 출력된다. "4"를 입력하면, 먼저 피벗 변환을 수행할 카테고리의 목록이 출력되며, 카테고리의 번호를 입력하면, 수행할 수 있는 피벗 변환의 목록이 뜬다. 수행할 함수의 번호를 입력하면, 피벗 테이블이 정렬되어 출력된다.

프로그램 디렉토리에는 각각 다른 객체를 선언해 놓은 세 개의 헤더파일(List.hpp, Student.hpp, Node.hpp), 멤버함수들의 정의와 메인함수가 정의되어 있는 main.cpp, 각 코드파일들에 대한 설명이 되어 있는 README.txt, 그리고 소스 코드를 컴파일하여 executable file을 만드는 등 다양한 기능을 make 명령어로 쉽게 실행할 수 있도록 하는

makefile이 포함되어 있다. makefile의 경우, make all 이외에도, 실행 파일을 실행시키는 make run, executable 파일을 지울 수 있는 make clean, gcc 옵션 -w과 -Wall을 이용하여 에러 혹은 경고 메세지 확인을 위한 test 컴파일을 진행할 수 있도록 하는 make testComp등 다양한 기능을 할 수 있도록 구현하였다.

2. 프로그램의 구조 및 알고리즘

먼저 각 class에 대한 설명을 하겠다.

첫 번째 클래스는 Student.hpp에 선언된 student class이다. 이 클래스는 private 멤버로 각 학생의 이름, 과, 성별을 저장하는 string 변수와 나이를 저장하는 int age를 가진다. Public 멤버 함수와 operator overlooading은 다음과 같다.

1. string getname() const, string getdept() const, string getgender() const, int getage()

네 멤버함수는 모두 학생의 정보를 return하는 함수들이다. const가 붙었기 때문에 학생 정보의 수정은 불가능하다. Data hiding의 측면에서 좋지 못한 구현법일 수 있지만, 학생 정보의 수정이 불가능하도록 하여 프로그램 구현이 안정화되도록 하는 것에 초점을 맞추었다. 이들은 모두 인라인 함수로 Student.hpp에 정의되었다.

2. bool operator==(const student& rhs) const

이 멤버함수는 연산자 "=="를 오버로딩한 것이다. Student 객체 간의 비교를 할 때, 비교 연산자가 student 객체의 모든 멤버변수를 비교하도록 하였다.

3. Int Input_info()

이 멤버함수는 정보를 입력받아 student 객체에 저장하는 함수이다. Dept, name, gender의 경우 getline으로 string을 입력을 받고 해당 입력이 invalid한지 확인하도록 하였다. 빈 칸이 포함되어있거나, gender가 "M" 혹은 "F"가 아니거나, dept에 대문자 알파벳이 아닌 것이 입력되거나, name에 대문자 혹은 소문자 알파벳이 아닌 무언가가 입력되었을 때 각 상황에 맞는 에러 메세지를 출력하도록 하였다. Age의 경우에도 먼저 getline을 통해 string으로 입력받고, 해당 입력이 범위 내의 정수인지 확인한 후, int로 변환하여 저장하도록 하였다. 이 때 age를 string으로 받기 위해 새로운 string 변수 ageIn을 선안하여 사용하였고, 입력이 범위 내의 정수인지 indicate하는 checker인 int errorchk도 선언하고 0으로 초기화하여 사용하였다.

두 번째 클래스는 Node.hpp에 선언된 node class이다. 이 클래스는 private 멤버로 각 노드의 student 타입 data 변수, 이전 노드를 가리키는 포인터 prev, 다음 노드를 가리키는 포인터 next를 가진다. Public constructor로 input argument가 없는 node()를 가지는데, 이

는 처음 노드가 생성될 때 prev와 next를 모두 nullptr로 초기화하도록 한다. Public 멤버 함수들은 다음과 같다. 아래 멤버함수들은 모두 inline function으로 Node.hpp에 정의되었다.

1. Student getdata() const, node* getnext() const, node* getprev() const: 이들은 각각 데이터와 양쪽 노드를 가리키는 포인터를 return한다. 구현의 안정성을 위해 const로 지정해주었다.

2. void setstudent(const student* Student), void setnext(node* Next), void setprev(node* Prev): 이들은 각각 node 객체의 데이터와 양쪽 노드를 가리키는 포인터를 input argument로 초기화해주는 함수들이다.

세 번째 클래스는 List.hpp에 선언된 list class이다. List 클래스는 private member variable로 각각 학생 수와 department의 수가 저장되어 있는 int count와 int dept_cnt, department의 종류가 저장된 string의 배열인 dept[9], head node를 가리키는 포인터를 가진다.

List의 생성자는 count와 dept_cnt를 0으로 초기화하고, head를 nullptr로 초기화하도록 하였다.

Private member function들은 다음과 같다.

1. void getDeptTable(int func) const, void getGenTable(int func) const, void getDeptGenTable(int func) const:

이들은 피벗 테이블 생성시 사용되는 함수들로, 무슨 func의 테이블을 출력할 것인지 argument로 받고, 피벗 테이블을 출력한다. 먼저 각 분류의 age를 보두 더해 넣을 int adder, 각각 최댓값, 최솟값을 계속 update해가며 저장할 int Max와 int Min, 각 분류의 노드 갯수를 저장한 counter를 선언하고 초기화해준다. getGenTable과 getDeptGenTable의 경우, mAdder와 fAdder, mMax와 mMin 등을 성별로 구분하여 선언하고 초기화한다. 이후 getDeptTable과 getDeptGenTable의 경우, dept의 갯수만큼 for loop을 실행, getGenTable의 경우 두 번 실행하여 내부 2차 loop에서는 temp 노드의 위치를 head 에서 prev로 끝까지 이동해 가며 매번 분류에 matching되는 temp node의 age를 adder에 더하고, 만약 해당 age가 이제까지 최댓값인 경우 Max에 update, 최솟값이 경우 Min에 update한다. 이후 counter를 1만큼 increment해준다. 이후 입력받은 func에 따라 형식에 맞춰 average, Max, Min을 출력한다.

Public member function들은 다음과 같다.

1. void deallocate(): list가 빈 list가 아닐 때 temp 노드 포인터를 선언하여 for문 내에서 각 놑에 접근하여 delete를 이용해 각 노드의 메모리를 deallocate해준다. 이는

sorting함수에서 각 노드를 array에 저장한 이후에, 그리고 main함수에서 프로그램이 종료되기 직전에 사용된다.

2. void insert(const student* Student): 이는 새로운 학생을 list에 노드로 추가하는 함수이다. 추가하기 이전에 먼저 이번에 추가하는 학생의 department 내에 존재하는 학생수가 10000명인지(각 dept 내에는 10000명을 넘는 학생이 존재할 수 없다), 그리고 새로 추가하는 학생의 dept가 혹시 10번째 dept가 될 것인지 확인하여 에러체킹을 하고, 이후 중복되는 데이터를 가지학생이 있는지 확인하여 예외처리를 한 후에, 새로운 node를 동적할당하여 list의 head 다음에 연결하고, head로 지정한다. 이후 성공적인 추가를 알리는 안내를 출력하고, dept array와 dept_cnt를 최신화하는 함수 resetDept를 호출한다. 이 함수에서는 에러체킹을 위해 선언한 정수 변수들이 있지만, 모두 단지 0이면 정상, 1이면 에러 등으로 간단히 표시하는 변수이기에 따로 설명하지 않는다.

3. void Delete(const student* Student): 이는 입력받은 학생 데이터에 일치하는 학생을 list에서 제거하는 함수이다. 만약 list가 비어있지 않다면, checker node pointer를 선언하여 for loop에서 각 노드를 옮겨다니며 일치하는 학생을 찾고, 일치하는 학생이 있는 경우, 해당 노드를 연결 해제한 후 delete해주었다. 성공적인 deletion이었을 경우, 안내문을 출력하고 resetDept()를 호출해 주었다. 이 함수에서는 에러체킹을 위해 선언한 정수 변수들이 있지만, 모두 단지 0이면 정상, 1이면 에러 등으로 간단히 표시하는 변수이기에 따로 설명하지 않는다.

4. Void resetDept(): 이 함수에서는 먼저 dept array와 dept_cnt를 모두 비워준 후, for loop에서 각 노드를 탐색하며 dept array와 dept_cnt를 다시 초기화해 주었다. 이후 사전식 순서에 맞게 dept array를 for문에서 insertion sort 해주었다.

5. Void sort(string metric): 이 함수에서는 list가 비어 있거나 하나의 노드만을 가지지 않는다면, 각 metric에 따라 insertion sort를 진행한다. 하지만 sorting을 linked list에서 직접 구현하는 것은 비효율적이라 생각해, list의 모든 노드를 한 student array에 저장하고, 이 배열에서 sorting을 진행해 주었다. 이후 해당 배열에서 다시 linked list를 recreate해주었다. Student array는 동적으로 할당해주었으며, sort 함수 마지막에 delete해 주었다. 또한, 배열을 만든 이후, 원래 있던 list의 모든 노드는 deallocate해 주었다.

6. Void Display_list() const: 노드의 모든 학생을 형식에 맞게 출력하는 함수이다.

7. Void Pivot_table(int cate, int func): 이는 위에 설명해 놓았던 세 private member function을 카테고리에 맞게 호출하는 함수이다.

마지막으로, 메인 함수의 동작은 다음과 같다. 먼저 메뉴를 저장할 int menu변수와 menu

를 getline으로 입력받을 것이기 때문에 필요한 string menuIn, 학생들의 데이터를 저장할 list List를 선언한다. Menu가 5가 아닐 동안 while loop가 계속 돌아가도록 하였다. Loop 내에서의 동작은 다음과 같다. 먼저 메뉴를 출력하고, getline으로 menuIn을 입력받아 예외처리 해준 후 int menu에 저장하고, menu가 5인 경우 반복문을 break하고, 1인 경우 새로운 학생 object를 할당받아 Input_info()와 insert()를 호출하고 예외처리해준다. 2인 경우 새로운 학생 객체를 할당받아 삭제할 학생의 데이터를 Input_info()로 받고, Delete를 호출한다. 이후 학생 객체는 할당해제하여준다. 3인 경우 age, name, gender, dept순으로 sort() 한 후 Displat_list를 호출한다. 4인 경우 카테고리와 func를 출력하고 입력받은 후 Pivot_table을 호출한다.

3. 토론 및 개선

이번 과제를 통해 class를 이용해 data 중심으로 프로그램을 개발하는 것이 어떤 것인지 조금이나마 깨달을 수 있었다. 먼저 class를 통한 캡슐화로 데이터와 함수를 묶어 프로그램을 나누었고, private과 public을 구분해 주어 정보를 숨기고, 안정적인 구현이 가능하도록 하였다. 추가 기능 구현을 하고싶었지만 하지 못한 부분은, 상속에 관한 개념을 먼저 배우고 구현하였다면, 더욱 간결한 코드를 짤 수 있었을 것이라는 아쉬움이 있다. 상속에 대한 개념이 없어 불필요하게 선언된 함수들이 많았다. 또한, 변수를 guideline에 맞춰 선언하려다 보니, redundunt하게 선언된 변수들이 있었다. 이 역시 멤버 변수를 더 선언하여 해결할 수 있을 것이다. Guideline과 다르게 private과 public을 구분하여 코드를 작성하니 좋았던 점은, 각 클래스별로 철저히 기능을 구분해 생각할 수 있었고, 데이터가 내 의지와 는 다르게 변형될 염려를 하지 않아도 되었다는 것이다. 또한, 다양한 알고리즘에 대한 지식이 있다면 좋을 것 같다는 생각을 하였다. 특히 sorting에 시간을 많이 투자하였는데, 재귀함수 등을 이용한 다양한 알고리즘에 대한 지식이 있다면 더욱 간결한 코드를 작성할 수 있었을 것이다.

아직 그래픽에 대한 지식이 없어 못하지만, student데이터들을 활용하여 다양한 시각적 자료를 툴력하는 기능을 추가할 수 있을 것이다.

4. 참고 문헌

https://www.geeksforgeeks.org/insertion-sort/ 이 사이트에서 array를 insertion sort하는 방법에 대한 힌트를 얻었음.