

2023 Spring OOP Assignment Report

과제 번호 : 2-1
학번 : 20220124
이름 : 김문겸
Povis ID : kkomy

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

- 해당 프로그램은 학생의 학과, 성별, 이름, 나이를 정보로 입력받아 정렬하고, 필요한 정보만을 피벗 변환하여 출력해주는 프로그램이다. 프로그램은 학생 정보를 입력받거나 삭제하며, 입력받은 모든 학생 정보들을 주어진 순서에 따라 정렬하여 출력하기도 한다. 또는 학과, 성별에 따른 학생들의 나이에 대한 평균, 최댓값, 최솟값을 피벗을 통해 보여준다.
- 프로그램 구현을 위해 3가지 class를 선언하였다. 학생들의 정보를 담는 곳, 해당 정보를 추가/삭제/출력 등 관리하는 곳, 해당 기능을 수행하는 곳을 별도로 분리하여 객체지향적 프로그래밍을 하고자 노력했다.
- 해당 프로그램을 실행시키면 아래와 같은 메뉴가 보여진다.

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

1번은 학생정보를 입력, 2번은 삭제, 3번은 모든 학생들의 리스트를 출력, 4번은 원하는 정보를 보여주는 피벗, 5번은 프로그램 종료로 구성되어 있다.

2. 프로그램의 구조 및 알고리즘

□ 해당 프로그램은 3개의 cpp 파일과 3개의 hpp 파일로 구성되어 있다

hpp파일

- list.h : 리스트를 구현하기 위한 클래스 "list"가 선언되어 있다.
- node.h : 리스트의 구성요소인 노드를 구현하기 위한 클래스 "node"가 선언되어 있다.
- student.h : 학생들의 정보를 담은 데이터를 다루기 위한 클래스 "student"가 선언되어 있다.

cpp파일

- assignment2.cpp : 메인함수가 존재하는 파일이다. 다른 파일에 존재하는 클래스에 대한 멤버함수의 기능이 여기서 사용된다.
- list.cpp : 리스트에 관한 함수들이 정의되어 있다. 리스트의 노드를 삽입, 삭제하거나 정렬, 출력, 피벗들을 구현하는 함수에 대한 정의가 여기에 있다.
- student.cpp : 학생들의 정보를 처리하기 위한 함수들이 여기에 있다

□ student.h

헤더파일 <string>을 사용한다.

- private

dept : 학과 / name : 이름 / gender : 성별 / age : 나이

⇒ 해당 정보들의 경우 노드가 가진 고유의 정보이기 때문에 변경되면 안되므로 외부에서 접근을 하지 않는 것이 좋다.

- public

input_info() : 학생 정보를 입력받는 함수

show_dept() : private에 있는 dept의 값을 출력

show_name() : name 값 출력

show_gender() : gender값 출력

show_age() : age 출력

blank_check() : dept, gender, name의 경우 공백이 들어가면 안되기 때문에 해당 예외 처리를 하기 위해 공백확인을 하는 기능을 가진 함수이다.

□ node.h

- public

data : 학생정보를 저장한다.

next : 리스트 상에서 다음 순번의 노드를 가리킨다.

□ list.h

- private

count : 노드 개수(= 학생 정보 수)

dept_cnt : 학과 개수 (9개 이하여야 한다)

dept[9] : 학과 종류를 저장하는 배열

dept_num[9] : dept[i]라는 학과의 인원 수는 dept_num[i]에 저장된다.

⇒ 해당 정보는 외부에 의해 변경되지 않는 것이 좋으므로 private에 선언해놓는다.

- public

head : 리스트의 가장 앞부분 노드를 가리킨다.

tail : 리스트의 가장 끝 노드를 가리킨다.

inval : false로 초기화되어 있다가, 예외처리가 필요할 시 true로 만들어서 메인함수에서 예외처리를 수행하도록 만든다.

save_node(student) : 학생 정보 저장

delete_node(student) : 학생 정보 삭제

print_list() : 리스트 출력

pivot(int, int) : 피벗 기능을 수행하기 위한 함수

□ student.cpp

- input_info()

공백 예외처리를 위해 각 정보를 getline을 통해 입력받는다. 이때 cin.ignore()를 getline 전에 실행해야 하는데, 그 이유는 앞전에 cin을 실행하면서 'wn' 등이 처리되

지 못하고 버퍼에 남아있을 수 있기 때문이다. 이를 비워주지 않으면 getline에서 버퍼의 쓰레기값들을 처리하여 에러가 발생할 수 있다.

□ list.cpp

- save_node(student)

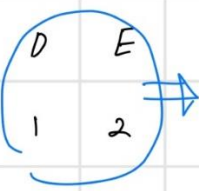
함수의 본격적인 기능을 수행하기 전에, 입력받은 정보에 예외처리를 해야 하는 부분이 있는지 검사한다. 해당 예외처리에 대한 내용은 주어진 조건을 따랐다.

노드 추가에 따른 dept[]와 dept_cnt[]는 아래와 같이 변화한다.

i	0	1	2	3	4	5	6	7	8
dept[i]	A	B	D	E					
dept_cnt[i]	2	1	1	2					

4개의 학과가 존재한다.

i	0	1	2	3	4	5	6	7	8
dept[i]	A	B	D	E					
dept_cnt[i]	2	1	1	2					



이때 새로운 학과("C"라고 하자)가 들어오게 되면, 사전순서에 따라 새로운 학과보다 뒤순서인 D,E는 한칸씩 밀린다.

i	0	1	2	3	4	5	6	7	8
dept[i]	A	B	C	D	E				
dept_cnt[i]	2	1	1	1	2				

새로운 빈칸에 C가 들어가게 된다.

- delete_node(student)

save_node와 같이 수행 전 예외처리를 진행한다.

삭제 시 dept[]와 dept_cnt[]는 아래와 같이 변화된다.

i	0	1	2	3	4	5	6	7	8
$dept[i]$	A	B	C	D	E				
$dept_cnt[i]$	2	1	1	1	2				

B를 삭제한다고 하자.

i	0	1	2	3	4	5	6	7	8
$dept[i]$	A	:	C	D	E				
$dept_cnt[i]$	2	:	1	1	2				

B가 삭제되면, B보다 사전순 기준 뒤에 위치한 C,D,E가 한칸씩 당겨진다.

i	0	1	2	3	4	5	6	7	8
$dept[i]$	A	C	D	E					
$dept_cnt[i]$	2	1	1	2					

최종적으로 이렇게 된다.

굳이 $dept[]$ 와 $dept_cnt[]$ 의 배열을 사전순으로 처리하는 이유는 이후에 구현할 pivot에서 편리하게 작용하기 때문이다.

- sort()

sort은 save_node 함수가 실행된 후 바로 실행된다. 따라서 새로운 노드가 유입될 때마다 바로바로 정렬을 해주기 때문에, 우리는 새 노드에 대해서만 정렬을 생각하면 된다. save_node에 의해 새 노드는 head에 들어온다. 따라서 본 프로그램에서는 head의 정보를 head를 제외한 전체 리스트와 비교하면서 새 노드를 사전순서 및 내림차순에 따른 적절한 위치를 찾아주고 해당 위치에 삽입한 후, head의 노드를 삭제하는 방식으로 진행할 것이다.

본 프로그램에서는 insertion sort를 사용했다.

NewNode : C, M, minsu, 20

List				
head	C	M	minsu	20
	A	...		
	A	...		
	B	...		
	C	...		
	C	...		
	C	...		
	D	...		

(C,M,minsu,20)인 노드를 리스트에 넣으려고 한다.

head	C	M	minsu	20
	A	...		
	A	...		
	B	...		
	C	F	sunny	23
	C	M	chulsu	25
	C	M	minsu	24
	D	...		

먼저 리스트에 속한 노드의 학과를 순서대로 체크하면서, 새 노드와 같거나 큰 사전 순서의 학과가 나타날 때까지 다음 노드로 넘어간다.

주어진 그림에서 C에 도달하면, 학과가 같으므로 다음 우선순위인 성별을 조사한다.

F를 지나치고, M에 도달하면 같으므로 다음 우선순위인 이름을 조사한다. "chulsu"는 사전순으로 앞이므로 지나친다.

같은 이름인 minsu에 도착하면, 이제 마지막 비교대상인 나이를 비교할 차례이다. 나이가 작다면, (C,M,minsu,24)의 앞에 삽입되고, 나이가 더 많다면 뒤에 삽입된다. 나이가 같다면 동일한 학생이므로 예외처리가 진행된다.

- pivot(int, int)

여기서 피벗의 대상은 나이이다. 먼저 학과에 대해서 조사할 것인지, 성별에 따라 조사할 것인지, 학과와 성별 둘다 조사할 것인지 조사한다. 이후 평균값을 출력할 것인지, 최댓값인지, 최솟값인지 선택한 후 선택한 내용에 따라 출력한다.

□ assignment2.cpp

변수 설명

- menu : 메뉴를 입력받기 위한 변수이다.
- cat, fun : 4번 피벗 기능에서 카테고리 and function을 입력받기 위한 변수
- stu : 학생들의 정보
- lst : 학생들의 정보가 담긴 노드를 저장하는 리스트

가장 먼저 메뉴가 실행되고, 1번부터 5번까지의 메뉴가 존재한다.

- 1번

list의 함수 save_node를 실행한다.

이때 save_node에서 예외처리가 발생하면, lst.inval의 값이 true가 된다.

해당 true에 대해 예외처리를 했다는 메시지를 출력한 후 처음으로 돌아간다.

정상적으로 save_node가 실행되었다면, list의 sort함수를 실행한다.

- 2번

list 함수 delete_node를 실행한다.

역시 예외처리가 진행되었다면 메시지 출력 후 처음으로 돌아간다.

- 3번

lst.print_list함수를 실행하여 리스트를 출력한다.

- 4번

카테고리 메뉴를 보여준 후 어떤 카테고리를 선택할 것인지 입력받는다. 이때 1,2,3번이 아닌 다른 숫자를 입력할 경우 예외처리를 한다.

다음 function 메뉴를 보여준 후 어떤 함수를 선택할 것인지 입력받는다. 역시 다른 숫자를 입력할 경우 예외처리를 한다.

- 5번

프로그램을 종료한다.

- 추가 기능)

1~5번이 아닌 다른 숫자를 메뉴에 입력할 경우 ERROR 메시지를 출력하고 다시 입력하도록 구현하였다. 일종의 예외처리이다.

3. 토론 및 개선

- 수업시간에 배운 class를 통해서 프로그래밍을 하면서, 객체지향의 의미를 좀 더 직접적으로 이해할 수 있는 기회가 되었다. class에 속한 멤버함수와 메소드들의 기능을 곱씹어보면서, public과 private을 설정하였고, 이는 객체지향프로그래밍을 하는 가장 중요한 부분이라는 것을 깨달았다.
- class의 멤버함수들을 초기화하는 부분이 있는데, 이 부분을 constructor을 통해 구현할 수 있을 것이다.
- 해당 프로그램에서는 3개의 class가 선언된다. 그 중에서도 객체지향프로그래밍이 잘 나타나는 class는 student와 list이다. student의 경우 입력받는 정보들은 고유의 값이므로 외부로부터 보호하기 위해 encapsulation을 진행했다. 즉 private을 설정하였다. 또한 list의 경우 student의 정보를 저장한 노드들을 관리하는 다양한 함수들이 존재한다. 리스트 내의 노드들을 삭제하고, 추가하고, 정렬한다. 하지만 메인함수에서는 이 기능들이 어떻게 진행되는지 모르고, 알 필요 또한 없다. list에 대한 함수는 list 내에 정의되어 있다. main함수는 해당 함수를 사용하는 곳으로만 사용된다. 또한 list에 관련된 것들은 list에, student에 관련된 것들은 student에 선언 및 정의되어 있기 때문에, 오류가 생겼을 경우 해당 파일만 수정하면 된다. 이렇게 객체지향 프로그래밍의 특징을 잘 드러나도록 프로그래밍을 진행하였고, 객체지향적인 프로그래밍이 어떤 것인지 더 잘 이해하게 되었다.

4. 참고 문헌

-