Object-Oriented Programming Report

Assignment 3-3

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2022202065
Name	박나림
Class (Design / Laboratory)	2 / C
Submission Date	2023. 5. 20

Program 1

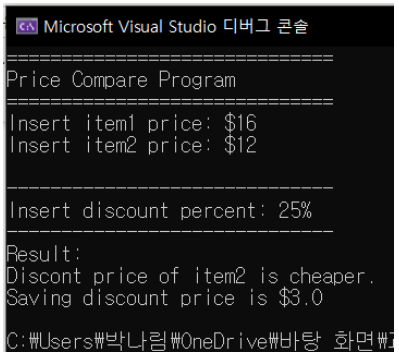
❑ 문제 설명

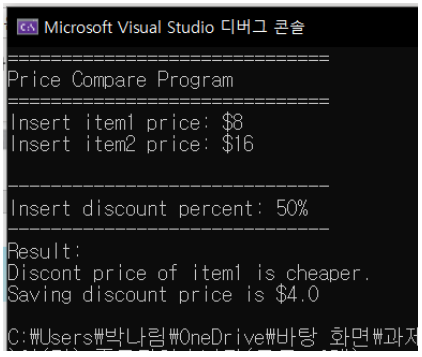
가격을 비교하는 프로그램을 구현하는 문제로, Sale 클래스와 이를 상속받는 클래스를 만들어야 한다. Sale 클래스에서는 가격 멤버 변수를 가지며 '<'연산자로 할인된 가격을 비교하는 역할을 한다. Bill 함수는 가상함수로 선언되어 있어야 한다. 이 함수는 자식 클래스에서 정의하도록 하며, 자식 클래스의 객체는 그 가상함수 정의를 사용하게 된다. 사용자로부터 처음에 두 물건의 가격을 입력 받고 할인율을 입력 받으면, 각 물건의 할인된 가격을 비교하여 더 싼 물건과 얼마나 더 싼 가격인지 출력하는 것으로 프로그램을 구현하도록 한다.

-구현 방법

Sale 클래스의 각 멤버함수들에서, get 은 가격을 반환하고 Bill 은 가상 함수로써 자식 클래스의 것이 사용되고 여기에선 무시되기 때문에 아무 값이나 반환하도록 한다. Savings 함수는 두 가격의 차이를 계산하고 반환하며 연산자는 두 가격을 비교할 수 있도록 정의한다. 이러한 Sale 을 public 으로 상속받는 DiscountSale 클래스를 만들어서, Discount 를 멤버 변수로 가지고 멤버 함수로는 비슷하게 구현한다. 여기서의 Bill 함수는 할인 가격을 계산한 후 반환하도록 정의한다. Savings 와 연산자 함수는 Bill 함수의 값을 이용하여 계산한 후 반환한다. 메인 함수 내에서는 사용자로부터 두 물건의 가격과 할인율을 입력 받고, 각각 DiscountSale 클래스 객체를 생성하여 가격과 할인율 변수를 인수로 넘겨준다. 할인율을 입력 받을 땐 문자열로 받고나서, '%'를 제외하기 위해 atof 함수를 사용하여 숫자만 저장하도록 한다. 그렇게 생성된 객체를 if 문으로 연산자 '<'을 사용하여 비교한다. 이때 '<'만 정의했으므로 '>'과 같은 연산자는 사용이 불가하다. 따라서 해당 값의 순서만 바꾸는 형식으로 조건문을 쓴다. 그래서 더 싼 물건과 얼마만큼 돈을 절약할 수 있는지 할인된 두 가격의 차이를 Savings 함수를 불러와서 같이 출력한다. 이때 더 큰 값을 객체로써 불러오고, 작은 값을 인수로 전달한다.

❑ 결과 화면





차례대로 1 번째 값이 더 큰 경우, 2 번째 값이 더 큰 경우, 두 가격이 동일한 경우로 나누어서 각각 테스트해 본 결과이다.

❑ 고찰

Sale 클래스와 DiscountSale 클래스의 멤버 함수들이 비슷해 보이지만 정의를 다르게 해야 한다는 점이 약간 헷갈렸었다. 하지만 둘이 전달받는 매개변수의 타입 자체가 다르므로, 연산을 다르게 해야 한다는 것을 깨달았다. 또한 결국 이 문제에서 사용되는 것은 DiscountSale 의 함수들이므로, 이 문제에서 오버로딩과 오버라이딩에 대해 더 생각해볼 수가 있었다. 또한 연산자 오버로딩을 통해 함수를 정의해두면 메인 함수내에서 객체끼리의 연산이 가능하다는 것을 확실히 알 수 있었다. 여기서 할인된 가격을 비교하여 그 차이를 출력하는 부분에서 시행착오가 많았는데, 다른 새로운 멤버 변수를 만들어서 값을 저장하고 반환하는 방법을 시도해봤었다. 하지만 더 복잡해지는 것 같아 다른 방법을 고민하다가, Bill 함수에서 할인된 값을 계산하고 반환하여 이 함수를 Savings 와 연산자 오버로딩 함수에서 사용하면 가능하다는 점을 깨달았다.

Program 2

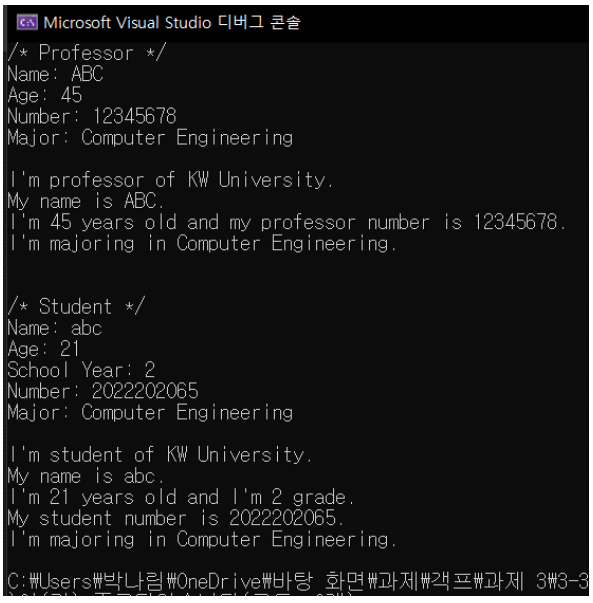
❑ 문제 설명

Professor, Student 클래스를 각각 만들어서 모두 Person 클래스를 상속받는 형식으로 구현한다. Person 클래스는 나이, 이름, Say() 함수를 가지며, 이때 함수는 순수 가상함수로써 자식 클래스에서 오버라이딩을 강제화 하도록 한다. Professor 클래스는 교수 번호, 전공을 추가 멤버 변수로 가지며 Student 클래스는 학생 번호, 전공, 학년을 추가 멤버 변수로 가진다. 이 클래스들을 구현하고 적절히 테스트해보는 문제이다.

-구현 방법

먼저 각 자식 클래스는 Person 클래스를 Public 으로 상속받도록 하고, 멤버 변수는 protected 로 한다. 이 외에 멤버 함수로는 get, set 함수를 구현하여 set 함수는 메인 함수에서 정보를 저장하는 데 사용하고, get 함수는 Say 함수에서 정보들을 불러올 때 사용하도록 한다. 메인 함수 내에서는 정보들을 입력 받고 클래스 객체를 생성하여 정보를 설정하고 Say 함수를 쓴다. 이때 정보는 공백 포함으로 입력 받아야 하므로 scanf_s 를 사용한다.

❑ 결과 화면



Professor, Student 의 각 정보들을 차례로 입력하면 총 정보를 출력하는 형식으로 테스트해 본 결과이다.

❑ 고찰

클래스에서 상속을 받을 때 그 타입마다 사용 방법이 다르다는 점을 이번 문제를 통해 더 잘 알 수 있었다. 또한 Say 함수에서 처음엔 get 함수를 사용하지 않고 멤버 변수 그대로 불러오는 형식으로 했었는데, 이렇게 해도 출력은 정상적으로 나왔다. 하지만 다른 형식으로 테스트를 할 때 결과가 달라질 수도 있으니 get 함수를 사용하는 형식으로 바꾸었다. 메인 함수에서 테스트를 할 때에는 이름이나 전공 등에서 공백을 포함해야 한다는 점을 깨달아서 scanf_s 로 바꾸었다. 주로 cin 을 쓰다가 이 함수를 쓰니 방법이 익숙하지 않아 꽤 시행착오를 겪었지만 여러 오류를 고쳐 나가면서 해결할 수 있었다.

Program 3

❑ 문제 설명

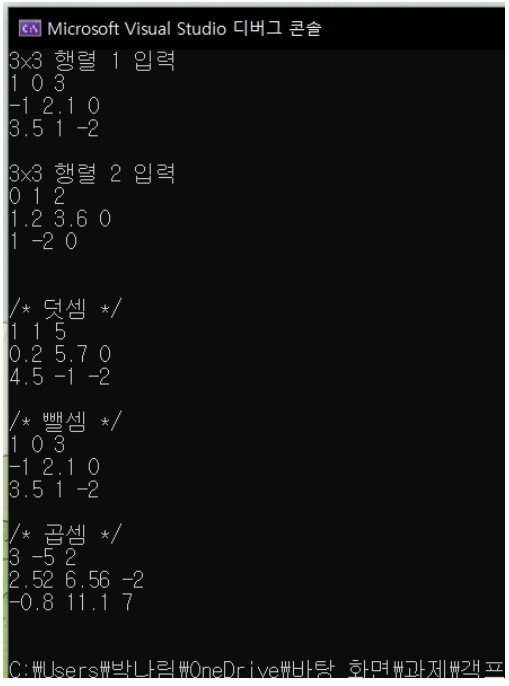
연산자 오버로딩을 사용하는 문제로, 3x3 크기의 행렬에서 덧셈, 뺄셈, 곱셈을 하기 위해 구현하는 문제이다. Matrix 클래스를 생성하여 double 타입의 요소들을 생성자에서 초기화 해야 하는데, 이때 아무것도 입력되지 않은 경우를 생각하여 기본 값을 넣도록 해야 한다. Public 으로는 행렬 출력 함수, 3 개의 연산자 오버로딩 함수들을 멤버 함수로 가져야 한다.

-구현 방법

먼저 생성자를 2 개 만들어서 기본 생성자는 행렬 값을 기본 0 값으로 초기화 하고, 초기화를 하는 생성자는 매개변수로 전달받은 행렬 값으로 초기화 하는 형식으로 한다. 그리고 void 형의 출력 함수를 만들어서 행렬의 값을 공백과 개행을 포함하여 출력하도록 만든다. 연산자 오버로딩에서는, += 과 -= 연산자들은 기존 객체의 행렬과 새로 매개변수로 전달받은 행렬을 사용하여 총 2 개의 행렬을 더하거나 빼서 대입하는 형태로 저장한다. 그리고 대입 결과인 행렬을 반환해준다. * 연산자에서는 결과 값을 저장할 객체를 새로 하나 더 생성한다. 이때 결과 행렬은 아무 값도 전달하지 않았기 때문에 기본 0 값으로 초기화 된다. 그러면 이 행렬에 +=을 사용하여 기존 행렬과 매개변수 행렬을 곱하는 형식으로 하여 저장한 다음 결과 행렬을 반환해준다.

메인 함수에서는 사용자로부터 두 행렬을 입력 받아서 각각 행렬 객체를 생성해준 다음 덧셈, 뺄셈, 곱셈 연산을 하면서 출력함수를 객체로써 불러와준다.

❑ 결과 화면



사용자로부터 행렬 두개를 입력 받으면, 먼저 행렬 1+=행렬 2 의 결과를 출력한다. 그 다음 다시 행렬 1-=행렬 2 의 결과를 출력하므로 결과는 처음 행렬 1 과 동일한 값이 나온다. 그 상태에서 행렬 1 * 행렬 2 의 결과를 출력하는 것으로 테스트를 마친 모습이다.

❑ 고찰

연산자 오버로딩을 하면서, 새로운 연산자를 정의하더라도 그건 객체로써의 연산을 정의하는 것일 뿐, 원래 요소끼리의 연산은 가능하다는 점을 깨달았다. 그래서 연산자 오버로딩 함수를 쓸 때에도 간단하게 기존 연산자를 이용하여 쓸 수 있었다. 연산자 중 특히 곱셈 연산자를 오버로딩 할 때 헷갈렸었던 것 같다. 다른 연산자는 대입 연산자가 결합된 형태이므로 두 개의 객체로도 연산이 가능했었는데, 곱셈은 새로운 객체가 필요해서 거기에 += 연산자를 같이 써야 했었다. 결국 원래 행렬 계산하는 것과 다를 것이 없었지만 처음에는 바로 생각하지를 못해서 다른 변수를 추가해서 하는 등 시행착오를 겪었었다. 그리고 반환할 땐 해당 객체 또는 객체에서 행렬 변수를 불러온 형태 둘 다 정상적인 결과 값이 나온다는 것을 새롭게 알 수 있었다.

Program 4

❑ 문제 설명

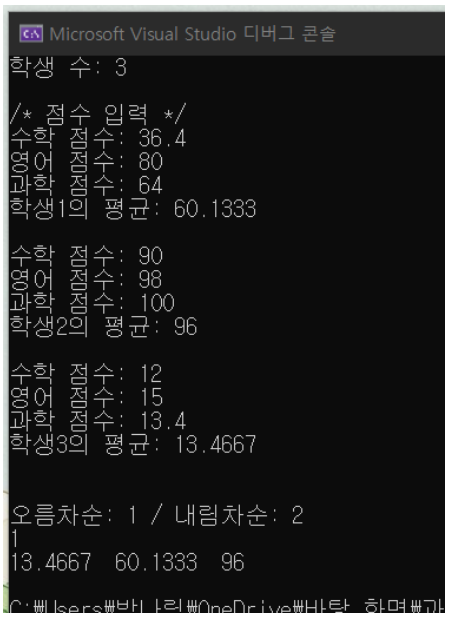
학생들의 성적을 관리하는 프로그램으로, 이중 연결 리스트를 사용하는 문제이다. main 함수 내에서 학생들의 수학, 영어, 과학 점수를 받고 평균 점수를 계산한 뒤 Score 클래스에 정보를 전달하여 저장시킨다. 이때 이 클래스는 평균 점수를 저장하는 double 형 변수와 이전, 다음을 가리키는 포인터를 가진다. StudentSocreList 클래스에서는 Score 형의 처음과 끝을 가리키는 포인터를 가지며, 멤버 함수로는 Score 노드를 오름차순으로 추가하는 Insert 함수와 매개변수 isAscending 이 true 면 오름차순, flase 면 내림차순으로 출력하는 함수를 가져야 한다.

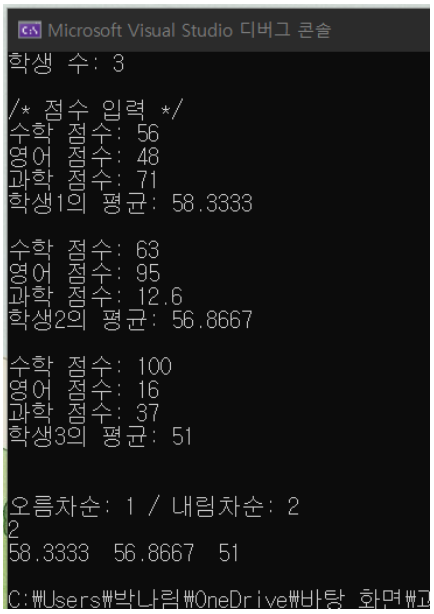
-구현 방법

main 함수에서 먼저 사용자로부터 학생 수를 입력 받아 그 수만큼 Score 객체 배열을 동적할당 시킨다. 또한 그 수만큼 반복하여 3 과목의 점수를 입력 받고 평균을 계산한 뒤 객체의 Set 함수로 평균 점수를 설정하며 list 객체에 추가하는 형식으로 한다. 입력이 끝나면 사용자의 메뉴 입력에 따라 오름차순 또는 내림차순으로 스위치문을 통해 구분하여 출력하도록 한다.

Score 클래스에서는 각각 Set 함수로 매개변수의 값을 클래스 멤버 변수에 저장하며, Get 함수로는 해당 멤버 변수를 반환하도록 한다. 그 다음 StudentScoreList 클래스에서는 먼저 Insert 함수를 구현한다. 처음 들어오는 경우와 이어서 들어오는 경우로 크게 if else 문으로 나누어서, 처음 들어오는 경우(head 포인터가 nullptr 인 경우)에 매개변수로 받은 pScore 을 각각 head 와 tail 에 저장한다. 이어서 들어오는 경우에는 현 위치를 가리킬 pCurr 을 생성하고 head 값으로 초기화 시킨다. 그리고 while 문으로 끝까지 탐색하여 만약 pCurr 이 가리키는 평균 값보다 pScore 이 가리키는 평균 값이 더 크다면 오름차순에 알맞게 뒤에 추가해야 되는 경우이므로 pCurr 을 다음 위치로 옮긴다. 반복하다가 현 위치 값이 더 커질 경우 break 로 while 문을 벗어난다. 현 위치 설정 작업이 끝나면 if 조건문을 이용하여 처음이나 중간에 들어가는 경우, 끝에 들어가는 경우로 나누어서 해당 위치에 각각 포인터를 이용해 이중 연결시킨다.
마지막으로 출력 함수에서는 매개변수 값에 따라 if 문으로 나누어서 true 인 경우 이미 오름차순으로 저장되어 있으므로 현 위치 노드를 생성하여 head 값으로 초기화 시킨 뒤 처음부터 끝까지 값을 출력한다. false 가 들어온 경우에는 내림차순이라 역순으로 출력되어야 하므로 현 위치 노드를 생성하여 tail 값으로 초기화 시킨 뒤 끝부터 처음까지 값을 출력한다.

❑ 결과 화면





학생 수에 따라 점수를 반복해서 입력 받고 평균을 계산하여 노드에 추가한다. 그리고 원하는 출력 순서를 입력하면 그에 알맞은 순서로 출력하도록 한 모습이다.

❑ 고찰

지금까지 한 연결리스트와 달리 이중으로 연결하는 리스트라서 처음엔 어떤 식으로 구현을 해야 할지 어려웠었다. 하지만 다시 공부를 해 보면서, prev 포인터를 이용하여 이전 값과 현재 값을 같이 설정해주면 된다는 점을 깨달았다. 이렇게 연결을 하니 오히려 출력할 때 역순으로 출력하는 경우, 단일 연결리스트였으면 다시 탐색하면서 더 복잡한 코드가 나왔을 것인데 여기서는 역순으로 이동할 수 있어서 더 간단해진 걸 알 수 있었다. 그리고 main 에서 객체 배열로 값을 추가할 때 여러 오류가 발생하였었는데, 이는 주소 값을 전달하는 것으로 하여 해결할 수 있었다.