객체지향프로그래밍

Assignment Report 3-1

2019202050 이강현

1. Class Person은 사람이름, 나이, 거주지 정보를 가지고 있는 Class이 다. 아래 main함수가 예시와 같이 동작하도록 Class를 구현하고 동작시키시오.

필요한 개념: class 사용법, <<,>>연산자 오버로딩 하는법

문제 설명: 클래스 자체를 cout과 cin을 이용하여 입출력을 하려면 연산자를 추가 정의 해야한다.

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

<입출력 결과>

고찰: 함수에 대한 오버로딩을 배우고 오버로딩이라는 것이 단순히 이름만 같아도 무방하도록 하는 것인줄 알았지만 이렇게 연산자에 추가적인 기능을 부여할 수 있다는 점에서 정말 중요도가 높다는 것을 깨달았다.

1. Class Polygon과 Triangle, Square는 아래그림(Feature 1)과 같은 상속관계를 가진 다. 각각의 Class는 Feature 1에 있는 멤버 변수, 멤버 함수만 가지고 있으며 생성자와 소멸자를 추가할 수 있다. 프로그램의 main함수가 정상적으로 동작해서 예시와 같이 Area(넓이)와 Perimeter(둘레)가 출력되어 나오도록 Class들을 구현하시오. 텍스트, 전자기기이(가) 표시된 사진

   자동 생성된 설명

필요한 개념: 클래스 상속에 대한 이해, 생성자 사용법, 생성자 오버로딩등

문제 설명: polygon에 넓이와 둘레를 구할 수 있는 함수를 마련해 둔 뒤 public 접근 수준으로 각각 triangle, square에게 상속을 해준다. 생성자를 통해 자식 클래스에서 부모 클래스에게 접근하여 생성자를 호출한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<결과>

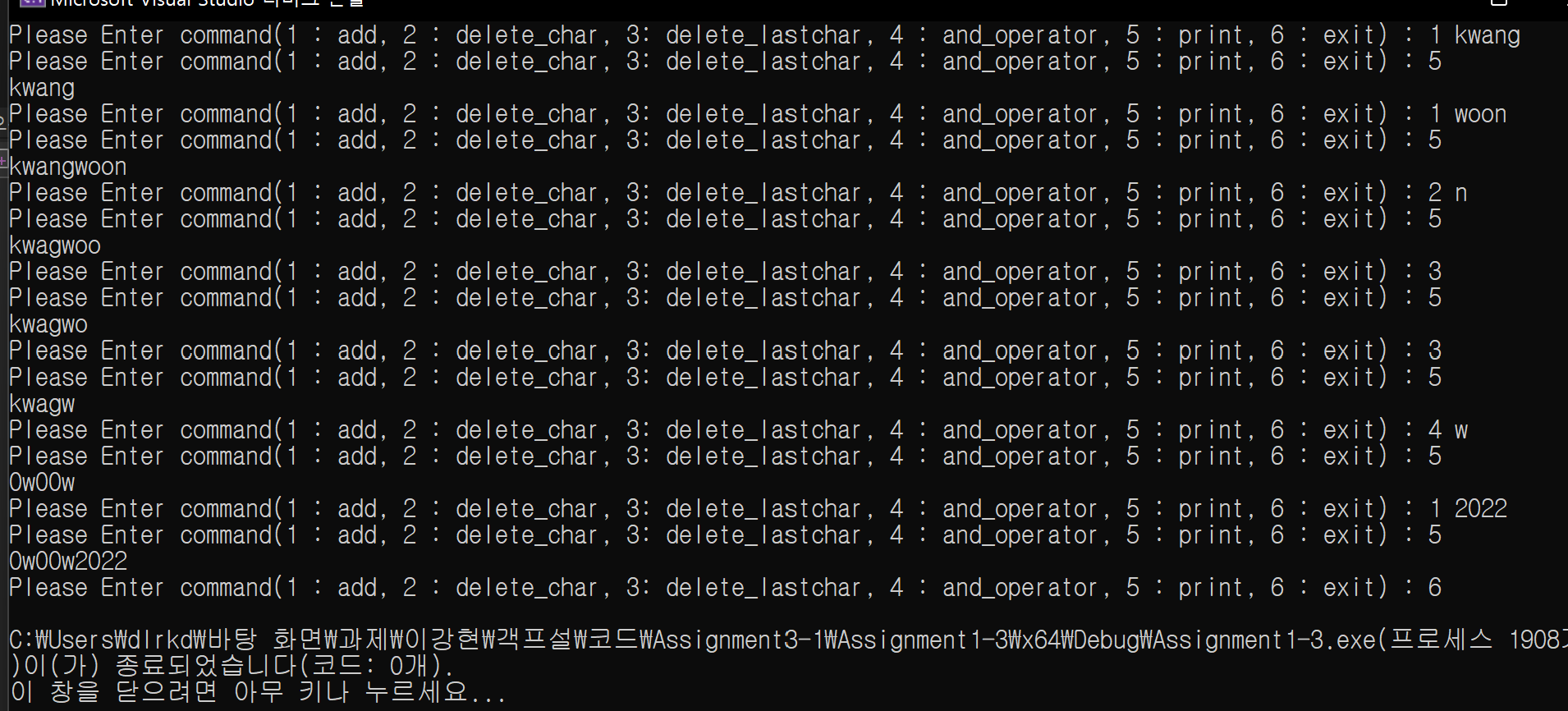
고찰: 클래스에서 다른 클래스에게 접근 권한을 주고 public 수준의 멤버함수를 이용할 수 있게 한다는 점이 코드가 중복되는 것을 쉽게 해결해 주었다. 비슷한 객체들끼리 서로연관되게 만들어 추상적인 객체간의 구조를 잘 이해하게 해주는 개념이었다.

1. mystring class는 private권한의 char \*형 멤버변수 string과 public권한의 print함수 그리고 생성자와 소멸자만 멤버로 가지고 있으며 +=, -=, --, && operator에 대해서 table 1과 같은 동작을 하는 class이다. 프로그램은 table 2와 같은 command를 가지며 exit명령을 읽을 때 까지 반복해서 command를 입력받는다. 테이블이(가) 표시된 사진

   자동 생성된 설명

필요한 개념: 연산자 오버로딩, friend를 이용한 클래스 접근

문제 설명: friend선언을 통해 연산자 오버로딩을 거친 operator가 클래스의 멤버 변수에 접근이 가능하게 만든다. 연산자마다 반환형식이 클래스인 Operator를 정의한 후 main에서 명령어에 맞게 사용한다.



<결과>

고찰: 1번 문제와 마찬가지로 연산자를 오버로딩하는 문제였는데 문자열과 문자열간의 연산을 사용자가 직접 설정할 수 있다는 것이 편리했다. 문자열과 문자열을 더한다는 개념을 덧붙이기 혹은 글자수를 더해서 반환하기등 여러가지로 생각하는대로 추가 정의만 해주면 가능했다. 흔히 편하게 사용하는 strcat,strcpy같은 편리한 함수들을 이 연산자 오버로딩을 통해 만들 수 있겠다고 생각했다.

1. 번호와 이름을 저장, 출력, 정렬, 삭제하는 프로그램을 구현하시오. 번호와 이름은 Node class에 저장되어 있어야 하며 doubly linked list에 저장되어 관리되어야 한다. Doubly linked list란 그림과 같이 Node가 양방향으로 연결되어 있는 구조를 말한다. 프로그램은 table1과 같은 command를 가지며 exit 명령을 받을 때까지 반복해서 command를 입력받는다.

테이블이(가) 표시된 사진

자동 생성된 설명

<결과>

필요한 개념: 단방향 연결리스트와 양방향 연결리스트의 차이, 노드의 생성,제거,정렬,접근

문제 설명: 노드마다 데이터와 앞 노드와 뒤 노드의 주소값을 저장해두고 노드를 추가 제거 정렬 삭제를 할 수 있게 한다. 앞과 같은 행동을 하기위해 서로 연결되어 있는 노드에 접근하기 위한 head라는 노드 포인터를 이용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<결과>

고찰: 앞에서 뒤로만 접근할 수 있는 단방향 연결리스트와는 다르게 양방향 연결리스트는 뒤에서도 앞으로 접근할 수 있어 편했다. 하지만 노드의 수정시 앞노드와 뒤노드의 주소값의 변경도 이루어져 수정할 때는 더 힘들었다. 앞에서 뒤로 흘러가는 흐름이 일정하다면 단방향, 일정하지 않다면 양방향으로 이어주는게 좋을 것 같다고 생각했다.

노드가 이어져있는 연결리스트라는 개념이 뭔가 명확하진 않았다. 새 노드를 추가할 때 동적할당을 통해 노드들을 생성해주었고 head같은 포인터가 없다면 노드들은 서로 이어져있기만 할 뿐 어디에 있는지 알 수 없었기 때문이다. 포인터에 의존하기에 head값을 잃어버리면 코드가 동작하지 않는 문제점도 자주 발생했다. 하지만 배열과는 다르게 원할 때 리스트의 길이를 수정하거나 필요없다면 메모리를 많이 사용하지 않도록 해제하거나 하는 것이 가능하기에 linked list의 필요성을 느낄 수 있었던 예제였다.