202284023 송성진
객체지향프로그래밍 과제
소스 구현 설명

문제 정의)
이 문제는 사용자가 콘솔을 통해 그래픽 도형을 추가하고 관리할 수 있는 간단한 그래픽 편집기를 구현하는 것이 목표입니다. 프로그램은 도형 삽입, 도형 삭제, 모든 도형 보기, 프로그램 종료의 네 가지 주요 기능을 제공합니다.
이 프로그램은 객체 지향 프로그래밍의 클래스 상속과 다형성을 활용하여 다양한 도형(Line, Circle, Rectangle)을 효율적으로 관리하는 구조를 갖습니다.

문제 해결방법)
첫 번째로 객체 지향 설계입니다.
shape 추상 클래스: 모든 도형의 공통적인 기능(인터페이스 역할)을 정의하고, Draw()라는 순수 가상 메서드를 포함하며, 자식 클래스에서 반드시 구현하도록 강제합니다.
Shape를 상속받아 각각의 Draw()메서드를 구현합니다.
이를 통해 프로그램은 구체적인 도형 클래스의 타입에 의존하지 않고 Shape 인터페이스를 통해 도형을 관리할 수 있습니다.

두 번째 데이터 관리 구조입니다.
벡터(std::vector):
동적 배열로, 도형 객체를 관리하는 컨테이너 역할을 합니다.
삽입, 삭제, 조회와 같은 작업을 효율적으로 처리할 수 있습니다.
각 도형은 Shape*포인터로 저장되며, 다형성을 통해 Draw() 메서드를 호출합니다.

세 번째 기능 구현입니다
도형 삽입(InsertShape): 사용자 입력에 따라 적절한 도형 객체를 동적으로 생성하고, 벡터에 저장합니다.
도형 삭제(DeleteShape): 사용자 입력으로 삭제할 도형의 인덱스를 받아 벡터에서 해당 객체를 제거하고 메모리를 해제합니다.
도형 목록 출력(ShowShapes):  터에 저장된 모든 도형을 순회하며 Draw()메서드를 호출해 도형 이름을 출력합니다.
프로그램 종료(Run):  용자가 종료를 선택하면, 프로그램이 종료되며 동적으로 생성된 모든 객체의 메모리를 안전하게 해제합니다.

아이디어 평가)

도형의 공통 인터페이스를 제공하여 각 도형(Line, Circle, Rectangle)의 개별 구현을 가능하게 했습니다. 그리고, 모든 도형 객체를 일관되게 처리하기 위해 벡터를 활용하여 Shape*로 관리하였고, 새로운 도형 추가 시 최소한의 코드 변경만으로 기능 확장이 가능하도록 설계하였습니다.
객체 생성 및 삭제시 동적 메모리 관리를 통해 안정성을 확보하였습니다.

알고리즘)
삽입: 사용자 입력에 따라 도형(Line, Circle, Rectangle)을 동적으로 생성.
생성된 객체를 벡터에 추가.
삭제: 사용자로부터 삭제할 도형의 인덱스를 입력받음.
입력값이 벡터의 유효 범위 내일 경우, 해당 객체를 삭제하고 메모리를 해제.
모두 보기: 벡터에 저장된 모든 도형 객체를 순회하며, 각 도형의 Draw() 메서드를 호출해 출력.
벡터가 비어 있으면 "저장된 도형이 없습니다" 메시지 출력.
종료:벡터에 남아 있는 모든 도형 객체를 순회하며 메모리를 해제. 프로그램 종료.