1. 문제 정의

문제 정의는 Shape 클래스를 기반으로 한 간단한 그래픽 편집기를 콘솔 환경에서 구현하는 것dl다.

주요 요구사항:

1. 다양한 도형(Circle, Rect, Line)을 상속 구조로 설계하고, 다형성을 활용해 도형 객체를 관리함.
2. Shape를 기반으로 한 도형의 삽입, 삭제, 출력 기능을 구현함.
3. 사용자는 콘솔 메뉴를 통해 도형을 삽입, 삭제, 전체 출력, 프로그램 종료를 선택할 수 있어야 함.

2. 문제 해결 방법

1. **클래스 설계**:
    a. Shape: 추상 클래스. 모든 도형의 공통 인터페이스(draw())를 제공.
    b. Circle, Rect, Line: Shape를 상속받아 각각의 draw()를 구현.
    c. GraphicEditor: Shape 객체들을 관리하고, 메뉴에 따라 도형을 삽입, 삭제, 출력할 수 있는 메인 클래스.
2. **다형성과 컨테이너**:
    a. std::vector<Shape*>를 사용하여 다양한 도형 객체를 동적으로 관리.
    b. 다형성을 통해 도형 객체에 대한 공통 인터페이스로 작업.
3. **메뉴 기반 인터페이스**:
    a. 사용자가 콘솔을 통해 메뉴를 선택하고 작업을 수행할 수 있도록 switch문과 루프를 설계.
4. **메모리 관리**:
    a. 동적으로 생성된 도형 객체를 안전하게 관리하고 삭제 시 메모리를 해제.

3. 아이디어 평가

1. **클래스 설계**:
    a. Shape와 이를 상속받는 도형 클래스 구조는 구현이 직관적이며 확장성이 뛰어남. 새로운 도형을 추가하려면 Shape를 상속받아 새로운 draw()를 구현하면 된다.
2. **다형성과 컨테이너**:
    a. std::vector<Shape*>를 사용하여 도형 객체들을 유연하게 관리할 수 있다. 동적 다형성 덕분에 코드가 깔끔해졌으며, 각 도형의 구체적인 세부사항을 GraphicEditor가 알 필요가 없어 관리가 용이하다.
3. **메뉴 기반 인터페이스**:
    a. 사용자가 작업을 명확히 선택하고 쉽게 이해할 수 있도록 구성되었다.
    b. 다만, 입력 에러 처리가 부족하다는 점에서 개선이 필요하다.
4. **메모리 관리**:
    a. delete를 통해 메모리를 해제하여 메모리 누수를 방지했다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

키 아이디어

1. **다형성을 활용한 유연한 구조**:
    a. Shape를 추상 클래스로 설계하여 모든 도형이 공통적으로 사용할 draw() 메서드를 선언했다.
    b. 이를 상속받은 Circle, Rect, Line 클래스는 도형별로 다르게 구현된 draw() 메서드를 제공한다.
2. **벡터 컨테이너 활용**:
    a. std::vector<Shape*>를 사용하여 도형 객체를 동적으로 관리했습니다. 삽입, 삭제, 출력 작업을 유연하게 처리할 수 있다.

알고리즘 설명

1. **삽입 (Insert)**:
    a. 사용자가 선택한 도형(`Circle`, `Rect`, `Line`)에 따라 객체를 생성하고, 이를 벡터에 추가한다.
    b. 사용자가 잘못된 입력을 하면 오류 메시지를 출력한다.
2. **삭제 (Delete)**:
    a. 사용자로부터 삭제할 도형의 인덱스를 입력받고, 벡터에서 해당 객체를 삭제한다.
    b. 삭제 시 `delete`를 호출하여 메모리를 해제한다.
3. **모두 보기 (View All)**:
    a. 벡터를 순회하며 각 도형 객체의 `draw()` 메서드를 호출하여 콘솔에 도형 정보를 출력한다.
4. **종료 (Exit)**:
    a. 프로그램 종료 시 모든 동적 메모리를 해제한다.