1. 문제 정의:

이 프로그램의 목표는 사용자가 도형을 삽입, 삭제하고 이를 출력할 수 있는 **그래픽 에디터**를 구현하는 것입니다. 도형으로는 **선(Line)**, **원(Circle)**, **사각형(Rect)**이 제공되며, 각각의 도형을 생성하고 관리하는 기능을 구현합니다. 도형은 **연결 리스트** 또는 **배열**(여기서는 std::vector)로 관리되며, 사용자는 도형을 추가하거나 삭제할 수 있고, 모든 도형을 출력할 수 있어야 합니다.

2. 문제 해결 방법:

문제 해결을 위해서는 도형들을 동적으로 생성하고 관리할 수 있는 데이터 구조가 필요합니다.

도형들은 **std::vector<Shape*>**를 사용하여 관리됩니다. 벡터를 사용하는 이유는 동적으로 크기가 변화하고, 순차적으로 접근이 가능하기 때문입니다. 벡터에 도형을 추가하거나 삭제하는 것은 매우 간단하고 효율적으로 처리할 수 있습니다.

3. 아이디어 평가:

이전 과제의 연결 리스트 방식에서 Shape 객체들을 연결하여 관리했으나, std::vector로 변경한 이유는 다음과 같습니다.

벡터 사용의 장점:

- **동적 크기 관리**: 연결 리스트처럼 메모리를 직접 관리할 필요 없이, 벡터는 크기가 자동으로 조정됩니다.
- **편리한 인덱스 접근**: 벡터는 인덱스로 직접 요소에 접근할 수 있기 때문에, 도형의 삽입, 삭제 및 탐색이 더 직관적이고 효율적입니다.
- **메모리 관리**: 벡터의 크기가 자동으로 조정되므로, 도형을 추가하거나 삭제할 때 별도의 메모리 관리가 필요 없습니다.

문제 해결 평가:

- **도형 추가**: 도형을 벡터에 push_back으로 추가하므로, 추가가 직관적이고 빠릅니다.
- **도형 삭제**: 벡터에서 도형을 삭제할 때 erase를 사용하여 인덱스 기반으로 간단하게 삭제할 수 있습니다. 삭제한 도형은 delete를 사용하여 메모리에서 해제합니다. 다만, 삭제 시 인덱스가 변경되므로 삭제 후에는 벡터가 재정렬됩니다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명:

핵심 아이디어는 std::vector<Shape*>를 사용하여 도형을 동적으로 관리하는 것입니다. 이를 통해 도형을 추가하거나 삭제하는 작업을 보다 직관적이고 효율적으로 처리할 수 있습니다. 벡터를 사용함으로써 코드가 간결해지고 유지보수가 용이해졌습니다.