1. 문제 정의

이 프로그램은 도형을 관리하고 처리하는 시스템을 구현하는 문제입니다. 주요 요구사항은 다음과 같습니다.

- 사용자가 도형(선, 원, 사각형)을 추가할 수 있어야 한다.
- 추가된 도형들을 삭제할 수 있어야 한다.
- 현재까지 추가된 모든 도형들을 볼 수 있어야 한다.
- 프로그램은 사용자와의 상호작용을 통해 동작해야 한다.

따라서, 도형을 나타내는 'Shape' 클래스를 기본으로 하여 여러 종류의 도형('Line', 'Circle', 'Rect')을 상속받아 다형성을 구현하고, 이를 연결 리스트로 관리하는 방식이 필요합니다.

2. 문제 해결 방법

문제 해결을 위한 주요 아이디어는 연결 리스트를 사용하여 도형들을 관리하는 것입니다. 각 도형 객체는 다음 도형을 가리키는 'next' 포인터를 가집니다. 이를 통해 도형을 추가하거나 삭제할 때, 리스트의 끝 또는 중간에서 도형을 처리할 수 있습니다.

주요 아이디어:

1. 도형 클래스 구조 설계:
   - 기본 도형 클래스를 'Shape'로 정의하고, 이를 상속받는 'Line', 'Circle', 'Rect' 등의 구체적인 도형 클래스를 만들어 각 도형에 대해 'draw' 메소드를 오버라이딩합니다.

2. 연결 리스트:
   - 도형을 연결 리스트 형태로 저장합니다. 각 도형은 'Shape' 클래스의 'next' 포인터를 통해 다음 도형을 가리킵니다. 새로운 도형을 추가할 때는 리스트의 끝에 추가합니다.

3. UI 클래스와 사용자 상호작용:
   - 사용자로부터 입력을 받아 도형을 추가하거나 삭제하는 기능을 제공합니다. 'UI' 클래스는 사용자와의 상호작용을 담당하며, 각 기능을 수행하는 메소드('func1', 'func2')를 통해 도형을 추가하고 삭제합니다.

4. 삭제 기능:
   - 도형을 삭제하는 과정에서 연결 리스트에서 특정 위치의 도형을 제거하고 나머지 도형들이 올바르게 연결되도록 합니다.

3. 아이디어 평가

아이디어 1: 연결 리스트로 도형 관리

- 장점: 연결 리스트를 사용하면 도형을 동적으로 추가하고 삭제할 수 있어 메모리 효율적입니다. 도형의 수가 변할 때마다 리스트의 크기를 재조정할 필요가 없고, 삭제가 비교적 간단합니다.
- 단점: 연결 리스트에서는 특정 인덱스에 접근할 때 선형 탐색이 필요하므로, 인덱스가 매우 클 경우 성능이 저하될 수 있습니다.

아이디어 2: 다형성 활용
- 장점: 'Shape'라는 기본 클래스를 정의하고, 이를 상속받은 여러 도형 클래스들('Line', 'Circle', 'Rect')을 사용함으로써 다형성을 구현하여 코드의 확장성과 유연성을 높일 수 있습니다. 새로운 도형을 추가할 때 기존 코드 변경 없이 새로운 클래스만 추가하면 되므로 유지보수가 용이합니다.
- 단점: 상속과 다형성의 사용으로 인해 코드가 복잡해질 수 있으며, 특히 자식 클래스에서 부모 클래스의 메소드를 정확히 오버라이딩하지 않으면 예기치 않은 동작을 할 수 있습니다.

4. 문제를 해결한 키 아이디어 및 알고리즘 설명

주요 알고리즘:

1. 도형 추가:
   - 사용자가 도형을 추가할 때, 선택한 도형('Line', 'Circle', 'Rect')에 해당하는 객체를 생성하고 이를 리스트에 추가합니다. 리스트가 비어 있으면 새로 생성된 도형이 첫 번째 도형이 되고, 그렇지 않으면 기존 리스트의 끝에 추가됩니다.

2. 도형 삭제:
   - 사용자가 삭제할 인덱스를 입력하면, 해당 인덱스에 해당하는 도형을 리스트에서 찾아 삭제합니다. 삭제 시, 해당 도형을 이전 도형과 연결하거나, 첫 번째 도형을 삭제하는 경우에는 'pStart' 포인터를 갱신합니다.

3. 도형 출력:
   - 모든 도형을 출력할 때는 'pStart'부터 시작하여 연결된 각 도형을 출력합니다. 각 도형은 'paint' 메소드를 통해 그려집니다. 'paint' 메소드는 내부적으로 'draw' 메소드를 호출하여 도형을 실제로 그립니다.

구현 세부사항:

- 연결 리스트 처리:
  'Shape::add()' 메소드는 'next' 포인터를 통해 새로운 도형을 리스트에 추가하는 역할을 합니다. 삭제 시에도 연결 리스트 특성에 맞게 이전 도형과 다음 도형을 올바르게 연결해야 합니다.

- 사용자 인터페이스:

‘UI’ 클래스는 사용자로부터 명령을 입력받고, 이를 처리하는 로직을 제공합니다. 사용자 인터페이스는 1, 2, 3, 4 번호로 메뉴를 제공하며, 각 메뉴에 따라 도형을 추가하거나 삭제하는 기능을 호출합니다.

<u>알고리즘 흐름:</u>

1. 도형 추가:
   - 사용자가 도형 종류를 선택하면 해당 도형 객체가 생성되고, ‘pStart’에 추가됩니다.
2. 도형 삭제:
   - 사용자가 삭제할 인덱스를 입력하면, 연결 리스트에서 해당 인덱스를 찾아 삭제합니다.
3. 도형 보기:
   - 모든 도형을 출력할 때, ‘pStart’부터 시작하여 연결된 각 도형을 순차적으로 출력합니다.
4. 종료:
   - 프로그램 종료 시, 동적으로 할당된 메모리(‘new’로 생성된 객체)는 ‘delete’를 통해 해제합니다.