문제 정의와

저번 주에 풀었던 문제와 동일한 문제이지만 이 문제는 벡터를 사용하여 문제를 풀어야 합니다. 사용자는 삽입 메뉴에서 선(Line), 원(Circle), 사각형(Rect) 중 하나를 선택하여 리스트에 삽입할 수 있어야 합니다. 삽입된 도형은 Shape라는 기본 클래스를 기반으로 만들어지고, 벡터(std::vector)에 추가됩니다. 각각의 삽입, 삭제, 도형보기, 종료 기능의 정의는 저번주와 같기 때문에 간단하게 설명하자면 Shape는 추상 클래스이며 상속받은 파생 클래스에서 draw() 가상 함수를 이용하여 오버라이드하여 각 도형의 출력 동작을 정의합니다. 저번 주에는 연결 리스트를 이용하여 데이터를 관리하였습니다. 그러나 이번 주는 벡터를 사용해 관리하도록 하였습니다.

문제 해결 방법 및 알고리즘 설명

이 프로그램에서 추상 클래스 Shape는 도형의 공통적인 동작을 정의합니다. Shape 클래스는 paint()는 공통 인터페이스로 사용되고 내부적으로 draw() 메서드를 호출해 도형을 그립니다. draw() 메서드는 가상 함수로 선언되어 있어 반드시 이를 상속받은 자식 클래스에서 구현해야 합니다.
도형 클래스인 Circle, Rect, Line은 Shape 클래스를 상속받아 구현되며 각각의 draw() 메서드는 해당 도형을 출력하는 동작을 정의합니다. paint() 메서드를 통해 도형을 출력할 수 있습니다. 새로운 도형을 추가할 때 Shape 클래스를 상속받아 새로운 도형 클래스를 만들고 draw() 메서드를 구현합니다.
UI 클래스는 사용자와 상호작용을 처리하며, 입력과 출력을 담당합니다. 사용자로부터 메뉴 선택, 삽입할 도형 선택, 삭제할 도형의 인덱스 입력 등을 처리하는 메서드를 제공합니다. 메뉴 선택에 따라 삽입, 삭제, 출력, 종료와 같은 작업을 수행하며, 잘못된 입력이 들어올 경우 오류 메시지를 출력합니다.

GraphicEditor 클래스는 도형 객체를 관리하는 클래스입니다. 이 클래스는 vector<Shape*>를 사용하여 동적으로 도형을 관리하며, 삽입, 삭제, 출력의 기능을 수행할 수 있습니다. 도형 삽입 시 동적으로 메모리를 할당하고 삭제 시 해당 도형의 메모리를 해제하여 메모리 누수를 방지합니다. 삽입된 도형은 벡터에 저장되며 이를 통해 도형을 추가하거나 삭제하는 작업이 이루어집니다. 도형을 출력할 때는 벡터를 돌며 각 도형의 paint() 메서드를 호출합니다.
동적 메모리 관리를 통해 필요할 때만 메모리를 사용하고 삭제 시에는 메모리를 해제하여 동작할 수 있습니다. 이 프로그램은 추상 클래스와 다형성, 동적 메모리 관리를 활용하여 Shape 클래스를 기반으로 하여 벡터를 활용한 동적 관리 방식을 사용했습니다.