문제 정의

사용자로부터 선택받은 도형을 프로그램에 추가/삭제하고
보여줄 수 있어야한다. 세 도형 클래스는 Shape 클래스를 상속받고 GraphicEditor 클래스는 동적으로 생성하여 도형이 늘어나도 오류가 발생하지 않도록 한다. pStart와 pLast 포인터를 사용해야하므로 도형들은 Shape* next 포인터를 이용한 연결 리스트를 사용한다.

문제 해결 방법

도형을 Shape라는 추상 클래스로 정의하고 각 세 도형(Line, Circle, Rectangle) 클래스는 Shape 클래스를 상속받는다. 각 도형의 draw() 메서드를 다 다르게 구현하여, Shape* 포인터로 여러 도형을 처리할 수 있게한다. 이때 다형성을 활용하여 paint() 메서드가 호출되면, 각 도형의 draw()가 실행된다. 삽입/삭제 한 도형을 관리하는 것은 GraphicEditor 클래스에서 수행하게 하고 연결 리스트를 사용하여 각 도형(동적으로 생성) 객체를 연결시켜주고, 삽입/삭제할 때 효율적일 수 있도록 한다.

아이디어 수행 평가/결과

가장 먼저 Shape를 추상 클래스로, shape 포인터로 각 도형을 처리할 수 있도록 하였다. 주요한 문제인 동적 도형 생성과 연결 리스트 관리인데 GraphicEditor 클래스 자체를 동적으로 생성하고 pStart와 pLAst 포인터로 리스트의 첫, 끝 도형을 가리킨다. 삽입할 경우에는 pLast 뒤에 새로운 도형을 연결하고 pLast가 새롭게 들어온 도형을 가리키게 만든다. 삭제할 경우에는 삭제할 도형을 바로 삭제하는 것이 아닌 이전 도형을 찾아 next 포인터를 삭제할 도형의 next 포인터로 연결하여 잘못된 포인터 접근을 막아준다.

알고리즘 설명

사용자가 메뉴에서 삽입을 선택하면, insertItem(int type) 함수가 호출된다. 그리고 도형의 타입을 입력 받는데 타입값에 따라 Line, Circle, Rectangle 중 하나의 객체를 동적으로 생성하고 생성된 객체는 연결 리스트에 삽입이 된다. 첫번째 도형인 경우 pStart와 pLast를 새로 생성된 도형 객체로 설정한다. 이때 리스트의 시작과 끝을 모두 해당 객체로 설정한다.
도형 삭제는 deleteItem(int index) 함수가 호출되어 삭제할 도형이 첫 번째일 경우 pStart를 두 번째 도형으로 갱신하고, 삭제된 도형은 메모리에서 해제한다. 삭제할 도형이 첫 번째 도형이 아닌 경우, 삭제할 도형의 이전 도형(pre)을 찾고
이전 도형(pre)의 next 포인터를 삭제할 도형의 next 포인터로 연결하여, 삭제할 도형을 연결 리스트에서 제거한 후 삭제된 도형은 메모리에서 해제한다.
모두보기는 show() 함수가 호출되어 pStart부터 시작해서 각 도형을 순차적으로 출력한다. 각 도형은 paint() 메서드를 호출하여 그 도형의 이름을 출력하고 순차적으로 getNext()를 호출하여 다음 도형으로 넘어간다. 그리고 각 도형은 paint() 메서드를 통해 이름을 출력하는데, 이 메서드는 각 도형의 draw() 메서드를 호출하여 도형의 이름을 출력하게 된다.