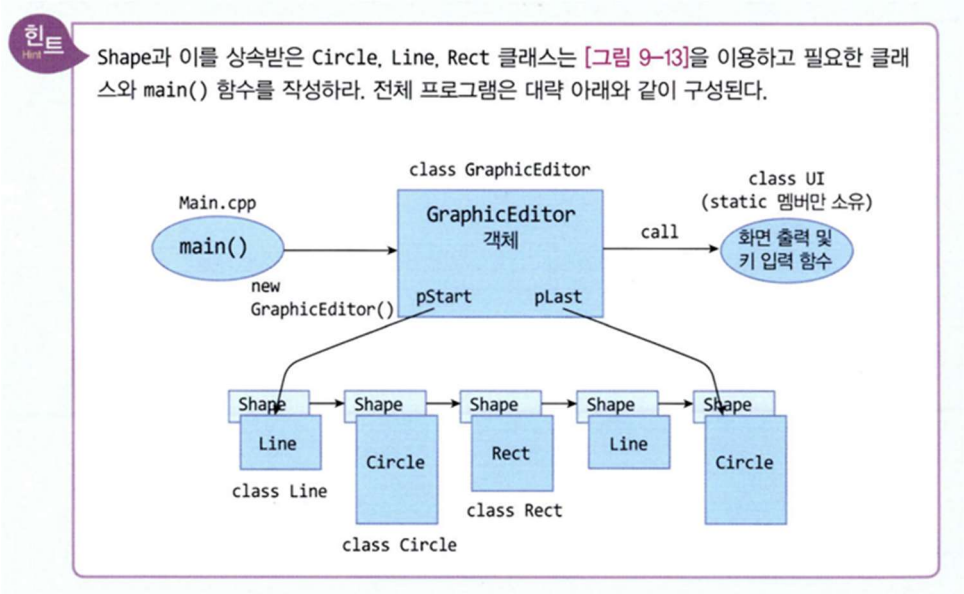
과제#5

1. 문제 정의

주어진 문제는 **간단한 그래픽 편집기**를 콘솔 기반에서 구현하는 것입니다. 그래픽 편집기의 주요 기능은 도형(선, 원, 사각형)을 삽입하고, 삭제하며, 현재 삽입된 도형들을 모두 볼 수 있도록 하는 것입니다.

힌트:



주요 요구 사항>
삽입: 도형을 삽입하는 기능 (선, 원, 사각형)
삭제: 삽입된 도형을 삭제하는 기능
모두보기: 현재 저장된 모든 도형을 출력하는 기능
종료: 프로그램을 종료하는 기능

위에 힌트와 교재 [그림 9-13]의 코드를 활용하여 동적으로 도형을 추가하거나 삭제하는 방법을 사용하는 것입니다.

2. 문제 해결 방법

삽입 함수>

- **아이디어: 삽입 함수는 기본적으로 [그림9-13]을 활용한 연결 리스트에서 새 도형을 삽입하는 방식으로 구현됩니다..**
- **[그림9-13]을 활용한 객체 생성:**
    - **삽입 시 첫 번째 도형을 삽입하는 경우와 이후 도형을 삽입하는 경우를 구분하여 처리합니다.**
    - **처음 도형이 삽입될 때는 pStart와 pLast 포인터가 동시에 새로 생성된 도형을 가리키게 됩니다. 이후 삽입되는 도형은 pLast->add() 메서드를 통해 현재 도형에 이어서 삽입됩니다. 이때 pLast는 새로 추가된 도형을 가리키게 되어 리스트의 끝을 관리합니다.**

```
public:
    GraphicEditor() { count = 0; }
    void getGraphic(int n) {//원하는 객체 생성
        if (count == 0) {//첫번째 객체 생성
            if (n == 1) {
                pStart = new Line();
                pLast = pStart;
                count++;
            }
            else if (n == 2) {
                pStart = new Circle();
                pLast = pStart;
                count++;
            }
            else if (n == 3) {
                pStart = new Rect();
                pLast = pStart;
                count++;
            }
        }
        else
        {//첫번째 이후에 객체 생성
            if (n == 1) {
                pLast = pLast->add(new Line());
                count++;
            }
            else if (n == 2) {
                pLast = pLast->add(new Circle());
                count++;
            }
            else if (n == 3) {
                pLast = pLast->add(new Rect());
                count++;
            }
        }
    }
```

삭제 함수>

- **아이디어: 삭제 함수는 주어진 인덱스를 기준으로 도형을 찾아 삭제하는 방식입니다. 그러나 도형이 삭제된 후, 리스트에서 그 위치가 잘리면서 나머지 도형들이 잘못 연결되는 문제를 해결해야 합니다. 따라서 삭제 후 연결을 처리하는 과정이 필요합니다.**
- **[그림9-13]에서처럼 객체를 삭제:**
    - **삭제할 도형을 찾은 후에는 그 도형을 메모리에서 해제합니다. 중요한 점은, 삭제 후 남은 도형들을 연결하는 작업입니다. 그렇지 않으면, 삭제된 도형 이후의 리스트가 끊어지게 되어, 나중에 모두보기 함수에서 삭제된 도형 이후의 도형들이 출력되지 않습니다.**
    - **이를 해결하기 위해 첫 번째 도형을 삭제한 경우와 중간의 도형을 삭제한 경우를 다르게 처리해야 합니다. 첫 번째 도형이 삭제되면 pStart를 갱신하고, 중간 도형이 삭제되면 이전 도형의 next 포인터를 삭제된 도형의 다음 도형으로 변경해야 합니다**

```
void delGraphic(int n) {
    p = pStart;
    int i = 0;
    Shape* prev = NULL;  // 이전 객체를 추적하는 포인터

    while (p != NULL) {
        if (i == n) {
            Shape* q = p->getNext();  // 다음 객체를 미리 추적
            delete p;  // 현재 객체 삭제

            //객체 리스트에서 삭제되었을때 나머지들을 연결하는 작업
            if (prev != NULL) {//리스트 중간에 것을 삭제한 경우
                prev->getShape(q);
                // 중간에 객체를 삭제하고 다음에 객채가 저장된 q를
                // prev에 저장
                // prev = p처럼 다음 객체를 이전으로 설정하여 객체 리스트를 연결함
            }
            else {
                pStart = q;  // 첫 번째 객체를 삭제한 경우 pStart를 갱신
            }

            break;  // 삭제 후 루프 종료
        }
        i++;
        prev = p;  // 이전 객체를 갱신, 지금의 객체를 이전 객체로 변경
        p = p->getNext();  // 다음 객체를 가리킴
    }
```

모두보기 함수>

- 아이디어: 모두보기 함수는 현재 리스트에 있는 모든 도형을 출력하는 기능입니다. 연결 리스트에서 각 도형을 순차적으로 출력해야 하므로, 리스트의 첫 번째 도형부터 시작하여 끝까지 출력합니다.
- [그림9-13]에 있는 전체 보여주는 함수 활용:
    - [그림9-13]에 있는 전체 보여주는 함수 활용하여 구현한다. 거기에, 도형의 번호를 출력하기 위해 int i = 0;을 추가하고 while문에서 각 도형을 출력할 때마다 i를 증가시켜 번호를 함께 출력합니다.

```
void show() {
    p = pStart;
    int i = 0;
    while (p != NULL) {
        cout << i << " : ";
        p->paint();
        p = p->getNext();
        i++;
    }
}
```

3. 아이디어 평가

Shape, Line, Circle, Rect 클래스는 GraphicEditor에서 삽입/삭제 작업을 관리할 수 있도록 설계되었습니다. GraphicEditor 클래스는 도형을 삽입할 때, Shape 클래스의 add 메소드를 이용해 새 도형을 리스트에 추가하고, 삭제 시 getShape를 이용해 연결을 변경합니다. UI 클래스는 사용자와 상호작용하며, 메뉴에 따라 기능을 호출합니다.

이것으로 도형들의 순차적인 삽입과 삭제를 효율적으로 처리할 수 있습니다. 또한 각 도형을 독립적으로 설계하여 코드의 재사용성과 확장성을 높일 수 있습니다.

4. 문제를 해결한 키 아이디어 또는 알고리즘 설명

알고리즘 정리>>

삽입:

- **첫 번째 도형이 삽입되면 pStart와 pLast를 새 도형으로 설정.**
- **두 번째 이후 도형은 pLast->add() 메서드를 통해 추가.**

삭제:

- **삭제할 도형을 찾고 해당 도형을 삭제.**
- **삭제 후 이전 도형과 이후 도형을 연결하여 리스트가 끊어지지 않도록 처리.**
- **삭제가 첫 번째 도형일 경우 pStart를 갱신하고, 중간 도형일 경우 이전 도형의 next 포인터를 업데이트.**

모두보기:

- **연결 리스트의 모든 도형을 순차적으로 출력.**
- **각 도형에 번호를 붙여 출력.**