– 문제 정의 :

'virtual void get()'이라는 함수를 가진 기본 클래스 'Shape'를 활용하여 파생 클래스의 'Line', 'Rect', 'Circle'에 있는 'void get()'으로 동적 바인딩에 의해 오버라이딩(재정의)을 하면서 함수를 호출한다.

GraphicEditor 클래스에는 삽입(insert), 삭제(remove), 모두 보기(show), 종료(break)를 구현하여, main함수에서 이를 호출하면서 실행한다.

※본래는 연결 리스트(Linked List)의 개념을 활용하여 객체를 연결하려고 하여 구현해야 하지만, 해당 자료구조에 대한 이해도 부족으로 구현하지는 못하였다.

– 문제 해결 방법 : 문제를 해결하기 위한 아이디어들

문제 해결을 위해서 중요한 관건은 가상 함수와 동적 바인딩의 개념보다는 연결 리스트를 사용하지 않는 우회하는 방안 내에서 삽입과 삭제를 구현해내는 것이 관건이었다.

첫 번째 아이디어 : while문과 if문을 활용한 main()함수 구현

main 함수는 while(1)로 무한 루프를 돌리고,
'cout<<삽입:1, 삭제:2, 모두보기:3, 종료:4 >> ";' 호출하고, cin을 통해 값을 받아서
그 값이 if문을 통해서 1이면 insert를 호출하고, 2면 remove를 호출하고, 3이면 show를 호출하고, 4면 break를 해서 전체적인 while문의 틀을 잡았다.

두 번째 아이디어 : 구현부를 위한 빌드업

구현부를 짜기 위해서 메인 함수에서 적절한 Index값을 넘길 필요가 있었다.
삽입 부분에서는 shapeIndex를 통해서 도형이 무엇인지를 확인하고 구현부로 넘겼다.
: insert(shapeIndex)
삭제 부분에서도 어떤 index를 삭제할 것인지 확인하고 구현부로 넘겼다.
: remove(index)

세 번째 아이디어 : Shape* p를 배열로 선언

p를 배열로 선언하고 배열 인덱스를 size로 선언해서,
배열의 크기를 늘리고, 줄이고, 삭제하고 생성을 할 수 있게 세팅하였다.

네 번째 아이디어 : 삽입과 삭제부분

삽입과 삭제 부분이 가장 관건이었다.

삽입 부분은 우선 *newShape를 생성하고,
메인함수에서 받아온 shapeIndex값을 if문으로 받아서 1, 2, 3일때
각각 new를 통해서 Line, Rect, Circle를 생성하였다.

또한, 이 newShape는 p[size]에서 size ++ 해서 p 배열에 삽입하였다.

삭제 부분도 이와 비슷한 원리로 구현하였다.
우선 index에 해당하는 배열 속 도형을 삭제하고,
for문을 통해서 비어있는 그 다음 인덱스의 값들을 1칸씩 당겨 온다.

그리고 한 칸씩 당겼으면, 마지막 size를 -1를 해서 size를 알맞게 설정한다.

다섯 번째 아이디어 : 가상함수를 이용한 show

Line, Rect, Circle에는 show()함수를 위해서 미리 'void get()'이라는 함수에 각각의 이름을 출력하였다.

for문을 통해서 "i : get()" 형식으로 출력하였다.

– 아이디어 평가 : 문제 해결을 위해 제시한 아이디어들 수행 평가와 결과

첫 번째 아이디어인 **'while문과 if문을 통한 main()함수 구현'**은 어렵지 않게 생각해낼 수 있었다. 두 번째 아이디어인 **'구현부를 위한 빌드업'**도 자연스럽게 작성할 수 있었다.

이 문제에서 가장 핵심 부분을 꼽으라고 하면 **세 번째와 네 번째 아이디어**이다.
해당 아이디어를 3일동안 고민하였으며, 결국 Gpt와 함께 해결하였다는 점이 아쉬웠다.

처음에 insert를 예제 문제의 'Shape* add(Shape* p)'처럼 구현하려고 하였다. 잘되지 않았고, pLast와 pStart까지 쓰려고 하니까 생각하는 것이 복잡하였다.
또한, 예제처럼 next를 통해서 객체를 받고 연결 리스트를 통해서 잇고 싶었다. 지금 newShape랑 비슷한 것 같지만, 아직도 해당 개념에 대한 이해가 완벽히 되지 못한 상태인 것이 아쉬웠다.

삭제를 하는 것이 연결 리스트의 꽃 파트라고 생각하는데, 삭제한 부분을 비우고 해당 pLast가 뭘 설정하고, 다음 next는 뭐고, 이러한 것을 설정하는 것이 복잡하였다.
따라서 배열로 한칸씩 당기는 방법을 (Gpt와 함께)고안하였다.

다섯 번째 아이디어인 get은 생각한대로 잘 구현이 되었다.

- 문제를 해결한 키 아이디어 또는 알고리즘 설명

문제를 해결한 키는 단연코 삽입 삭제 부분이다.

```
void insert(int shapeIndex) {
    Shape* newShape = NULL;

    if (shapeIndex == 1) newShape = new Line();
    else if (shapeIndex == 2) newShape = new Rect();
    else if (shapeIndex == 3) newShape = new Circle();

    if (newShape != NULL) {
        p[size++] = newShape; // 배열에 추가
    }
}
```

삽입 부분을 보면, newShape 객체가 예제의 next 역할을 하고,
newShape에 객체를 입력받고,
size++ 하면서 공간을 늘리고, newShape를 p에 넣는다.

```
void remove(int index) {
    delete p[index];

    for (int i = index; i < size - 1; i++) {
        p[i] = p[i + 1];
    }

    p[size - 1] = NULL; // 마지막 자리를 초기화
    --size;
}
```

remove는 해당 index p값을 제거하고(new로 생성했던 도형들)
그후 for루프 size-1까지 값들을 한칸씩 앞으로 당긴다.
그리고 마지막에 당겨서 쓰레기 값을 NULL로하고,
도형이 담긴 size를 마이너스 한다.