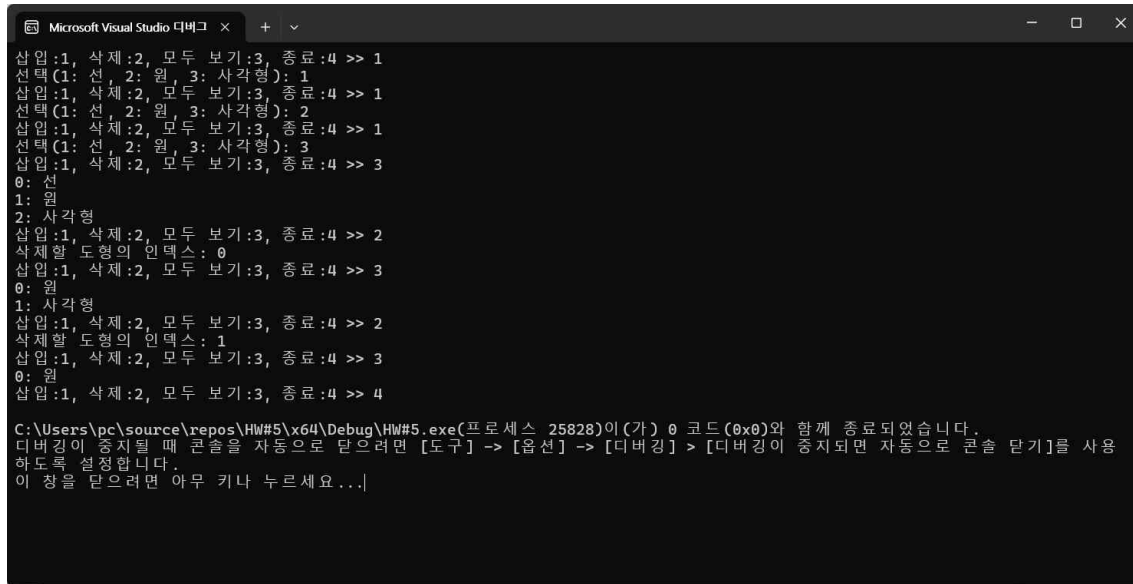


1) 소스코드 수행결과 화면 캡처



```
Microsoft Visual Studio 디버그 콘솔
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 1
선택(1: 선, 2: 원, 3: 사각형): 1
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 1
선택(1: 선, 2: 원, 3: 사각형): 2
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 1
선택(1: 선, 2: 원, 3: 사각형): 3
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 3
0: 선
1: 원
2: 사각형
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 2
삭제할 도형의 인덱스: 0
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 3
0: 원
1: 사각형
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 2
삭제할 도형의 인덱스: 1
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 3
0: 원
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 4
C:\Users\pc\source\repos\HW#5\x64\Debug\HW#5.exe(프로세스 25828)이(가) 0 코드(0x0)와 함께 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요 ...
```

2) 소스 구현 설명

2-1) 문제 정의

간단한 그래픽 편집기를 설계하여 도형(선, 원, 사각형)을 삽입, 삭제, 조회할 수 있는 프로그램을 구현한다. 사용자는 콘솔에서 명령어를 입력하여 프로그램과 상호작용하며, 입력된 도형들을 동적으로 관리해야 한다.

요구사항

사용자는 Line, Circle, Rect를 삽입할 수 있어야 한다.

삽입된 도형은 삭제가 가능해야 한다.

전체 도형을 목록으로 볼 수 있어야 한다.

프로그램은 명령에 따라 종료되어야 한다.

제약 조건

객체 지향 프로그래밍(OOP)을 기반으로 클래스를 설계한다.

동적 메모리를 활용하여 유연한 도형 관리를 지원한다.

삭제 시 메모리 누수를 방지해야 한다.

2-2) 문제 해결 방법

1. 추상 클래스 설계

모든 도형의 공통된 속성 및 기능을 제공하기 위해 Shape 추상 클래스를 설계했다. 추상 클래스는 인터페이스 역할을 하며, 자식 클래스(Line, Circle, Rect)는 이를 기반으로 구체적인 동작을 구현한다.

2. 상속을 통한 다형성

Shape를 상속받는 Line, Circle, Rect 클래스는 각각 draw() 메서드를 오버라이딩하여 자신만의 출력 형식을 제공한다.

3. 도형 관리 클래스

GraphicEditor 클래스를 설계하여 도형의 삽입, 삭제, 조회 기능을 관리한다. 벡터를 사용하여 동적으로 도형 객체를 저장하고 관리한다. 사용자가 입력한 명령어에 따라 적절한 작업을 수행하도록 구현한다.

4. 메모리 관리

삽입 시 동적 메모리 할당(new), 삭제 시 메모리 해제(delete)를 통해 동적 객체 관리의 안정성을 유지한다.

2-3) 아이디어 평가

1. 도형 관리의 유연성

도형을 삽입, 삭제, 조회하는 과정에서 벡터를 사용하여 도형의 순서를 동적으로 관리함으로써 구현의 단순성과 유연성을 확보했다. 다형성을 통해 Shape 포인터를 사용하여 다양한 도형 객체를 동일한 방식으로 처리할 수 있었다.

2. 코드 확장성

새로운 도형 클래스를 추가하려면 Shape를 상속받아 draw()를 구현하면 된다. 기존 코드를 수정하지 않아도 새로운 기능을 추가할 수 있는 구조를 가졌다.

3. 메모리 안정성

동적으로 할당된 메모리를 delete와 소멸자를 통해 적절히 관리하여 메모리 누수를 방지했다.

2-4) 문제를 해결한 키 아이디어 또는 알고리즘 설명

1. 추상 클래스와 다형성의 활용

Shape 추상 클래스의 draw 메서드는 도형의 출력 형식을 지정하기 위한 인터페이스 역할을 수행하며, 이를 상속받은 Line, Circle, Rect 클래스는 각각 자신만의 구체적인 출력 방식을 제공한다. Shape 포인터를 통해 다양한 도형 객체를 동일한 방식으로 처리할 수 있었다.

2. 동적 메모리와 벡터 활용

도형 객체를 동적으로 생성하여 벡터에 저장함으로써 삽입과 삭제가 유연하게 이루어진다. 사용자가 삭제 명령을 내릴 때, 특정 인덱스에 위치한 도형 객체를 제거하고 메모리를 해제하여 자원 누수를 방지한다.

3. 명령어 기반의 사용자 입력 처리

사용자가 선택한 명령(삽입, 삭제, 조회, 종료)에 따라 적절한 메서드를 호출한다. 잘못된 입력에 대해 적절한 에러 메시지를 출력하여 프로그램의 안정성을 유지한다.

4. 소멸자를 통한 메모리 관리

GraphicEditor의 소멸자에서 벡터에 저장된 모든 도형 객체를 순회하며 메모리를 해제함으로써 프로그램 종료 시 동적 메모리 누수를 방지한다.