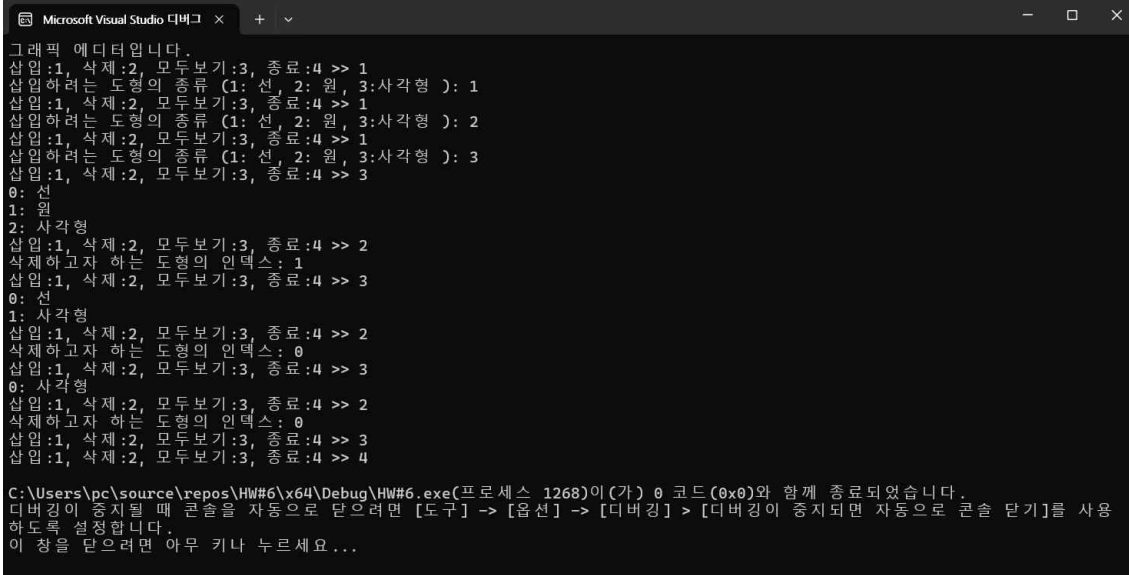


1) 소스코드 수행결과 화면 캡처



```
Microsoft Visual Studio 디버그
그래픽 에디터입니다.
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 1
삽입하려는 도형의 종류 (1: 선, 2: 원, 3:사각형 ): 1
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 1
삽입하려는 도형의 종류 (1: 선, 2: 원, 3:사각형 ): 2
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 1
삽입하려는 도형의 종류 (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
삭제하고자 하는 도형의 인덱스: 1
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 3
0: 선
1: 사각형
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 2
삭제하고자 하는 도형의 인덱스: 0
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 3
0: 사각형
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 2
삭제하고자 하는 도형의 인덱스: 0
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 3
삽입:1, 삭제:2, 모두 보기:3, 종료:4 >> 4
C:\Users\pc\source\repos\HW#6\HW#6.exe(프로세스 1268)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

문제는 그래픽 에디터 프로그램을 구현하는 것이다. 이 프로그램은 다음의 기능을 가져야 한다.

삽입 : 사용자가 입력한 도형(Line, Circle, Rectangle)을 메모리에 동적으로 생성하여 관리한다.

삭제 : 사용자가 입력한 특정 인덱스의 도형을 제거하고 메모리를 해제한다.

모두 보기 : 현재 저장된 모든 도형을 출력한다.

종료 : 프로그램을 종료하기 전, 동적으로 생성된 모든 도형의 메모리를 해제한다.

2-2) 문제 해결 방법

문제를 해결하기 위해 다음의 접근 방식을 채택했다.

1. 클래스 계층 구조 설계

Shape 클래스를 기반으로, Line, Circle, Rect와 같은 세부 클래스를 구현한다.

Shape 클래스는 추상 클래스이며, draw()라는 순수 가상 함수를 정의한다. 이를 통해 세부 클래스마다 고유한 동작을 구현한다.

2. 다형성과 동적 바인딩 활용

Shape 클래스의 포인터를 std::vector에 저장하여, 동적 바인딩으로 다양한 도형 객체를 관리한다.

도형 출력 시에는 Shape 클래스의 paint() 메서드를 호출하여, 실제 객체의 draw() 메서드를 실행하도록 한다.

3. 메모리 관리

삽입 시 new 연산자로 객체를 동적으로 생성한다.

삭제 시 delete를 통해 메모리를 해제하고, 벡터에서 해당 포인터를 제거한다.

종료 시 모든 남아 있는 객체의 메모리를 해제하여 메모리 누수를 방지한다.

4. 사용자 인터페이스 설계

사용자 입력을 기반으로 삽입, 삭제, 조회, 종료의 4가지 주요 동작을 처리한다.

입력 오류(예: 잘못된 인덱스 입력)에 대해 적절한 예외 처리를 제공한다.

2-3) 아이디어 평가

1. std::vector<Shape*>를 활용한 동적 객체 관리

std::vector를 사용하여 도형 객체를 관리하는 방법은 삽입과 삭제가 간단하며 다형성을 통해 다양한 도형을 처리할 수 있다는 장점이 있다. 하지만 메모리 누수를 방지하기 위한 철저한 메모리 관리가 필요하다.

2. 추상 클래스를 이용한 다형성 구현

Shape를 추상 클래스로 설계해 공통 인터페이스를 제공하며, 각 도형 클래스에서 고유 동작을 구현할 수 있다. 이는 코드의 확장성과 유지보수성을 높이는 장점이 있지만, 객체 생성과 메서드 구현에 추가 작업이 필요하다.

3. 동적 메모리 관리

프로그램 종료 시 동적으로 생성된 모든 객체를 삭제하여 메모리 누수를 방지하는 것은 필수적이다. 이는 안정적인 프로그램 실행을 보장하기 위한 중요한 요소이다.

2-4) 문제를 해결한 키 아이디어 또는 알고리즘 설명

1. 다형성과 추상 클래스

Shape 클래스는 순수 가상 함수 draw()를 포함한 추상 클래스다. 이를 상속받은 Line, Circle, Rect 클래스는 각 도형에 맞는 draw()를 재정의하고 paint() 메서드는 draw()를 호출하며, 다형성(polymorphism)을 통해 각 도형 클래스의 draw()가 실행된다.

2. 벡터를 이용한 객체 관리

vector<Shape*> shapes는 Shape 객체의 포인터를 저장한다. 동적으로 생성된 Shape 객체는 삽입 시 push_back()으로 추가되고, 삭제 시에는 인덱스를 통해 객체를 제거하고, 메모리를 해제한 뒤 벡터에서 삭제한다.

3) 사용자 입력 기반 동작

삽입 : 사용자 입력에 따라 Line, Circle, Rect 객체를 생성하고 벡터에 추가한다.

삭제 : 사용자로부터 입력받은 인덱스를 확인하고, 해당 도형을 삭제 및 메모리 해제한다.

모두 보기 : 벡터에 저장된 모든 도형의 paint()를 호출하여 출력한다.

종료 : 벡터를 순회하며 동적으로 생성된 모든 객체를 삭제한 후 프로그램을 종료한다.

4) 메모리 관리

모든 삽입된 객체는 프로그램 종료 시 반드시 삭제하여 메모리 누수를 방지한다. 이는 delete 연산과 벡터의 clear()를 사용하여 구현한다.