202304105 ICT융합공학부 이상은

문제 정의

이 프로그램의 목표는 콘솔 기반의 간단한 그래픽 편집기를 만드는 것이다. 사용자 입력을 통해 도형을 삽입, 삭제하고, 현재 삽입된 모든 도형을 확인할 수 있다.

1. **삽입**: 선, 원, 사각형과 같은 도형을 추가할 수 있어야 한다.
2. **삭제**: 특정 도형을 인덱스를 통해 삭제할 수 있어야 한다.
3. **모두보기**: 현재 저장된 도형의 목록을 표시할 수 있어야 한다.
4. **종료**: 사용자가 프로그램을 종료할 수 있어야 한다.

문제 해결 방법

1. **객체 지향 프로그래밍(OOP) 활용**:

도형을 **클래스**로 정의하고, 공통 인터페이스를 사용하여 각 도형을 개별적으로 관리한다. 이를 통해 도형 관리가 직관적이고 확장 가능해진다.

   a. `Shape` 클래스를 추상 클래스(인터페이스)로 정의하고, 이를 상속받는 `Line`, `Circle`, `Rectangle` 클래스를 생성했다.

2. **STL `vector` 활용**:

도형 목록은 가변적인 크기를 가지므로 동적 배열과 같은 동작을 지원하는 `vector`를 사용하여 저장하고 관리했다.

3. **사용자 인터페이스(UI)**:

   a. 콘솔 기반 메뉴 시스템을 사용하여 사용자가 각 기능(삽입, 삭제, 모두보기, 종료)을 선택할 수 있게 했다.
   b. 입력 유효성 검사를 통해 잘못된 입력을 처리했다.

4. **메모리 관리**:

   a. 도형 객체를 동적으로 생성하고 `vector`에 저장하므로, 프로그램 종료 시 동적 메모리를 해제한다.

아이디어 평가

b. 객체 지향 설계

a. **장점**: 도형 클래스들을 독립적으로 관리할 수 있으므로 유지보수가 쉽고, 새로운 도형을 추가할 때 최소한의 변경으로 확장이 가능하다.
b. **단점**: 단순한 프로그램이지만 클래스 계층 구조를 구성해야 하므로 약간의 복잡도가 추가된다.

c. STL vector 사용

a. **장점**: 삽입과 삭제가 간단하고, 자동으로 크기를 관리하므로 효율적이다.
b. **단점**: 삭제 시 도형 객체의 메모리를 수동으로 해제해야 하는 부담이 있다.

d. 콘솔 기반 UI

a. **장점**: 사용자와의 상호작용이 간단하며, 실시간으로 명령어를 수행할 수 있다.
b. **단점**: 콘솔 UI는 그래픽적 표현이 어렵고, 실제 그래픽 편집기와는 차이가 있다.

문제를 해결한 키 아이디어 또는 알고리즘 설명

1. 객체 지향 설계 (Shape 인터페이스 및 상속 구조)

a. `Shape` 추상 클래스는 `getName`이라는 순수 가상 함수를 정의하여 모든 도형에서 이름을 반환하도록 했다.
b. `Line`, `Circle`, `Rectangle` 클래스는 각각 `Shape`를 상속받아 `getName`을 구현하여 각 도형의 이름을 반환한다. 이를 통해 다양한 도형 타입을 `Shape*` 형태로 통합 관리할 수 있게 했다.

2. STL vector를 활용한 도형 저장 및 관리

a. 삽입: 사용자 입력에 따라 적절한 도형 객체를 동적으로 생성한 뒤, 이를 `vector`에 추가했다.
b. 삭제: 사용자 입력으로 받은 인덱스를 기준으로 `vector`에서 해당 도형

객체를 삭제하고, 메모리를 해제했다.

c. 모두보기: vector를 순회하면서 저장된 도형들의 이름과 인덱스를 출력한다.

3. 콘솔 메뉴 시스템

a. 주요 명령(삽입, 삭제, 모두보기, 종료)을 반복문으로 구현하여 사용자가 원하는 작업을 수행할 수 있도록 한다.

b. 메뉴 선택과 입력 처리 로직을 분리하여 가독성을 높였다.

4. 메모리 관리

a. 동적으로 생성된 도형 객체들은 프로그램 종료 시 소멸자를 통해 모두 해제하여 메모리 누수를 방지했다.

결론

이 설계는 간단하면서도 확장 가능한 방식으로 문제를 해결했다. 객체 지향 설계를 통해 도형 관리가 직관적이고, vector를 사용하여 삽입/삭제 작업을 쉽게 처리했다.