전공 : 컴퓨터공학과 학년 : 4 학번 : 20212020 이름 : 박민준

1. 주어진 문제 및 5-1의 문제 해결에 관한 내용을 이해하고 이 문제를 효율적으로 해결하기 위한 방법을 생각하여 이를 1쪽 이내로 요약하여 제출하시오.

이번 3주차 실습의 목적은 C++ 객체지향 프로그래밍의 개념 중 특히 ‘클래스 상속’을 활용하여 기초 클래스인 Array 클래스로부터 RangeArray 클래스를 구현하는 것이다. 클래스를 구현하기에 앞서 클래스, 상속, 캡슐화, 다형성과 관련된 개념은 다음과 같다.

1) 클래스

클래스는 한 마디로 객체를 생성하기 위한 템플릿이다. 객체지향 프로그래밍에서 클래스는 데이터와 데이터를 조작하는 함수를 하나의 단위로 묶는 역할을 한다. 클래스에는 두 가지 주요 구성 요소가 있는데, 속성과 메서드이다. 먼저 속성은 클래스에 속하는 데이터의 특성을 정의한다. 이러한 속성은 변수로 표현되며, 객체의 상태를 저장하는 데 사용된다. 그리고 메서드는 클래스가 수행할 수 있는 작업을 정의한다. 메서드는 클래스 내에 정의된 함수로서 클래스의 속성에 접근하고 조작할 수 있도록 도와주는 역할을 한다.

또한, 클래스를 정의함으로써 해당 클래스의 인스턴스를 생성할 수 있는데, 각 인스턴스는 해당 클래스의 구조를 따르지만 인스턴스마다 속성의 값이 다를 수 있기 때문에 각각 독립된 상태를 가지게 된다.

2) 상속

상속은 한 클래스의 속성과 메서드를 다른 클래스가 물려받아 사용할 수 있게 해주는 방식이다. 상속을 통해 기존 코드를 재사용하여 새로운 클래스를 생성할 수 있으며, 기존 클래스를 수정하지 않고도 새로운 기능을 추가할 수 있다. 이 때, 속성과 메서드를 물려주는 클래스를 부모 클래스라고 하며, 부모 클래스를 물려받는 클래스를 자식 클래스라고 한다. 클래스의 상속을 이용하게 되면 공통적인 속성과 메서드를 가진 여러 클래스에 대해 중복 코드를 줄일 수 있으며, 기존 클래스의 기능을 확장하거나 변경이 필요할 때 상속을 통해 새로운 클래스를 쉽게 만들 수 있다는 장점이 있다.

3) 캡슐화

캡슐화는 객체 내부 데이터와 해당 데이터를 조작하는 메서드를 하나로 묶는 것을 의미한다. 이를 통해 객체의 구체적인 구현 내용을 숨기고 객체 외부에서는 제공되는 인터페이스를 통해서만 객체와 상호작용할 수 있도록 한다. 캡슐화의 주요 목적은 정보 은닉과 인터페이스 제공이다. 정보를 은닉함으로써 객체의 내부 상태를 외부로부터 보호하며, 객체의 상태가 예상치 못한 방식으로 변경되는 것을 방지할 수 있다. 또한, 객체가 수행할 수 있는 연산을 명확하게 정의함으로써 객체 사용자는 객체가 어떻게 구현되었는지 정확히 알지 못해도 해당 객체를 사용할 수 있다. 캡슐화를 구현하기 위해서는 클래스 내부의 변수 속성을 private 또는 protected 접근 지정자로 선언하여 외부 접근으로부터 보호하고, 필요한 메서드를 public으로 선언하여 객체와의 상호 작용 방법을 제공하게 된다.

4) 다형성

다형성은 객체지향 프로그래밍에서 한 객체가 여러 타입의 인스턴스로서 행동할 수 있는 능력을 의미한다. 클래스의 다형성은 주로 상속과 인터페이스를 통해 구현되며, 같은 메서드 호출에 대해 다른 행동을 하게 한다. 가장 대표적인 특징으로 메서드 오버라이딩과 메서드 오버로딩이 있는데, 메서드 오버라이딩은 상속받은 자녀 클래스가 부모 클래스에서 정의된 메서드를 재정의하는 것을 말한다. 반면 메서드 오버로딩은 같은 이름의 메서드를 매개변수의 유형이나 개수를 다르게 하여 여러 버전으로 정의하는 것을 말한다. 클래스의 다형성은 공통 인터페이스를 사용함으로써 상속과 마찬가지로 여러 클래스에 대한 코드를 중복 없이 작성할 수 있으며, 기존 코드를 변경하지 않고 새로운 기능을 추가하거나 확장할 수 있다는 장점이 있다.

이러한 개념적인 내용을 토대로 5-1 문제 해결에 관한 내용은 다음과 같다.

Array 클래스는 정수 타입의 1차원 배열을 동적으로 생성하며 배열의 인덱스 접근을 위해 연산자 오버로딩을 제공한다. RangeArray 클래스는 Array 클래스의 자녀 클래스로서 모든 기능을 상속받으며, 추가적으로 생성자에서 지정된 임의의 범위로 인덱스가 시작되는 특징을 갖는다. 예를 들어, RangeArray A(-10, 10)은 인덱스가 -10에서 시작하여 10에서 끝나는 배열을 의미하며 총 21개의 원소를 저장할 수 있는 배열이다. RangeArray 클래스를 구현하는 과정에서 상속, 캡슐화, 다형성 등의 객체지향 프로그래밍 원리가 적용된다.

해당 문제를 효율적으로 해결하기 위해서는 먼저 Array 클래스를 구현해야 한다. Array 클래스는 배열의 생성, 접근, 소멸 등의 기본적인 기능을 제공하는데, 배열의 크기와 동적 할당된 배열 데이터를 관리해야 하며, 배열의 원소에 접근하거나 수정할 수 있는 연산자 오버로딩과 배열 내용을 출력하는 함수를 포함해야 한다. Array 클래스를 구현한 이후, RangeArray 클래스는 Array 클래스를 상속받아 추가적으로 배열의 시작과 끝 인덱스를 관리하는 범위 기반의 인덱스 접근 기능을 구현해주면 된다. Array 클래스의 기능을 RangeArray에서 확장하면서 상속과 다형성을 통해 코드의 재사용성과 유지보수성을 높일 수 있다.

2. 문제 해결을 위한 간단한 단계별 수행 내용, 자료구조 등을 기술하시오.

해당 문제 해결을 위해 주로 활용되는 자료구조는 동적 배열이다. 동적 배열을 기반으로 Array 및 RangeArray 클래스를 구현하면 되는데, 가장 먼저 Array 클래스를 설계하고 구현한다. 배열 데이터를 저장하기 위한 포인터(int\* data)와 배열의 길이를 저장하는 변수(int len)를 정의한다. 이후 생성자에서 배열 크기를 매개변수로 받아 해당 크기의 배열을 동적 할당하고, 소멸자에서 동적 할당된 메모리를 해제한다. C++에서는 ‘new’ 와 ‘delete’ 연산자를 사용해 동적 배열의 메모리 할당 및 해제를 관리하는데, 클래스의 생성자에서는 ‘new’를 통해 메모리를 할당하고, 소멸자에서는 ‘delete’를 사용해 할당 받은 메모리를 해제하여 메모리 누수를 방지한다. 이를 통해 배열의 크기를 결정하고, 필요한 메모리 자원을 효율적으로 사용할 수 있다. 또한, 배열의 길이를 반환하는 length() 함수를 구현하고 ‘[ ]’ 연산자 오버로딩을 통해 배열 원소에 접근하고 수정할 수 있는 기능을 구현해준다. 마지막으로 배열의 모든 원소를 출력해주는 print() 함수를 구현해주면 된다.

두 번째로, RangeArray 클래스를 설계하고 구현한다. RangeArray 클래스는 Array 클래스를 상속받아 기본 기능을 사용하므로 배열의 시작과 끝 인덱스를 나타내는 데이터 멤버를 추가적으로 정의해주면 된다. RangeArray 클래스의 생성자에서 시작과 끝 인덱스를 매개변수로 받고, 이를 기반으로 부모 클래스인 Array 클래스의 생성자를 호출하여 배열을 동적 할당해준다. 또한 인덱스 범위를 체크하는 로직을 추가하여, ‘[ ]’ 연산자 오버로딩을 재정의해준다. 마지막으로 시작과 끝 인덱스를 반환하는 함수를 추가해주면 RangeArray 클래스를 최종적으로 구현할 수 있다.

마지막 단계로, 테스트 및 디버깅을 해준다. 제공된 main() 함수를 사용해 Array와 RangeArray 객체의 생성, 원소 접근, 수정, 출력 등의 기능을 테스트한다. 또한, 해당 문제 해결을 위해 동적 배열을 사용할 때, 배열의 범위를 벗어난 접근을 시도할 수 있다. 이를 관리하기 위해 배열 접근 시 범위 검사를 수행하고, 만약 잘못된 인덱스에 접근하는 경우 예외 처리나 에러 메시지를 출력하는 기능을 수행한다.