CSE3013 (컴퓨터공학 설계 및 실험 I)
CPP-2 예비 보고서

서강대학교 컴퓨터공학과 박수현 (20181634)

서강대학교 컴퓨터공학과

1 목적

실험에서 제시한 문제를 이해하고, 이를 해결하기 위한 알고리즘 및 자료 구조를 구상한다.

2 문제

2.1 문제 이해

Print가 미구현된 LinkedList 클래스를 완성하고, 템플릿 자료형을 사용할 수 있도록 확장한다. 이를 상속받는 Stack 클래스를 작성한다.

2.2 구현 방법 구상

- **Print()**: 링크드 리스트의 원소를 순회하면서 노드의 값을 차례로 출력하면 된다.
- **템플릿 클래스로 확장**: 선언부에 `template <typename T>` 혹은 `template <class T>`를 노드와 링크드 리스트 모두에 추가한다. 그리고 기존에 데이터를 담고 있던 자료의 자료형 `int`를 전부 템플릿 변수 T로 바꿔 준다.
- `class Stack`: `class LinkedList`를 상속받는다. `virtual bool Delete(T& element)`로 Delete()를 재정의한다.
  링크드 리스트의 맨 앞 원소 삭제에 성공 시 `true`, 실패 시 `false`를 반환하며, 삭제에 성공했을 경우 삭제된 원소를 element에 저장한다. 또한 first는 first->link가 되며 current_size는 1 감소하게 된다.

3 예비 학습

3.1 다형성

다형성polymorphism은 프로그래밍 언어들 또는 유형론에서, 각 요소들(상수, 변수, 식, 오브젝트, 함수, 메소드 등)이 다양한 자료형에 속하는 것이 허가되는 성질을 말한다. 반의어로는 **단형성**monomorphism이 있다. 다형성에는 **임시 다형성**ad hoc polymorphism, **변수 다형성**parametric polymorphism, **서브타입 다형성**subtype polymorphism 등이 존재한다.

임시 다형성은 같은 이름을 가진 함수가 특정 자료형에 따라 다른 기능을 가질 수 있게 하는 성질이다. 예를 들어, C++에서 `int`에서의 '+' 연산자와 `std::string`에서의 '+' 연산자의 정의는 다르다. 맥락에 따라 전자는 두 `int`의 값을 더하는 역할을 하지만 후자는 두 `std::string`을 이어붙이는 역할을 한다. 또한 임시 다향성은 **오버로딩**overloading의 기반이다.

변수 다형성은 자료형이나 함수를 일반적으로 정의할 수 있게 하는 성질이다. C++의 **템플릿**template, Java의 **제네릭**generic을 뒷받침하는 성질이다. C++에서 `std::pair<class T1, class T2>`는 `T1`과 `T2` 자료형에 상관없이 `first`, `second` 등의 이름으로 멤버 변수에 액세스할 수 있다. 또한

```
template <class T>
T max(T a, T b) {
    return a > b ? a : b;
}
```

와 같은 함수는 `T`에 상관없이 `a`와 `b` 중 큰 값을 반환한다. 변수 다향성은 컴파일 타임에 일어나기 때문에, **컴파일 타임 다형성**compile-time polymorphism이라고도 불린다.

서브타입 다형성 또는 유형론에서 **포함 다형성**inclusion polymorphism은 변수 다향성과 비슷하지만, 변수 다향성이 자료형에 구애받지 않고 적용 가능한 성질이었다면 서브타입 다향성은 어떤 자료형을 상속받는 자료형에만 적용되는 개념이다. 어떤 다형성 함수를 호출할지는 런타임에서 결정되기 때문에 변수 다향성과는 반대로 **런타임 다형성**runtime polymorphism이라고 불린다.

3.2 캡슐화

캡슐화encapsulation는 OOP에서 객체의 속성과 메서드를 하나로 묶고, 구현 내용 일부를 외부에서 액세스할 수 없게 하는 것이다.

캡슐화를 통해 객체의 내부 구현은 숨기면서 꼭 필요한 필드와 메서드들만 다른 코드에서 참조하게 만들어 코드의 복잡도를 줄일 수 있다. 또한 객체의 내부를 숨김으로써 다른 코드가 객체의 내부 데이터를 일관성 없는 상태로 수정하는 것을 방지할 수 있고, 이는 객체의 무결성을 보호한다. 여러 객체 지향 프로그래밍 언어들은 캡슐화를 위한 **접근 지정자**access specifiers들을 제공한다.

3.3 재정의

재정의override는 위에서 언급한 **서브타입 다형성**subtype polymorphism에 해당된다. 상위 클래스에서 상속받은 메서드를 하위 클래스에서 다른 동작으로 재정의하는 것을 말한다.