CSE3013 (컴퓨터공학 설계 및 실험 I)
CPP-1 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

`Array`, `RangeArray` 클래스를 구현하고, 동작에 맞는 멤버 함수들과 연산자들을 구현한다.

2.2 구현 방법 구상

```
class Array
```

- `int *data` (protected): 런타임에 메모리를 할당하여 자료가 저장되는 변수이다.
- `int len` (protected): 배열의 크기를 저장하는 변수이다.
- `Array(int size)` (public): 크기가 `size`인 배열을 생성한다.
- `~Array()` (public): 배열을 메모리에서 제거한다.
- `int length()` (public): 배열의 크기를 반환한다.
- `int& operator[](int i)` (public): $0 \leq$ `i` $<$ `len`일 경우, 배열의 `i`번째 위치에 값을 할당한다.
- `int operator[](int i) const` (public): $0 \leq$ `i` $<$ `len`일 경우, 배열의 `i`번째 위치의 값을 반환한다.
- `void print()` (public): 배열의 모든 원소를 출력한다.

또한 `cout`은 `std::ostream`이므로 `print()` 대신 다음과 같은 함수를 정의하면 `cout << array`와 같은 형식으로 배열의 모든 원소를 출력할 수 있을 것이다.

```
std::ostream& operator<<(std::ostream& os, const Array &input) {
    for (int i = 0; i < input.length(); i++) {
        os << input[i] << ' ';
    }
    return os;
}
```

```
class RangeArray : public Array
```

- `int low` (protected): 배열의 시작 인덱스를 지정한다.
- `int high` (protected): 배열의 끝 인덱스를 지정한다.
- `RangeArray(int low, int high)` (public): 크기가 $\texttt{high} - \texttt{low} + 1$인 배열을 생성한다.
- `~RangeArray()` (public): 배열을 메모리에서 제거한다.
- `int baseValue()` (public): 배열의 시작 인덱스를 반환한다.
- `int endValue()` (public): 배열의 끝 인덱스를 반환한다.
- `int& operator[](int i)` (public): $\texttt{low} \leq \texttt{i} \leq \texttt{high}$일 경우, 배열의 `i`번째 위치에 값을 할당한다.
- `int operator[](int i) const` (public): $\texttt{low} \leq \texttt{i} \leq \texttt{high}$일 경우, 배열의 `i`번째 위치의 값을 반환한다.

이 때 `class RangeArray`는 `class Array`를 상속받는다.

3 예비 학습

3.1 Visual Studio의 단축 키

단축 키	동작
ctrl + C	복사하기
ctrl + V	붙여넣기
ctrl + F7	컴파일
ctrl + F5	디버깅 없이 시작
F5	디버깅 시작
alt + F9	중단점 창
F9	중단점 설정 / 해제
F10	프로시저 단위 실행
F11	한 단계씩 코드 실행
ctrl + F10	커서까지 실행
⇧ + F11	프로시저 나가기

3.2 OOP

- **OOP**Object Oriented Programming: **객체 지향 프로그래밍**이라고도 한다. OOP는 컴퓨터 프로그램을 명령어의 절차로 보기보다는 객체의 집합으로 보는 패러다임이다. 자료 추상화를 기초로 하여, 상속, 다형 개념, 동적 바인딩 등의 특징이 존재한다.
- **객체**object: 변수, 자료 구조, 함수 또는 메소드가 될 수 있는 공간이다. OOP 패러다임에서는 클래스의 인스턴스를 일컫는 데에 쓰인다.
- **클래스**class: 객체를 정의하기 위한 속성attribute과 기능method들의 집합이다. 속성과 기능들은 객체를 설계하지만, 클래스 자체가 객체인 것은 아니다. 비유하자면 클래스는 객체의 설계도인 셈이다.
- **인스턴스**instance: 클래스의 정의에 따라 만들어진 객체들이다.
- **상속**inheritance: 새로운 클래스가 기존의 클래스의 자료와 기능들을 사용할 수 있게 하는 기능이다. 상속을 받는 클래스를 하위 클래스, 또는 자식 클래스라고 하며 새로운 클래스가 상속하는 기존의 클래스를 상위 클래스, 또는 부모 클래스라고 한다.

3.3 OOP를 사용하는 이유

- 개발과 보수가 편리해지며, 추상화를 통해 직관적인 코드 분석이 가능해진다.

- 코드의 구조가 유연해지고 수정이 용이해지기 때문에 대규모 프로젝트에 적합하다.
- 라이브러리를 만들어 여러 다른 프로젝트에서 활용하기에 좋다.