3주차 결과보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호



텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 1]

[사진 1]은 메인함수가 들어있는 c++파일을 캡처한 것이다. Array 객체를 정의하는데, 각각 크기가 10, 5인 객체를 생성한다. 생성자는 양의정수의 크기가 입력되지 않은 경우는 에러메세지를 출력하게 하고, 양의정수일 경우에만 배열을 동적할당 해준다. 크기가 10인 a객체에는 1부터 10까지의 수를 저장하고, 크기가 5인 b객체에는 0부터 2씩 증가하는 수를 저장한다. 이후 Array객체에 있는 멤버함수인 print()를 통해 객체에 할당된 전체 배열을 출력해주는 과정을 통해 잘 저장이 되었나 확인한다. 그 이후 Array 객체의 인덱스범위에 포함되지 않는 값인 -1을 대입해주어 벗어났을 경우 에러메세지가 잘 출력되는지도 확인한다. 그 이후 x, y에 각각 a[0]과 b[0]을 대입해주는데, a와 b는 배열이 아니기때문에 바로 []연산자를 사용하게 되면 오류가 뜨게된다. 저런 형태의 연산을 추가로 정의해준 이후에 써야 잘 작동하는데, 그때 사용하는 방법이 연산자 오버라이딩이다. Array.cpp에 int operator=(int i)와 int& operator=(int i)형식으로 정의된 함수가 바로 연산자 오버라이딩이다. 이 부분을 통해 =연산자가 재정의되어 배열의 원소를 뽑아오는 형태로 객체 a의 멤버변수 배열의 원소를 뽑아올 수 있게 된다.

Rangearray 클래스는 Array 클래스를 기반 클래스로 하는 파생 클래스이다. Rangearray 에서는 Array의 기본적인 멤버함수나 오버라이딩 된 연산자는 모두 사용이 가능하고, 인덱스의 범위를 변경해주는 코드만 추가적으로 작성하였다. Rangearray 클래스를 사용해 객체를 정의하면 그는 Array객체를 활용해 배열이 정의된다. 예를 들어, Rangearray ex(-4,5)라고 정의가 되었다면 현재 ex 객체의 인덱스는 -4부터 5까지 접근이 가능하여 총 10개의 원소가 저장이 가능하다. 이를 구현할 때, 인덱스가 0부터 9까지인 array 객체를 Rangearray 객체 내부에서 생성한다. 이러한 과정을 위해 인덱스의 왼쪽끝을 나타내는 변수로 low, 인덱스의 오른쪽끝을 나타내는 변수로 high를 설정하였다. 또한, 연산자 []를 재정의하여, ex[-3] 같은 것을 구하기 위해 내부 array 객체의 1번 인덱스값을 불러오게 만들었다. 따라서, 인덱스 범위를 사용자가 지정할 수 있는 배열을 만들었다.

이러한 재정의 덕분에 [사진 1]에서 c[-1]형태로 c 객체에 []연산자를 이용해 접근이 가능해진다. 그러나 c[-2]와 같은 식은 인덱스 범위에서 벗어나므로 에러메세지가 출력된다.

파이썬의 리스트의 경우 인덱스값이 음수인 경우에도 접근이 가능하다. 어떻게 구현이 되었는지 잘 가늠이 되지 않았었는데 이러한 연산자 재정의를 사용하면 충분히 구현이 가능하다는 것을 알게 된 실습이었다.

1. 일반적으로 문자열의 대입은 c언어에서 불가능하다. 이를 위해서는 strcpy함수를 이용할 수도 있다. 하지만 정수나 실수를 변수에 대입하는 것처럼 =연산자로 문자열의 대입을 하고 싶기 때문에 이를 구현하는 프로그램을 작성하는 과제이다.

이를 위해 Str이라는 클래스를 구현하여, 멤버변수로 len과 str를 주었다. 여기서 len은 문자열의 길이를 담고, str배열에는 문자열의 내용을 담는다. 내부 멤버함수는 아래와 같다.

* 생성자 : 길이나 내용을 입력받아서 새로운 객체를 생성함. 길이는 양수만 받고, 음수를 받을 시 에러메세지를 출력함.
* 소멸자 : str 배열 메모리를 해제해줌
* Int length() : 문자열의 길이를 리턴하는 함수
* Char \*contents() : 문자열의 내용을 리턴하는 함수
* Int compare(class Str&) : 현재 객체와 비교하려는 객체의 문자열을 strcmp했을 때의 리턴값을 리턴하는 함수
* Int compare(const char \*) : 현재 객체의 문자열과 입력 문자열을 strcmp했을 때의 리턴값을 리턴하는 함수
* Void operator=(const char \*) : =연산자 오버라이딩으로, 입력 문자열의 내용을 현재 객체에 대입한다. 대입이란, 객체의 len변수와, str배열을 업데이트 해주는 것을 의미한다. 이때 len변수는 입력문자열의 길이로 주고, str배열은 원래 있는 값이 있으니 메모리 해제를 해주고 새롭게 str배열을 new를 이용해 길이 len+1짜리 배열로 만들어 준 후, str에 문자열을 대입해준다. 이때 대입은 strcpy를 이용한다.
* Void operator=(class Str&) : =연산자 오버라이딩으로, 입력 객체의 내용을 현재 객체에 대입한다. 이때 대입은, 입력객체의 len값을 현재 객체의 len값에 대입해주고, 입력 객체의 str의 내용을 현재 객체의 str내용으로 업데이트 해준다. 이때 문자열 업데이트는 strcpy를 이용한다.

이러한 Str 이라는 객체를 위와 같이 생성하면, 우리가 원하는 기능인 “=연산자를 활용한 문자열 대입” 구현할 수 있다. 문제의 입력으로 코드를 돌리면 객체가 알아서 메모리 해제가 된다. 그 이유는 new를 이용하여 객체를 생성해준것이 아닌 대입을 통해서 생성해주었기 때문이다. 객체를 예시코드와 같이 생성하면 객체가 생성된 함수가 종료되면 객체도 같이 메모리 해제가 된다.

원래 코드와 다르게 const를 추가해준 부분이 있다. 객체 초기화 과정에서 string literal을 char\*에 대입해주게 되는 연산이 strcpy에 의해 발생하기 되는데 이는 deprecated한 과정이라고 warning 메세지가 뜬다. 이를 해결하기 위해 const를 추가해주어 string literal이 잘 대입될 수 있도록 코드를 약간 수정해주었다.

클래스의 선언과 실제 함수 구현은 다른 파일에 하는 것이 좋으므로 Str.h와 Str.cpp를 따로 구현하였다. 또한, makefile을 이용하여 프로그램이 실행되도록 하게 했다.