자료구조및프로그래밍 HW3

C211171 최후락

2025 10 01

1 개요

강의에서 배운 ArrayBag 클래스를 구현한다. 기존의 기능에 추가로 두 ArrayBag의 합집합과 교집합, 차집합을 구현한다. 두 ArrayBag을 키보드로 입력받는 것이 아닌 test.in파일을 읽는다. 출력또한 test.out파일로 생성한다.

2 공부한 C++문법

반복문for를 사용할 때 증감식을 i++로 쓰는 습관이 있었다. i+=1이나 i=i+1보다 간결하다고 생각했기에 i++을 즐겨 사용했다. 이번과제를 수행하면서 미리 작성된 코드를 살펴보던중 for문의 증감식에 i++이 아닌 ++i가 써있어 두개의 차이점이 궁금해졌고 찾아보았다. 두 증감식 모두 i값을 하나 증가시킨다는 동일한 기능이지만, 성능면에서 아주 약간의 차이가 있다. i++로 작성을 하면 임시로 값을 저장해두었다가 다시 사용해야하기에 ++i에비해서 약간의 성능적 저하가 있다. 현대의 컴파일러에서 간단한 자료형을 다룰때는 유의미한 차이를 만들지 못하지만, 벡터나 리스트를 사용할때 iterator가 복잡해진다면 유의미한 성능차이가 발생할 수도 있다. 이 때문에 ++i로 사용하는 경향이 있다.

이번 과제를 수행하면서 헤더파일을 두 파일로 나누어서 선언하였다. Arraybag.h(이하 .h)와 Arraybag.cpp(이하 .cpp) 두 개의 파일로 나누어서 만들었고 .h파일에서는 멤버변수와 멤버함수의 이름들과 시그니처작성까지만 하였고 .cpp파일에서 함수의 구현방식과 세부사항들을 작성했다. 이 두 파일을 살펴보던중 의문점이 생겨서 찾아보았다. .cpp파일에서는 int ArrayBag ¡T ¿:: get_index_of (const T& target) const로 함수의 시그니처가 작성되었지만, .h파일에서는 int get_index_of (const T&) const로 선언되어 있다. 두 시그니처의 소괄호를 살펴보면 .h에서는 괄호안에 들어간 인자를 레퍼런스변수를 사용해서 별명을 붙여서 사용할 예정이다 까지만 선언되어 있고 .CPP에서는 별명을 붙여서 사용할 예정인데 이제 그 별명이 target이다 까지 선언한다. 처음 .h파일의 시그니처를 보았을 때 뭔가 구성요소가 빠진것이 아닌가라는 생각을 했지만, 검색결과 어차피.cpp에서 상세하게 기입할 예정이므로 .h에서는 레퍼런스변수 사용예정만 알리면 된다.

Arraybag의 멤버함수중 벡터로 기존 값들을 전부 받는 함수가 있다. 이를 구현하며 cpp에서 벡터의 사용법을 공부하던 중, 초기화방법중 vec(5,10)이 5와 10을 인자로 하는 두 칸짜리 벡터가 아니라 5칸을 10으로 채워넣으라는 초기화 방법인 것을 배웠다. vec5,10꼴과 사용에 유의한다.

std :: vector¡T¿ ArrayBag ¡T¿:: to_vector () const 이 함수의 시그니처를 살펴보던 중 const가 붙은 위치가 의아해서 찾아보았다. 기존의 자료형에 const를 붙여 int const처럼 사용하거나, 포인터에 const를 붙여서 const int* p혹은 int* const p처럼 사용하던 것이 생각났다. 이와 유사한 기능으로 멤버함수의 const는 이 함수를 호출한 객체의 값을 바꾸지 않기위해 안전장치로써 const를 사용한다. 이는 멤버함수에서만 쓸 수 있는 특별한 문법이고 this포인터의 존재 때문에 가능하다. 우리가 멤버함수를 호출하면 자기자신을 가리키는 보이지않는 this포인터가 존재하고 이 this포인터를 상수화해서 this포인터가 가리키는 것을 수정하지 않도록 하는데에 그 목적이 있다. 이 때문에 멤버함수가 아닌 함수에서는 사용이 불가하다.

for문을 벡터와 함께 사용시 괄호내부에 (i=0;i¡k;++i)와 같은 방법으로 기입하는 것이 아닌 (const T& item : vector)처럼 사용할 수도 있다.

3 책에 없는 함수 구현 및 설명

합집합: 중복이 불가능한 수학적 집합과 다르게 과제에서는 집합내부에서 중복을 허용하였다. 이에 따라 중복값을 고려하지 않고 두 bag에서 전부 복사하여 새로운 벡터에 전부 복사한다.

교집합: for문의 반복자로 하나의 bag을 벡터로 반환해서 사용한다. 그 후 조건문의 조건을 두 집합 모두 들어가 있다면 신규벡터에 추가하는 방식으로 한다. 이 과정에서 위의 합집합과 동일하게 중복을 허락하는 집합이므로, 신규 집합에 포함된 어떤 원소의 갯수가 원래 집합에 들어있는 같은 원소의 갯수보다 적다면 계속 추가한다.

차집합:A-B와 B-A가 다른 결과를 만드므로 합집합 교집합과는 다르게 멤버함수로 작성한다. 자기자신은 this포인터로 받고 또다른 집합은 인자로 받아서 처리한다. 벡터로 반복을 돌리면서 제거하는 과정에서 remove함수를 만들 때 존재하지 않는 원소를 제거하지는 않도록 만들었으므로 전체 원소를 반복을 돌리면서 빼도 문제는 발생하지 않는다.

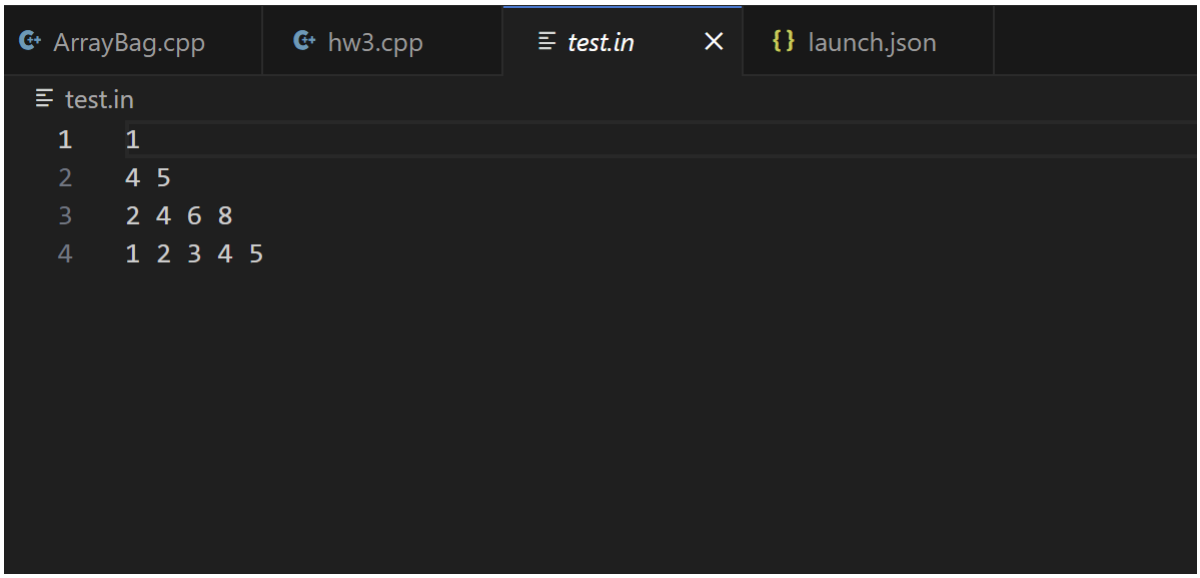

Figure 1: test.in

4 실행결과

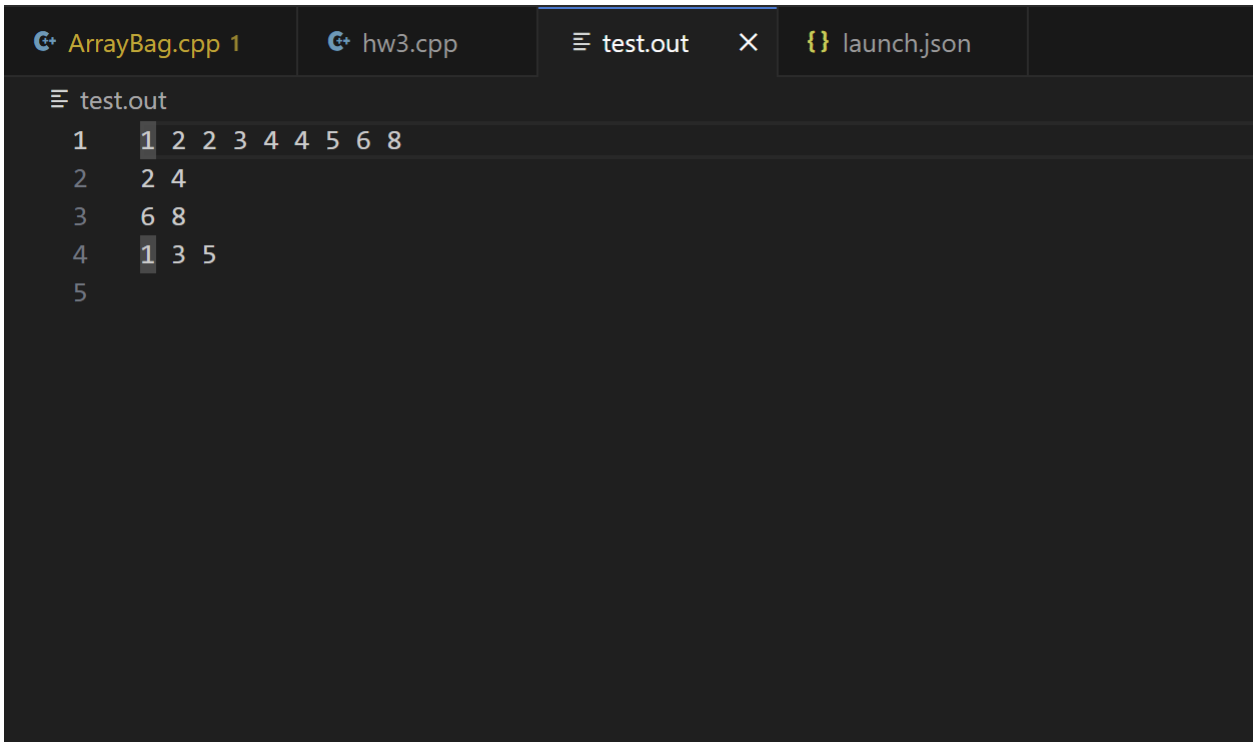

Figure 2: test.out

우리가 평소에 실행했던 프로그램들과는 다르게 키보드를 사용하여 값을 입력하고 모니터상에 출력하여 확인하는 것이 아니라 파일로 출력값을 생성한다. 위 그림 1을 보면 미리 생성한 test.in이라 이름붙인 파일에 미리 입력양식에 맞는 값들을 기입해놓았다. launch.json에 미리 어떻게 이 파일을 실행할 것인지 방법을 명시해 놓았기에 프로그램 실행시 test.in(그림: 1)을 읽고 test.out(그림: 2)파일로 결과값을 출력한다.

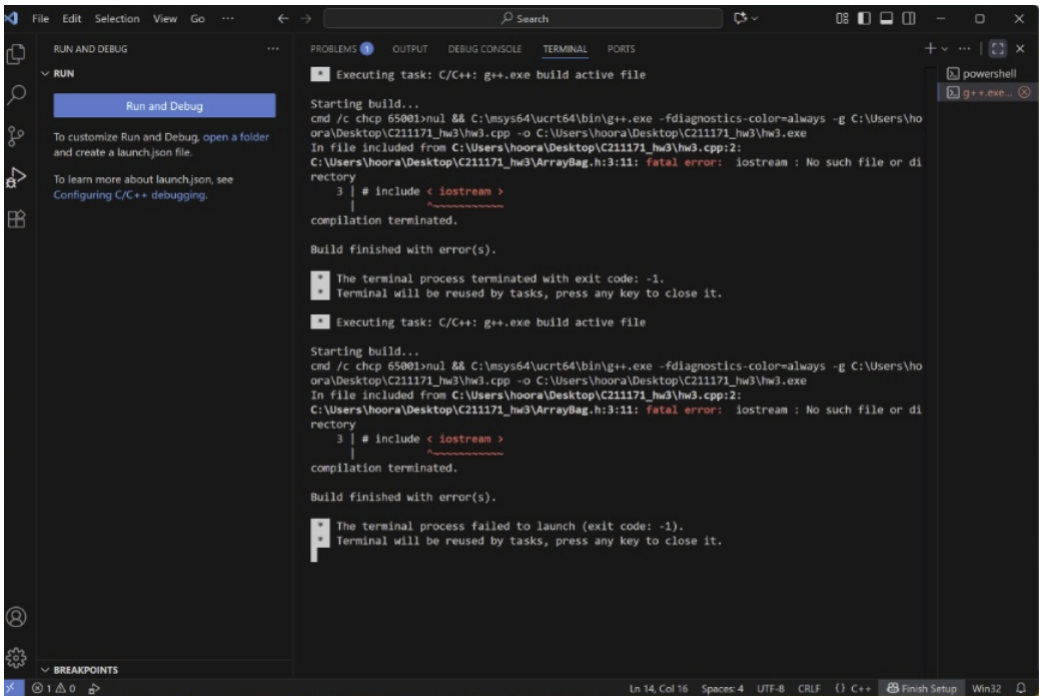

Figure 3: iostream인식안됨

5 어려웠던 점

함수구현을 하는 부분에서는 배운내용과 인터넷검색등을 적절히 사용하여 비교적 짧은 시간내에 해결할 수 있었다. 하지만 정말 시간이 오래걸리는 부분은 컴파일이 안되는 문제를 해결하는 것이었다.
처음 코드작성을 마쳤을 때는 f5를 눌러서 실행하려고 해도 되지 않았고 무엇이 문제인지 몰라서 평소에는 노트북으로 작업하다가 데스크탑으로 작업환경이 바뀌어서 그런가 생각이 들어 파일만 그대로 옮겨서 해보자라며 일단 접어두었다. 하지만 노트북으로 옮겨서 실행하려해도 되지 않았다. 지난 실습인 hw2는 잘 돌아갔기에, 작업환경이 아닌 파일의 문제인지 생각해보게 되었다.
이후 LLM의 도움을 받아 컴파일문제를 해결해보려 시도했지만 많은 방법이 실패했고 결국 오랜시간이 지나고 해결법을 찾아냈다. iostream을 인식하지 못하는 오류가 계속 발생하고 이는 visual studio에서 설치된 c++컴파일러를 visual studio code가 인식하지 못해서 생기는 문제였다. 이를 해결하기 위하여 x64 Native Tools Command Prompt for VS 2022 를 실행한 뒤 cd C:
Users
hoora
Desktop
C211171_hw3 명령어를 통해서 파일로 이동하고 code .을 입력하여 위치를 제대로 인식한 vscode를 실행하였다. 이후 f5를 눌러 실행시 정상적으로 실행되는 모습을

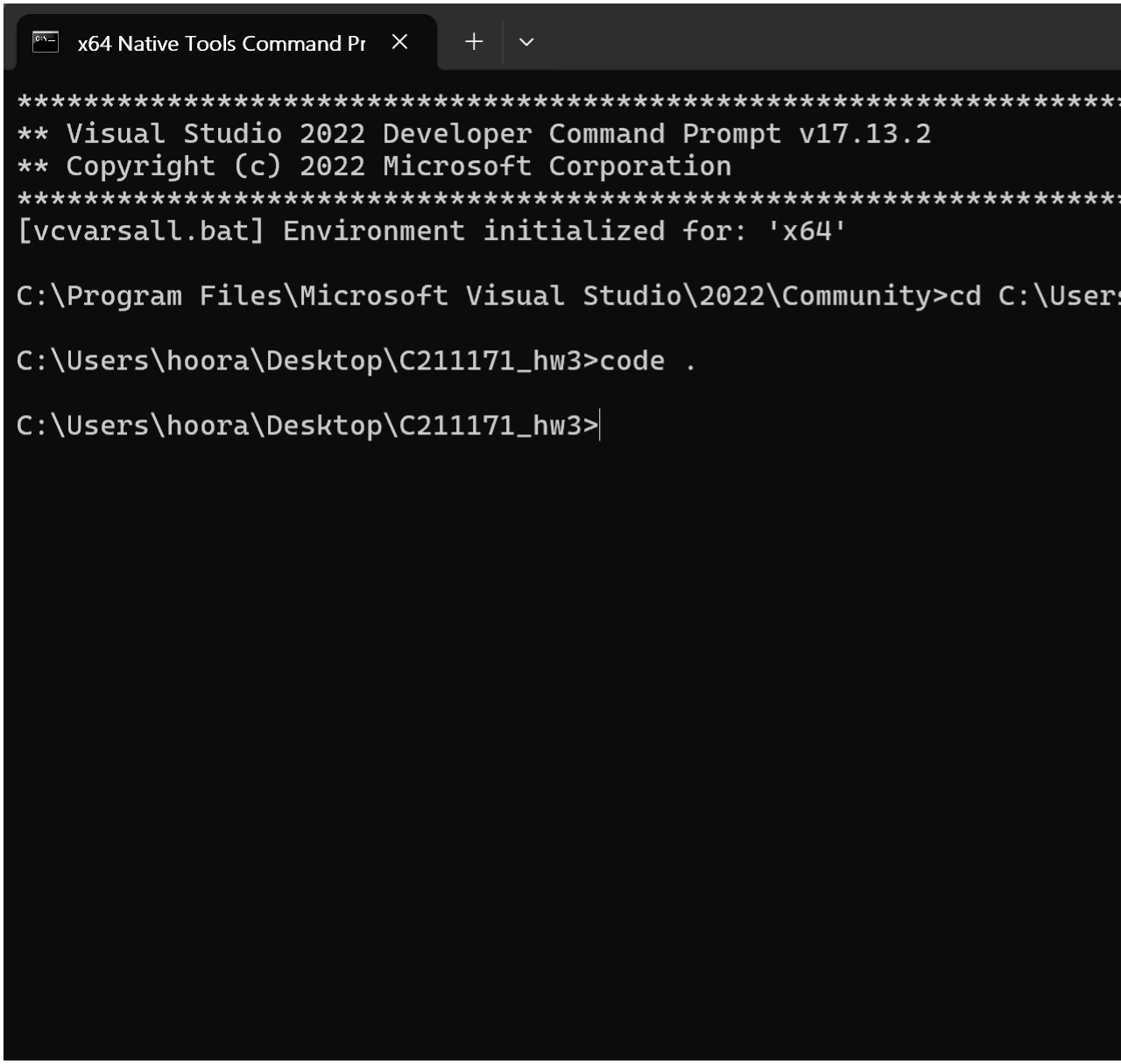

Figure 4: 직접 찾아서 열기

보여주었다.