2019년-Spring

객체지향프로그래밍

Assignment #1

학번: 20180085

학과: 무은재학부

이름:송수민

POVIS ID: songsm921

명예서약(Honor code)

"나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다."

Problem 1: 보이는 나무막대기 개수 세기

I. 프로그램 개요

첫번째 프로그램은 일렬로 나무 막대기가 늘어서 있을 때, 각 왼쪽, 오른쪽에서 보았을 때 보이는 나무막대기의 개수를 각각 더해서 이를 출력하는 프로그램이다.

예를 들어, 나무막대기가 5개 배열 되어있고, 이 막대기의 높이는 각각 다음과 같다고 하자.

4 6 5 8 6

왼쪽에서 보았을 때는 3개, 오른쪽에서 보았을 때는 2개가 보이므로, 5가 출력되어야 할 것이다.

II. 프로그램 구조 및 알고리즘

이 프로그램의 전체적 구조는 다음과 같다.

1)입력부 : 기둥의 개수, 기둥의 높이를 사용자가 입력한다.

2)처리부 : 사용자가 입력한 각각 변수에 따라, 배열을 동적할당 하고, 이에 따라 보이는 막대기의 개수를 계산한다.

3)출력부 : 문제에서 요구하는 답안(왼쪽에서 보이는 막대기 수 + 오른쪽에서 보이는 막대기 수)을 출력한다.

이에 따라 이 프로그램의 알고리즘은 다음과 같다 할 수 있다.

1)한 쪽 끝에서부터 각 원소의 값을 비교한다. 검사한 INDEX까지의 MAX값이랑 비교를 하여, MAX보다 크다면 그것은 눈에 보일 것이므로, Count변수를 1씩 늘려준다.

2)최종 count값이 한 쪽에서 보이는 최종 값이다.

3)양쪽에서 나온 count를 더하여 출력한다.

III. 프로그램 실행 방법 및 실행 결과

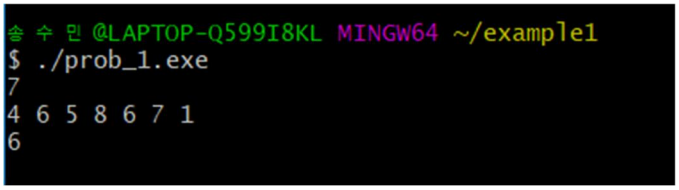
1) 사용자 입력

난이도를 선택하면, 나무 막대기의 개수를 입력한다. 그 후 각 나무막대기의 높이를 입력한다.

2) 결과 출력

1)에서 입력한 값을 바탕으로 답안이 계산되고, 이를 출력한다.

예시 출력 결과.



왼쪽에서 본 값 : 3

오른쪽에서 본 값 : 3

결과 : 6

Problem 2 : 보이는 나무막대기 개수의 최대값

I. 프로그램 개요

이 프로그램은 1번문제의 상황과 같은 상태에서, 막대기를 원하는 수 만큼 제거하여 최대로 보이는 개수를 구하는 것이다. 예를 들어, 다음과 같은 상황이 주어진다고 하자.

4 6 5 8 6 7 1

이와 같은 경우에서, 왼쪽에서부터 1,3,5,6,번째의 나무막대기만 남길 경우 왼쪽에서 보았을 때 나무막대기의 수가 최대가 된다. 값은 4가 출력 될 것이다.

II. 프로그램 구조 및 알고리즘

본 프로그램의 전체적 구조는 다음과 같다.

1)입력부 : 기둥의 개수, 기둥의 높이를 사용자가 입력한다.

2)처리부 : 사용자가 입력한 각각 변수에 따라, 배열을 동적할당 하고, 이에 따라 나무 막대기의 개수가 최대로 보이게 계산한다.

3)출력부 : 문제에서 요구하는 답안(왼쪽에서 보이는 최대 나무 막대기 개수)을 출력한다.

이와 같은 결과를 처리하기 위한 알고리즘은 다음과 같다.

1) 배열을 처음부터 입력 받기 시작한다.

2)각 원소 값의 Boundary를 정하여, 그 Boundary보다 미만이면, 그 수로 변환하고, 그 Boundary 초과이면 그 수 뒤에 값을 추가한다.

Assn1문서에서 제시된 예시로 통해 알아보자.

4 6 5 8 6 7 1이 차례로 입력된다고 하자.

초기값 4 이것의 lower_boundary -> 3

다음이 6이므로 4 초과.

4 6 lower_boundary -> 5

4 5 이의 lower_boundary -> 4

4 5 8 ->이의 lower_boundary -> 7

4 5 6 -> 5

4 5 6 7 이와 같은 형식으로 처리된다.

III. 프로그램 실행 방법 및 실행 결과

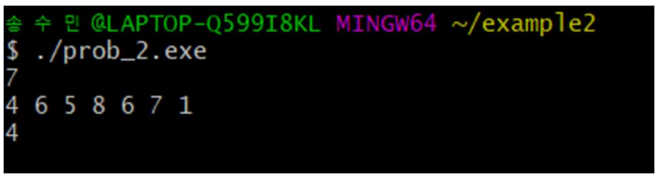
1) 사용자 입력

난이도를 선택하면, 나무 막대기의 개수를 입력한다. 그 후 각 나무막대기의 높이를 입력한다.

2) 결과 출력

1)에서 입력한 값을 바탕으로 답안이 계산되고, 이를 출력한다.

예시 출력 결과.



4 5 6 7 - > 4개라는 값 출력

Problem 3 : 빙하의 개수 세기

I. 프로그램 개요

이 프로그램은 y년 후 녹은 빙하의 덩어리 수를 계산하는 프로그램이다.

다음과 같은 상황이 있다고 가정하자.

0	1	0	4	0
0	0	3	5	0
0	1	0	0	2
5	4	0	2	1

이와 같은 경우 4덩이로 계산되며, 4라는 값이 출력된다.

II. 프로그램 구조 및 알고리즘

본 프로그램의 전체적 구조는 다음과 같다.

1)입력부 : 난이도, 빙하의 높이, 빙하의 개수를 사용자가 입력한다.

2)처리부 : 사용자가 입력한 각각 변수에 따라, 2차원 배열을 동적할당 하고, 이에 따라 값이 입력되고 답안이 처리된다.

3)출력부 : 문제에서 요구하는 답안(왼쪽에서 보이는 최대 나무 막대기 개수)을 출력한다.

이와 같은 결과를 처리하기 위한 알고리즘은 다음과 같다.

1) y년 후 빙하의 높이를 계산하여, 0보다 작거나 같은 곳은 모두 0으로 처리한다.

2) Test array에서는 이 배열을 검사하였는지 검사하고, 본 array에서는 값을 검사한다.

3) xy-component를 이용하여, 접근한 배열의 4방향을 검사하고, 그 곳이 이어져 있다고 판별되면, 그곳으로 접근하여 계속 이 2,3)을 반복한다. 그러다 4방향 모두 끊어져 있으면, main으로 돌아가 빙하의 덩어리를 카운트하고, 다시 배열을 탐색하여, 빙하가 있는 곳에서부터 다시 2,3)을 시작한다.(Depth-First Search 방식)

III. 프로그램 실행 방법 및 실행 결과

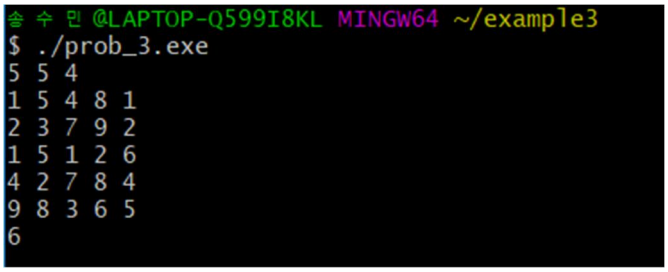
.1) 사용자 입력

난이도를 선택하면, 빙하의 가로, 세로, 년수를 입력한다. 그 후 각 빙하의 높이를 입력한다.

2) 결과 출력

2)에서 입력한 값을 바탕으로 답안이 계산되고, 이를 출력한다.

예시 출력 결과.



IV. 토론

C에서 C++로 언어를 바꾸면서, 가장 편리하다고 느낀 점이 동적할당이다. C에서는 malloc함수를 사용하여야 했는데, C++에서는 new라는 객체와 delete를 객체를 이용하면 간단히 해결된다.

2번 문제에서, 문헌조사 결과, LIS문제를 해결하는 과정이 시간 복잡도에 따라 두가지 경우가 있는 것을 알았다. 본인의 처음 해결방법은 $O(N^2)$이였다. 하지만 나중에 생각과 조사를 한 결과, 십만X십만의 개수를 처리하여야 할 때, 이 방법은 10초만에 결과가 나오지 않을 수도 있다는 생각과, 교수님의 조언에 따라 $O(nlogN)$의 방식으로 전환하였다.

3번 문제에서는, 정사각형 Array에서는 정상적으로 문제가 해결되지만, 직사각형 Array에서는 정상적으로 문제가 해결이 안되는 문제점이 발생하였다. 디버깅을 한 결과, 할당 되지 않은 배열에 접근하려 한다는 문제점을 발견 하였고, 이를 if를 사용하여 예외처리를 하였으나 또한 문제가 해결되지 않았다. 다시 디버깅을 한 결과, 아예 배열에 성분을 집어 넣는 작업부터 가로, 세로를 반대로 한 코드를 발견 하였고, 이를 수정하여 문제를 해결하였다.

V. 결론

본 문제에서 C에서의 기본적인 문법과 C++에서의 새로운 문법을 익히어 C보다 C++에서 편리한점을 알게 되었다. 또한 Data Structure에서 배울 항목을 접목하여 문제 해결을 시도 하였다. 특히, 이번 과제에서 함수 내에서 자기 함수를 호출하는 방식을 자주 사용하여 문제를 해결한 것이 이 때까지 잘 시도하지 않았던 방법이었으나, 이를 시도하여 문제를 해결하였다.

VI. 개선사항

이번 과제에서는 별다른 예외처리를 요구하지 않아 처리하지 않았으나, 일단 난이도에 따라 입력 받는 범위를 제한을 두어야 할 것 같다. IF문을 사용하여 범위 밖의 값을 입력하면 재입력을 요구하는 등의 방법으로 예외처리를 하면, 보다 사용자를 배려하는 프로그램이 될 것 같다. 또한 3번과제에서 발생하였던 문제를 보아, 정사각형에서는 문제가 되지 않았으나, 직사각형에서 발생하는 어레이 접근 문제에 대해서는, 기존에 설정하였던 Struct를 수정하면, if라는 연산을 거치지 않고 효율적으로 프로그램을 실행 할 수 있을 것이라고 생각한다.

VII. 참고문헌

1) 2번 문제 : https://ko.cppreference.com/w/cpp/algorithm/lower_bound

2) 2번 문제 : 포항공과대학교 동아리 포스캣 DP세미나 자료

3) 3번 문제 : https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/