자료구조 오픈랩5

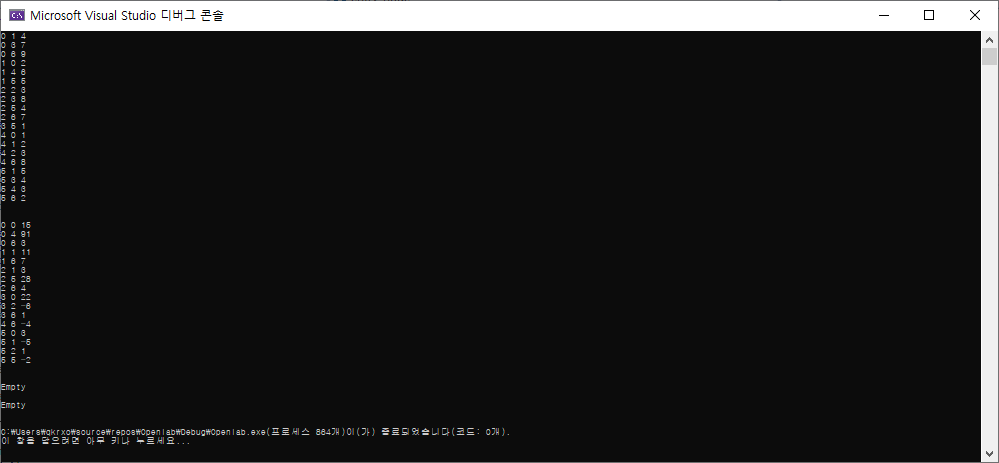
컴퓨터공학과 20171646 박태윤

**- 코드 실행 결과**

다음과 같이 A.txt와 B.txt를 입력으로 받았다.

|  |  |
| --- | --- |
|  |  |

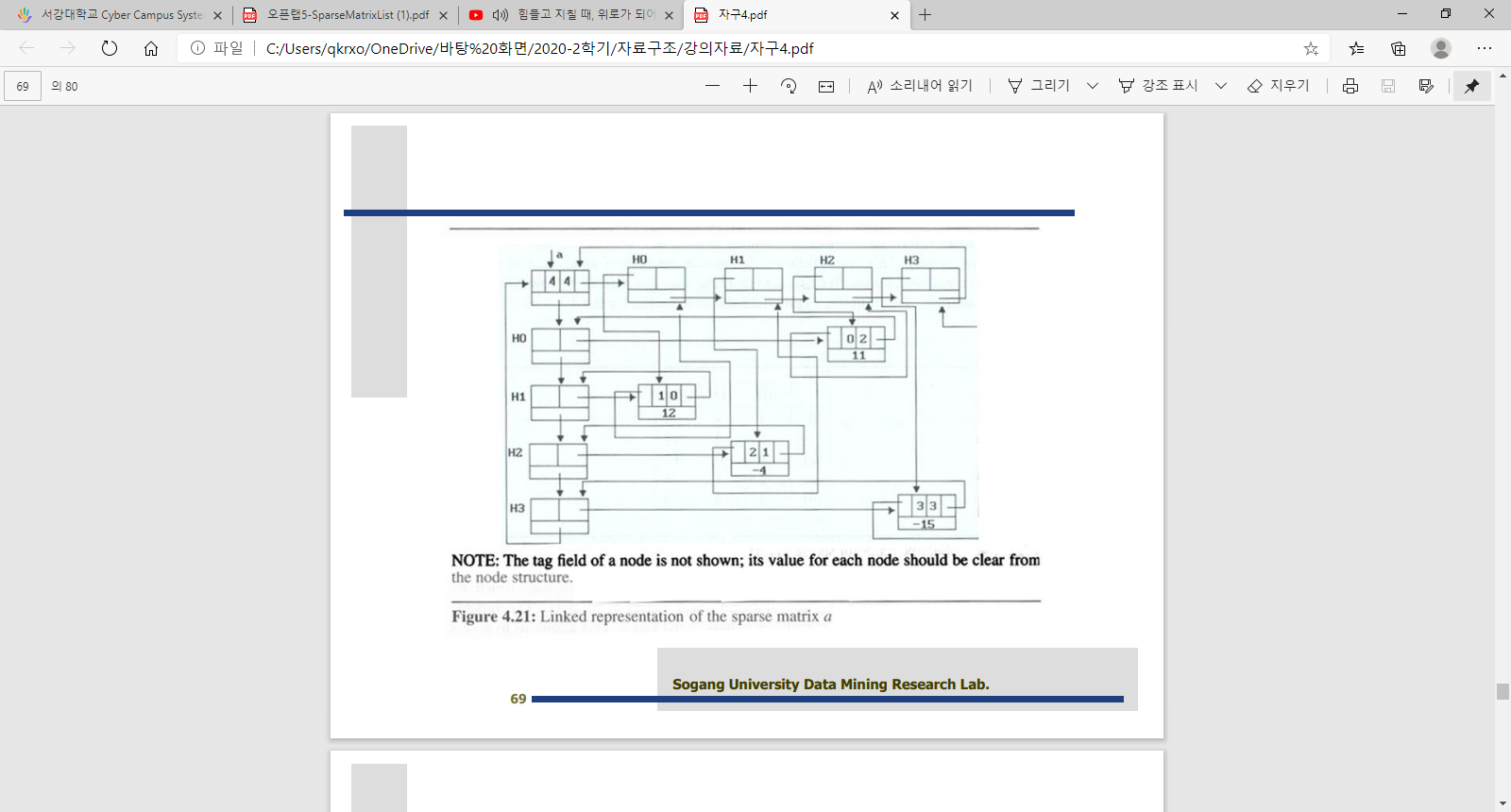
코드를 실행시켜서 다음과 같은 결과를 얻었다.



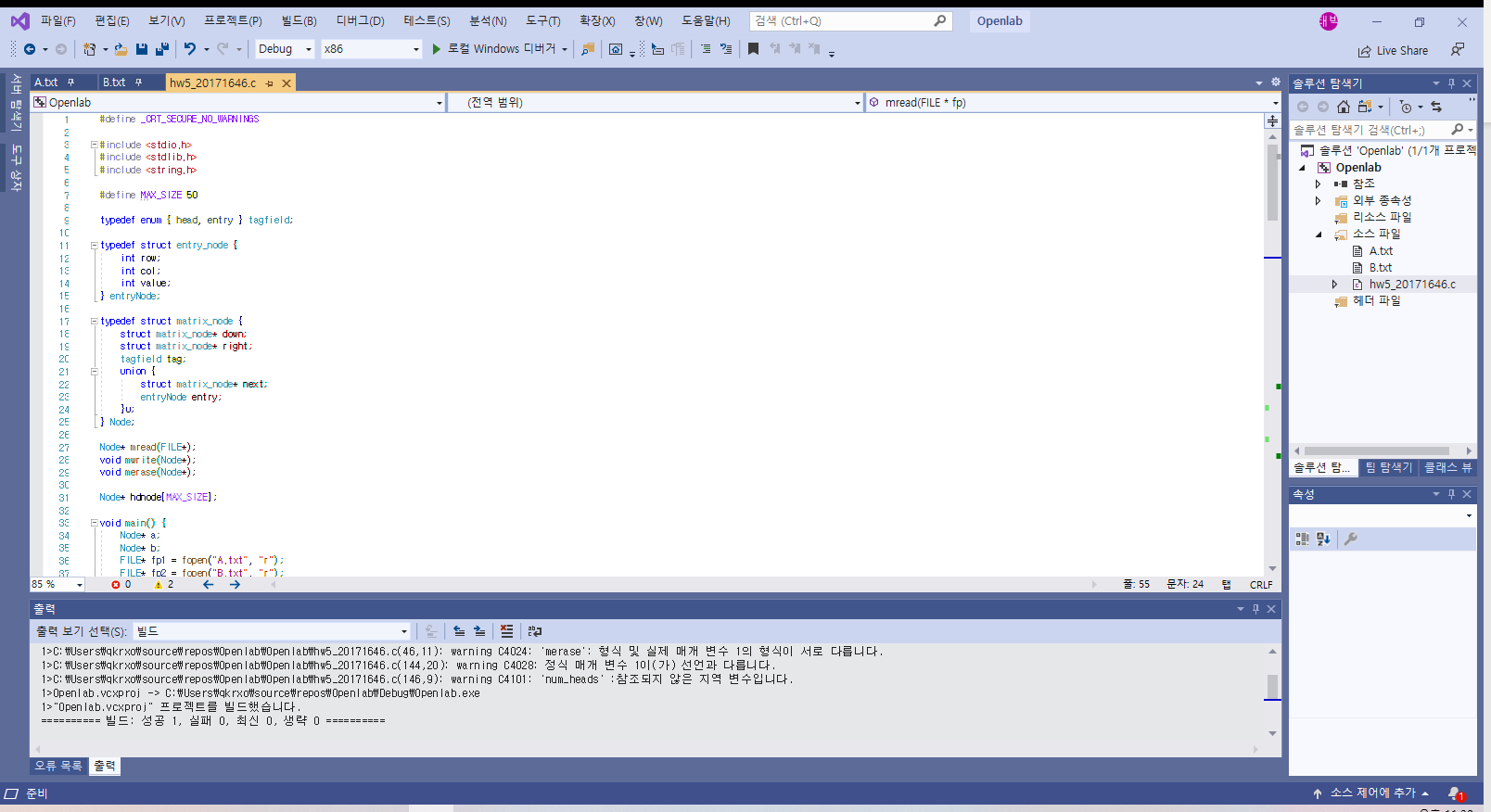
각 줄에 (행, 열, value)가 출력된다. merase(메모리해제)를 하고 난 이후 출력을 하면 Empty를 출력한다.

**- 코드 & 알고리즘 설명**

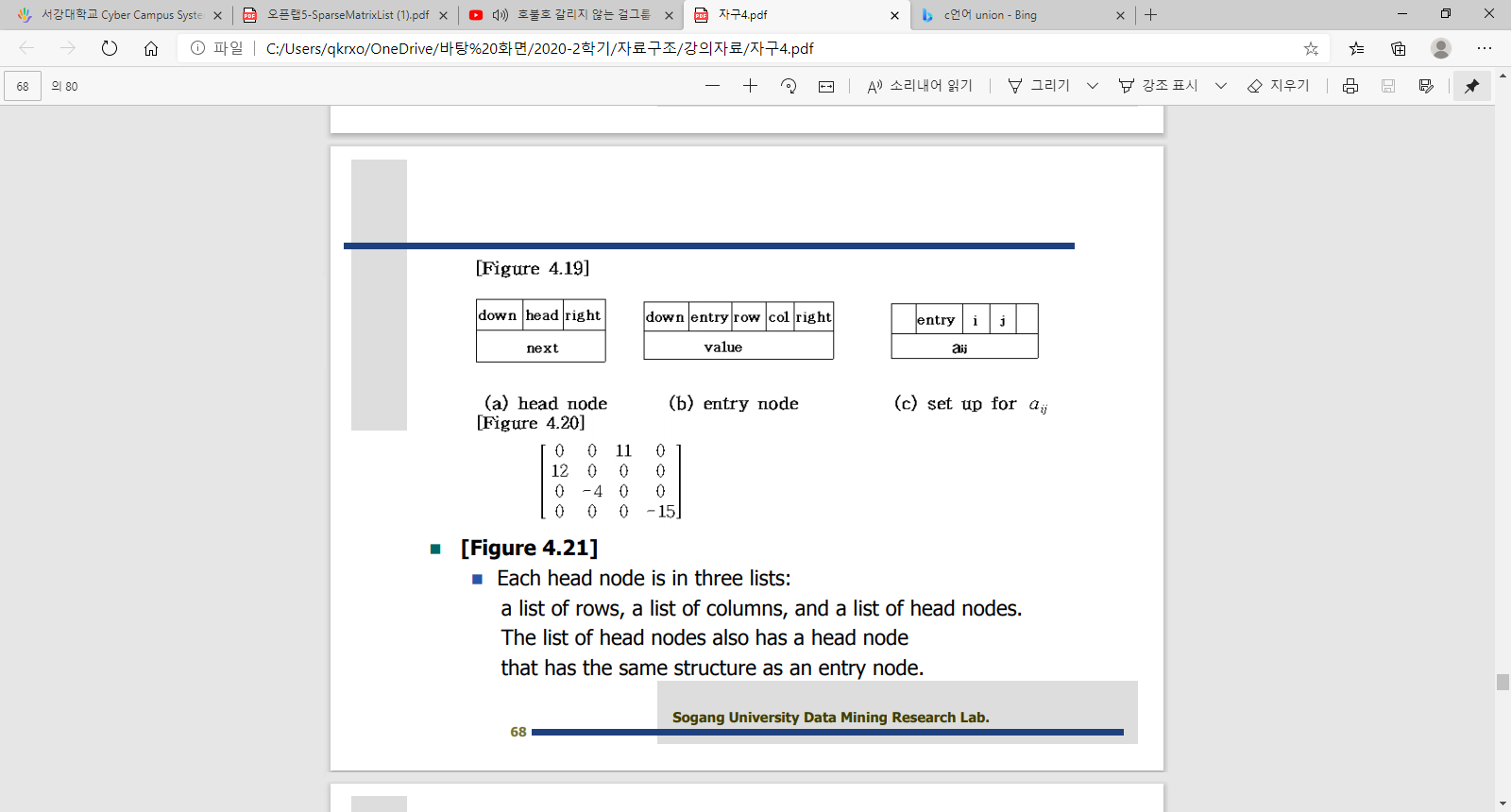
텍스트 파일에 저장된 배열을 읽어와 Linked List에 0이 아닌 값만 저장한다.



그림과 같이 Linked List를 만든다. 빨간색 원은 Special head node로 값이 존재하는 전체 행의 길이, 전체 열의 길이, 0이 아닌 value의 개수를 저장한다. 파란색 원은 각 head node를 의미하는데, 그림에서는 head node가 행 따로 열 따로 존재하는 것처럼 보이지만 실제로는 행과 열의 head node는 같은 node이며 연결하는 link인 right(행) 또는 down(열)에 따라 구분만 해 놓은 것이다. head node끼리도 next link를 이용하여 연결해준다.



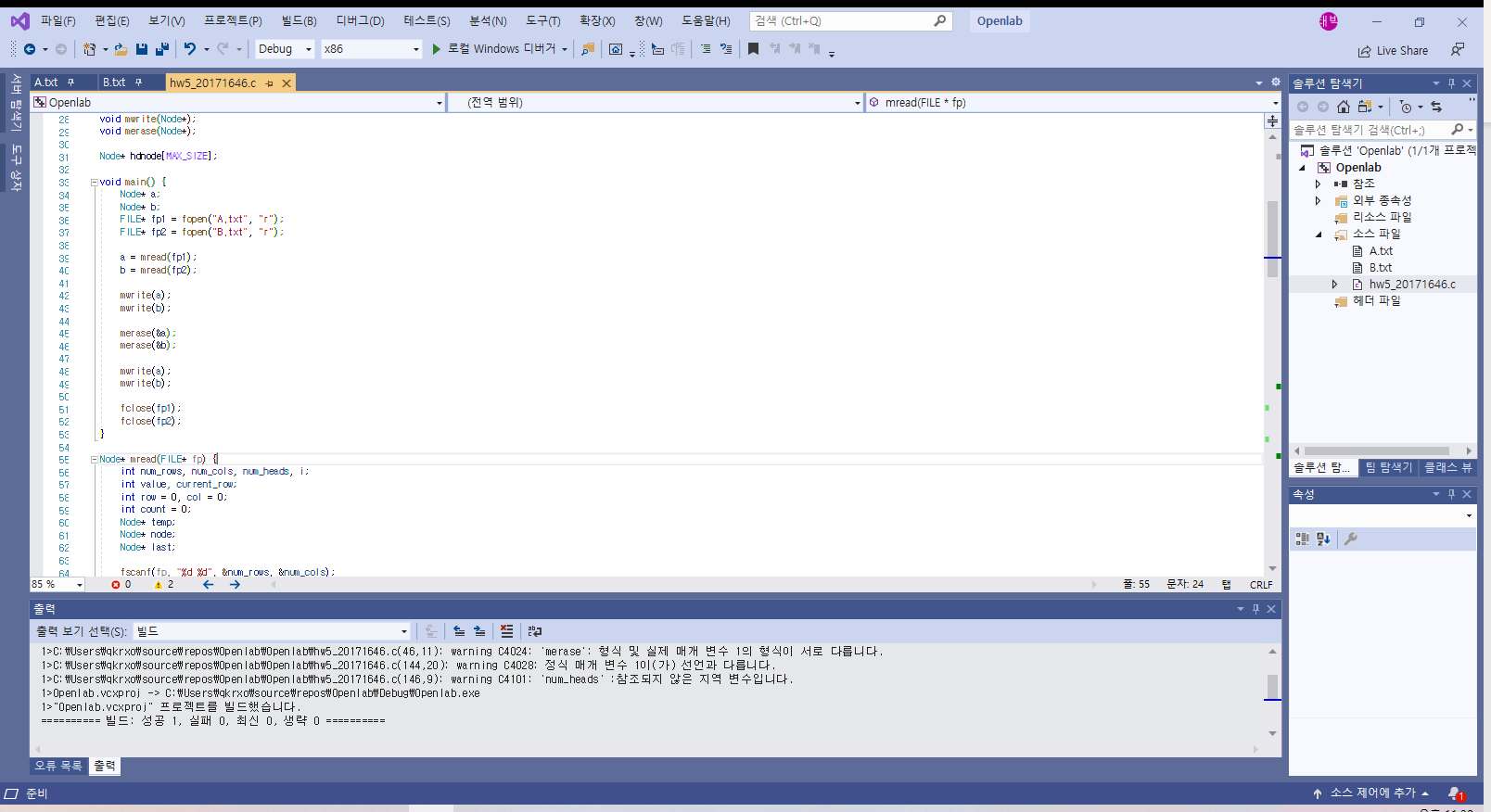
처음에 tagfield라는 열거형을 선언하였다. 이는 head노드인지, data값이 들어가있는 entry노드인지를 구분해준다. entry노드라면 노드에 값으로 row, col, value를 저장하기 위해 구조체 entry\_node를 선언하였다. 전체 node는 matrix\_node로 선언하여 link로 down, right를 가지게 선언하였으며 앞에서 말했던 tagfield열거형을 사용하기 위해 tagfield tag라는 변수를 넣어주었다. u라는 공용체를 선언하였는데, 이는 head node와 entry node의 생김새가 다르지만 따로 node구조체를 선언하지 않기 위함이다.



공용체를 이용하여 위의 그림에서 빨간색 동그라미로 표시한 부분을 같은 노드 구조체를 이용하여 표현할 수 있다.

함수로 Node\*를 return하는 mread, void형 mwrite와 merase를 선언하였다. 각각 Sparse Matrix Linked List를 만들어주는 역할, 만든 Sparse Matrix Linked List를 출력하는 역할, 메모리 해제를 하는 역할을 한다.

Node\*로 hdnode[MAX\_SIZE]를 선언하였는데, 이는 헤드 노드를 의미한다.



다음과 같이 메인 함수를 선언하였다. Node\* a와 b를 선언하였는데 이는 각기 “A.txt”에서 불러오는 Matrix와 “B.txt”에서 불러오는 Matrix를 나타낸다. mread함수를 이용하여 만들어진 Linked List의 Special head node를 각각 a와 b에 assign해주었다. 이후 mwrite를 이용하여 Linked List를 출력하였으며 중간에 한번 merase를 이용하여 a와 b가 나타내는 Linked List의 메모리 해제를 진행하였고 다시 한번 mwrite를 이용하여 출력하였다.

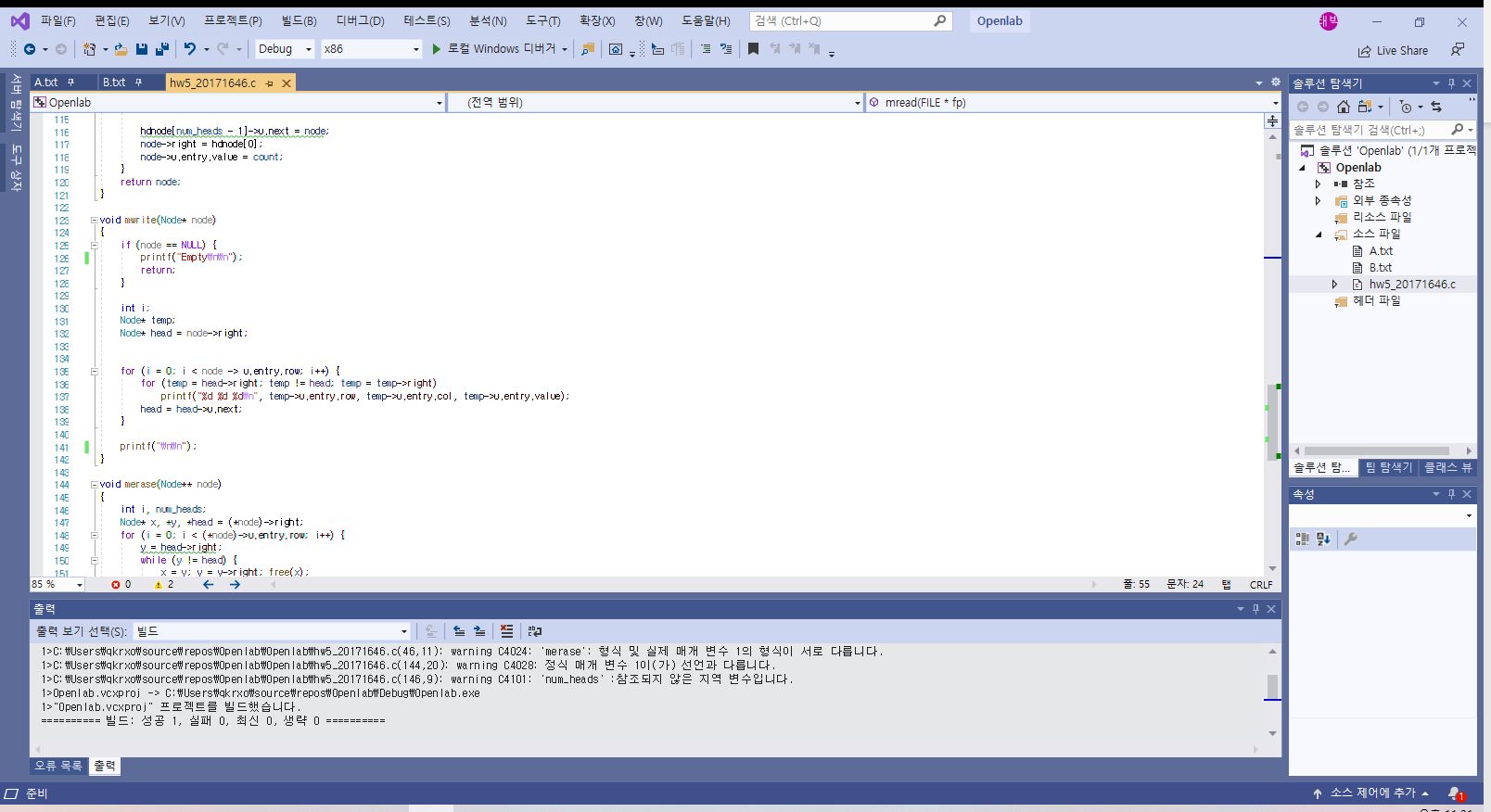
|  |  |
| --- | --- |
|  |  |

위와 같이 mread함수를 만들었다. 매개변수로 FILE\*형 변수를 받아 함수 내에서 텍스트 파일에 접근하여 배열을 read하였다. 텍스트 파일 첫 줄에 쓰여진 row와 col의 개수를 읽어 각각 num\_rows, num\_cols변수에 저장하였다. 이후 num\_heads변수에 num\_rows가 큰지 num\_cols가 큰지를 비교하여 큰 값을 assign하였다. 이후 Node\* node를 동적할당하였고 tag를 entry로 설정하였으며 행과 열의 개수를 저장하였다. node는 special head node를 나타내는데, tag를 head가 아닌 entry로 설정하여 행과 열, 0이 아닌 값의 개수를 저장할 수 있도록 구현하였다.

이후 if조건문을 사용하였는데 처음 if구문은 num\_head가 0일 때, 즉 행도 0이고 열도 0인 size가 없는 배열일 경우 node의 right link가 자기 자신을 가리키도록 하는 역할을 한다. else문에서는 size가 있는 배열을 처리한다. else문 처음에 for반복문을 사용하였는데, 이 반복문은 num\_heads만큼 Linked List에서 head노드를 생성하는 역할을 한다. 각 반복마다 temp라는 노드를 동적할당하여 tag를 head로 설정하였고 앞에서 선언한 hdnode배열에 temp를 assign하였으며 right와 next link가 자기 자신을 가리키도록 구현하였다. 이후 current\_row를 0으로, last를 hdnode[0]으로 설정하였는데 각각 현재 보고 있는 row와 노드를 의미한다.

그 다음 while반복문을 사용하였다. 반복문의 조건을 row < num\_rows로 설정하였다. row변수는 행렬 elements의 행을 의미한다. fscanf를 이용하여 텍스트파일에서 하나씩 값을 읽었다. 처음 if조건문은 col이 num\_cols랑 같은 경우, 즉 값을 읽다가 해당 행의 마지막 열에 도달하였을 때 last노드의 right를 같은 행의 head node와 연결시켜준다. 이후 current\_row를 하나 증가시키며 현재 보고 있는 node인 last를 다음 head node인 hdnode[current\_row]로 설정하고 row를 하나 증가시킨 뒤 col을 0으로 assign한다. 결국 최종적으로 Circular하게 Linked List를 구현하는 역할을 하는게 첫번째 if문이다. 두 번째 if문은 텍스트 파일을 읽다가 0이 아닌 값을 받았을 시 노드를 만들어 Linked List에 연결해주는 역할을 한다. temp노드를 동적할당하여 tag를 entry로 설정하였고, temp노드의 u.entry.row와 u.entry.col에 각각 현재 보고 있던 행과 열의 값을 나타내는 row와 col을 assign해주었으며 u.entry.value에 value를 넣어주었다. 이후 last와 temp노드를 연결해주고 last를 temp로 assign해준다. 그 다음 hdnode[col]->u.next와 temp노드를 연결해주는데, hdnode[col]->u.next는 temp와 같은 col에서 temp바로 위에 있는 노드를 가리킨다. 연결해준 뒤 hdnode[col]->u.next를 temp로 설정한다. 마지막으로 count를 1증가시키는데, count는 0이 아닌 elements의 개수를 의미한다. while문을 돌 때마다 항상 col을 1증가시키는데 Matrix에서 row major로 다음 elements를 보기 위함이다.

while문을 빠져나오면 나머지 처리들을 해준다. 우선 처음에 for반복문을 통해 hdnode[i]->u.next->down = hdnode[i]를 해주는데 이는 col기준으로 Circular하게 만들어주는 역할을 한다. 그 다음 for문을 통해 head node들을 연결해주며 마지막에 hdnode[num\_heads – 1]->u.next = node를 통해 마지막 헤드노드가 special head node와 연결되게 하여 헤드노드끼리도 Circular할 수 있도록 구현하였다. 이후 special head node의 right링크를 첫번째 헤드노드로 연결시키고 value에 0이 아닌 elements의 개수인 count를 assign한 뒤 최종적으로 special head node인 node를 return해준다.



위의 코드는 만들어진 Linked List를 출력해주는 mwrite함수이다. special head node인 node를 매개변수로 받는다. 처음에 node == NULL일 때, 즉 이미 메모리 해제가 모두 완료되어 special head node가 NULL일 때 “Empty”를 출력하고 함수를 종료시킨다. 그렇지 않다면 정상적으로 출력을 시켜주는데 row major로 출력하며 이중for문을 사용한다. 처음 for문은 i = 0 ~ node->u.entry.row, 즉 0번째 row부터 마지막 row까지 반복을 하며 두번째 for문은 Node\* temp를 사용하여 해당 row에서 첫 번째 element를 가리키는 head->right에서부터 다시 head로 돌아올 때까지 right링크를 타며 출력해준다. 해당 row에 있는 entry node들을 모두 출력하였다면 다음 row로 넘어가기 위해 head를 u.next링크를 이용하여 옮겨준다. 출력은 (row, col, value)순으로 출력이 되게 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Linked List의 메모리 해제를 해주는 merase함수이다. 처음에 Node\* head는 첫 번째 헤드노드를 가리킨다. 이후 for반복문이 나오는데 0번째 row부터 마지막 row까지 반복한다. y는 head->right 즉, 해당 row의 첫 번째 entry node를 가리키며 y가 right링크를 타면서 head로 돌아올 때까지 while문을 반복한다. while문 안에서 y의 이전 노드를 가리키는 x가 free되면서 해당 row의 entry node들이 모두 삭제되고 while문을 빠져나오면 최종적으로 x가 해당 row의 head node를 가리키게 한 뒤 메모리 해제를 해준다. 즉, 메모리 해제는 우선적으로 entry node들을 전부 free시킨 뒤 head node들을 free시키는 과정을 거친다. for문을 빠져나오면 남은 헤드노드들을 free시켜준다. y는 head로 assign해주는데, 남아있는 헤드노드가 존재한다면 while문을 통해 메모리 해제를 할 것이고 그렇지 않다면 head노드들 또한 Circular하기 때문에 y가 special head node인 \*node를 가리켜 while문을 실행하지 못할 것이다. 최종적으로 special head node를 free해주고 \*node = NULL을 해준다.