실습과제 4.1.1

[과제 설명]

원형 연결리스트란 단순 연결리스트에서 마지막 노드의 링크가 첫번째 노드를 가리키고 있는 연결리스트다. 원형 연결리스트를 구성하고, 출력한다. 이때, 자료구조(node)의 형태는 단순 연결리스트와 동일하나, 함수를 생성하고 삽입할 때 차이가 발생한다. 마지막 노드가 항상 첫 노드를 가리키고 있어야 하기 때문인데, 이것은 노드가 하나일 때도 마찬가지다. 따라서, 노드가 하나면, 그 노드가 자기 자신을 가리키고 있어야 한다.

[코드]

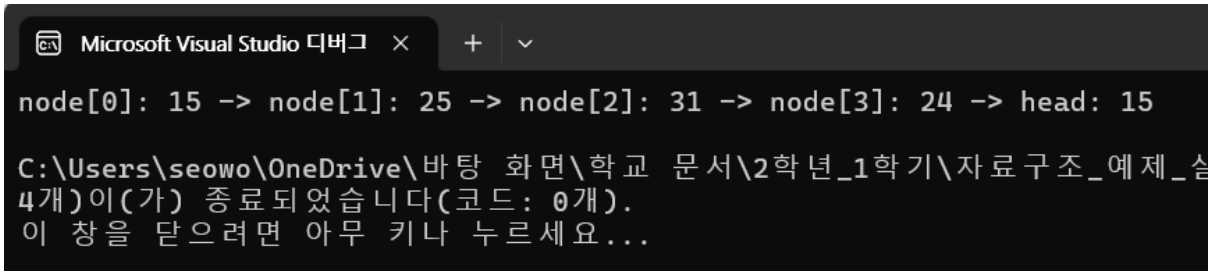
헤더 파일을 분리할 정도로 함수를 많이 사용하지 않았기도 하고 보고서를 작성할 때 필요 이상으로 길어지는 것을 방지하기 위해 이번 실습에 쓰지 않은 함수를 제외하고 한 코드로 첨부한다.

또한 이전 실습에서도 사용한 함수는 주석을 생략한다.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
  int data;
  struct Node* link;
}Node;

Node* Create_Node(int newData) {
  Node* newNode = (Node*)malloc(sizeof(Node));
  newNode->data = newData;
  newNode->link = NULL;

  return newNode;
}

void Append_Node(Node** head, Node* newNode) {
  if ((*head) == NULL) {
    //현재 head가 비어있으면, 새로운 노드를 head로 설정
    *head = newNode;
```

```
    (*head)->link = newNode; //원형 리스트기 때문에 자기 자신을 다시 가리킨다.
  }
  else {
    //비어있지 않다면, 노드의 제일 끝부분까지 탐색해서 새로운 노드 추가
    Node* tail = (*head);
    while (tail->link != (*head)) {
      tail = tail->link;
    }
    tail->link = newNode; //맨 마지막 노드가 새 노드를 가리키게 한다.
    newNode->link = (*head); //새 노드가 head를 가리키게 한다.
  }
}

void Print_Circluar_Linked_List(Node* head) {
  if (head == NULL) return; //head가 비어있으면, 함수 종료

  Node* iter = head;
  int i = 0;

  do {
    printf("node[%d]: %d", i, iter->data);
    iter = iter->link;
    if (iter != NULL) printf(" -> ");
    i++;
  } while (iter != head); //head로 돌아올 때까지 반복

  printf("head: %d", iter->data);
  //마지막 노드가 head로 연결되었는지 확인
  puts("");
}

int main(void) {
  Node* head = NULL;

  //원형 연결리스트 구성
  Append_Node(&head, Create_Node(15));
  Append_Node(&head, Create_Node(25));
  Append_Node(&head, Create_Node(31));
  Append_Node(&head, Create_Node(24));

  //원형 연결리스트 출력
  Print_Circluar_Linked_List(head);
}
```

[실행 결과]



[과제에 대한 고찰]

주변에서 봤던 프로그램의 기능 중에 원형 연결리스트가 사용되었을 거라고 떠오르는 것이 생각보다 많이 있었는데, 게임용 툴을 제공하는 사이트의 하나의 덱을 다 쓴 후 다음 덱 채우기, 한 바퀴를 돌면 다시 처음으로 돌아가는 전체 반복을 하는 음악 재생 앱 같은 상황에서 많이 사용되었던 것 같다.

이전에 자료구조응용실습 시간에서 원형 연결리스트를 먼저 해보았는데, 그 때는 일반 포인터 형식으로 구현했었다. 이번에는 더블 포인터를 이용하는 방식으로 구현해보았는데, 노드를 추가하는 코드는 일반 포인터로 한 코드보다 단순해서, 삭제하는 코드도 더 간단해질지 궁금해졌다. 나중에 조금 더 찾아보거나 학습해서 일반 연결리스트처럼 특정 노드를 삭제하거나 역순으로 만드는 코드도 구현해보고 싶다.