**Chapter 04 연결 리스트(Linked List)2**

**04-1 연결 리스트의 개념적인 이해**

: Ch03에서의 배열 은 메모리의 특성이 정적이므로 메모리의 길이 변경 불가능

* 필요로 하는 메모리 크기에 유연하게 대처할 수 있는 방안 필요

이를 보안하는 것이 연결리스트!

* 필요할 때마다 바구니의 역할을 하는 구조체 변수(노드)를 하나씩 동적 할당해서 이들을 연결
* 노드& 연결표현(데이터 저장 장소와 다른 변수를 가리키기 위한 장소로 구분하여 표현)

data1

텍스트이(가) 표시된 사진

자동 생성된 설명

\*LikedRead.c 예제 분석

#include <stdio.h>

#include <stdlib.h>

typedef struct \_node

{

int data;

struct \_node \* next;

} Node;

int main(void)

{

**연결리스트의 삼형제**

**head**: 첫번째 노드 가리키기 위함, 반드시 필요

**tail**: 마지막 노드 가리키기 위함, 없을 수도

**cur**: 순차접근 위함/ 저장된 데이터 조회에 사용

Node \* head = NULL; // NULL 포인터 초기화

Node \* tail = NULL;

Node \* cur = NULL;

Node \* newNode = NULL;

int readData;

/\*\*\*\* 데이터를 입력 받는 과정 \*\*\*\*/

while(1)

{

printf("자연수 입력: ");

scanf("%d", &readData);

if(readData < 1) // 탈출조건 :data 값이 <1 이면 노드 생성X

break;

/\*\*\* 노드의 추가과정 \*\*\*/

newNode = (Node\*)malloc(sizeof(Node)); // 노드 생성

newNode->data = readData; // 노드에 데이터 값 저장

newNode->next = NULL; // 노드의 next를 NOLL로 초기화

if(head == NULL) // 첫번쨰 노드라면!

head = newNode; //첫번째 노드가 head가리키게 함

else // 두번째 이후 노드라면!

tail->next = newNode; //tail이 가리키는 노드의 뒤에 새로운 노드 연결

tail = newNode; // 노드의 끝을 tail이 가리키게 함

}

printf("\n");

cur이 다음과 같이 모든 노드를 가리키며 이동

/\*\*\*\* 입력 받은 데이터의 출력과정 \*\*\*\*/

printf("입력 받은 데이터의 전체출력! \n");

if(head == NULL)

{

printf("저장된 자연수가 존재하지 않습니다. \n");

}

else

{

cur = head; // cur이 첫번째 노드 가리킴

printf("%d ", cur->data); // 첫 번째 데이터 출력

while(cur->next != NULL) // 연결된 노드가 존재한다면 (두 번째 이후의 데이터 출력)

{

cur = cur->next; //cur이 다음 노드 가리키게 함

printf("%d ", cur->data); //cur이 가리키는 노드 출력

}

}

printf("\n\n");

/\*\*\*\* 메모리의 해제과정 (데이터 삭제) \*\*\*\*/

if(head == NULL)

{

return 0; // 삭제할 노드가 존재하지 않는다.

}

else

{

Node \* delNode = head; // 두개의 포인터 변수 추가선언 -> head가 가리키는 노드 그냥 삭제하면

Node \* delNextNode = head->next; 그 다음 노드 접근 불가능 하므로

printf("%d을(를) 삭제합니다. \n", head->data);

free(delNode); // 첫 번째 노드의 삭제

while(delNextNode != NULL) // 두번째 이후의 노드 삭제위한 반복문

{

delNode = delNextNode; // dNN에 있는 값 dN에 저장

delNextNode = delNextNode->next;

printf("%d을(를) 삭제합니다. \n", delNode->data);

free(delNode); // 두 번째 이후의 노드 삭제

}

}

return 0;

}