자료구조 실습과제

과일 리스트 관리 프로그램

5671144 강범창

1.1 양방향 연결리스트 로 과일 리스트 관리 프로그램을 작성하기 위하여 우선 구조체를 작성(구현)합니다. 이 때 양방향 리스트 기에 앞(prev) 뒤(next) data(문자)으로 구성되어 있습니다.

```
typedef struct Node {
        char* data;
        struct Node* prev;
        struct Node* next;
} Node;
```

1.2 현재 선택된 아이템을 확인하기 위해 currentNode를 선언합니다.

```
Node* currentNode = NULL;
```

1.2 양방향 리스트를 구현합니다.

insert 함수입니다. 우선 양방향 리스트에서 node를 추가 하기위해 이전 node와 새롭게 추가할 값을 받습니다. 이후 새로운 노드를 만든 후 text 데이터를 새로운 노드의 data으로 넣어준 다음, 이 새로운 노드를 이전의 노드 위치 다음에 넣어 줍니다. 이때 이전의 노드의 다음 노드는 이 노드가 되며 이전 노드의 다음 노드 위치는 새롭게 만든 노드의 다음 노드가 되며 노드 리스트 중간에 새로운 노드를 추가합니다.

```
Node* insert(Node* prevNode, char* data) {
        Node* newNode = (Node*)malloc(sizeof(Node));
        newNode->data = malloc(strlen(data) + 1); // 문자열 마지막에 ₩0 이 있기에 +1 을 해줍니다.
        strcpy(newNode->data, data);

        // 리스트 중간에 값을 넣습니다.
```

```
        newNode->prev = prevNode; // new node 의 이전 node 를 현재
연결하고자하는 노드로 지정합니다.
        newNode->next = prevNode->next;

        prevNode->next->prev = newNode;
        prevNode->next = newNode;

        return newNode;
}
```

delete 함수입니다. Node 의 head 와 removed 즉 지워질 node를 입력 받으며 만약 head 와 removed 가 같다면 head 를 지워버리기에 삭제 함수를 종료합니다 아니라면, removed의 왼쪽 오른쪽에 있는 node 들끼리 연결을 시켜준 다음 지워질 노드를 삭제합니다.

```
void delete(Node* head, Node* removed) {
        if (removed == head) return;
        removed->prev->next = removed->next;
        removed->next->prev = removed->prev;
        free(removed);
}
```

Init 함수입니다. HeadNode를 입력 받으며 headnode를 초기화 시켜줍니다;

```
void init(Node* head) {
        head->prev = head;
        head->next = head;
}
```

Print List 함수입니다. 리스트의 노드를 하나 하나 다음으로 넘기면서 출력하며 만약 current 지금 선택된 노드일 경우 [0] 텍스트를 추가로 출력시켜줍니다.

```
void printList(Node* head) {
        Node* p;
```

```
        for (p = head->next; p != head; p = p->next) {
                printf("%s%s₩n", p->data, p == currentNode ? " [O]" : "");
        }
}
```

1.3 메인에서 입력을 받아 처리하는 부분입니다.

n의 경우 – 다음 node를 current node에 넣어줘 current node를 수정합니다.

p의 경우 – 이전 node를 current node에 넣어줘 current node를 수정합니다.

d의 경우 – current node를 다음 node로 변경 후 이전 node를 delete함으로써 현제 current node를 삭제합니다.

i의 경우 – 문자열을 입력 받아 처음 과일 이 추가된 경우 그 과일을 current node로 둡니다 아니라면 current node 이후에 문자열 데이터를 가진 node를 추가시켜줍니다.

o의 경우 – 리스트를 출력합니다 current node의 경우 [0]을 추가적으로 출력 을 하여 선택 되어있는 node를 확인할 수 있습니다.

e의 경우 – 프로그램을 종료합니다.

나머지의 경우 – Invalid menu. 메시지와 함께 다시 메뉴를 출력합니다.

```
int main() {
        Node* head = (Node*)malloc(sizeof(Node));
        char command;
        char data[100];
        init(head);

        while (true)
        {
                printf("==== Menu ====₩n");
                printf("n) next fruit₩n");
                printf("p) previous fruit₩n");
```

```
		printf("d) delete the current fruit₩n");
		printf("i) insert fruit after current fruit₩n");
		printf("o) output the fruit list ( current fruit [0] )₩n");
		printf("e) exit the program₩n");
		printf("==============₩n");
		printf("Select a menu : ");
		scanf(" %c", &command); // 공백을 추가하여 개행문자를 삭제합니다!

		switch (command)
		{
		case 'n':
			if (currentNode) {
				currentNode = currentNode->next;
				break;
			}
			break;
		case 'p':
			if (currentNode) {
				currentNode = currentNode->prev;
				break;
			}
			break;
		case 'd':
			currentNode = currentNode->next;
			delete(head, currentNode->prev);
			break;
		case 'i':
			printf("Enter the name of the fruit to add : ");
			scanf("%s", data);
			if (!currentNode) { // 처음 과일을 추가하면 해당 과일이
선택된 상태로 과일 리스트를 관리합니다.
				currentNode = insert(head, data);
				break;
			}
			insert(currentNode, data);
```

```
                    break;
                case 'o':
                    printList(head);
                    break;
                case 'e':
                    exit(0);
                    break;
                default:
                    printf("Invalid menu. ₩n");
                    break;
                }
                printf("₩n");
        }
}
```

아래는 실행 결과입니다.

https://youtu.be/0asTCDgO1tY