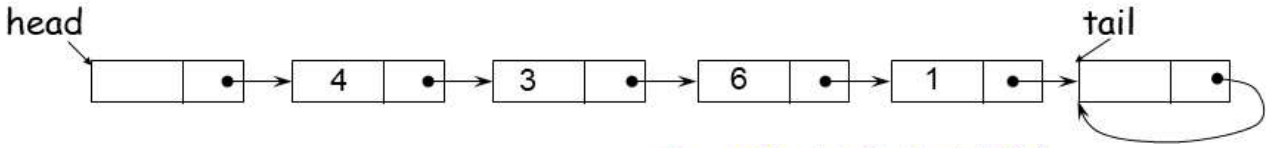
자료구조 3주차 과제

제출일: 2022-09-19
학번/이름: 2016121150 / 윤준영

문1) 다음과 같은 단순연결리스트를 만들고, 리스트 내의 원소들을 출력한 후 주어진 리스트에서 가장 큰 정수값을 출력하는 프로그램을 작성하시오. 정수 단순연결리스트의 헤드 노드 주소를 받아 가장 큰 원소의 값을 반환하는 함수인 int max_element(node *head)를 작성하여 사용하시오. (비어있는 리스트이면 -999999를 반환함)



HW3 > C hw3_1.c > ...

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

typedef struct _node {
    int key;
    struct _node *next;
} node;

node* init_list(void) {
    node *head = (node*)malloc(sizeof(node));
    node *tail = (node*)malloc(sizeof(node));
    // malloc이 NULL을 리턴할 경우 처리 방법은?
    if (head==NULL || tail==NULL) return NULL;
    head->next = tail;
    tail->next = tail;
    printf("리스트 생성 완료!\n");
    return head;
}

node *insert_after(int k, node* t) {
    node *s;
    if (!t || t->next == t) return t;
    s = (node*)malloc(sizeof(node));
    if (s==NULL) return s;
    s->key = k;
    s->next = t-> next;
    t->next = s;
    return s;
}

void print_list(node* h) {
    h = h->next;
    printf("리스트 원소: ");
    while(h != h->next) {
        printf("%d ", h->key);
        h = h-> next;
    }
    printf("\n");
}

int max_element(node *h) {
    int max = -999999;
    h = h->next;
    while (h->next != h){
        if (h->key >= max) max = h->key;
        h = h->next;
    }
    printf("가장 큰 원소의 값: %d\n", max);
    return max;
}

int main() {
    node *head = init_list();
    //node *p;
    insert_after(1,head);
    insert_after(6,head);
    insert_after(3,head);
    insert_after(4,head);

    print_list(head);
    max_element(head);
}
```

max_element 함수는 head node를 h로 받아 사용한다. 먼저 max값을 -999999로 설정한 후 h=h->next를 이용하여 리스트의 첫 원소부터 탐색을 시작한다. 탐색하는 원소(key)가 max 값 보다 클 경우, max는 해당 원소의 key로 대체되고 h=h->next를 이용하여 다음 원소를 탐색할 준비를 한다. 만약 h가 tail일 경우, h->next=h가 되므로 while문을 빠져나가고

max를 반환한다. 만약 빈 리스트일 경우, 첫 while문에서 h=tail이기 때문에 max는 대체되지 않고 처음 지정된 -999999를 반환하게 된다.

프로그램 시행 결과는 다음과 같다.

```
리스트 생성 완료!
리스트 원소: 4 3 6 1
가장 큰 원소의 값: 6
[*] 터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

빈 리스트의 경우 다음과 같다.

```
리스트 생성 완료!
리스트 원소:
가장 큰 원소의 값: -999999
[*] 터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

문2) 숫자를 입력받아 이 숫자로 이루어진 단순연결 리스트를 만들고, 리스트 원소들의 합을 계산하는 프로그램을 작성하시오.(비어있는 리스트이면 0을 리턴함)

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

typedef struct _node {
    int key;
    struct _node *next;
} node;

node* init_list(void) {
    node *head = (node*)malloc(sizeof(node));
    node *tail = (node*)malloc(sizeof(node));
    if (head==NULL || tail==NULL) return NULL;
    head->next = tail;
    tail->next = tail;
    printf("리스트 생성 완료! ");
    return head;
}

node *insert_after(int k, node* t) {
    node *s;
    if (!t || t->next == t) return t;
    s = (node*)malloc(sizeof(node));
    if (s==NULL) return s;
    s->key = k;
    s->next = t-> next;
    t->next = s;
    return s;
}

void print_list(node* h) {
    h = h->next;
    printf("(리스트 원소: ");
    while(h != h->next) {
        printf("%d ", h->key);
        h = h-> next;
    }
    printf(")\n");
}

int sum_list(node *h) {
    int sum = 0;
    h = h->next;
    while (h->next != h){
        sum += h->key;
        h = h->next;
    }
    printf("원소의 합: %d\n", sum);
    return sum;
}

node* make_list(void) {
    int n;
    printf("원소의 개수를 입력하세요: ");
    scanf("%d", &n);
    int a[n];
    printf("숫자를 입력하세요: ");
    for (int i = 0; i < n; i++) scanf("%d", &a[i]);
    node *head = init_list();
    node *l = head;
    for (int i = 0; i < n; i++){
        //printf("%d, %d ", i, a[i]);
        l = insert_after(a[i],l);
    }
    print_list(head);
    return head;
}

int main() {
    node *head = make_list();
    sum_list(head);
}
```

원소의 개수와 숫자를 입력하면 리스트를 만들어주는 make_list함수를 작성하여서 사용하였다. 원소의 개수를 입력하면, for문을 통하여 입력받은 원소의 개수만큼 scan을 하여 array로 저장한 후, 문1)에서 사용한 init_list()와 insert_after를 이용하여 list를 생성하고 각 list의 원소를 array의 원소를 이용하여 list를 확장하였다.
또한 sum_list함수를 작성하여, 노드 h가 tail이 될 때까지 첫 원소부터 마지막까지 sum에 더하여 저장하고 반환하였다. 만약 빈 리스트이면, while문에 진입하는 h는 tail이 되기 때문에 초기화 한 sum=0이 그대로 반환된다.

실행한 결과는 다음과 같다.

```
원소의 개수를 입력하세요: 5
숫자를 입력하세요: 4 5 6 7 10
리스트 생성 완료! (리스트 원소: 4 5 6 7 10 )
원소의 합: 32
* 터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

원소의 개수를 0으로 시행한 결과는 다음과 같다.

```
원소의 개수를 입력하세요: 0
숫자를 입력하세요: 리스트 생성 완료! (리스트 원소: )
원소의 합: 0
* 터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

문3) 1부터 10까지의 숫자로 이루어진 이중연결 리스트를 만들고, 이중연결 리스트의 노드들의 순서를 반대로 하는 reverse_list 함수를 작성하여 위 리스트에 적용한 후 결과를 출력하시오. (head와 tail을 전역변수로 사용)

HW3 > C hw3_3.c > ...

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

typedef struct _dnode {
    int key;
    struct _dnode *prev;
    struct _dnode *next;
} dnode;

dnode *head, *tail;

void init_dlist(void) {
    head = (dnode*)malloc(sizeof(dnode));
    tail = (dnode*)malloc(sizeof(dnode));
    head->next = tail;
    head->prev = head;
    tail->next = tail;
    tail->prev = head;
    printf("리스트 생성 완료! \n");
}

dnode *insert_dnode_ptr(int k, dnode *t) {
    /* insert k, before t */
    dnode *i;
    if (t==head) return NULL;
    i = (dnode*)malloc(sizeof(dnode));
    i->key = k;
    t->prev->next = i;
    i->prev = t->prev;
    t->prev = i;
    i->next = t;
    return i;
}

void print_dlist(dnode *p) {
    while(p != tail) {
        printf("%d ", p->key);
        p = p->next;
    }
    printf("\n");
}
```

```
43
44  void reverse_list(dnode *h){
45      // find number elements of dlist: n
46      int n = 0;
47      while (h->next != tail){
48          h = h->next;
49          n = n+1;
50      }
51      // now h = tail->prev
52      // make reverse_array
53      int *s = (int *)calloc(n, sizeof(int));
54      for (int j = 0; j<n; j++){
55          s[j] = h->key;
56          h = h->prev;
57      }
58      // now h = head
59      h = h->next;
60      for (int j = 0; j<n; j++){
61          h->key = s[j];
62          h = h->next;
63      }
64      free(s);
65  }
```

```
66
67  int main() {
68      int i;
69      init_dlist();
70      for (i=1; i<=10; i++) {
71          insert_dnode_ptr(i,tail);
72      }
73      printf("리스트 원소: ");
74      print_dlist(head->next);
75      reverse_list(head);
76      printf("reverse_list 결과: ");
77      print_dlist(head->next);
78
79  }
```

reverse_list함수를 사용하여 list의 원소의 개수를 구하고, 역순으로 array에 저장한 후, 다시 역순으로 저장된 array의 값을 list의 key에 저장하는 함수를 만들었다.
reverse_list함수는 head를 입력받아 h로 사용한다. 먼저 원소의 개수를 나타내는 n을 0으로 초기화 한 후, h를 next로 이동하며, tail이 되기 전까지 n을 +1시킨다. 그렇게 되면, h는 tail 전의 마지막 node를 의미하게 된다. 이제 마지막 원소부터 앞으로 이동하며, 할당된 array s에 저장한다. 그러면 h는 head node가 되고 h=h->next를 이용하여 첫 원소로 h의 위치를 변경하고, 다시 반복문을 이용하여 처음 원소부터 뒤로 이동하여 array의 원소를 h의 key로 저장한다. 마지막으로 동적 할당된 array s의 할당을 풀어준다.

프로그램 실행 결과는 다음과 같다.

```
리스트 생성 완료!
리스트 원소: 1 2 3 4 5 6 7 8 9 10
reverse_list 결과: 10 9 8 7 6 5 4 3 2 1
* 터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```