|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 6, 7장 실습( O ) 과제( ) | 제출일자 | 2022.  11 .    03 . |
| 학번 | 201911608 | 이름 | 김지환 |

|  |
| --- |
| 6-1. 프로그램 6.9  #include <stdio.h>  #include <stdlib.h>  typedef struct ListNode {  int coef;  int expon;  struct ListNode\* link;  } ListNode;  typedef struct ListType {  int size;  ListNode\* head;  ListNode\* tail;  } ListType;    void error(char\* message) {  fprintf(stderr, "%s\n", message);  exit(1);  }  ListType\* create() {  ListType\* plist = (ListType\*)malloc(sizeof(ListType));  plist->size = 0;  plist->head = plist->tail = NULL;  return plist;  }  void insert\_last(ListType\* plist, int coef, int expon) {  ListNode\* temp = (ListNode\*)malloc(sizeof(ListNode));  if (temp == NULL) error("메모리 할당 에러");  temp->coef = coef;  temp->expon = expon;  temp->link = NULL;  if (plist->tail == NULL) plist->head = plist->tail = temp;  else {  plist->tail->link = temp;  plist->tail = temp;  }  plist->size++;  }  void poly\_add(ListType\* plist1, ListType\* plist2, ListType\* plist3) {  ListNode\* a = plist1->head;  ListNode\* b = plist2->head;  int sum;  while (a && b) {  if (a->expon == b->expon) {  sum = a->coef + b->coef;  if (sum != 0) insert\_last(plist3, sum, a->expon);  a = a->link; b = b->link;  }  else if (a->expon > b->expon) {  insert\_last(plist3, a->coef, a->expon);  a = a->link;  }  else {  insert\_last(plist3, b->coef, b->expon);  b = b->link;  }  }  for (; a != NULL; a = a->link) insert\_last(plist3, a->coef, a->expon);  for (; b != NULL; b = b->link) insert\_last(plist3, b->coef, b->expon);  }  void poly\_print(ListType\* plist) {  ListNode\* p = plist->head;  printf("polynomial = ");  for (; p; p = p->link) printf("%d^%d + ", p->coef, p->expon);  printf("\n");  }  int main(void) {  ListType\* list1, \* list2, \* list3;  list1 = create();  list2 = create();  list3 = create();  insert\_last(list1, 3, 12);  insert\_last(list1, 2, 8);  insert\_last(list1, 1, 0);  insert\_last(list2, 8, 12);  insert\_last(list2, -3, 10);  insert\_last(list2, 10, 6);  poly\_print(list1);  poly\_print(list2);  poly\_add(list1, list2, list3);  poly\_print(list3);  free(list1); free(list2); free(list3);  }  실행결과    -> 다항식을 가진 각 노드 간의 덧셈 연산 결과이다.  expon이 같은 coef만 연산을 수행한다.  연산 수행 후 빈 노드에 수행 결과를 insert한다. |
| 7-1. 프로그램 7.3  #include <stdio.h>  #include <stdlib.h>  typedef int element;  typedef struct ListNode {  element data;  struct ListNode\* link;  } ListNode;  void print\_list(ListNode\* head)  {  ListNode\* p;  if (head == NULL) return;  p = head->link;  do {  printf("%d->", p->data);  p = p->link;  } while (p != head);  printf("%d->", p->data);  }  ListNode\* insert\_first(ListNode\* head, element data) {  ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));  node->data = data;  if (head == NULL) {  head = node;  node->link = head;  }  else {  node->link = head->link;  head->link = node;  }  return head;  }  ListNode\* insert\_last(ListNode\* head, element data) {  ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));  node->data = data;  if (head == NULL) {  head = node;  node->link = head;  }  else {  node->link = head->link;  head->link = node;  head = node;  }  return head;  }  int main(void) {  ListNode\* head = NULL;  head = insert\_last(head, 20);  head = insert\_last(head, 30);  head = insert\_last(head, 40);  head = insert\_first(head, 10);  print\_list(head);  return 0;  }  실행결과    -> 원형 리스트의 결과대로 잘 출력되었다.  20  20, 30  20, 30, 40  10, 20, 30, 40 |
| 7-2. QUIZ (227쪽) 2번 문제를 작성해서 테스트  02. 원형 연결 리스트에 존재하는 노드의 수를 계산하는 함수 get\_length()를 작성하라.  #include <stdio.h>  #include <stdlib.h>  typedef int element;  typedef struct ListNode {  element data;  struct ListNode\* link;  } ListNode;  void print\_list(ListNode\* head)  {  ListNode\* p;  if (head == NULL) return;  p = head->link;  do {  printf("%d->", p->data);  p = p->link;  } while (p != head);  printf("%d->", p->data);  }  ListNode\* insert\_first(ListNode\* head, element data) {  ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));  node->data = data;  if (head == NULL) {  head = node;  node->link = head;  }  else {  node->link = head->link;  head->link = node;  }  return head;  }  ListNode\* insert\_last(ListNode\* head, element data) {  ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));  node->data = data;  if (head == NULL) {  head = node;  node->link = head;  }  else {  node->link = head->link;  head->link = node;  head = node;  }  return head;  }  int get\_length(ListNode\* head) {  int cnt;  element data = head->data;  head->data = 9999;  element end = head->data;  for (head->link, cnt=1; head->link->data != 9999; head = head->link, cnt++);  head->link->data = data;  return cnt;  }  int main(void) {  ListNode\* head = NULL;  head = insert\_last(head, 20);  head = insert\_last(head, 30);  head = insert\_last(head, 40);  head = insert\_first(head, 10);  head = insert\_last(head, 20);  head = insert\_last(head, 30);  head = insert\_last(head, 40);  head = insert\_first(head, 10);  print\_list(head);  printf("\nlist의 size = %d\n", get\_length(head));  return 0;  }  실행결과    -> getlength(list, 0) ADT구현  head와 tail의 값을 가져와서 시작과 끝을 계산,  head와 tail이 같거나, 리스트 중간에 head와 tail의 값과 같은 값이 존재할 경우  리스트의 size가 아닌 head부터 head와 같은 값까지의 size를 구하는 반례가 존재.  head값을 임의로 변경. 실제에서는 해쉬값으로 변경할 수 있음. |
| 7-3. 프로그램 7.4  #include <stdio.h>  #include <stdlib.h>  typedef char element[100];  typedef struct ListNode {  element data;  struct ListNode\* link;  } ListNode;  typedef struct CListType {  ListNode\* head;  } CListType;  void print\_list(CListType\* L) {  ListNode\* p;  if (L->head == NULL) return;  p = L->head->link;  do {  printf("%s->", p->data);  p = p->link;  } while (p != L->head);  printf("%s->\n", p->data);  }  void insert\_first(CListType\* L, element data) {  ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));  strcpy(node->data, data);  if (L->head == NULL) {  L->head = node;  node->link = L->head;  }  else {  node->link = L->head->link;  L->head->link = node;  }  }  int main(void) {  CListType list = { NULL };  insert\_first(&list, "KIM");  insert\_first(&list, "PARK");  insert\_first(&list, "CHOI");  ListNode\* p = list.head;  for (int i = 0; i < 10; i++) {  printf("현재 차례=%s \n", p->data);  p = p->link;  }  return 0;  }  실행결과    ->  멀티게임의 구현 구조가 원형리스트 이므로 KIM->CHOI->PARK이 반복된다. |
| 7-4. 위 7-1번 문제에서 만들어진 원형 연결 리스트에서 처음과 끝에서 노드를 삭제하는 함수와 테스트 프로그램  ADT 구현  ListNode\* delete\_last(ListNode\* head);  ListNode\* delete\_first(ListNode\* head);  -> last, first 구현 후 free  include <stdio.h>  #include <stdlib.h>  typedef int element;  typedef struct ListNode {  element data;  struct ListNode\* link;  } ListNode;  void print\_list(ListNode\* head)  {  ListNode\* p;  if (head == NULL) {  printf("LIST가 비었습니다.\n");  return;  }  p = head->link;  while (p != head) {  printf("%d->", p->data);  p = p->link;  }  printf("%d\n", p->data);  }  ListNode\* insert\_first(ListNode\* head, element data) {  ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));  node->data = data;  if (head == NULL) {  head = node;  node->link = head;  }  else {  node->link = head->link;  head->link = node;  }  return head;  }  ListNode\* insert\_last(ListNode\* head, element data) {  ListNode\* node = (ListNode\*)malloc(sizeof(ListNode));  node->data = data;  if (head == NULL) {  head = node;  node->link = head;  }  else {  node->link = head->link;  head->link = node;  head = node;  }  return head;  }  int get\_length(ListNode\* head) {  int cnt;  element data = head->data;  head->data = 9999;  element end = head->data;  for (head->link, cnt=1; head->link->data != 9999; head = head->link, cnt++);  head->link->data = data;  return cnt;  }  ListNode\* delete\_last(ListNode\* head) {  ListNode\* p = head;  int i = 0;  if (head == NULL) {  printf("삭제할 LIST가 없습니다.\n");  return NULL;  }  if (get\_length(head) == 1) return NULL;  for (i, head; i < get\_length(head) - 1; i++, head = head->link);  head->link = p->link;  free(p);  return head;  }  ListNode\* delete\_first(ListNode\* head) {  ListNode\* p = head;  if (head == NULL) {  printf("삭제할 LIST가 없습니다.\n");  return NULL;  }  if (get\_length(head) == 1) return NULL;  head->link = head->link->link;  free(p);  return head;  }  int main(void) {  ListNode\* head = NULL;  head = insert\_last(head, 20);  head = insert\_last(head, 30);  head = insert\_last(head, 40);  head = insert\_first(head, 10);  print\_list(head);  head = delete\_last(head); print\_list(head);  head = delete\_first(head); print\_list(head);  return 0;  }  실행결과    LIST 값 변경 후    -> LIST가 하나 존재할 경우 HEAD = NULL, LIST가 존재하지 않을 경우까지 구현 |