|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 6장 실습( O ) 과제( ) | 제출일자 | 2022.  11 .    02 . |
| 학번 | 201911608 | 이름 | 김지환 |

|  |
| --- |
| 1. Lab 단어들을 저장하는 연결리스트 만들기 (200쪽)  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  typedef struct {  char name[100];  } element;  typedef struct {  element data;  struct ListNode\* link;  } ListNode;  void error(char\* message) {  fprintf(stderr, "$s\n", message);  exit(1);  }  ListNode\* insert\_first(ListNode\* head, element value) {  ListNode\* p = (ListNode\*)malloc(sizeof(ListNode));  p->data = value;  p->link = head;  head = p;  return head;  }  void print\_list(ListNode\* head) {  for (ListNode\* p = head;p != NULL;p = p -> link)  printf("%s -> ", p->data.name);  printf("NULL \n");  }  int main(void) {  ListNode\* head = NULL;  element data;  strcpy(data.name, "APPLE");  head = insert\_first(head, data);  print\_list(head);  strcpy(data.name, "KIWI");  head = insert\_first(head, data);  print\_list(head);  strcpy(data.name, "BANANA");  head = insert\_first(head, data);  print\_list(head);  return 0;  }  실행결과    -> 기존에 int 자료형으로 받던 element의 자료형을 문자열로 변경했다. |
| 2. Lab 특정한 값들을 탐색하는 함수를 작성 (202쪽)  #include <stdio.h>  #include <stdlib.h>  typedef int element;  typedef struct {  element data;  struct ListNode\* link;  } ListNode;  ListNode\* insert\_first(ListNode\* head, element value) {  ListNode\* p = (ListNode\*)malloc(sizeof(ListNode));  p->data = value;  p->link = head;  head = p;  return head;  }  void print\_list(ListNode\* head) {  for (ListNode\* p = head;p != NULL; p = p->link)  printf("%d -> ", p->data);  printf("NULL \n");  }  ListNode\* search\_list(ListNode\* head, element x) {  ListNode\* p = head;  while (p != NULL) {  if (p->data == x) return p;  p = p->link;  }  return NULL;  }  void search(ListNode\* l) {  if (search\_list(l, 30) != NULL)  printf("리스트에서 30을 찾았습니다. \n");  else  printf("리스트에서 30을 찾지 못했습니다. \n");  }  int main(void) {  ListNode\* head = NULL;  head = insert\_first(head, 10);  print\_list(head);  head = insert\_first(head, 20);  print\_list(head);  search(head);  head = insert\_first(head, 30);  print\_list(head);  search(head);  return 0;  }  실행결과  ]  -> list를 모두 탐색하며 찾는 값이 없을 경우 NULL을 return, 찾는 값이 존재할 경우 return 리스트  return 된 값이 null이 아니면 값을 찾은 것이고 null이면 못찾음.  리스트에 20->10 이 존재할 때와 30->20->10이 존재할 때 탐색을 비교하기 위해  void seach(list) ADT 구현 |
| 3. Lab 두 개의 리스트를 하나로 합치는 함수 작성 (204쪽)  #include <stdio.h>  #include <stdlib.h>  typedef int element;  typedef struct {  element data;  struct ListNode\* link;  } ListNode;  ListNode\* insert\_first(ListNode\* head, element value) {  ListNode\* p = (ListNode\*)malloc(sizeof(ListNode));  p->data = value;  p->link = head;  head = p;  return head;  }  void print\_list(ListNode\* head) {  for (ListNode\* p = head;p != NULL; p = p->link)  printf("%d -> ", p->data);  printf("NULL \n");  }  ListNode\* concat\_list(ListNode\* head1, ListNode\* head2) {  if (head1 == NULL) return head2;  if (head2 == NULL) return head1;  ListNode\* p;  p = head1;  while (p->link != NULL)  p = p->link;  p->link = head2;  return head1;  }  int main(void) {  ListNode\* head1 = NULL;  ListNode\* head2 = NULL;  head1 = insert\_first(head1, 10);  head1 = insert\_first(head1, 20);  head1 = insert\_first(head1, 30);  printf("head1\n"); print\_list(head1);  head2 = insert\_first(head2, 40);  head2 = insert\_first(head2, 50);  head2 = insert\_first(head2, 60);  printf("head1\n"); print\_list(head2);  head1 = concat\_list(head1, head2);  printf("concat\n"); print\_list(head1);  return 0;  }  실행결과    ->  head1과 head2 둘다 NULL일 경우 NULL 리턴 됨.  head1만 NULL일 경우 head2 리턴 됨  head2만 NULL일 경우 head1 리턴 됨  head1과 head2과 모두 존재할 경우 head1의 마지막 노드를 head2의 head와 연결 |
| 4. 연습문제 15  단순 연결 리스트가 정렬되지 않은 정수들의 리스트를 저장.  이 리스트에서 최대값과 최소값을 찾는 프로그램 작성.  #include <stdio.h>  #include <stdlib.h>  #include <limits.h>  #define max(a,b) (((a) > (b)) ? (a) : (b))  #define min(a,b) (((a) < (b)) ? (a) : (b))  typedef int element;  typedef struct {  element data;  struct ListNode\* link;  }ListNode;  ListNode\* insert\_first(ListNode\* head, element value) {  ListNode\* p = (ListNode\*)malloc(sizeof(ListNode));  p->data = value;  p->link = head;  head = p;  return head;  }  void print\_list(ListNode\* head) {  for (ListNode\* p = head;p != NULL; p = p->link)  printf("%d -> ", p->data);  printf("NULL \n");  }  element Find\_MAX(ListNode\* head) {  element maxInList = INT\_MIN;  for (ListNode\* p = head;p != NULL; p = p->link)  maxInList = max(maxInList, p->data);  return maxInList;  }  element Find\_MIN(ListNode\* head) {  element minInList = INT\_MAX;  for (ListNode\* p = head;p != NULL; p = p->link)  minInList = min(minInList, p->data);  return minInList;  }  int main() {  ListNode\* head = NULL;  srand(time(NULL));  for (int i = 0; i < 10; i++)  head = insert\_first(head, (rand()%100)+1);  print\_list(head);  printf("List 최대 값 == %d\n", Find\_MAX(head));  printf("List 최소 값 == %d\n", Find\_MIN(head));  return 0;  }  실행결과    ->  1. 매크로를 통해 최댓값, 최솟값을 찾는 함수 작성  2. list 구현  3. FIND\_MAX 구현 -- list에 존재하는 모든 값을 탐색 후 매크로를 통해 최댓값을 찾은 후 리턴  4. FIND\_MIN 구현 -- 3번과 동일  리스트에 랜덤 값 10개를 삽입한 후 최대값과 최소값 출력 완료 |