Day11 (20 01 31) \_ 자료구조

내가 구현한 search

|  |
| --- |
| //search - 현재 리스트에 내가 찾고자하는 함수가 있는지 . 있으면 o 없으면 x  void Search(Linklist\*\* head) {  int num;  int count = 1;  printf("찾고자 하는 수를 입력하세요 \n");  scanf("%d", &num);  if (num == (\*head)->data) {  printf("%d 가 존재합니다\n", (\*head)->data);  }  else if (num != (\*head)->data) {  printf("%d를 찾는 중입니다 \n", num);  Search(&(\*head)->link);  }else {  printf("%d는 존재하지 않습니다\n", num);  }    } |

선생님이 구현한 search

|  |
| --- |
| void Search(Linklist\*\* head, int data) {    if ((\*head)->data == data) {  printf("%d SEARCH! \n", data);  }  else {  if ((\*head)->link == NULL) {  printf("%d Not exit! \n", data);  return;  }  Search(&(\*head)->link, data);  }  } |

내가 구현한 add

|  |
| --- |
| void Add(Linklist \*\*head) {  int inum, wnum;  Linklist\* tmp ;  int true = 1;  printf("넣고자 하는 수를 입력하세요 ");  scanf("%d", &wnum);  printf("어떤 수의 뒤에 넣을까요? ");  scanf("%d", &inum);  if (inum == (\*head)->data) {  tmp = (\*head);  printf("%d 뒤에 %d를 넣겠습니다\n", (\*head)->data,wnum);  (\*head) = GetNode();  (\*head)->link = &(\*tmp);  (\*head)->data = wnum;  }  else {  Add(&(\*head)->link);  }      } |

선생님이 구현한 add

|  |
| --- |
| //add - 중간에 삽입하는 목적, 내가 원하는 위치랑 데이터 뒤..  void Add(Linklist \*\*head,int data1, int data2) {  Linklist\* tmp = NULL;    if (!(\*head)) {  printf("없어용 \n");  } else if ((\*head)->data == data1) {  Insert(&tmp, data2);  tmp->link = (\*head)->link;  (\*head)->link = tmp;  }  else {  Add(&(\*head)->link, data1, data2);  }      } |

선생님이 구현한 delete

|  |
| --- |
| //delete - 현재 리스트에 있는 것을 삭제 , 내가 원하는 값  void Delete(Linklist\*\*head, int data) {  Linklist\* tmp;  if (!(\*head)) {  printf("없음용 \n");  }  else if ((\*head)->data == data) {  tmp = \*head;  \*head = (\*head)->link;  free(tmp);  }  else {  Delete(&(\*head)->link, data);  }  } |

Double link list

-linked list의 탐색 기능을 개선한 자료구조로 양방향으로 탐색이 가능하다

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 10 |  | 서로 가르킴 |  | 20 |  | 서로가르킴 |  | 30 |  |

Head-> mid <-tail

{ Head(ptr) tail(ptr) 만들고,

총 node의 개수의 /2 = mid(ptr) 로 지정

해서 어디서부터 (head/tail) 찾아나갈지 결정 } \* 반복

* 그러면 계속 중간, 중간 , 중간해서 데이터를 찾아나간다
* 기존의 linked list보다 효율이 좋다