
CSE2010 자료구조론

# Week 3: ArrayList

ICT융합학부 한진영

# 리스트(List)의 정의

- 리스트(list), 선형 리스트(linear list): 순서를 가진 항목들의 모임
  - Cf. 집합, 항목간의 순서의 개념이 없음
- 리스트의 예
  - 요일: (일요일, 월요일, ..., 토요일)
  - 한글 자음의 모임: (ㄱ,ㄴ,...,ㅎ)
  - 카드: (Ace, 2,3,...,King)
  - 핸드폰의 문자 메시지 리스트

$$L = (item_0, item_1, ..., item_{n-1})$$

# 리스트(List) 연산 나열

- 새로운 항목을 리스트의 끝, 처음, 중간에 추가
- 기존의 항목을 리스트의 임의의 위치에서 삭제
- 모든 항목을 삭제
- 기존의 항목을 대치
- 리스트가 특정한 항목을 가지고 있는지를 살핌
- 리스트의 특정 위치 항목을 반환
- 리스트 안의 항목 개수를 셈
- 리스트가 비었는지, 꽉 찼는지를 체크
- 리스트 안의 모든 항목을 표시

# 리스트 ADT

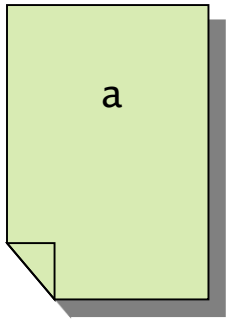
·객체:

n개의 element형으로 구성된 순서있는 모임

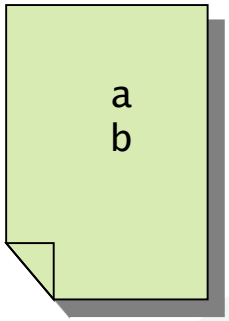
·연산:

- add_last(list, item) ::= 맨끝에 요소를 추가한다.
- add_first(list, item) ::= 맨끝에 요소를 추가한다.
- add(list, pos, item) ::= pos 위치에 요소를 추가한다.
- delete(list, pos) ::= pos 위치의 요소를 제거한다.
- clear(list) ::= 리스트의 모든 요소를 제거한다.
- replace(list, pos, item) ::= pos 위치의 요소를 item로 바꾼다.
- is_in_list(list, item) ::= item이 리스트안에 있는지를 검사한다.
- get_entry(list, pos) ::= pos 위치의 요소를 반환한다.
- get_length(list) ::= 리스트의 길이를 구한다.
- is_empty(list) ::= 리스트가 비었는지를 검사한다.
- is_full(list) ::= 리스트가 꽉찼는지를 검사한다.
- display(list) ::= 리스트의 모든 요소를 표시한다.

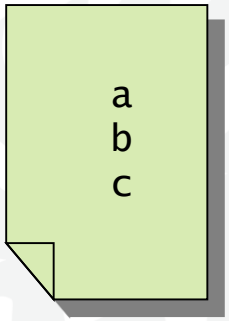
# 리스트 ADT 사용 예 #1



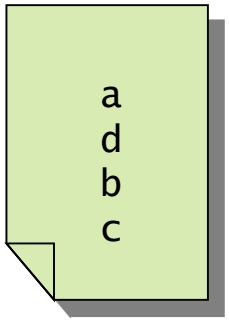

addLast(list1,a)



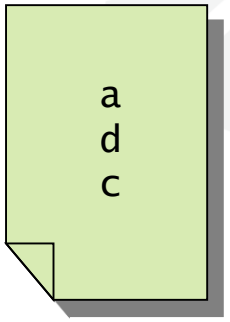

addLast(list1,b)



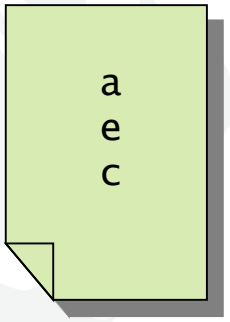

addLast(list1,c)




add(list1,1,d)




delete(list1,2)




replace(list1,1,e)

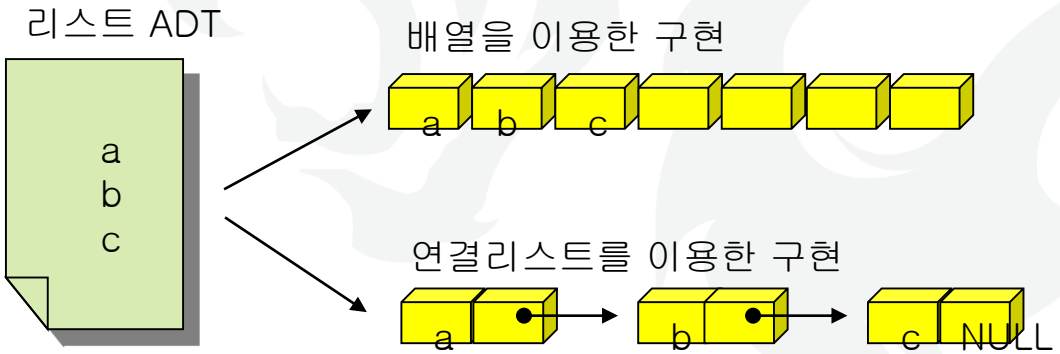
# 리스트 ADT 사용 예 #2

```
main()
{
        int i, n;

        // list2를 생성한다: 구현방법에 따라 약간씩 다름
        ListType list2;
        add_last (&list2,"마요네즈”);  // 리스트의 포인트를 전달
        add_last(&list2,"빵”);
        add_last(&list2,"치즈”);
        add_last(&list2,"우유”);
        n = get_length(&list2);
        printf("쇼핑해야할 항목수는 %d입니다.\n", n);
        for(i=0;i<n;i++)
          printf("%d항목은 %s입니다.i°, i, get_entry(&list2,i));
}
```

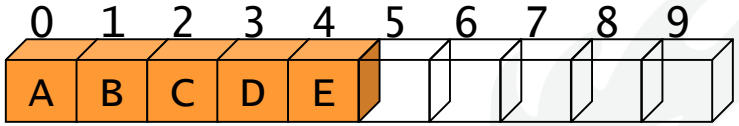
# 리스트 구현 방법

- 배열을 이용하는 방법
  - 구현 간단
  - 삽입, 삭제 시 오버헤드
  - 항목의 개수 제한
- 연결리스트를 이용하는 방법
  - 구현 복잡
  - 삽입, 삭제가 효율적
  - 크기가 제한되지 않음

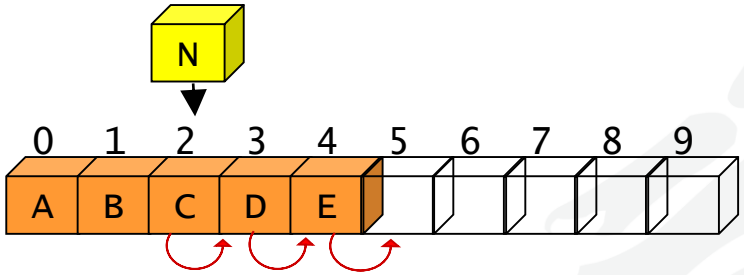


# 배열로 구현된 리스트

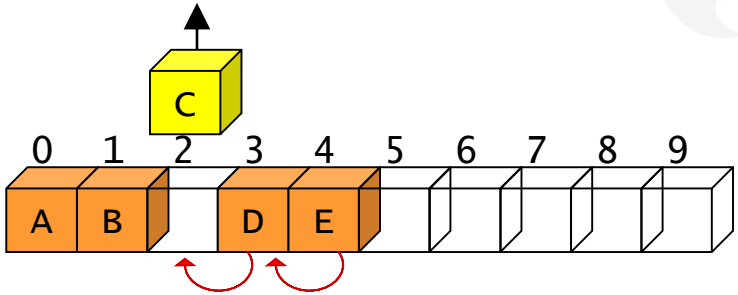
- 1차원 배열에 항목들을 순서대로 저장
  - L=(A, B, C, D, E)




- 삽입연산: 삽입위치 다음의 항목들을 이동해야 함




- 삭제연산: 삭제위치 다음의 항목들을 이동해야 함



# ArrayListType 정의

- 항목들의 타입은 element로 정의
- list라는 1차원 배열에 항목들을 차례대로 저장
- length에 항목의 개수 저장

```
typedef int element;
typedef struct {
        int list[MAX_LIST_SIZE];        // 배열 정의
        int length;             // 현재 배열에 저장된 항목들의 개수
} ArrayListType;
```

```
// 리스트 초기화
void init(ArrayListType *L)
{
        L->length = 0;
}
```

# ArrayListType 구현: is_empty, is_full

- is_empty와 is_full 함수의 구현

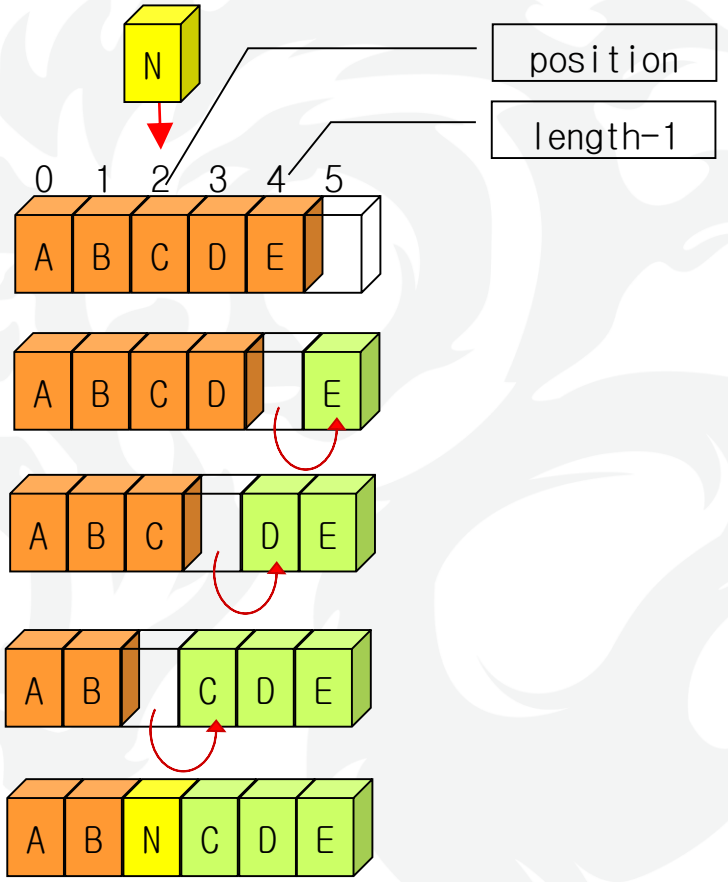
```
// 리스트가 비어 있으면(즉 length가 0이면) 1을 반환
// 그렇지 않으면 0을 반환
int is_empty(ArrayListType *L)
{
        return L->length == 0;
}
// 리스트가 가득 차 있으면 1을 반환
// 그렇지 않으면 0을 반환
int is_full(ArrayListType *L)
{
        return L->length == MAX_LIST_SIZE;
}
```

(참고) == 연산자의 결과 : 거짓(0), 참(1)

# ArrayListType 구현: add

- add 함수 구현
  - 먼저 배열이 포화상태인지를 검사하고 삽입 위치가 적합한 범위에 있는지를 검사
  - 삽입 위치 다음에 있는 자료들을 한칸씩 뒤로 이동

```
// position: 삽입하고자 하는 위치
// item: 삽입하고자 하는 자료
void add(ArrayListType *L, int position, element item)
{
  if( !is_full(L) && (position >= 0) && (position <= L->length) )
  {
     int i;
     for(i=(L->length-1); i>=position;i--)
         L->list[i+1] = L->list[i];
     L->list[position] = item;
     L->length++;
  }
}
```

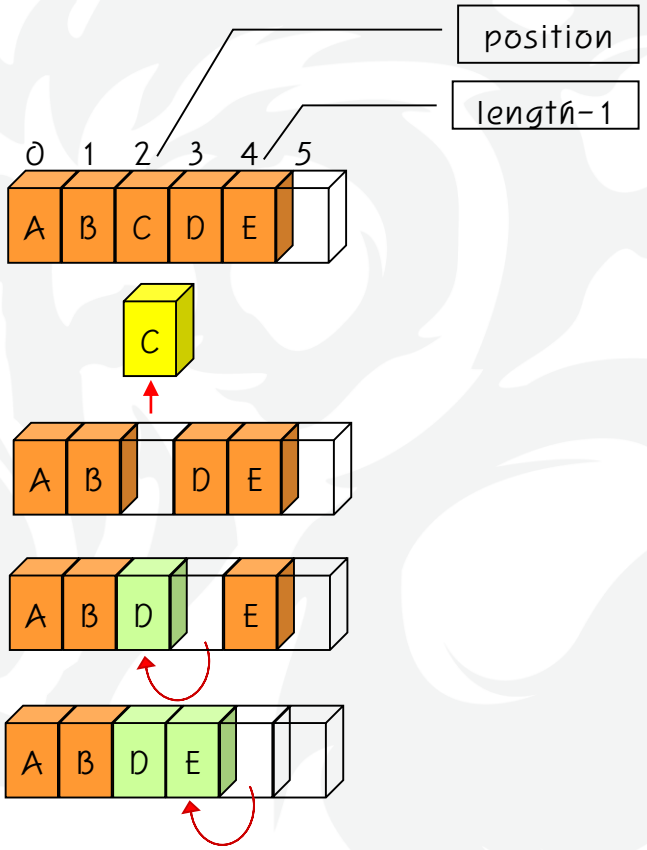


# ArrayListType 구현: delete

- delete 함수 구현
  - 삭제 위치를 검사
  - 삭제위치부터 맨끝까지의 자료를 한칸씩 앞으로 옮김

```
// position: 삭제하고자 하는 위치
// 반환값: 삭제되는 자료
element delete(ArrayListType *L, int position)
{
        int i;
        element item;

        if( position < 0 || position >= L->length )
                error("위치 오류");
        item = L->list[position];
        for(i=position; i<(L->length-1);i++)
                L->list[i] = L->list[i+1];
        L->length--;
        return item;
}
```



# Week 3: ArrayList