동국대학교
dongguk university

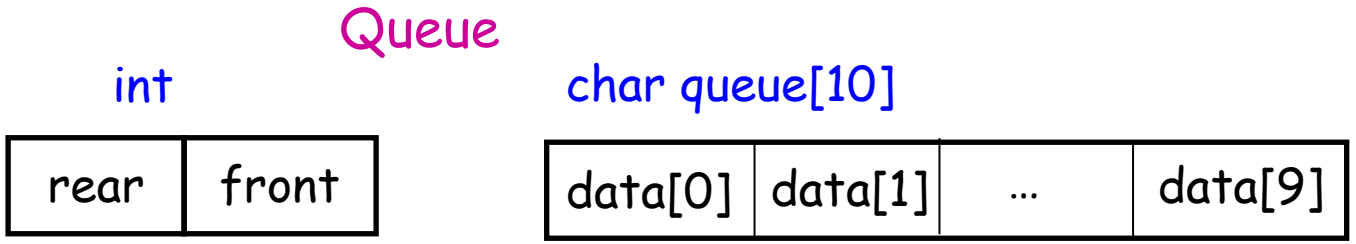
실습: Week 5

Data Structures

Contents

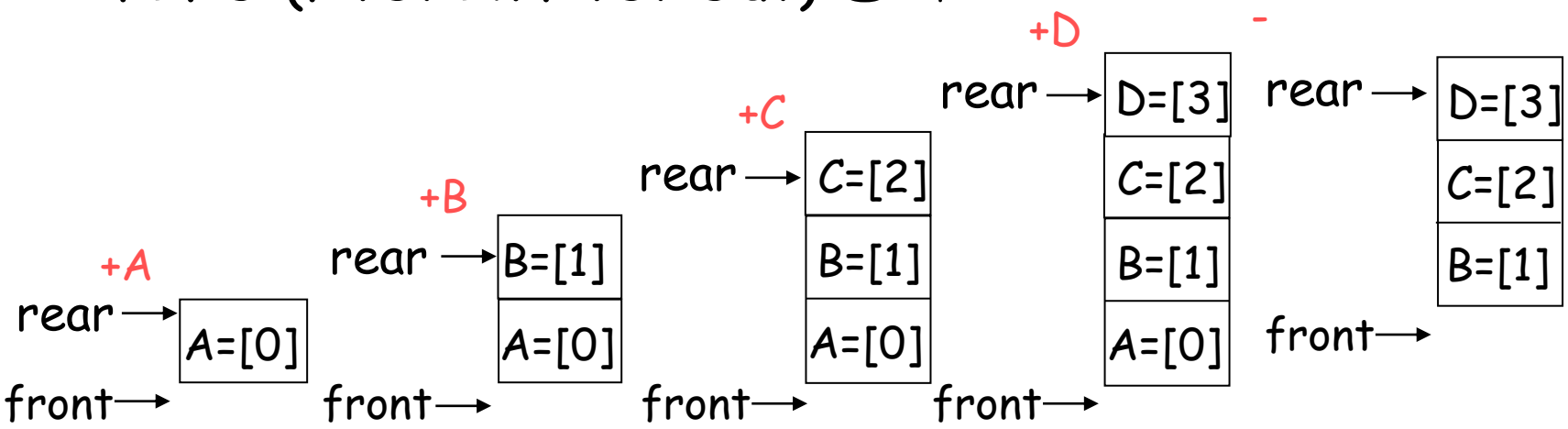
- Queue
  - Queue 구현
- List
  - Linked List 구현
- 실습
  - 실습 5-1. Queue (circular array)구현
  - 실습 5-2. Linked List (singly linked list) 구현

Queue 구조



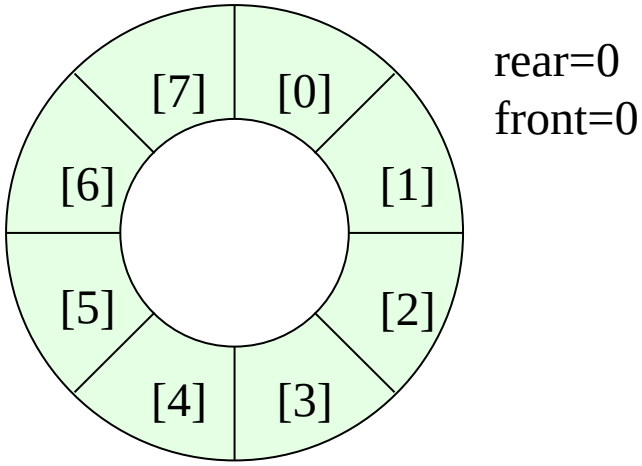
- front : 배열 Queue에서 처음 원소의 앞을 가리킴
- rear : 배열 Queue에서 마지막 원소를 가리킴

- FIFO (First In First Out) 형식



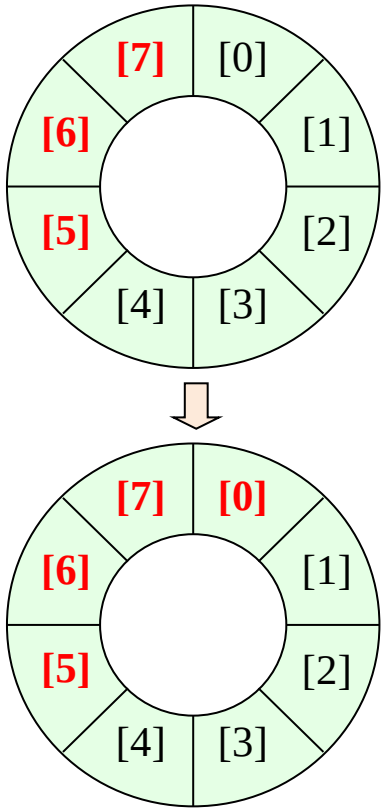
Circular Queue

- Circular array를 사용
  - front : 첫번째 원소의 하나 앞을 가리킴
  - rear : 마지막 원소를 가리킴
  - 삽입과 삭제: modular 연산자를 이용
    - rear = (rear+1) % Max_Size
    - front = (front+1) % Max_Size

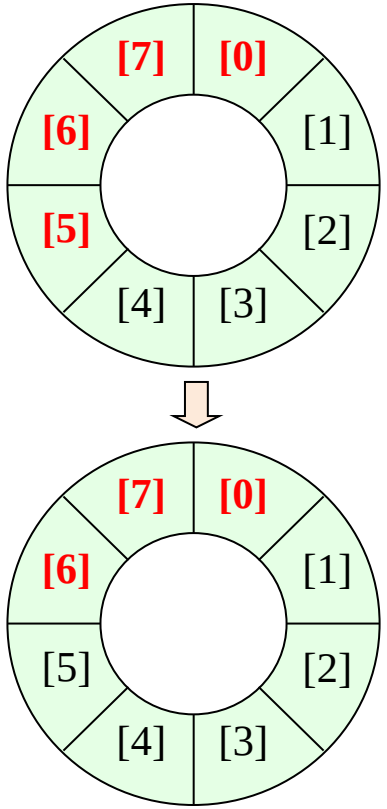


Circular Queue

- 삽입연산

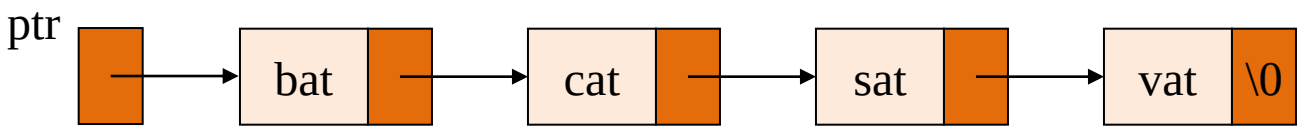


- 삭제연산



Linked Lists

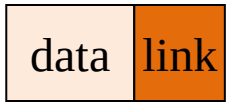
- 연결리스트
  - 화살표로 표시된 링크를 가진 노드들의 순열
  - 크기가 미리 정해지지 않음
  - 리스트의 구성 : data field & link
    - data field : 실제 자료 저장
    - link field : 다음 노드에 대한 주소 저장(pointer)



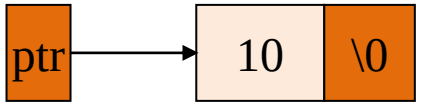
Linked Lists

- Example: 정수 연결 리스트

```
typedef struct list_node *list_pointer;
typedef struct list_node {
        int             data;
        list_pointer    link;
};
```

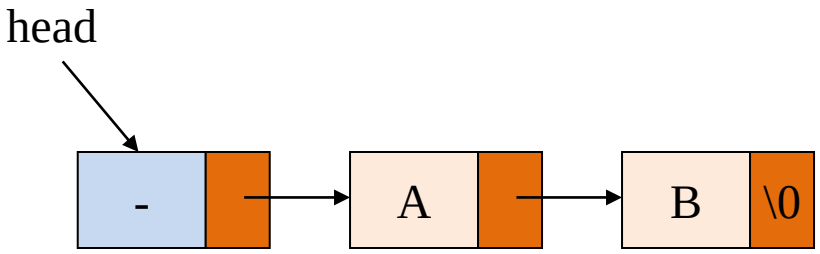



```
list_pointer ptr;
ptr=(list_pointer)malloc(sizeof(list_node));
ptr->data = 10;
ptr->link=NULL;
```



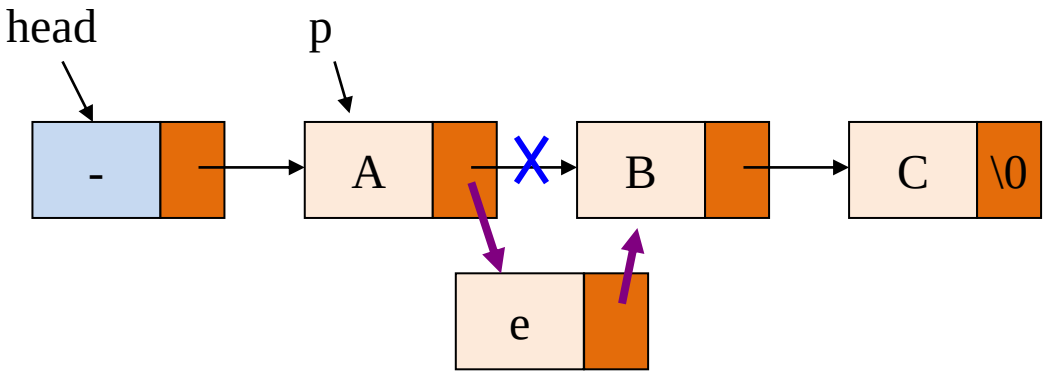
Use of Head Node

- 프로그램을 쉽게 하기 위하여 dummy head node 사용

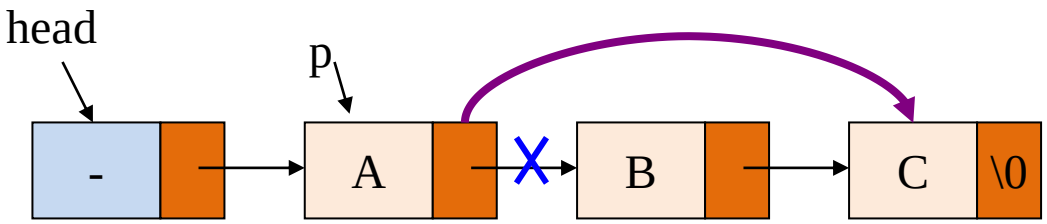


Linked List - 삽입, 삭제

- 삽입



- 삭제



실습 5-1. Array Queue 구현

- 문자들의 **Queue**를 테스트하는 프로그램 구현
- 명령어
  - +<c> : AddQ
  - - : DeleteQ
  - S : Show
  - Q : Quit

실습 5-1. 실행 예

```
******************command******************
+<c> : AddQ c , - : DeleteQ,
 S : show , Q : Quit
*******************************************

Command> +1
Command> +2
Command> +3
Command> +4
Command> +5
Command> +6
Command> +7
Command> +8
Command> +9
Command> +0
queue is full

Command> s
1 2 3 4 5 6 7 8 9
Command> -
 1
Command> -
 2
Command> -
 3
Command> -
 4
Command> -
 5
Command> -
 6
Command> -
 7
Command> -
 8
Command> -
 9
Command> -
queue is empty
```

```
Command> +2
Command> +4
Command> +6
Command> s
2 4 6
Command> -
 2
Command> -
 4
Command> -
 6
Command> -
queue is empty

Command>
```

자료구조 및 함수

- void addq(Element e)
  - Requires : Queue가 포화되지 않아야 함(full과 empty를 구분하기 위해 max-1개만 채움.)
  - Results : Queue에 e를 삽입
- Element deleteq()
  - Requires : Queue가 비어 있지 않아야 함
  - Results : Queue에서 원소를 반환
- void queue_show()
  - Results : Queue의 내용을 보여줌

arrayqueue.h

```
#include <stdio.h>
#define  MAX_SIZE 10
#define  boolean   int
#define  true      1
#define  false     0

typedef char Element;
// Global queue
Element queue[MAX_SIZE];
int front = 0;
int rear = 0;
boolean is_empty = false;

void addq(Element e);
Element deleteq();
void queue_show();
```

arrayqueue.c - main()함수

```
#include "arrayqueue.h"

void main() {

        char    c, e;

        printf("****************command*******************\n");
        printf("+<c> : AddQ c , - : DeleteQ, \n S : show , Q : Quit\n");
        printf("*****************************************\n");

        while (1) {
                printf("\nCommand> ");
                c = _getche();
                c = toupper(c);
```

arrayqueue.c - main()함수

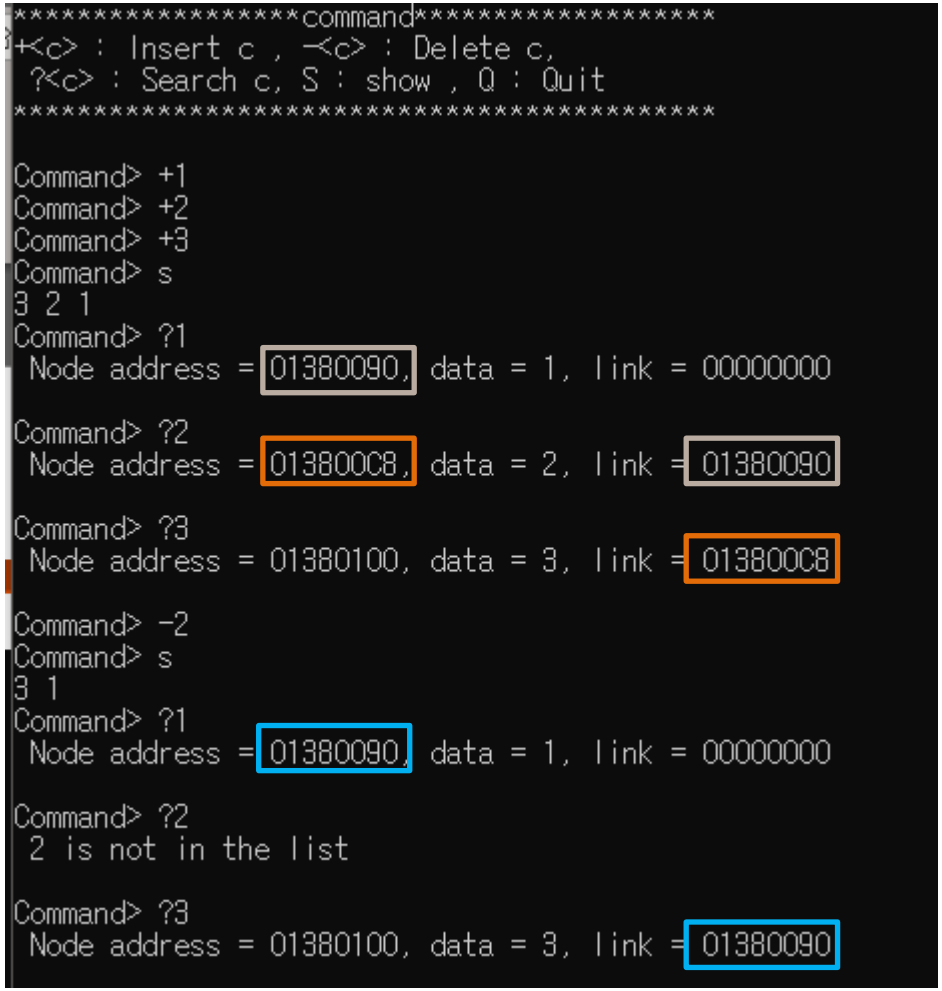
```
                switch (c) {
                        case '+':
                                e = _getche();
                                addq(e);
                                break;
                        case '-':
                                e = deleteq();
                                if (is_empty != true)
                                        printf("\n %c ", e);
                                is_empty = false;
                                break;
                        case 'S':
                                queue_show();
                                break;
                        case 'Q': printf("\n");
                                exit(1);
                        default: break;
                }
        }

}
```

실습 5-2. linked_list

- 문자들의 리스트를 linked_list로 구현.
- 명령어
  - +<c> : Insert c
  - -<c> : Delete c
  - ?<c> : Search c
  - S : Show
  - Q : Quit

실습 5-2. 실행 예


**********command**********
+<c> : Insert c , -<c> : Delete c,
?<c> : Search c, S : show , Q : Quit
****
Command> +1
Command> +2
Command> +3
Command> s
3 2 1
Command> ?1
Node address = 01380090, data = 1, link = 00000000
Command> ?2
Node address = 013800C8, data = 2, link = 01380090
Command> ?3
Node address = 01380100, data = 3, link = 013800C8
Command> -2
Command> s
3 1
Command> ?1
Node address = 01380090, data = 1, link = 00000000
Command> ?2
2 is not in the list
Command> ?3
Node address = 01380100, data = 3, link = 01380090

자료구조 및 함수

- void list_insert (list_pointer head, Element e)
  - Results : 문자 **e**를 담은 노드를 만들어 리스트의 앞에 삽입
- list_pointer list_search(list_pointer head, Element e)
  - Results : 문자 e를 head 리스트에서 찾아서 노드의 **주소 반환**
- void list_delete(list_pointer head, Element e)
  - Results : 문자 e를 head 리스트에서 찾아서 노드를 **삭제**
- boolean list_empty(list_pointer head)
  - Results : head 리스트가 비어있으면 true
- void list_show(list_pointer head)
  - Results : head 리스트의 내용을 앞에서 부터 출력

linked_list.h

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

#define  Element  char
#define  bool     int
#define  true     1
#define  false    0

typedef  struct list_node* list_pointer;
typedef  struct list_node{
         Element          data;
         list_pointer     link;
} list_node;

void list_insert(list_pointer head, Element e);
void list_delete(list_pointer head, Element e);
list_pointer list_search(list_pointer head, Element e);
bool list_empty(list_pointer head);
void list_show(list_pointer head);
```

linked_list.c - main()함수

```
#include "linked_list.h"

void main()
{
        char            c, e;
        list_pointer  head, p;

        // dummy head 노드
        head = (list_pointer)malloc(sizeof(list_node));
        head->data = NULL;
        head->link = NULL;
        printf("******************command******************\n");
        printf("+<c> : Insert c , -<c> : Delete c, \n ?<c> : Search c, S : show ,
Q : Quit\n");
        printf("*****************************************\n");

        while (1) {
                printf("\nCommand> ");

                c = _getche();
                c = toupper(c);
```

linked_list.c - main()함수

```
switch (c) {
        case '+':
            e = _getche();
            list_insert(head, e);
            break;
        case '-':
            e = _getche();
            list_delete(head, e);
            break;
        case '?':
            e = _getche();
            p = list_search(head, e);
            if (p) {
                printf(" Node address = %p, data = %c, link = %p \n",
                    p, p->data, p->link);
            }
            else printf("\n %c is not in the list \n", e);
            break;
```

linked_list.c - main()함수

```
        case 'S':
            list_show(head); break;
        case 'Q':
            printf("\n");
            exit(1);
        default:  break;
        }
    }
}
```

주의 사항

- Queue
  - circular queue여야 함
- List
  - dummy head 노드가 있음
  - insert는 새로운 노드를 만들어 삽입해야 함
  - insert → show → search → delete 순서로 구현할 것