동국대학교
dongguk university

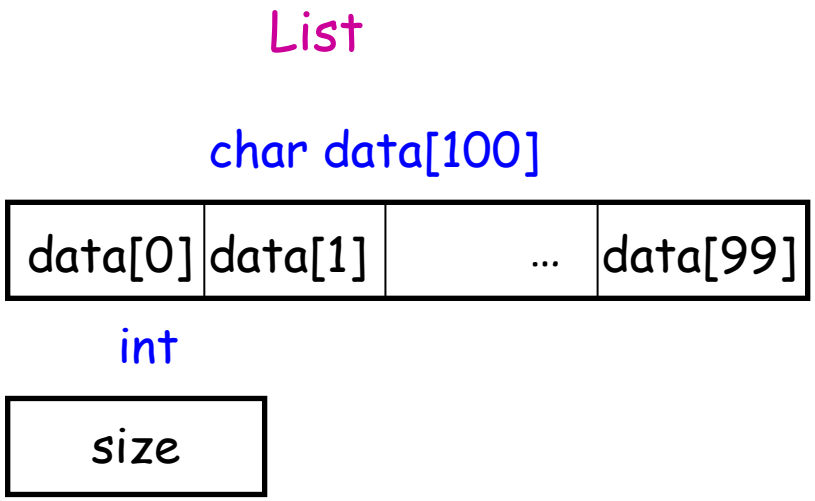
실습: Week 3

Data Structures

Contents

- Arrays
  - List (array)
  - Matrix (multidimensional array)
- 실습
  - 실습 3-1. List 구현
  - 실습 3-2. Matrix(2D array) add, mult, trans 구현

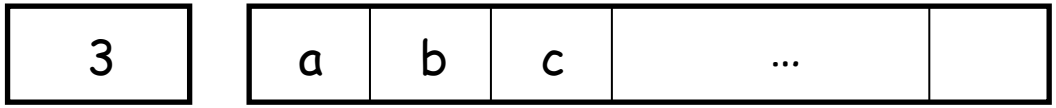
List 자료 구조



size : 배열 크기 (초기값: 0)

List 연산

- 삽입(ListInsert)
  - +a, +b, +c



- 삭제(ListDelete)
  - -b

Multidimensional Arrays

- 2차원 배열 → 배열의 배열

배열 선언의 예	설명
int a[100];	1차원 배열 (100 개)
int b[2][10];	2차원 배열 (2*10 = 20 개)
int c[5][3][2];	3차원 배열 (5*3*2 = 30 개)

```
int a[3][5];  // a[5]가 3개
```

	1 열	2 열	3 열	4 열	5 열
1 행	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
2 행	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
3 행	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

Matrix

- Matrix(행렬) – 2차원 배열로 표현
- 예: 행렬의 모든 원소의 합

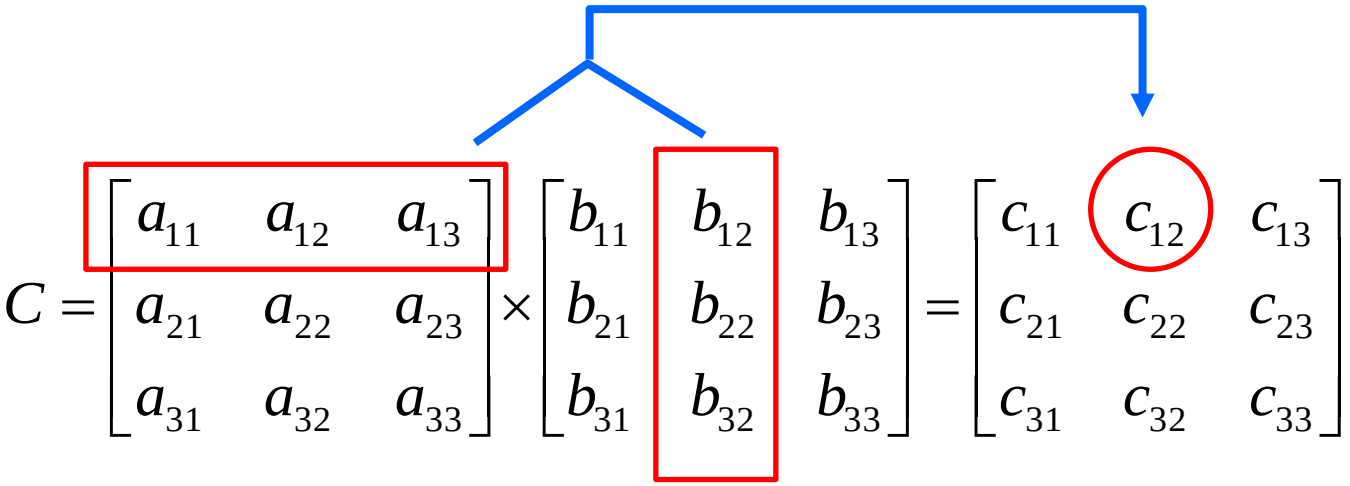
```
int m[4][3]={ {1,5,-3}, {9,0,0}, {2,3,1}, {0,3,8} };
int i, j, sum;
sum = 0;

for( i=0; i<4; i++ )
        for( j=0; j<3; j++ )
                sum = sum + m[i][j];

printf("The sum of all elements is %d\n", sum );
```

Matrix Mult

- A, B, C 가 3 x 3 행렬일 때
  - C = A x B



$$C = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

$$c_{12} = a_{11} \times b_{12} + a_{12} \times b_{22} + a_{13} \times b_{32}$$

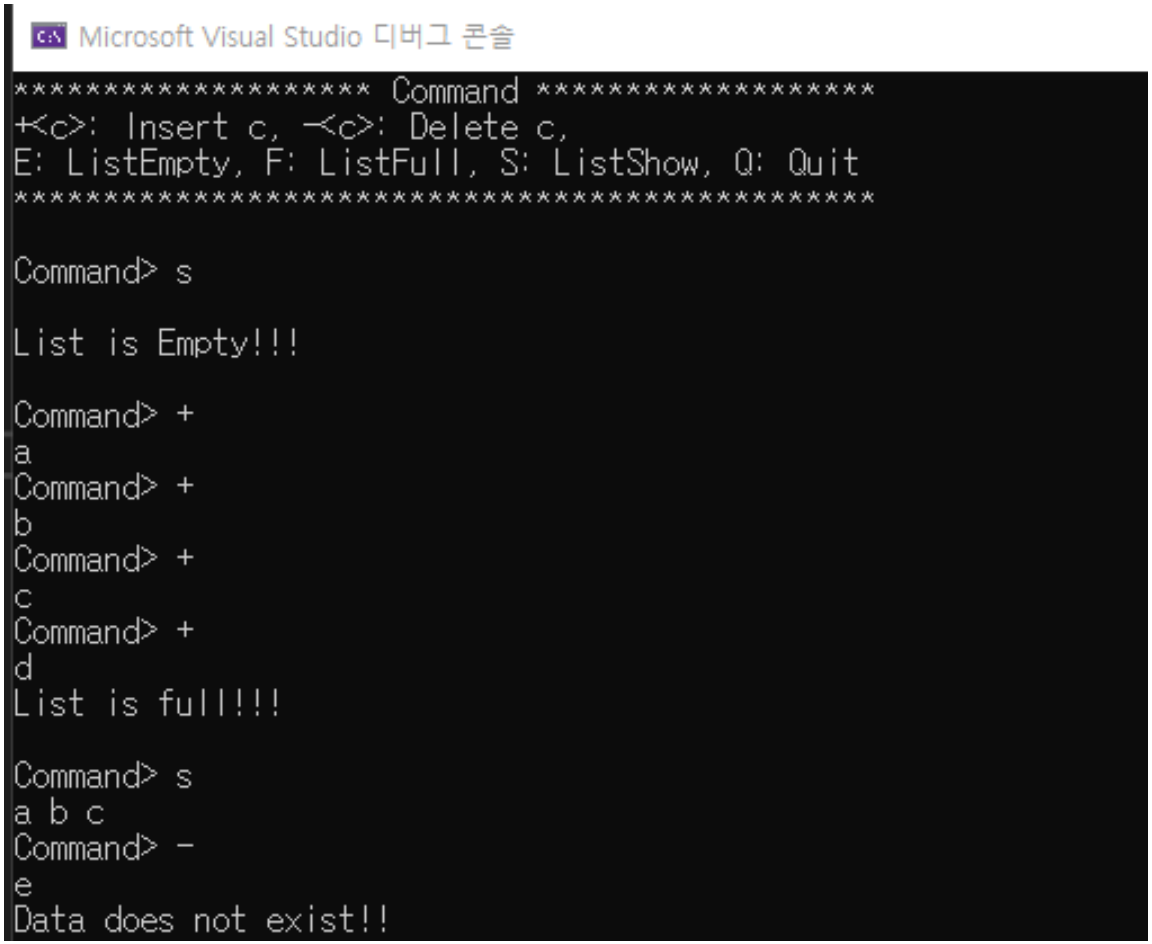
3-1. List 구현

- 문자들을 원소로 하는 리스트 프로그램 구현
- 명령어
  - 원소 삽입(+문자), 원소 삭제(-문자)
  - 리스트가 비어있는지 확인(E)
  - 리스트가 꽉 차있는지 확인(F)
  - 리스트 내용 출력(S)

List

- Objects
  - List
  - 문자들의 리스트. 사이즈 = MaxSize
- Functions
  - list_insert(c): 리스트에 문자 c 삽입
  - list_delete(c): 리스트에서 문자 c 삭제
  - list_full(): 리스트가 full이면 true
  - list_empty(): 리스트가 empty이면 true
  - list_show(): 리스트의 내용을 출력

실습 3-1. 실행 예(1)


Microsoft Visual Studio 디버그 콘솔
********** Command **********
+<c>: Insert c, -<c>: Delete c,
E: ListEmpty, F: ListFull, S: ListShow, Q: Quit
****
Command> s
List is Empty!!!
Command> +
a
Command> +
b
Command> +
c
Command> +
d
List is full!!!
Command> s
a b c
Command> -
e
Data does not exist!!

실습 3-1. 실행 예(2)

```
Command> s
a b c
Command> -
e
Data does not exist!!
Command> s
a b c
Command> -
b
Command> s
a c
Command> -
c
Command> s
a
Command> -
a
Command> s

List is Empty!!!

Command> e

TRUE

Command> q
```

자료구조 및 함수

- // Global로 선언 (main() 밖에 선언)
  Element List[MaxSize];
  int size = 0;

- void list_insert(Element e)
  - Requires : 리스트가 포화(Full)되지 않아야 함
  - Results : e 를 리스트의 마지막에 삽입

- void list_delete(Element e)
  - Requires : 리스트가 비어있지 않아야 함
  - Results : 리스트에서 e를 찾아 삭제 (삭제된 자리를 메꾸어야 함)

- boolean list_empty()
  - Results : 리스트가 비어 있다면 true, 그렇지 않으면 false를 리턴

- boolean list_full()
  - Results : 리스트가 포화상태라면 true, 그렇지 않으면 false를 리턴

- void list_show()
  - Results : 리스트에 있는 모든 원소를 출력

arraylist.h

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>

#define    MaxSize 3
#define    boolean     unsigned char
#define    true        1
#define    false       0

typedef char Element;

// Global로 선언한 List 자료구조
Element    List[MaxSize];
int        size = 0;

void list_insert(Element e);
void list_delete(Element e);
boolean list_empty();
boolean list_full();
void list_show();
```

arraylist.c - main() 함수

```
#include "arraylist.h"

int main(void)
{
    char  c;

    printf("********** Command **********\n");
    printf("+<c>: Insert c, -<c>: Delete c, \n");
    printf("E: ListEmpty, F: ListFull, S: ListShow, Q: Quit \n");
    printf("*****************************************\n");
    while (1) {
        printf("\nCommand> ");
        c = _getche();
        c = toupper(c);
        printf("\n");
```

arraylist.c - main() 함수

```
switch (c) {
        case '+':
                c = _getche();
                list_insert(c);
                break;
        case '-':
                c = _getche();
                list_delete(c);
                break;
        case 'E':
                if (list_empty()) printf("\nTRUE \n");
                else printf("\nFALSE \n");
                break;
        case 'F':
                if (list_full()) printf("\nTRUE \n");
                else printf("\nFALSE \n");
                break;
```

arraylist.c - main() 함수

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

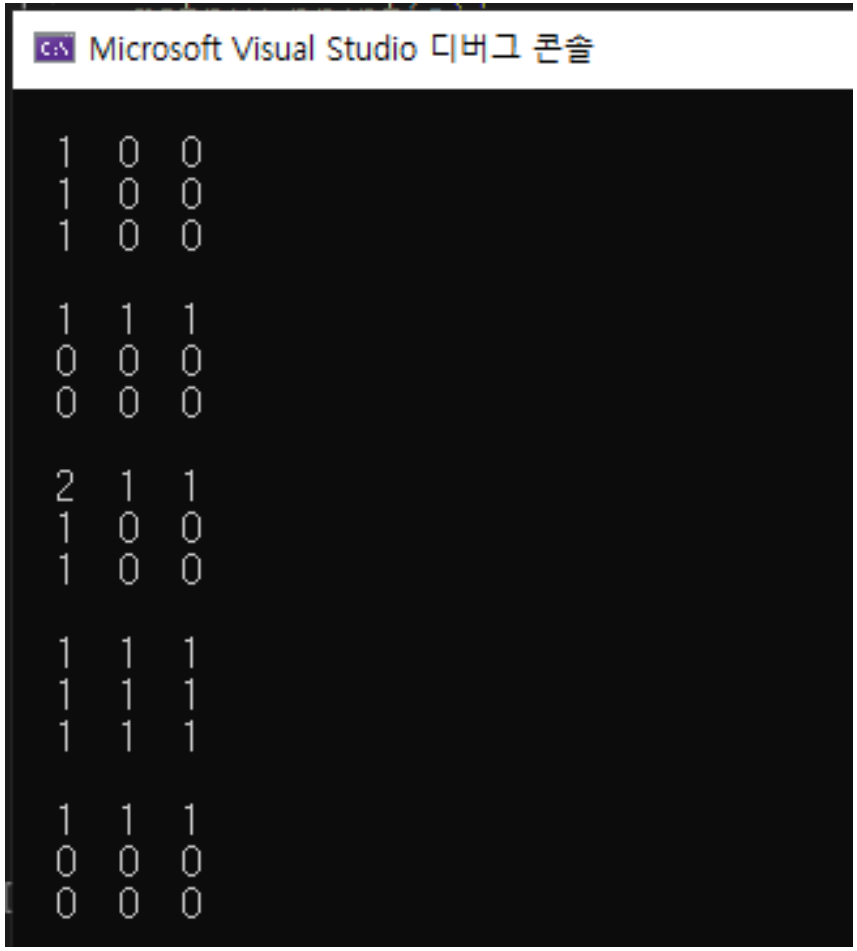
arraylist.c - list_full() 함수

```
boolean list_full()
{
    if(size == MaxSize)
        return true;     // 리스트가 가득 차 있으면 true
    else
        return false;
}
```

실습 3-2. Matrix

- 행렬 연산 함수 구현
- A, B, C 가 3 x 3 행렬일 때
  - C = A + B
  - C = A x B
  - C = A^T
  - C 출력

실습 3-2. 실행 예


Microsoft Visual Studio 디버그 콘솔
1 0 0
1 0 0
1 0 0
1 1 1
0 0 0
0 0 0
2 1 1
1 0 0
1 0 0
1 1 1
1 1 1
1 1 1
1 1 1
0 0 0
0 0 0

자료구조 및 함수

- int a[ROW][COL]={ {1,0,0},{1,0,0},{1,0,0} };
  int b[ROW][COL]={ {1,1,1},{0,0,0},{0,0,0} };

- void matrix_init(a)
  - matrix a의 원소를 모두 0으로
- void matrix_add(a, b, c)
  - a + b 를 수행하여 c 에 저장
- void matrix_mult(a, b, c);
  - a x b 를 수행하여 c 에 저장
- void matrix_trans(a, c);
  - a transpose 를 수행하여 c 에 저장
- void matrix_print(a);
  - 행렬 a 출력

matrix.h

```
#include<stdio.h>

#define ROW 3
#define COL 3

void matrix_init(int a[ROW][COL]);
void matrix_add(int a[ROW][COL], int b[ROW][COL], int c[ROW][COL]);
void matrix_mult(int a[ROW][COL], int b[ROW][COL], int c[ROW][COL]);
void matrix_trans(int a[ROW][COL], int c[ROW][COL]);
void matrix_print(int a[ROW][COL]);
```

matrix.c - main() 함수

```
#include "matrix.h"

void main()
{
	int a[ROW][COL] = { {1,0,0},{1,0,0},{1,0,0} };
	int b[ROW][COL] = { {1,1,1},{0,0,0},{0,0,0} };
	int c[ROW][COL] = { {0,0,0},{0,0,0},{0,0,0} };

	matrix_print(a);
	matrix_print(b);

	matrix_add(a, b, c);
	matrix_print(c);

	matrix_init(c);
	matrix_mult(a, b, c);
	matrix_print(c);

	matrix_init(c);
	matrix_trans(a, c);
	matrix_print(c);
}
```

matrix.c - matrix_init() 함수

```
void matrix_init(int a[ROW][COL])
{
    int i, j;

    for(i=0; i<ROW; i++) {
        for(j=0; j<COL; j++) {
            a[i][j] = 0;
        }
    }
}
```