

프로그래밍언어 (실습)

**실습 3 - (보충설명) 누적 날짜수 및 요일 확인,
2차원 배열을 사용한 행렬의 덧셈, 뺄셈, 곱셈 연산**



교수 김 영 탁

영남대학교 정보통신공학과

(Tel : +82-53-810-2497; Fax : +82-53-810-4742

<http://antl.yu.ac.kr/>; E-mail : ytkim@yu.ac.kr)

Outline

◆ 지정된 연월일이 서기 1년 1월 1일로부터 몇번째 날인지, 무슨 요일인지를 계산하여 출력하는 프로그램 작성

- `bool isLeapYear(int year);`
- `int getDaysFromJan01AD01(int year, int month, int day);`

◆ 배열 (Array)

- 배열 선언과 초기화
- 배열을 함수의 인수 (argument)로 사용

◆ 다차원 배열 (Multi-dimensional Array)

- 2차원 배열로 행렬 (matrix) 구현
- 행렬의 덧셈, 뺄셈, 곱셈 연산



```
getDaysFromJan01AD01()  
printCalendar()
```

지정된 연월일이 서기 (AD) 1년 1월 1일로 부터 몇 번째 날인지, 무슨 요일인지 출력

◆ 예)

- 서기 (AD) 1년 1월 1일은 1번째 날, 월요일
- 서기 1년 12월 31일은 365번째 날, 요일 ?
- 서기 2년 1월 1일은 ? 번째 날, ? 요일
- 2021년 1월 1일은 ? 번째 날, ? 요일
- 2021년 3월 19일은 ? 번째 날, ? 요일



지정된 연월일이 서기 (AD) 1년 1월 1일로 부터 몇 번째 날인지, 무슨 요일인지 출력

◆ **bool isLeapYear(int y);**

- 윤년이 되는 조건?
- 교재 그림 3.24

◆ **int getDaysFromJan01AD01(int year, int month, int day);**

- 서기 1년으로 부터 지정된 연 (year) 직전 연도까지의 날짜 수를 합산
- 지정된 월 (month) 직전까지의 월 별 날짜수를 합산
 - 윤년인 경우, 2월이 29일까지이며, 평년인 경우 28일까지 있음
 - 윤년인가를 확인하는 방법 => 교재 그림 3.24
 - 지정된 일자를 합산
- 교재 그림 3.25



getDaysFromJan01AD01()

```
int getDaysFromJan01AD01(int year, int month, int d)
{
    int daysFromAD01Jan01 = 0;
    int daysInYear;
    int daysInMonth[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int days = 0;
    for (int y = 1; y < year; y++)
    {
        daysInYear = isLeapYear(y) ? 366 : 365;
        // 이 부분의 코드는 직접 작성할 것
    }
    if (isLeapYear(year))
        daysInMonth[2] = 29;
    for (int m = 1; m < month; m++)
    {
        // 이 부분의 코드는 직접 작성할 것
    }
    daysFromAD01Jan01 += d;
    return daysFromAD01Jan01;
}
```



printCalendar()

```
#define WEEKDAY_AD01Jan01 MON // the weekday of AD Jan 1.
#define DAYS_PER_WEEK 7
enum WEEKDAY {SUN, MON, TUE, WED, THR, FRI, SAT };
enum MONTH { JAN = 1, FED, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT,
             NOV, DEC, NUM_MONTHS };
const char* weekDayName[DAYS_PER_WEEK] = { "SUN", "MON",
      "TUE", "WED", "THR", "FRI", "SAT" };
const char* monthName[NUM_MONTHS] = { "", "January", "February",
      "March", "April", "May", "June", "July", "August", "September", "October",
      "November", "December" };
```

void printCalendar(int year)

```
{
    int weekDay;
    int daysFromJan01AD01 = 0;
    int daysInMonth[13] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    daysFromJan01AD01 = getDaysFromJan01AD01(year, 1, 1);
    weekDay = (daysFromJan01AD01 - 1 + WEEKDAY_AD01Jan01)
              % DAYS_PER_WEEK;
    if (isLeapYear(year))
        daysInMonth[2] = 29;
```



```

printf(">>>>>> Calendar of %d <<<<<<<\n", year);
for (int month = JAN; month <= DEC; month++)
{
    printf("%s\n", monthName[month]);
    printf("=====\n");
    for (int wk = SUN; wk <= SAT; wk++)
    {
        printf("%5s", weekDayName[wk]);
    }
    printf("\n-----\n");
    // 각 달 별로 1일 ~ 마지막 날까지 날짜 별로 요일에 맞추어 출력
    // 이 부분의 코드는 직접 작성할 것

    printf("=====\n");
    if (((month % 3) == 0) && (month != DEC))
    {
        printf("Hit any key to continue to next 3 months :");
        _getch();
        printf("\n");
    }
}
}

```



◆ 실행 결과

>>>>>>> Calendar of 2021 <<<<<<<<<																											
January							April							July							October						
SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT
					1	2					1	2	3						1	2							
3	4	5	6	7	8	9		4	5	6	7	8	9	10							3	4	5	6	7	8	9
10	11	12	13	14	15	16		11	12	13	14	15	16	17							10	11	12	13	14	15	16
17	18	19	20	21	22	23		18	19	20	21	22	23	24							17	18	19	20	21	22	23
24	25	26	27	28	29	30		25	26	27	28	29	30								24	25	26	27	28	29	30
31																					31						
February							May							August							November						
SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT
	1	2	3	4	5	6		2	3	4	5	6	7	8													
7	8	9	10	11	12	13		9	10	11	12	13	14	15							7	8	9	10	11	12	13
14	15	16	17	18	19	20		16	17	18	19	20	21	22							14	15	16	17	18	19	20
21	22	23	24	25	26	27		23	24	25	26	27	28	29							21	22	23	24	25	26	27
28								30	31												28	29	30				
March							June							September							December						
SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT	SUN	MON	TUE	WED	THR	FRI	SAT
	1	2	3	4	5	6				1	2	3	4	5													
7	8	9	10	11	12	13		6	7	8	9	10	11	12							5	6	7	8	9	10	11
14	15	16	17	18	19	20		13	14	15	16	17	18	19							12	13	14	15	16	17	18
21	22	23	24	25	26	27		20	21	22	23	24	25	26							19	20	21	22	23	24	25
28	29	30	31					27	28	29	30										26	27	28	29	30	31	

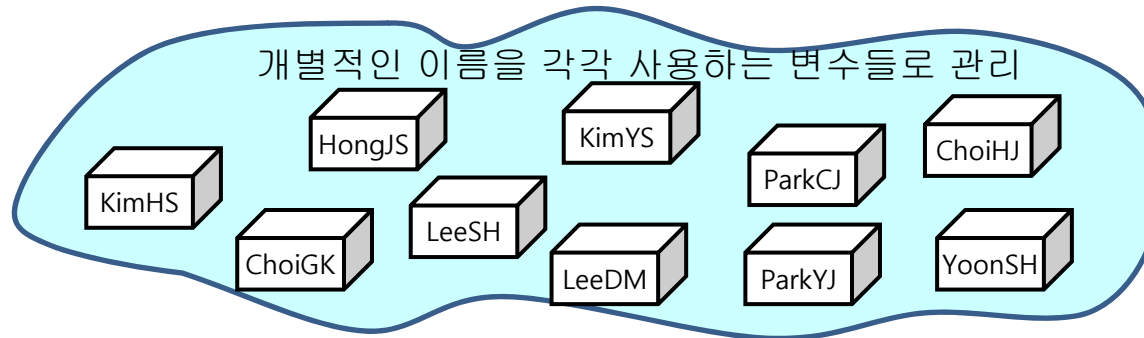
Hit any key to continue to next 3 monthsHit any key to continue to next 3 monthsHit any key to continue to next 3 months :



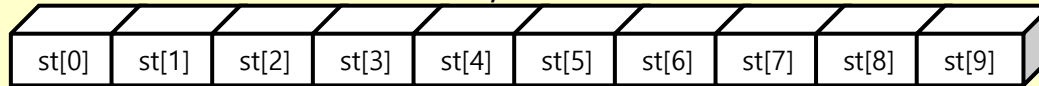
배열 (array)

배열이란?

◆ 여러 개의 데이터를 관리하는 방법

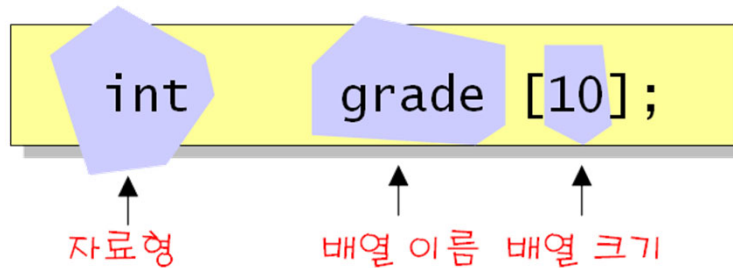


하나의 이름으로 배열을 관리, 개수가 증가해도 동일한 배열이름사용



- ◆ **배열(array):** 동일한 자료형의 데이터가 여러 개 저장되어 있는 데이터 저장 장소
- ◆ 배열 안에 들어있는 각각의 데이터들은 정수로 되어 있는 인덱스(첨자)에 의하여 접근
- ◆ 배열을 이용하면 동일한 자료형의 여러 개의 데이터 들을 하나의 이름으로 관리할 수 있다.

배열의 선언



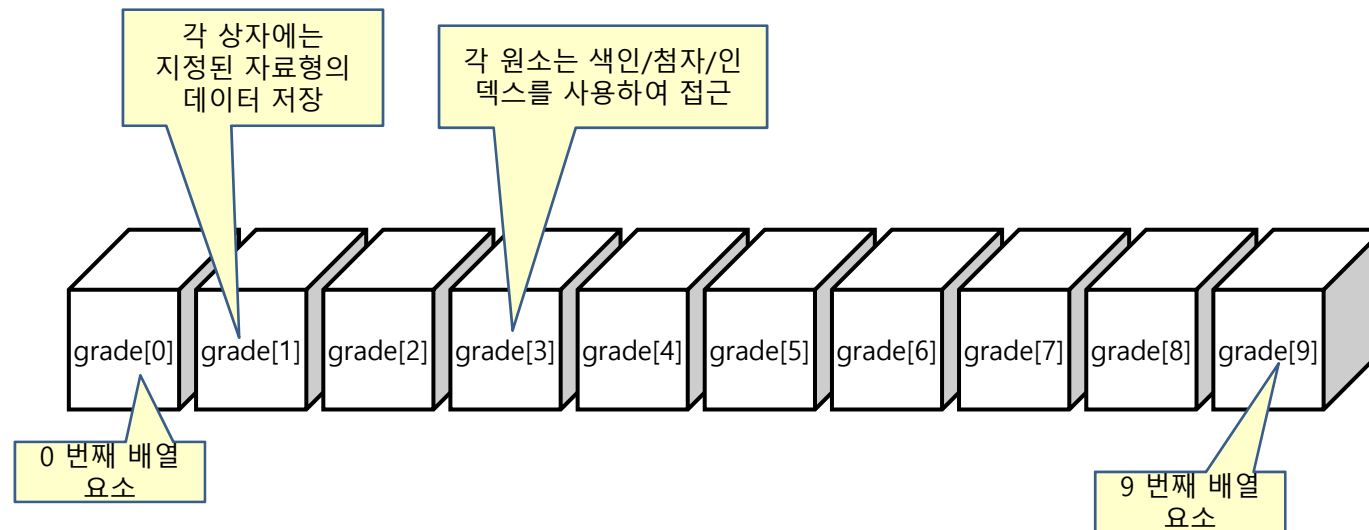
- ◆ 자료형: 배열 원소들이 **int**형라는 것을 의미
- ◆ 배열 이름: 배열을 사용할 때 사용하는 이름이 **grade**
- ◆ 배열 크기: 배열 원소의 개수가 **10**개
- ◆ 인덱스(배열 번호)는 항상 0부터 시작한다.

```
int score[60];           // 60개의 int형 값을 가지는 배열 grade
float cost[12];          // 12개의 float형 값을 가지는 배열 cost
char name[50];           // 50개의 char형 값을 가지는 배열 name
char src[10], dst[10];   // 2개의 문자형 배열을 동시에 선언
int index, days[7];      // 일반 변수와 배열을 동시에 선언
```

배열 원소와 인덱스

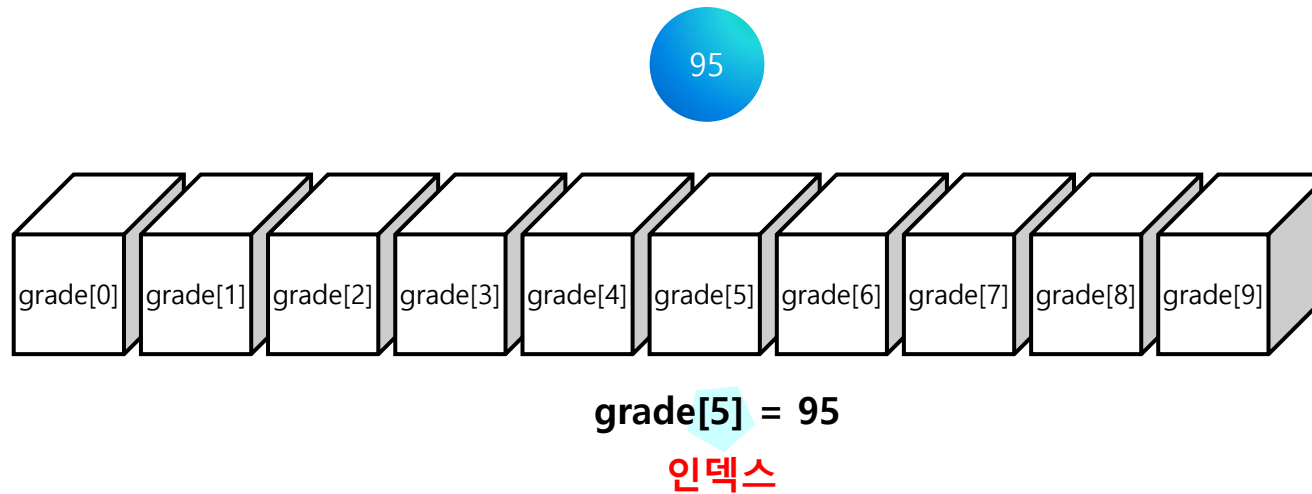
◆ 색인, 인덱스(index)

- 배열 원소의 색인 번호
- 총 N개의 배열 원소가 포함되어 있을 경우 인덱스는 0 ~ N-1



배열 원소 접근

- 배열 이름과 인덱스를 사용하여 배열 원소에 접근

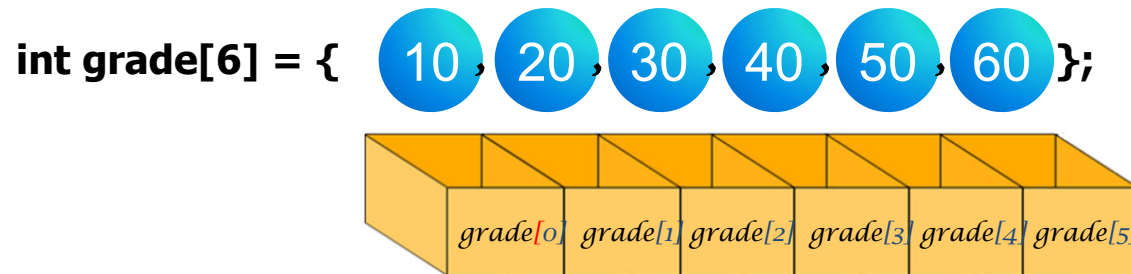


```
grade[5] = 95;
grade[1] = grade[0];
grade[i] = 100;      // i는 정수 변수
grade[i+2] = 100;    // 수식이 인덱스가 된다.
grade[index[3]] = 100; // index[]는 정수 배열
```

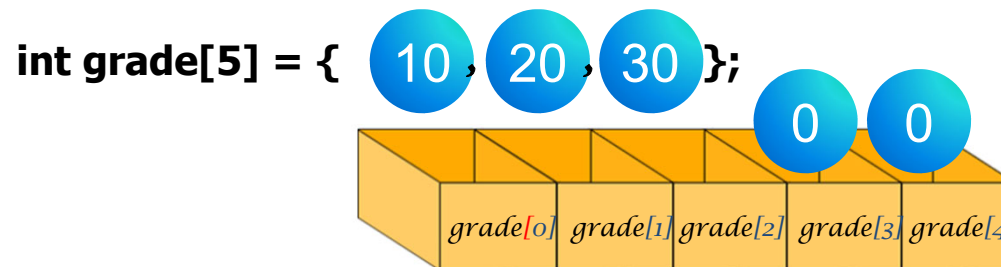


배열의 초기화

◆ `int grade[6] = { 10,20,30,40,50,60 };`



◆ `int grade[5] = { 10,20,30 };`



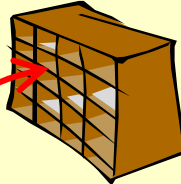
초기값을 일부
만 주면 나머지
원소들은 0으로
초기화됩니다.



배열을 함수의 인수 (argument)로 전달 (1)

- ◆ 배열의 경우에는 사본이 아닌 원본이 전달된다.

```
int main(void)
{
    ...

    get_average(  , int n);

    ...
}
```

```
int get_average(int score[], int n)
{
    ...

    sum += score[i];

    ...
}
```

배열 인수의
경우, 원본이
직접 참조됩
니다.



배열을 함수의 인수 (argument)로 전달 (2)

```
#include <stdio.h>
#define STUDENTS 5
int get_average(int score[], int n); // ①

int main(void)
{
    int grade[STUDENTS] = { 1, 2, 3, 4, 5 };
    int avg;

    avg = get_average(grade, STUDENTS);
    printf("평균은 %d입니다.\n", avg);
    return 0;
}

int get_average(int score[], int n) // ②
{
    int i;
    int sum = 0;

    for(i = 0; i < n; i++)
        sum += score[i];
    return sum / n;
}
```

배열이 인수인 경우,
참조에 의한 호출

배열의 원본이
score[]로 전달



배열이 함수의 인수인 예 (1)

```
#include <stdio.h>
#define SIZE 7

void square_array(int a[], int size);
void print_array(int a[], int size);

int main(void)
{
    int list[SIZE] = { 1, 2, 3, 4, 5, 6, 7 };

    print_array(list, SIZE);           // 배열은 원본이 전달된다. (인수 : 배열)
    square_array(list, SIZE);
    print_array(list, SIZE);

    return 0;
}
```



배열이 함수의 인수인 예 (2)

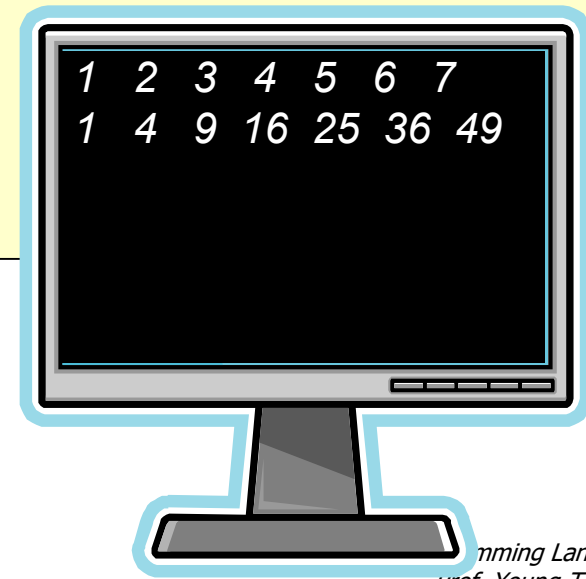
```
void square_array(int a[], int size)
```

```
{  
    int i;  
  
    for(i = 0; i < size; i++)  
        a[i] = a[i] * a[i];  
}
```

```
void print_array(int a[], int size)
```

```
{  
    int i;  
  
    for(i = 0; i < size; i++)  
        printf("%3d ", a[i]);  
    printf("\n");  
}
```

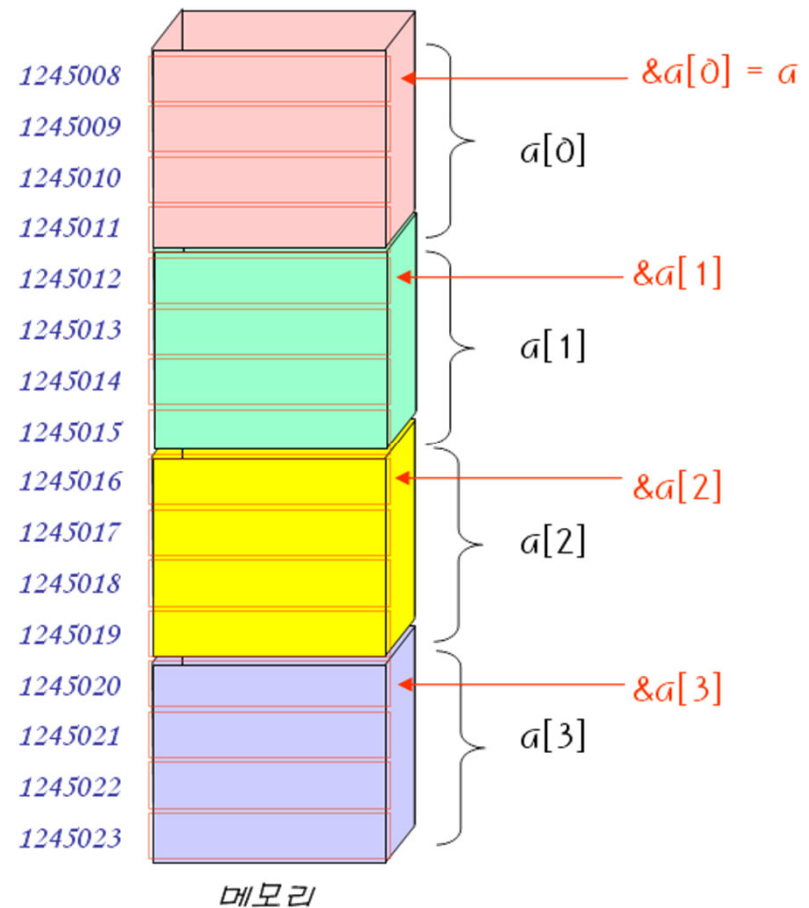
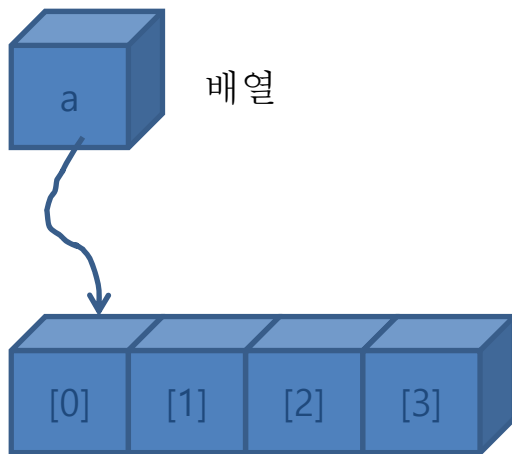
배열의 원본이
a[]로 전달



배열과 주소의 관계

◆ 배열의 각 요소들은 개별 주소가 지정된다.

```
int a[4];
```



Array.c

```
/* Array.c (1) */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "Array.h"
#include <math.h>

void printArray(int array[], int size, int line_size)
{
    for (int i = 0, count = 0; i < size; i++)
    {
        printf("%5d ", array[i]);
        count++;
        if (count % line_size == 0)
            printf("\n");
    }
    printf("\n");
}
```



```
/* Array.c (2) */
```

```
double sumArray(int array[], int size)
```

```
{  
    double sum = 0.0; // local variable  
    for (int i = 0; i < size; i++)  
        sum += array[i];  
    return sum; // return the result  
}
```

```
void genRandArray(int array[], int size)
```

```
{  
    int d;  
    char flag[MAX_ARRAY_SIZE] = { 0 };  
  
    srand((unsigned)time(NULL));  
  
    for (int i = 0; i < size; i++)  
    {  
        d = rand() % size;  
        while (flag[d] == 1) // while d was generated before, try new one  
            d = rand() % size;  
        flag[d] = 1; // mark d as generated  
        array[i] = d;  
    }  
}
```



```
/* Array.c (3) */
```

```
void getArrayStatistics(int data_array[], int num_data)
```

```
{
```

```
    int data, min, max;
```

```
    double sum = 0.0, var, diff, sq_diff_sum = 0.0, avg, std_dev;
```

```
    min = INT_MAX;
```

```
    max = INT_MIN;
```

```
    for (int i = 0; i < num_data; i++)
```

```
    {
```

```
        data = data_array[i];
```

```
        if (data < min)
```

```
            min = data;
```

```
        if (data > max)
```

```
            max = data;
```

```
    }
```

```
    sum = sumArray(data_array, num_data);
```

```
    avg = sum / num_data;
```

```
    sq_diff_sum = 0.0;
```

```
    for (int i = 0; i < num_data; i++)
```

```
    {
```

```
        diff = data_array[i] - avg;
```

```
        sq_diff_sum += diff * diff;
```

```
    }
```

```
    var = sq_diff_sum / num_data;
```

```
    std_dev = sqrt(var);
```

```
    printf("Total (%3d) integer data : \n", num_data);
```

```
    printArray(data_array, num_data, 10);
```

```
    printf("min (%3d), max (%3d), ", min, max);
```

```
    printf("sum (%8.2lf), average (%8.2lf), ", sum, avg);
```

```
    printf("variance (%8.2lf), standard deviation (%8.2lf)\n", var, std_dev);
```

```
}
```



```
/* Array.c (5) */
```

```
void suffleArray(int array[], int size)
```

```
{  
    int i1, i2, d;  
    srand((unsigned)time(NULL));  
  
    for (int i = 0; i < size / 2; i++)  
    {  
        i1 = rand() % size;  
        i2 = rand() % size;  
  
        /* suffle array*/  
        d = array[i1];  
        array[i1] = array[i2];  
        array[i2] = d;  
    }  
}
```



다차원 배열 **(Multi-dimensional Array)**

2차원 배열

```
int s[10];    // 1차원 배열  
int s[3][10]; // 2차원 배열  
int s[5][3][10]; // 3차원 배열
```

```
#define NUM_CLASSES 3  
#define NUM_STUDENTS 5  
int s[NUM_CLASSES][NUM_STUDENTS];
```

행(row)

열(column)

s[0][0]	s[0][1]	s[0][2]	s[0][3]	s[0][4]
s[1][0]	s[1][1]	s[1][2]	s[1][3]	s[1][4]
s[2][0]	s[2][1]	s[2][2]	s[2][3]	s[2][4]



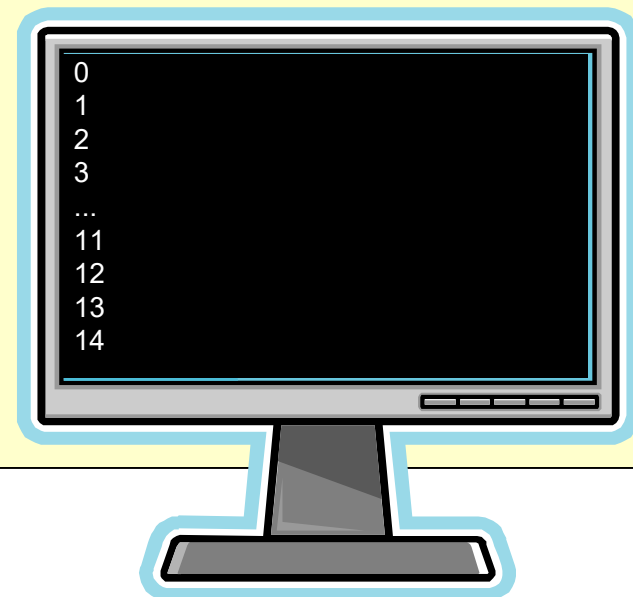
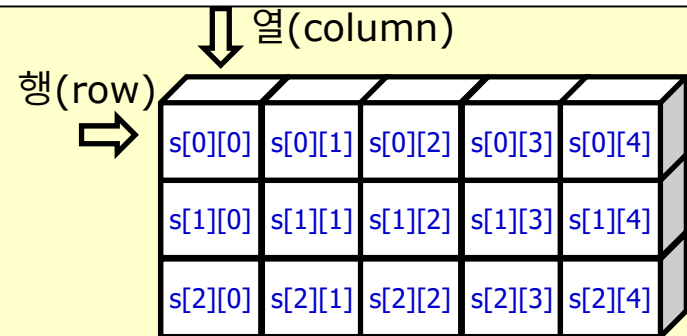
2차원 배열의 활용

```
#include <stdio.h>
#define ROWS 3
#define COLS 5
int main(void)
{
    int s[ROWS][COLS]; // 2차원 배열 선언
    int i, j;           // 2개의 인덱스 변수
    int value = 0;      // 배열 원소에 저장되는 값

    for(i=0;i<ROWS;i++)
        for(j=0;j< COLS;j++)
            s[i][j] = value++;

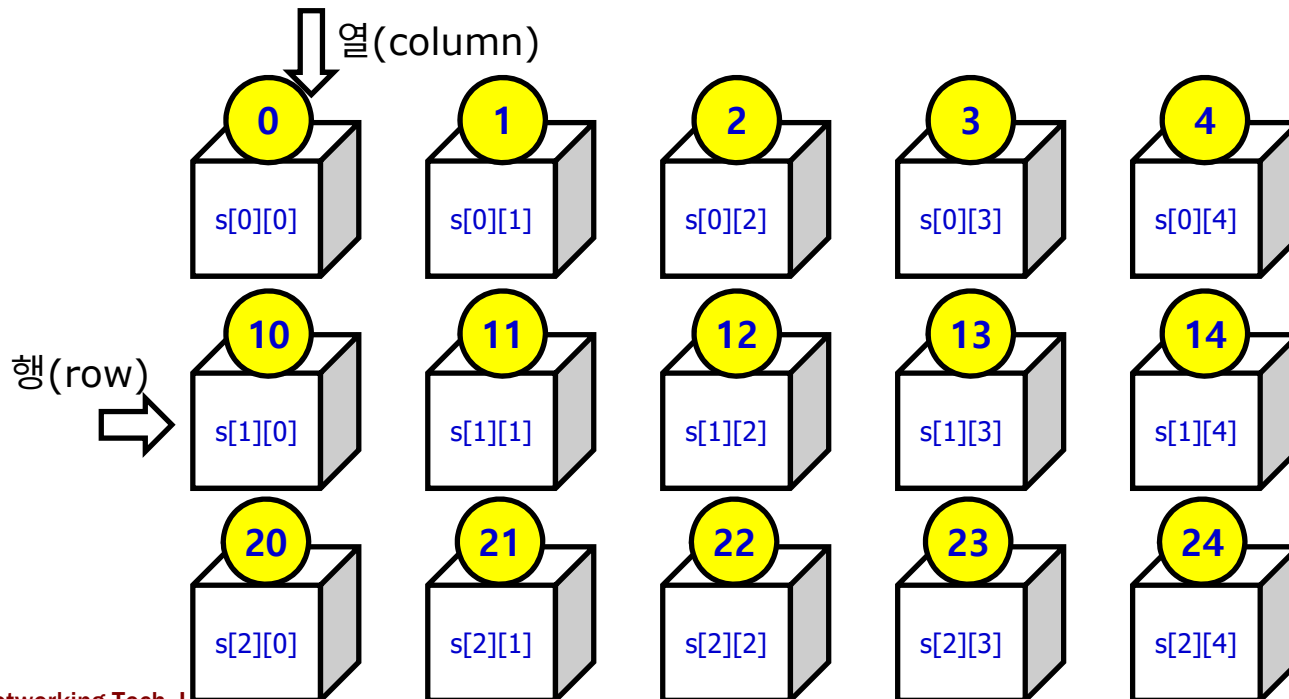
    for(i=0;i<ROWS;i++)
        for(j=0;j< COLS;j++)
            printf("%d\n", s[i][j]);

    return 0;
}
```



2차원 배열의 초기화

```
#define ROWS 3
#define COLS 5
int s[ROWS][COLS] = {
    { 0, 1, 2, 3, 4 }, // 첫 번째 행의 원소들의 초기값
    { 10, 11, 12, 13, 14 }, // 두 번째 행의 원소들의 초기값
    { 20, 21, 22, 23, 24 } // 세 번째 행의 원소들의 초기값
};
```



3차원 배열

```
int s [6][3][5];
```

첫번째 두번째 세번째
인덱스: 인덱스: 인덱스:
학년번호 학급번호 학생번호

```
#include <stdio.h>
int main(void)
{
    int s[3][3][3];    // 3차원 배열 선언
    int x, y, z;        // 3개의 인덱스 변수
    int i = 1;          // 배열 원소에 저장되는 값

    for(z=0;z<3;z++)
        for(y=0;y<3;y++)
            for(x=0;x<3;x++)
                s[z][y][x] = i++;

    return 0;
}
```



다차원 배열 인수

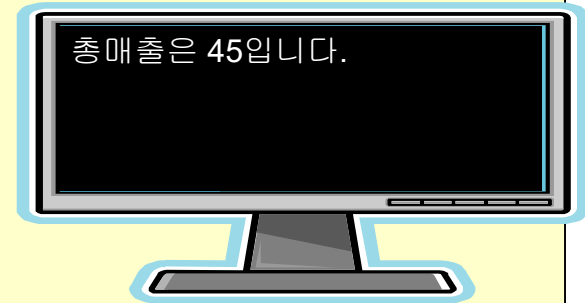
```
#include <stdio.h>
#define YEARS    3
#define PRODUCTS 5

int sum(int grade[][PRODUCTS]);

int main(void)
{
    int sales[YEARS][PRODUCTS] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
    int total_sale;
    total_sale = sum(sales);

    printf("총매출은 %d입니다.\n", total_sale);
    return 0;
}

int sum(int grade[][PRODUCTS])
{
    int y, p;
    int total = 0;
    for(y = 0; y < YEARS; y++)
        for(p = 0; p < PRODUCTS; p++)
            total += grade[y][p];
    return total;
}
```



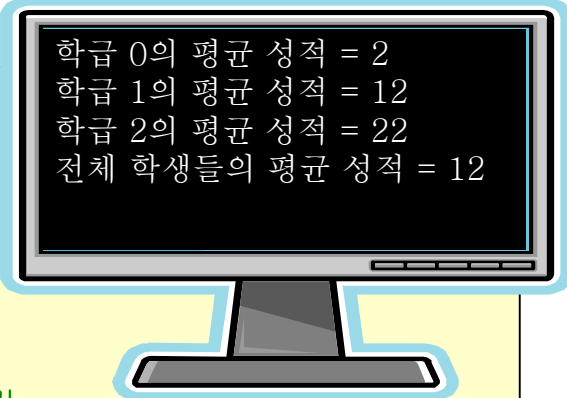
첫번째 인덱스의 크기는
적지 않아도 된다.



다차원 배열 예제

```
#include <stdio.h>
#define CLASSES 3
#define STUDENTS 5

int main(void)
{
    int s[CLASSES][STUDENTS] = {
        { 0, 1, 2, 3, 4 },    // 첫 번째 행의 원소들의 초기값
        { 10, 11, 12, 13, 14 }, // 두 번째 행의 원소들의 초기값
        { 20, 21, 22, 23, 24 }, // 세 번째 행의 원소들의 초기값
    };
    int clas, student, total, subtotal;
    total = 0;
    for(clas = 0; clas < CLASSES; clas++)
    {
        subtotal = 0;
        for(student = 0; student < STUDENTS; student++)
            subtotal += s[clas][student];
        printf("학급 %d의 평균 성적= %d\n", clas, subtotal / STUDENTS);
        total += subtotal;
    }
    printf("전체 학생들의 평균 성적= %d\n", total / (CLASSES * STUDENTS));
    return 0;
}
```



학급 0의 평균 성적 = 2
학급 1의 평균 성적 = 12
학급 2의 평균 성적 = 22
전체 학생들의 평균 성적 = 12



2차원 배열과 행렬 (Matrix)

- ◆ 행렬(matrix)은 자연과학에서 많은 문제를 해결하는데 사용

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 8 & 9 & 1 \\ 7 & 0 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 7 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}$$

Mathematics - ELEMENTARY MATRIX OPERATIONS

OPERATION: MULTIPLY EACH ELEMENT IN 2ND Row by 7:

$A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$

1) FIND E 2×2 $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix}$
 I E

2) PREMULT $\begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 1(0)+0(3) & 1(1)+0(4) & 1(2)+0(5) \\ 0(0)+7(3) & 0(1)+7(4) & 0(2)+7(5) \end{bmatrix}$



2차원 배열을 이용한 행렬의 표현

```
#include <stdio.h>
#define ROWS 3
#define COLS 3

int main(void)
{
    int A[ROWS][COLS] = { { 2,3,0 },
                          { 8,9,1 },
                          { 7,0,5 } };
    int B[ROWS][COLS] = { { 1,0,0 },
                          { 1,0,0 },
                          { 1,0,0 } };

    int C[ROWS][COLS];
    int r, c; // row, column
    // 두 개의 행렬을 더한다.
    for(r = 0; r < ROWS; r++)
        for(c = 0; c < COLS; c++)
            C[r][c] = A[r][c] + B[r][c];
    // 행렬을 출력한다.
    for(r = 0; r < ROWS; r++)
    {
        for(c = 0; c < COLS; c++)
            printf("%d ", C[r][c]);
        printf("\n");
    }
    return 0;
}
```

중첩 for 루프를 이용하여 행렬 A의 각
원소들과 행렬의 B의 각 원소들을 서
로 더하여 행렬 C에 대입한다.



4x4 행렬의 계산

```
/** matrix4_4.cpp */

#include <iostream>
#include <iomanip>
using namespace std;

#define SIZE_4 4

void printMtrx(int mA[][SIZE_4], int size_n);
void addMtrx(int mA[][SIZE_4], int mB[][SIZE_4], int mX[][SIZE_4], int size_n);
void subtractMtrx(int mA[][SIZE_4], int mB[][SIZE_4], int mX[][SIZE_4], int size_n);
void multiplyMtrx(int mA[][SIZE_4], int mB[][SIZE_4], int mX[][SIZE_4], int size_n);

int main()
{
    int mA[4][4] = {{1, 2, 3, 4},
                    {5, 6, 7, 8},
                    {9, 10, 11, 12},
                    {13, 14, 15, 16}};
    int mB[4][4] = {{1, 0, 0, 0},
                    {0, 2, 0, 0},
                    {0, 0, 3, 0},
                    {0, 0, 0, 4}};
    int mC[4][4];
    int mD[4][4];
    int mE[4][4];
}
```

Matrix mA:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Matrix mB:

1	0	0	0
0	2	0	0
0	0	3	0
0	0	0	4



4x4 행렬의 계산

```
/** matrix4_4.cpp (cont.) */

    printf("\n Matrix mA:\n");
    printMtrx(mA, 4);

    printf("\n Matrix mB:\n");
    printMtrx(mB, 4);

    addMtrx(mA, mB, mC, 4);
    printf("\n Matrix mC = mA + mB:\n");
    printMtrx(mC, 4);

    subtractMtrx(mA, mB, mD, 4);
    printf("\n Matrix mD = mA - mB:\n");
    printMtrx(mD, 4);

    multiplyMtrx(mA, mB, mE, 4);
    printf("\n Matrix mE = mA x mB:\n");
    printMtrx(mE, 4);

    printf("\n");

    return 0;

}
```



확장 완성형 코드를 사용한 4x4 행렬의 출력

```
void printMtrx(int mA[][SIZE_4], int size)
{
    unsigned char a6 = 0xA6, a1 = 0xA1, a2 = 0xA2;
    unsigned char a3 = 0xA3, a4 = 0xA4, a5 = 0xA5;

    for (int i=0; i< size_n; i++) {
        for (int j=0; j< size; j++)
        {
            if ((i==0) && (j==0))
                printf("%c%c%c%3d", a6, a3, mA[i][j]);
            else if ((i==0) && j==(size -1))
                printf("%3d%c%c", mA[i][j], a6, a4);
            else if ((i>0) && (i<size-1) && (j==0))
                printf("%c%c%c%3d", a6, a2, mA[i][j]);
            else if ((i>0) && (i<size-1) && (j== (size -1)))
                printf("%3d%c%c", mA[i][j], a6, a2);
            else if ((i==(size-1)) && (j==0))
                printf("%c%c%c%3d", a6, a6, mA[i][j]);
            else if ((i==(size-1)) && (j==(size -1)))
                printf("%3d%c%c", mA[i][j], a6, a5);
            else
                printf("%3d", mA[i][j]);
        }
        printf("\n");
    }
}
```

출력 결과	확장 완성형 코드
—	0xa6, 0xa1
	0xa6, 0xa2
┌	0xa6, 0xa3
┐	0xa6, 0xa4
└	0xa6, 0xa5
┘	0xa6, 0xa6

Matrix mA:

```
[ 1  2  3  4
  5  6  7  8
  9 10 11 12
 13 14 15 16]
```

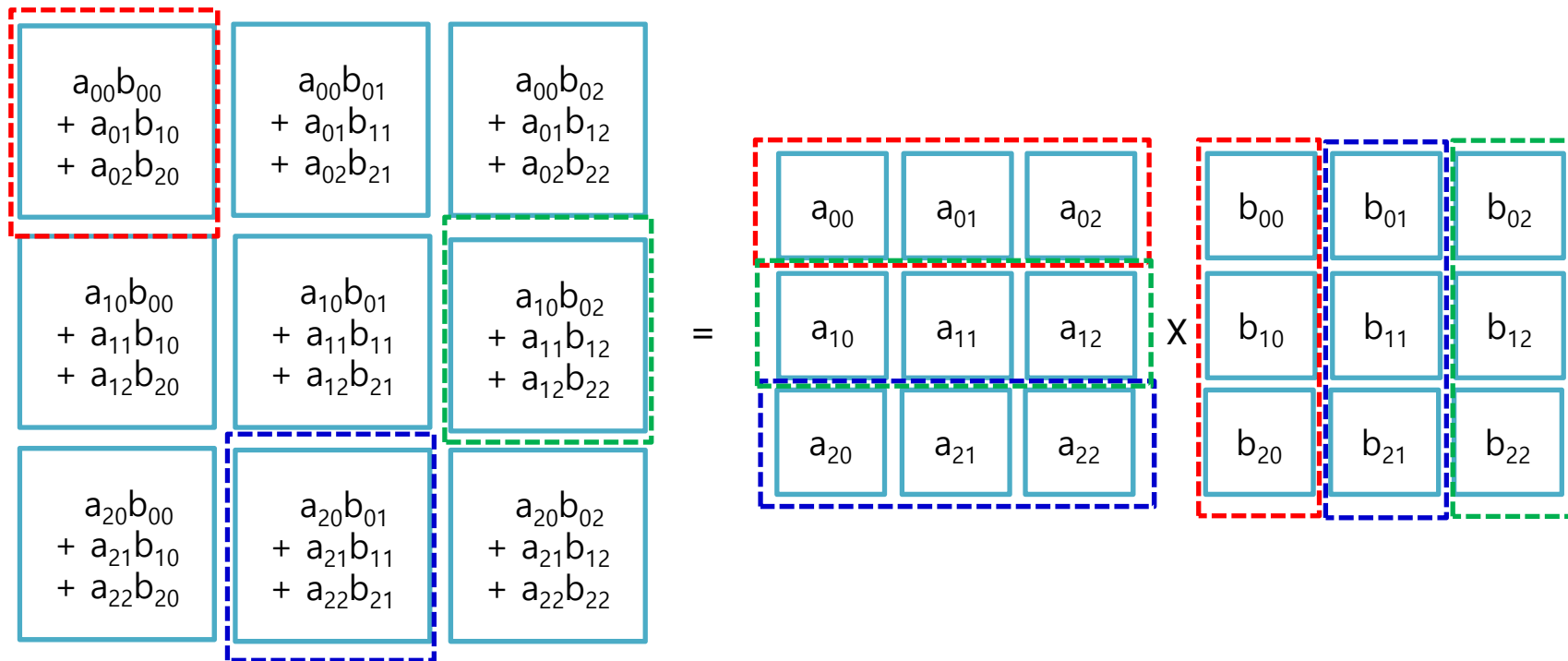
Matrix mB:

```
[ 1  0  0  0
  0  2  0  0
  0  0  3  0
  0  0  0  4]
```



행렬의 곱셈

◆ 행렬의 곱셈 계산



4x4 행렬의 덧셈, 뺄셈, 곱셈

```
void addMtrx(int mA[][SIZE_4], int mB[][SIZE_4], int mX[][SIZE_4], int size)
```

```
{  
    for (int i=0; i<size; i++)  
        for (int j=0; j<size; j++)  
            mX[i][j] = mA[i][j] + mB[i][j];  
}
```

```
void subtractMtrx(int mA[][SIZE_4], int mB[][SIZE_4], int mX[][SIZE_4], int size)
```

```
{  
    for (int i=0; i<size; i++)  
        for (int j=0; j<size; j++)  
            mX[i][j] = mA[i][j] - mB[i][j];  
}
```

```
void multiplyMtrx(int mA[][SIZE_4], int mB[][SIZE_4], int mX[][SIZE_4], int size)
```

```
{  
    for (int i=0; i<size; i++)  
        for (int j=0; j<size; j++)  
        {  
            mX[i][j] = 0;  
            for (int k=0; k<size; k++)  
                mX[i][j] += mA[i][k] * mB[k][j];  
        }  
}
```



실행결과

Matrix mA:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Matrix mB:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

Matrix mC = mA + mB:

$$\begin{bmatrix} 2 & 2 & 3 & 4 \\ 5 & 8 & 7 & 8 \\ 9 & 10 & 14 & 12 \\ 13 & 14 & 15 & 20 \end{bmatrix}$$

Matrix mD = mA - mB:

$$\begin{bmatrix} 0 & 2 & 3 & 4 \\ 5 & 4 & 7 & 8 \\ 9 & 10 & 8 & 12 \\ 13 & 14 & 15 & 12 \end{bmatrix}$$

Matrix mE = mA x mB:

$$\begin{bmatrix} 1 & 4 & 9 & 16 \\ 5 & 12 & 21 & 32 \\ 9 & 20 & 33 & 48 \\ 13 & 28 & 45 & 64 \end{bmatrix}$$



실습 3 문제 및 Oral Test

실습 3

3.1 지정된 연월일이 서기 1년 1월 1일로부터 몇 번째 날인지, 그리고 무슨 요일인지를 계산하여 출력하는 프로그램 작성

- 1) 연월일 3개 정수로 지정된 날짜가 서기 (AD) 1년 1월 1일로부터 몇 번째 날이며, 무슨 요일인가를 계산하여 출력하기 위한 함수 `bool isLeapYear(int y)`와 `int getDaysFromJan01AD01(int year, int month, int day)`를 작성하라. 참고로 서기 1년 1월 1일은 월요일이다.
- 2) 오늘의 연 월 일 3개 정수를 입력 받아 서기 (AD) 1년 1월 1일로부터 몇 번째 날이며, 무슨 요일인가를 계산하여 출력하라. 달에 해당하는 이름을 출력하고, 요일에 해당하는 이름을 각각 출력할 것.
- 3) 작성된 프로그램을 사용하여, 2011년 이후 올해까지의 매년 1월 1일이 서기 1년 1일로부터 몇 번째 날인지, 그리고 무슨 요일인가를 출력하라.



실습 3

◆ 실행 결과 (예시)

```
Input year, month, day to check : 2021 3 13
The week day of March 13, 2021 is 737862-th day from Jan 1, 1 (SAT)

The week day of January 1, 2011 is 734138-th day from Jan 1, 1 (SAT)
The week day of January 1, 2012 is 734503-th day from Jan 1, 1 (SUN)
The week day of January 1, 2013 is 734869-th day from Jan 1, 1 (TUE)
The week day of January 1, 2014 is 735234-th day from Jan 1, 1 (WED)
The week day of January 1, 2015 is 735599-th day from Jan 1, 1 (THR)
The week day of January 1, 2016 is 735964-th day from Jan 1, 1 (FRI)
The week day of January 1, 2017 is 736330-th day from Jan 1, 1 (SUN)
The week day of January 1, 2018 is 736695-th day from Jan 1, 1 (MON)
The week day of January 1, 2019 is 737060-th day from Jan 1, 1 (TUE)
The week day of January 1, 2020 is 737425-th day from Jan 1, 1 (WED)
The week day of January 1, 2021 is 737791-th day from Jan 1, 1 (FRI)
```



실습 3

3.2 5 x 5 크기의 행렬 A, B에 대한 덧셈, 뺄셈, 곱셈 연산

- 1) 2개의 5 x 5 크기의 행렬 A, B의 덧셈, 뺄셈, 곱셈을 계산하여 그 결과를 C, D, E에 저장하는 행렬 연산 함수 3개(addMtrx(), subtractMtrx(), multiplyMtrx())를 작성하라.

```
void addMtrx(double A[][SIZE], double B[][SIZE], double X[][SIZE], int size);  
void subtractMtrx(double A[][SIZE], double B[][SIZE], double X[][SIZE], int size);  
void multiplyMtrx(double A[][SIZE], double B[][SIZE], double X[][SIZE], int size);
```
- 2) 2차원 배열 A와 B를 각각 초기화하라. 초기화 데이터는 다음과 같이 설정할 것.
- 3) 초기화 된 2개의 행렬을 printMtrx(double M[][SIZE], int size)함수를 사용하여 출력할 것. 이 때, 행렬을 표시하기 위하여, 확장 완성형 코드를 사용할 것.
- 4) 행렬 A, B의 덧셈, 뺄셈, 곱셈을 위 1)에서 작성한 함수들을 사용하여 실행하라.
- 5) 행렬 연산을 총괄하는 main() 함수에서 2개의 행렬 초기화를 실행하고, printfMtrx(), addMtrx(), subtractMtrx(), multiplyMtrx() 함수를 호출하며, 행렬 준비, 행렬 연산 및 결과 출력을 실행하고, 그 결과를 확인할 것.



실습 3

◆ 실행결과 (예시)

```
double A[MTRX_SIZE][MTRX_SIZE] =
{
    {1, 2, 3, 4, 5},
    {6, 7, 8, 9, 10},
    {11, 12, 13, 14, 15},
    {16, 17, 18, 19, 20},
    {21, 22, 23, 24, 25}
};

double B[MTRX_SIZE][MTRX_SIZE] =
{
    {1, 0, 0, 0, 0},
    {0, 1, 0, 0, 0},
    {0, 0, 1, 0, 0},
    {0, 0, 0, 1, 0},
    {0, 0, 0, 0, 1}
};

double C[MTRX_SIZE][MTRX_SIZE] = { 0 };
double D[MTRX_SIZE][MTRX_SIZE] = { 0 };
double E[MTRX_SIZE][MTRX_SIZE] = { 0 };

printf("Matrix A : \n");
printMtrx(A, MTRX_SIZE);

printf("Matrix B : \n");
printMtrx(B, MTRX_SIZE);

printf("Matrix C = A + B: \n");
addMtrx(A, B, C, MTRX_SIZE);
printMtrx(C, MTRX_SIZE);

printf("Matrix D = A - B: \n");
subtractMtrx(A, B, D, MTRX_SIZE);
printMtrx(D, MTRX_SIZE);

printf("Matrix E = A * B: \n");
multiplyMtrx(A, B, E, MTRX_SIZE);
printMtrx(E, MTRX_SIZE);
```

Matrix A :

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

Matrix B :

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix C = A + B:

$$\begin{bmatrix} 2 & 2 & 3 & 4 & 5 \\ 6 & 8 & 8 & 9 & 10 \\ 11 & 12 & 14 & 14 & 15 \\ 16 & 17 & 18 & 20 & 20 \\ 21 & 22 & 23 & 24 & 26 \end{bmatrix}$$

Matrix D = A - B:

$$\begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 6 & 6 & 8 & 9 & 10 \\ 11 & 12 & 12 & 14 & 15 \\ 16 & 17 & 18 & 18 & 20 \\ 21 & 22 & 23 & 24 & 24 \end{bmatrix}$$

Matrix E = A * B:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$


Oral Test

Q3.1 enum을 사용하는 방법에 대하여 예를 들어 설명하라. enum으로 선언되는 기호 상수를 사용하여 요일 (weekday)의 이름과 달 (month)의 이름을 출력하는 방법에 대하여 설명하라.

(**Key points:** 요일과 달의 영어 이름을 각각 enum으로 열거)

Q3.2 반복문의 실행에서 continue와 break가 실행되면 어떤 결과가 나타나는지에 대하여 설명하라.

(**Key points:** for-loop내부에 continue와 break가 포함된 예를 구분하여 설명)

Q3.3 함수호출에서 2차원 배열 2개 (A, B)를 인수로 전달하고, 그 배열들에 대한 지정된 연산 (+, -, *)을 수행하여 또 다른 2차원 배열 X로 반환하기 위한 함수 matrixOperation() 구현 방법에 대하여 설명하라.

(**Key points:** 이 함수 호출에 사용되는 인수 (argument/parameter) 구성과 함수 본문에서의 행렬 연산을 위한 다중 for-loop 구성 방법을 상세하게 설명할 것.)

Q3.4 $N \times K$ 행렬 A와 $K \times M$ 행렬 B의 곱셈 계산을 하여 $N \times M$ 행렬 C에 저장하는 위한 반복문의 구성에 대하여 설명하고, 각 항목의 계산이 어떤 순서로 실행되는가에 대하여 그림으로 표현하고, 이를 설명하라.

(**Key points:** 3중 for-loop의 실행 단계별로 계산에 해당되는 배열 원소를 표시할 것)

