

10장 배열 연습문제 풀이

01 배열이 10개의 요소를 가진다면 첫 번째 요소의 배열 번호는?

- ① -1                      ② 0                      ③ 1                      ④ 0 또는 1

02 배열 char a[5][6]은 총 몇 개의 요소를 가지는가?

- ① 20                      ② 24                      ③ 30                      ④ 42

03 다음 중 잘못 선언된 배열을 모두 선택하라.

- ① int a[200+1];                      ② int a[MAX]; // MAX는 기호상수  
③ int a[v]; // v는 정수변수                      ④ int a[100.25];

04 int a[100] = { 10 }; 으로 선언된 배열에서 a[99]의 초기값은 무엇인가?

- ① 0                      ② 알 수 없다.  
③ 전역 변수이면 0, 지역 변수이면 알 수 없다.                      ④ 99

05 int a[2][2] = { 1, 2, 3, 4 }에서 a[1][1]의 초기값은?

- ① 1                      ② 2                      ③ 3                      ④ 4

06 다음 프로그램의 오류를 전부 수정하라.

```
#define MAX_SIZE 3
int main(void)
{
    int a[MAX_SIZE] = { 0, 1, 2, 3 };
    int b[3.0];

    for(i=0;i<=MAX_SIZE; i++)
        b[i]=a[i];
    return 0;
}
```

1. (2)

2. (3)

3. (3), (4)

4. (1)

5. (4)

6.

#define MAX\_SIZE 4

int main(void)

{

int a[MAX\_SIZE] = { 0, 1, 2, 3 };

int b[4];

int i;

for(i=0;i<MAX\_SIZE; i++)

b[i]=a[i];

return 0;

}

07 다음은 3명의 학생들의 성적을 입력받아서 평균을 구하는 프로그램의 일부이다.

```
float scores1, scores2, scores3;
float average;

printf("3명의 성적을 입력하시오: ");
scanf("%f %f %f", &scores1, &scores2, &scores3);

average = (scores1 + scores2 + scores3)/3;
```

- (a) 위의 프로그램을 배열을 사용하도록 수정하라.  
 (b) 사용자로부터 값을 입력받지 않고 배열의 초기값을 10.0, 20.0, 30.0으로 주도록 수정하라.  
 (c) 평균을 구하는 부분을 함수 get\_average(float a[], int n)으로 작성하고 이 함수를 호출하여서 평균을 구하도록 위의 프로그램을 수정하라.  
 (d) 다음과 같은 10점 단위의 점수대 분포도를 출력하도록 코드를 추가하라. 분포도도 배열로 구현하라.

점수대	인원수
0-10	0
11-20	1
...	...

08 다음 코드에서 잘못된 부분이 있으면 지적하고 수정하라.

- (a) 

```
int main(void)
{
    int scores[10][60];
    compute_avg(scores);
}
```

```
int compute_avg(int array[ ][ ]){ ... }
```

(b) 

```
float test[10];
scanf("%f", test[0]);
```

(c) 

```
int main(void)
{
    int x, y;
    int test[10][5];

    for(x = 0; x < 5; x++)
        for(y = 0; y < 10; y++)
            test[x][y] = 0;

    return 0;
}
```

7. (a) 

```
float grade[3];
float average, sum;
int i;
for(i=0;i<3;i++){
    printf("성적을 입력하시오: ");
    scanf("%f", &grade[i]);
}
sum = 0.0;
for(i=0;i<3;i++){
    sum += grade[i];
}
average = sum /3.0;
```

(b) 

```
float grade[3]={10.0, 20.0, 30.0};
float average, sum;
int i;
sum = 0.0;
for(i=0;i<3;i++){
    sum += grade[i];
}
average = sum /3.0;
```

(c) // 함수 버전  

```
float get_average(float a[], int n);
int main(void)
{
    float grade[3]={10.0, 20.0, 30.0};
    get_average(grade, 3);
    return 0;
}

float get_average(float a[], int n)
{
    float average, sum;
    int i;
    sum = 0.0;
    for(i=0;i<3;i++){
        sum += a[i];
    }
    average = sum /3.0;
}
```

(a) 2차원 배열을 전달하는 경우에 첫 번째 인덱스의 크기만 생략이 가능하다.  

```
int main(void)
{
    int grade[10][60];
    compute_avg(grade);
}

int compute_avg(int array[ ][60]) { ... }
```

(b) 배열 원소인 경우에는 주소를 보내주어야 한다.

```
float test[10];
scanf("%f", &test[0]);
```

(c) 

```
int main(void)
{
    int x, y;
    int test[10][5];

    for(x = 0; x < 10; x++)
        for(y = 0; y < 5; y++)
            test[x][y] = 0;

    return 0;
}
```

(d) 

```
void get_freq(float s[], int freq[], int n);
int main(void)
{
    float grade[3] = { 10.0, 20.0, 30.0 };
    int freq[10]={0};
    get_freq(grade, freq, 3);
    return 0;
}

void get_freq(float score[], int f[10], int n)
{
    int i, range;
    for(i=0;i<n;i++){
        range = score[i]/10;
        f[range]++;
    }
}
```

01 배열 days[]를 아래와 같이 초기화하고 배열 요소의 값을 다음과 같이 출력하는 프로그램을 작성하라.

31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

```
C:\WINDOWS\system32\cmd.exe
8월은 31일까지 있습니다.
9월은 30일까지 있습니다.
10월은 31일까지 있습니다.
11월은 30일까지 있습니다.
12월은 31일까지 있습니다.
```

**HINT** 배열을 초기화하려면 int x[] = { 1, 2 };와 같이 한다.

```
#include <stdio.h>
int days[] = { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
int main(void)
{
    int i;
    for (i = 0; i < 12; i++)
        printf("%d월은 %d일까지 있습니다.\n", i + 1, days[i]);
    return 0;
}
```

02 크기가 10인 1차원 배열에 난수를 저장한 후에, 최대값과 최소값을 출력하는 프로그램을 작성하라. 난수는 rand() 함수를 호출하여 생성하라.

```
C:\WINDOWS\system32\cmd.exe
최대값은 29358
최소값은 41
```

**HINT** x[i] = rand(); // 난수를 생성하여서 i번째 배열 요소에 대입한다.

```
#include <stdio.h>
int main(void)
{
    int list[10];
    int i, max, min;

    for (i = 0; i < 10; i++)
        list[i] = rand();
    max = min = list[0];
    for (i = 1; i < 10; i++) {
        if (list[i] < min)
            min = list[i];
        if (list[i] > max)
            max = list[i];
    }

    printf("최대값은 %d\n", max);
    printf("최소값은 %d\n", min);
    return 0;
}
```

03 2개의 정수 배열 a, b를 받아서 대응되는 배열 요소가 같은지를 검사하는 함수 array\_equal(int a[], int b[], int size)를 작성하고 테스트하라. 이 함수는 a[0]와 b[0], a[1]과 b[1], ... , a[size-1]와 b[size-1]가 같은지를 검사한다. 만약 전체 요소가 같다면 1을 반환하고 그렇지 않으면 0을 반환한다.

```
C:\WINDOWS\system32\cmd.exe
1 2 3 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
2개의 배열은 다름
```

**HINT** 반복 루프를 이용해서 배열의 각 요소가 같은지를 검사한다. 만약 하나라도 다르면 0을 바로 반환하면 된다.

```
#include <stdio.h>
#define N_DATA 10
int array_equal(int a[], int b[], int size);
void array_print(int a[], int size)
{
    int i;
    for (i = 0; i < size; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
int main(void)
{
    int A[N_DATA] = { 1, 2, 3 };
    int B[N_DATA] = { 0 };
    array_print(A, 10);
    array_print(B, 10);
    if (array_equal(A, B, N_DATA) == 1)
        printf("2개의 배열은 같음 \n");
    else
        printf("2개의 배열은 다름 \n");
    return 0;
}
int array_equal(int a[], int b[], int size)
{
    int i;
    for (i = 0; i < size; i++) {
        if (b[i] != a[i])
            return 0;
    }
    return 1;
}
```

04 2개의 정수 배열 a, b를 받아서 배열 a의 요소를 배열 b로 복사하는 함수 array\_copy(int a[], int b[], int size)를 작성하고 테스트하라. 이 함수는 a[0]를 b[0]에, a[1]를 b[1]에, ... ,a[size-1]을 b[size-1]에 대입한다. 이 함수의 반환값은 없다.

```
C:\WINDOWS\system32\cmd.exe
1 2 3 0 0 0 0 0 0
1 2 3 0 0 0 0 0 0
```

**HINT** 반복 루프를 이용해서 배열의 각 요소를 복사한다.

```
#include <stdio.h>
#define N_DATA 10
void array_copy(int a[], int b[], int size);
void array_print(int a[], int size);

int main(void)
{
    int A[N_DATA] = { 1, 2, 3 };
    int B[N_DATA] = { 0 };

    array_print(A, N_DATA);
    array_copy(A, B, N_DATA);
    array_print(B, N_DATA);
    return 0;
}

void array_copy(int a[], int b[], int size)
{
    int i;
    for (i = 0; i < size; i++) {
        b[i] = a[i];
    }
}

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

05 0부터 9까지의 난수를 100번 생성하여 가장 많이 생성된 수를 출력하는 프로그램을 작성하라. 난수는 rand() 함수를 사용하여 생성하라.

```
C:\WINDOWS\system32\cmd.exe
가장 많이 나온수=9
```

**HINT** 본문의 빈도수 구하는 예제를 참고한다. 0에서 9까지의 난수는 rand()%10으로 구할 수 있다.

```
int main(void)
{
    int freq[10] = { 0 };
    int i, max;

    for (i = 0; i < 1000; i++) {
        ++freq[rand() % 10]; // 빈도수를 구한다.
    }
    max = 0; // 0번이 제일 많이 나왔다고 가정
    printf("%d ", freq[0]);
    for (i = 1; i < 10; i++) {
        printf("%d ", freq[i]);
        if (freq[i] > freq[max])
            max = i;
    }
    printf("\n가장 많이 나온수=%d\n", max);
    return 0;
}
```

06 다음과 같은 2차원 표를 배열로 생성하고, 각 행의 합계, 각 열의 합계를 구하여 출력하는 프로그램을 작성하라.

12	56	32	16	98
99	56	34	41	3
65	3	87	78	21

```
C:\WINDOWS\system32\cmd.exe
0행의 합계: 214
1행의 합계: 233
2행의 합계: 254
0열의 합계: 176
1열의 합계: 115
2열의 합계: 153
3열의 합계: 135
4열의 합계: 122
```

**HINT** 2차원 배열을 주어진 표로 초기화한다. 각 행의 합계, 각 열의 합계를 중첩 반복문을 통하여 계산한다.

```
#include <stdio.h>

int main(void)
{
    int i, j, sum;
    int data[3][5] = { { 12, 56, 32, 16, 98 },
                       { 99, 56, 34, 41, 3 },
                       { 65, 3, 87, 78, 21 } };

    for (i = 0; i < 3; i++) {
        sum = 0;
        for (j = 0; j < 5; j++)
            sum += data[i][j];
        printf("%d행의 합계: %d\n", i, sum);
    }
    for (j = 0; j < 5; j++) {
        sum = 0;
        for (i = 0; i < 3; i++)
            sum += data[i][j];
        printf("%d열의 합계: %d\n", j, sum);
    }

    return 0;
}
```



07 1부터 10까지의 정수에 대하여 제곱값과 세제곱값을 계산하여 출력하는 프로그램을 작성하라. 10×3 크기의 2차원 배열을 만들고 첫 번째 열에는 정수를, 두 번째 열에는 제곱값을, 세 번째 열에는 세제곱값을 저장하라. 추가로 사용자에게 세제곱값을 입력하도록 하고 이 세제곱값을 배열에서 찾아서 그것의 세제곱근을 출력하도록 하여 보라.



**HINT** 배열을 순차탐색하여서 사용자가 입력한 세제곱값을 찾은 후에 첫 번째 열에 저장된 값을 출력하면 그것이 세제곱근이 된다. 세제곱근이 없는 경우도 처리하도록 하자.

```
int main(void)
{
    int list[3][10] = { 0 };
    int value, i;

    for (i = 1; i <= 10; i++) {
        list[0][i-1] = i;
        list[1][i-1] = i * i;
        list[2][i-1] = i * i * i;
    }

    printf("정수를 입력하시오:");
    scanf("%d", &value);

    for (i = 0; i < 10; i++) {
        if (value == list[2][i])
            printf("%d의 세제곱근은 %d\n", value, i+1);
    }

    return 0;
}
```



08 사용자가 입력하는 10개의 실수 자료의 평균과 표준 편차를 계산하는 프로그램을 작성하라. 자료들은 난수를 생성하여서 작성된다(정수로 생성하여서 실수로 변환하라). 평균은 n개의 실수가 주어져 있을 때, 다음과 같이 계산된다.

$$m = \frac{1}{n} \sum_{i=1}^n x_i$$

표준 편차는 분산의 양의 제곱근으로 분산은 다음과 같이 계산된다. 표준 편차는 자료가 평균값 주위에 어느 정도의 넓이로 분포하고 있는가를 나타내는 하나의 척도이다.

$$v = \frac{1}{n} \sum_{i=1}^n (x_i - m)^2$$

```
#include <stdio.h>
#define N_DATA 10
void get_data(double data[]);
double get_mean(double data[]);
double get_std_dev(double data[], double mean);

int main(void)
{
    double data[20];
    double mean;

    get_data(data);
    mean = get_mean(data);
    printf("평균값은 %f\n", mean);
    printf("표준편차값은 %f\n", get_std_dev(data, mean));
    return 0;
}

void get_data(double data[])
{
    int i;
    for (i = 0; i < N_DATA; i++) {
        printf("데이터를 입력하시오:");
        scanf("%lf", &data[i]);
    }
}
```

C:\WINDOWS\system32\cmd.exe

데이터를 입력하시오:10  
데이터를 입력하시오:20  
데이터를 입력하시오:30  
데이터를 입력하시오:40  
데이터를 입력하시오:50  
데이터를 입력하시오:60  
데이터를 입력하시오:70  
데이터를 입력하시오:80  
데이터를 입력하시오:90  
데이터를 입력하시오:100  
평균값은 55.000000  
표준편차값은 28.722813

HINT

사용자가 입력하는 자료 값들은 모두 배열에 저장한다. 평균과 표준편차를 구하는 함수를 작성하고 함수의 인수로 배열을 넘기도록 하자.

```
double get_mean(double data[])
{
    int i;
    double sum = 0.0;
    for (i = 0; i < N_DATA; i++) {
        sum += data[i];
    }
    return sum / N_DATA;
}

double get_std_dev(double data[], double mean)
{
    int i;
    double sum = 0.0;
    for (i = 0; i < N_DATA; i++) {
        sum += (data[i] - mean) * (data[i] - mean);
    }
    return sum / N_DATA;
}
```

- 09 학생들의 시험 점수를 통계 처리하는 프로그램을 작성하여 보라. 한 학급은 최대 10명까지의 학생들로 이루어진다. 각 학생들은 3번의 시험을 치른다. 학생들의 성적은 난수를 생성하여서 얻는다. 각 시험에 대하여 최대점수, 최저점수를 계산하여 출력한다.

학번	시험 #1	시험 #2	시험 #3
1	30	10	11
2	40	90	32
3	70	65	56
4	70	43	32
5	80	10	89

```
C:\WINDOWS\system32\cmd.exe
시험 #0의 최대점수=17
시험 #0의 최저점수=1
시험 #1의 최대점수=19
시험 #1의 최저점수=3
시험 #2의 최대점수=96
시험 #2의 최저점수=22
```

**HINT** 2차원 배열을 주어진 표로 초기화한다. 각 열의 최소값, 최대값을 반복문을 통하여 계산한다.

```
#include <stdio.h>

void get_stat_per_test(int score[][3])
{
    int i, k, min, max, sum;
    for (k = 0; k < 3; k++) {
        min = max = score[0][k];
        sum = 0;
        for (i = 0; i < 10; i++) {
            if (score[i][k] < min) min = score[i][k];
            if (score[i][k] > min) max = score[i][k];
            sum += score[i][k];
        }
        printf("시험 #%d의 최대점수=%d\n", k, max);
        printf("시험 #%d의 최저점수=%d\n", k, min);
    }
}

int main(void)
{
    int i, k;
    int score[10][3];
    for (i = 0; i < 10; i++) {
        for (k = 0; k < 3; k++) {
            score[i][k] = rand() % 100 + 1;
        }
    }
    get_stat_per_test(score);
    return 0;
}
```

10 벡터에 대한 연산을 배열을 이용하여서 작성하여 보자.

(a) 2개의 벡터를 더하는 연산은 다음과 같이 정의된다. 2개의 벡터를 더하는 함수인 `vector_add()`를 작성하라. 이 함수를 테스트하기 위한 코드도 작성하라.

$$\vec{X} + \vec{Y} = (x_1 + y_1, x_2 + y_2, x_3 + y_3)$$

```
C:\WINDOWS\system32\cmd.exe
벡터의 합 = [ 2.000000 4.000000 6.000000]
```

**HINT** 하나의 벡터를 크기가 3인 배열로 나타내자. `vector_add(double x[], double y[], double z[])`는 2개의 배열을 받아서 반복 구조를 이용하여 벡터의 합을 계산한 후에 매개 변수 `z`를 통하여 벡터의 합을 반환한다.

```
#include <stdio.h>
void vector_add(double x[], double y[], double z[]);
int main(void)
{
    double x[3] = { 1.0, 2.0, 3.0 };
    double y[3] = { 1.0, 2.0, 3.0 };
    double z[3];

    vector_add(x, y, z);
    printf("벡터의 합 = [ %f %f %f] \n", z[0], z[1], z[2]);
    return 0;
}
void vector_add(double x[], double y[], double z[])
{
    z[0] = x[0] + y[0];
    z[1] = x[1] + y[1];
    z[2] = x[2] + y[2];
}
```

(b) 벡터의 내적(dot product)를 계산하는 함수인 `vector_dot_prod()`를 작성하라. 이 함수를 테스트하기 위한 코드도 작성하라. 벡터의 내적은 다음과 같이 정의된다.

$$\vec{X} \cdot \vec{Y} = (x_1 y_1 + x_2 y_2 + x_3 y_3)$$

```
C:\WINDOWS\system32\cmd.exe
벡터의 내적=14.000000
```

**HINT** 벡터의 내적을 구하는 것이 더 쉽다. 하나의 벡터를 크기가 3인 배열로 나타내고 `double vector_dot_prod(double x[], double y[])`는 2개의 배열을 받아서 반복 구조를 이용하여 내적은 계산한 후에 스칼라 값을 반환한다.

```
#include <stdio.h>
double vector_dotp(double x[], double y[]);
int main(void)
{
    double x[3] = { 1.0, 2.0, 3.0 };
    double y[3] = { 1.0, 2.0, 3.0 };
    double value;

    value = vector_dotp(x, y);
    printf("%f\n", value);
    return 0;
}
double vector_dotp(double x[], double y[])
{
    int i;
    double result;

    result = 0.0;
    for (i = 0; i < 3; i++)
        result += x[i] * y[i];
    return result;
}
```

- 11 아주 간단한 재고 관리 시스템을 만들어보자. 상품마다 상품 번호가 붙어 있고 상품 번호를 사용자가 입력하면 물품이 어디 있는지를 알려주는 번호를 출력한다. 상품 번호는 1부터 10까지로 하고 장소를 나타내는 번호는 1부터 5까지라고 가정한다. 1차원 배열을 사용하여 미리 상품 번호마다 장소를 저장해놓고 사용자로부터 상품 번호를 받아서 장소를 출력한다.



```
C:\WINDOWS\system32\cmd.exe
상품 번호를 입력하시요:1
상품 번호 1의 위치는 1입니다.
```

**HINT** 본문에 있는 영화관 예약 시스템을 참조하여서 작성한다.

```
#include <stdio.h>
int location[] = { 1, 1, 2, 5, 3, 3, 1, 1, 2, 4 };
int main(void)
{
    int product;
    printf("상품 번호를 입력하시요:");
    scanf("%d", &product);
    if (product <= 1 || product > 10)
        printf("잘못된 상품 번호입니다.\n");
    else
        printf("상품 번호 %d의 위치는 %d입니다.\n", product, location[product-1]);
    return 0;
}
```

12 2차원 행렬(matrix)에 대한 다음과 같은 함수를 작성하고 테스트하여 보라. 행렬의 크기는 3×3으로 가정하라.

(a) scalar\_mult(int a[][3], int scalar)

$$2 \cdot \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$



```
#include <stdio.h>

void smult(int A[][3], int b, int C[][3])
{
    int r, c, k;
    for (r = 0; r < 3; r++) {
        for (c = 0; c < 3; c++) {
            C[r][c] = b * A[r][c];
        }
    }
}

void print(int A[][3])
{
    int r, c;
    for (r = 0; r < 3; r++) {
        for (c = 0; c < 3; c++)
            printf("%d ", A[r][c]);
        printf("\n");
    }
}

int main(void)
{
    int A[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
    int C[3][3];

    smult(A, 2, C);
    print(C);
    return 0;
}
```

(b) transpose((int a[][3], int b[][3])

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$



**HINT** 하나의 행렬을 2차원 배열로 나타내자. void transpose(int a[][3], int b[][3])로 선언하여서 a[][]를 전치하여서 b[][]로 반환한다. 배열은 원본이 함수로 전달되는 것에 유의하자.

```
#include <stdio.h>

void transpose(int a[][3], int b[][3])
{
    int r, c;
    for (r = 0; r < 3; r++)
        for (c = 0; c < 3; c++)
            b[r][c] = a[c][r];
}

void print(int a[][3])
{
    int r, c;
    for (r = 0; r < 3; r++) {
        for (c = 0; c < 3; c++)
            printf("%d ", a[r][c]);
        printf("\n");
    }
}

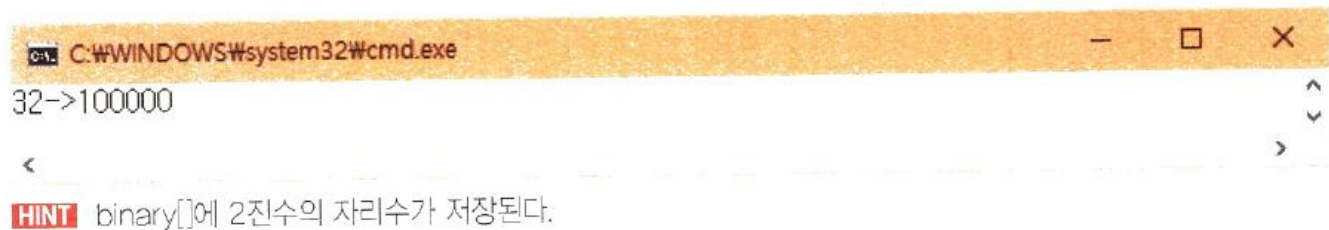
int main(void)
{
    int A[3][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
    int B[3][3];

    transpose(A, B);
    print(B);
    return 0;
}
```



- 13 10진수를 2진수로 변환하여 출력하는 프로그램을 작성하여 보자. 최대 32자리까지 변환이 가능하도록 하라. 변환된 자리수를 저장하는데 배열을 사용하라. 10진수를 2로 나누어서 생성된 나머지를 역순으로 나타내면 2진수로 표현할 수 있다.

```
for(i = 0; i < 32 && n > 0; i++)
{
    binary[i] = n % 2;
    n = n / 2;
}
```



```
C:\WINDOWS\system32\cmd.exe
32->100000
```

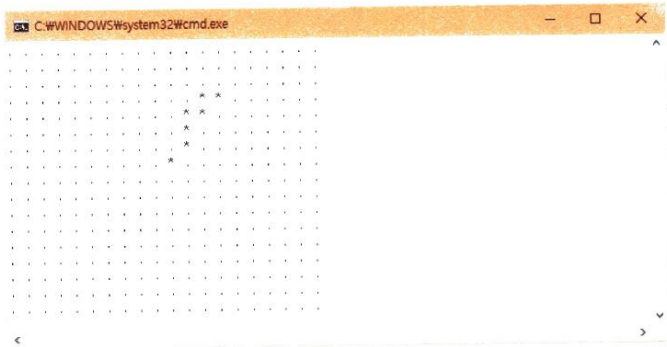
**HINT** binary[]에 2진수의 자리수가 저장된다.

```
#include <stdio.h>

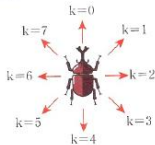
void show_digit(int n)
{
    int i;
    int binary[32] = { 0 };
    for (i = 0; i < 32 && n > 0; i++)
    {
        binary[i] = n % 2;
        n = n / 2;
    }
    for (i--; i >= 0; i--)
    {
        printf("%d", binary[i]);
    }
    printf("\n");
}

int main(void)
{
    show_digit(32);
    return 0;
}
```

14 수학에서의 "random walk"라 불리는 문제를 배열을 이용하여 프로그래밍하여 보자. 문제는 다음과 같다. 술에 취한 딱정벌레가  $n \times m$ 크기의 타일로 구성된 방안에 있다. 딱정벌레는 임의의(랜덤) 위치를 선택하여 여기저기 걸어 다닌다. 현재의 위치에서 주의의 8개의 타일로 걸어가는데 확률은 동일하다고 가정하자. 딱정벌레가 이동하는 경로를 다음과 같이 표시하라.



**TIP** 방 전체를 2차원 배열 tile[n][m]로 모델링을 하고 처음에는 딱정벌레가 배열의 중앙에 있다고 가정하라. tile[i][j]의 초기값은 0이다. 딱정벌레가 타일을 지나갈 때마다 2차원 배열의 값을 1로 만들어서 딱정벌레가 지나갔음을 나타낸다. 0부터 7까지의 랜덤한 숫자를 생성하여 다음과 같이 움직인다. 즉 0이면 북쪽으로 이동하고 4이면 남쪽으로 이동한다.



number = rand() % 8;

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

#define ROWS 20
#define COLS 20

int grid[ROWS][COLS];
int ro, co;

void mark_grid(int y, int x)
{
    if (x >= 0 && y >= 0 && y < ROWS && x < COLS) {
        ro = y;
        co = x;
        if (grid[y][x] == '.')
            grid[y][x] = '*';
    }
}

void print_grid()
{
    int r, c;
    system("cls");
    for (r = 0; r < ROWS; r++) {
        for (c = 0; c < COLS; c++) {
            printf("%c ", grid[r][c]);
        }
        printf("\n");
    }
}
```

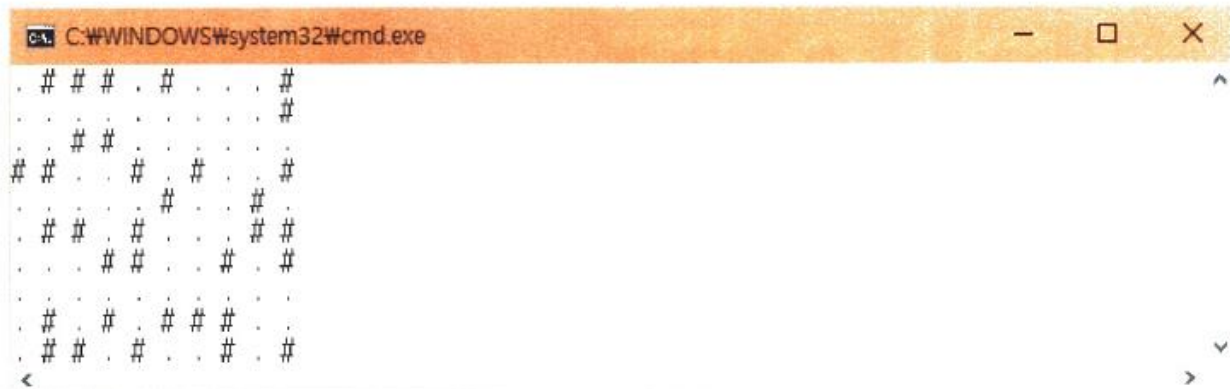
```
int main() {
    int r, c;
    int i;
    char letters = '*';
    for (r = 0; r < ROWS; r++) {
        for (c = 0; c < COLS; c++) {
            grid[r][c] = '.';
        }
    }
    srand((unsigned)time(NULL));
    ro = ROWS / 2;
    co = COLS / 2;
    grid[ro][co] = letters;

    for (i = 0; i < 100000; i++) {
        int move = rand() % 8;
        switch (move) {
            case 0: mark_grid(ro + 1, co); break;
            case 1: mark_grid(ro + 1, co + 1); break;
            case 2: mark_grid(ro, co + 1); break;
            case 3: mark_grid(ro - 1, co + 1); break;
            case 4: mark_grid(ro - 1, co); break;
            case 5: mark_grid(ro - 1, co - 1); break;
            case 6: mark_grid(ro, co - 1); break;
            case 7: mark_grid(ro + 1, co - 1); break;
        }
        print_grid();
        if (getch() == 'q') break;
    }

    return 0;
}
```



- 15 지뢰찾기는 예전에 윈도우에 무조건 포함되어 있어서 상당히 많은 사람들이 즐겼던 프로그램이다. 2차원의 게임판 안에 지뢰가 숨겨져 있고 이 지뢰를 모두 찾아내는 게임이다. 지뢰가 없는 곳을 클릭했을 때 숫자가 나오면 주변칸에 지뢰가 숨겨져 있다는 것을 의미한다. 예를 들어서 숫자가 2이면 주변칸에 지뢰가 두개 있다는 의미가 된다. 이 문제에서는 지뢰찾기 게임을 위한 기초 작업을 하여 보자. 10×10 크기의 2차원 배열을 만들고 여기에 지뢰를 숨긴다. 지뢰가 아닌 곳은 .으로 표시하고 지뢰인 곳은 #로 표시하여 보자.



```
C:\WINDOWS\system32\cmd.exe
. # # . # . . . #
. . # . . . . . #
. . # . . . . . #
# # . . # # . . #
. # # . # . . . #
. . # # . . # #
. # # . # # # .
. # # . # . . #
. # # . # . . #
. # # . # . . #
```

**HINT** 어떤 칸이 지뢰일 확률은 난수를 발생시켜서 결정한다. 전체의 30%를 지뢰로 하고 싶으면 0부터 99 사이의 난수를 생성하여서 30보다 적은 경우에만 지뢰를 놓으면 된다.

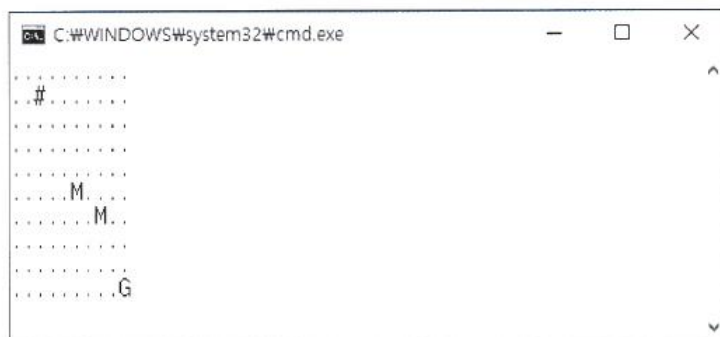
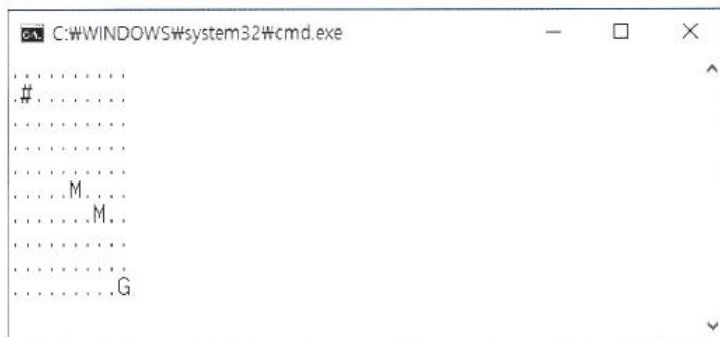
```
#include <stdio.h>
#include <time.h>

int main()
{
    int board[10][10] = { 0 };
    srand(time(NULL));

    for (int i = 0; i < 10; i++)
        for (int j = 0; j < 10; j++)
            if ((rand() % 100) < 30)
                board[i][j] = 1;

    for (int i = 0; i < 10; i++) {
        for (int j = 0; j < 10; j++)
            if (board[i][j])
                printf("# ");
            else
                printf(". ");
        printf("\n");
    }
    return 0;
}
```

- 16 간단한 텍스트 기반의 게임을 작성해보자. 주인공은 '#'로 표시되어 있다. 주인공이 금 'G'를 찾으면 게임이 종료된다. 중간에 몬스터 'M'가 있어서 금을 찾는 것을 방해한다. 주인공은 'w', 's', 'a', 'd' 키를 이용하여 상하좌우로 움직일 수 있다.



**HINT** 사용자로부터 실시간으로 키를 받으려면 `_getch()` 함수를 사용해야 한다. 몬스터들은 난수만큼 자동으로 이동하도록 하라. 최대한 게임을 재미있게 만들어보자. 화면을 지우려면 `system("cls")`을 실행한다.

```
char board[10][10];
int xpos = 1, ypos = 1;
int k, i;
int ch;
```

// 보드를 초기화한다.

```
for (int y = 0; y < 10; y++)
    for (int x = 0; x < 10; x++) board[y][x] = '.';
board[ypos][xpos] = '#';
board[9][9] = 'G';
board[5][5] = 'M';
board[6][7] = 'M';
```

```
printf("왼쪽이동:a, 오른쪽 이동:d 위쪽 이동:w, 아래쪽 이동:s\n");
// 사용자로부터 위치를 받아서 보드에 표시한다.
```

```
while (1) {
    int x, y, z;
    system("cls");
    for (int y = 0; y < 10; y++) {
        for (int x = 0; x < 10; x++) printf("%c", board[y][x]);
        printf("\n");
    }
    board[ypos][xpos] = '.';
    ch = _getch();
    if (ch == 'a') xpos--;
    else if (ch == 's') ypos++;
    else if (ch == 'w') ypos--;
    else if (ch == 'd') xpos++;
    if (board[ypos][xpos] == 'G') {
        printf("\nWin!!\n");
        break;
    }
    board[ypos][xpos] = '#';
}
return 0;
```