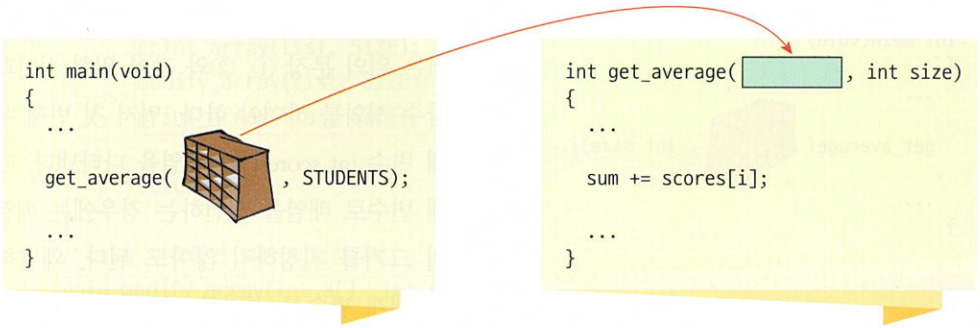


배열도 함수로 전달할 수 있다. 하지만 변수를 전달할 때와는 다르다. 우리는 함수 호출 시에 "값에 의한 호출"이 이루어진다는 것을 알고 있다. "에 의한 호출"이란 함수를 호출할 때 인수의 값이 매개 변수로 복사되는 것을 의미한다. 그러나 배열의 경우에는 단순한 "값에 의한 호출"이 아니다. 배열이 인수인 경우에는 배열의 원본이 매개 변수를 통하여 전달된다. 여기에 대한 설명은 포인터를 학습해야만 완전해진다. 지금으로는 그냥 배열의 경우에는 원본이 전달된다고만 알아두자.

실습 1

간단한 예를 들어 보자. 학생들의 성적을 저장하고 있는 정수 배열을 만들고 평균을 계산하는 함수를 작성하여 호출해보자.

```c
#include <stdio.h>
#define STUDENTS 5

int get_average(int scores[], int size);      // ①

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

        avg = get_average(scores, STUDENTS);
        printf("평균은 %d입니다.\n", avg);

        return 0;
}
// 배열에 들어 있는 값들의 평균을 계산하는 함수
int get_average(int scores[], int size)       // ②
{
        int i;
        int sum = 0;

        for(i = 0; i < size; i++)
           sum += scores[i];

        return sum / size;
}
```

원형을 정의한다. 배열을 받는 매개 변수는 크기를 적어주지 않아도 된다.

원본 배열이 전달된다.

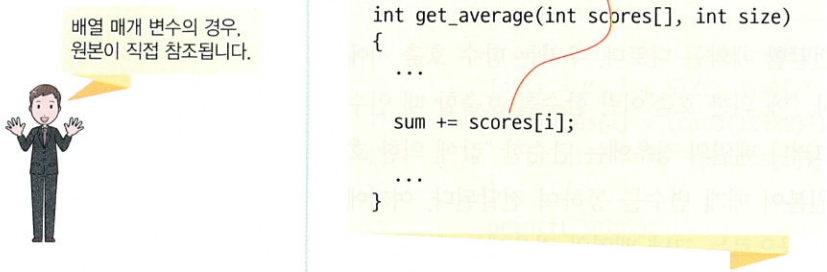
반복하면서 배열의 모든 요소들의 합을 계산한다.

실행결과

```
평균은 3입니다.
```

실습1 설명

그림 10-10
배열 매개 변수



먼저 함수가 배열을 매개 변수로 받기 위해서는 위의 문장 ①, ②와 같이 원형 선언과 함수 정의를 하여야 한다. 먼저 첫 번째 매개 변수 int scores[]가 배열을 나타낸다. 매개 변수로 배열을 선언하는 경우에는 배열의 크기를 지정하지 않아도 된다. 왜냐하면 매개 변수를 선언할 때, 실제로 배열이 생성되는 것이 아니기 때문이다. 우리는 매개 변수 scores를 통하여 원본 배열 scores를 참조한다. 따라서 정확한 크기는 필요하지 않다. 물론 크기를 지정하여도 문제는 없다. get_average()가 호출되면, 인수로 전달되는 scores 배열과 매개 변수 scores 배열은 같아진다.

두 번째 매개 변수 int size는 배열의 크기를 받는 매개 변수이다. 호출된 함수에서는 scores가 배열이라는 것만 알 수 있고 배열의 크기는 모른다. 배열의 크기 정보가 있어야 만이 배열의 올바르게 데이터를 처리할 수 있다. 따라서 일반적으로 배열의 크기를 별도의 매개 변수로 전달받는다. 물론 배열의 크기가 항상 일정하다면 그럴 필요가 없을 것이다.

원본 배열의 변경

배열은 매개 변수를 통하여 원본을 참조하는 것이기 때문에 항상 조심을 하여야 한다. 만약 함수 안에서 매개 변수를 통하여 배열 요소를 변경한다면 이것은 원본 배열을 변경시키는 결과를 가져온다. 하나의 예로 1차원 배열을 조작하는 다음과 같은 함수를 작성하고 사용하여 보자.

실습 2

modify.c

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

void modify_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 } ;

```

```
11        print_array(list, SIZE);
12        modify_array(list, SIZE);    // 배열은 원본이 전달된다.
13        print_array(list, SIZE);
14
15        return 0;
16  }
17  // a[]를 변경하면 원본이 변경된다.
18  void modify_array(int a[], int size)
19  {
20        int i;
21
22        for(i = 0; i < size; i++)
23           ++a[i];
24  }
25  // 배열 요소들을 화면에 출력한다.
26  void print_array(int a[], int size)
27  {
28        int i;
29
30        for(i = 0; i < size; i++)
31           printf("%3d ", a[i]);
32        printf("\n");
33  }
```

배열을 함수로 전달한다.

modify_array()는 배열을 인수로 받아서 배열 내의 모든 요소를 하나 증가시킨다. 배열은 원본이 전달되므로, 호출된 함수가 배열의 요소를 수정하면 원본 배열의 내용도 동시에 수정된다.

실행결과

```
  1   2   3   4   5   6   7
  1   4   9  16  25  36  49
```

modify_array()는 배열 a[]를 받아서 배열 요소를 하나 증가하고 있다. 배열은 원본이 전달되므로, 호출된 함수가 배열의 요소를 수정하면 원본 배열의 내용도 동시에 수정된다.

오류 주의

배열 요소를 인수로 하여서 함수를 호출하면 복사본이 전달된다. 배열은 원본이 전달되지만 배열 요소는 복사본이 전달되므로 착각하면 안 된다.

```
int main(void)
{
   square_element(list[2]);
}
void square_element(int e)
{
   e = e * e;
}
```

복사본 전달

예제 #2

실습 3

행렬(matrix)은 자연과학에서 많은 문제를 해결하는데 사용된다.

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

행렬을 어떻게 표현할 것인지를 생각해보자. 일반적으로 행렬을 표현하는 자연스러운 방법은 2차원 배열을 사용하는 것이다. 이 예제에서는 두개의 행렬을 더하는 프로그램을 작성하여 보자.

matrix.c

```
1  #include <stdio.h>
2  #define ROWS 3
3  #define COLS 3
4
5  int main(void)
6  {
7        int A[ROWS][COLS] = { { 2,3,0 },
8                              { 8,9,1 },
9                              { 7,0,5 } };
10       int B[ROWS][COLS] = { { 1,0,0 },
11                             { 1,0,0 },
12                             { 1,0,0 } };
13       int C[ROWS][COLS];
14       int r,c;
15
16       // 두개의 행렬을 더한다.
17       for(r = 0;r < ROWS; r++)
18          for(c = 0;c < COLS; c++)
19             C[r][c] = A[r][c] + B[r][c];
20
21       // 행렬을 출력한다.
22       for(r = 0;r < ROWS; r++)
23       {
24          for(c = 0;c < COLS; c++)
25             printf("%d ", C[r][c]);
26          printf("\n");
27       }
28
29       return 0;
30 }
```

크기가 ROWS × COLS인 2차원 정수 배열 A[][], B[][], C[][]를 선언하였다. A[][]와 B[][]는 초기값이 주어져 있다.

행렬 요소들의 인덱스로 사용될 변수 r과 c를 선언한다. 행렬에서는 보통 행(row), 열(column)이라는 용어를 사용한다. r과 c는 이것들의 약자이다.

중첩 for 루프를 이용하여 행렬 A의 각 요소들과 행렬의 B의 각 요소들을 서로 더하여 행렬 C에 대입한다.

중첩 for 루프를 이용하여 행렬 C의 각 요소들의 값을 화면에 출력한다. 26번째 라인은 내부 for 루프가 끝난 후에 화면에 줄바꿈 문자를 출력하기 위한 것이다.

실행결과

```
3 3 0
9 9 1
8 0 5
```

2차원 배열을 함수로 전달하기

2차원 배열도 1차원 배열과 마찬가지로 함수에 인수로 전달할 수 있다. 2차원 배열도 원본이 전달된다. 함수를 사용하여 2차원 배열의 요소들의 합을 구하는 예제를 살펴보자.

중간점검

1 다차원 배열 int a[3][2][10]에는 몇개의 요소가 존재하는가?

2 다차원 배열 int a[3][2][10]의 모든 요소를 0으로 초기화하는 문장을 작성하시오.

참고사항

C에서는 2차원 배열, 3차원 배열 등 일반적으로 n차원의 배열이 가능하다. 사실 C에서 가질 수 있는 차원의 수에는 아무런 제한이 없다.

```
int s[3][3][5]; // 3차원 배열
```

그러나 다차원이 되면 필요한 메모리의 양이 급격하게 늘어나게 되므로 주의하여야 한다. 보통 특별한 경우를 제외하고는 3차원 이상의 다차원 배열은 피하는 것이 좋다. 예를 들어서 100개의 정수를 저장할 수 있는 1차원 배열은 하나의 정수가 4바이트이므로 400바이트면 되지만 의 2차원 배열은 40000바이트를 필요로 하고 의 3차원 배열은 4000000바이트를 필요로 한다. 따라서 필요이상으로 차원을 늘리지 않도록 주의하여야 한다.

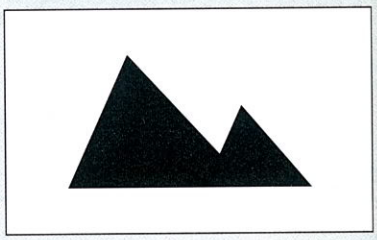
LAB

실습 4

영상 처리

영상 처리(image processing)이란 카메라를 통해서 입력받은 디지탈 영상을 컴퓨터를 이용해서 처리하는 분야이다. 최근 화두가 되고 있는 자율 주행 자동차에서도 영상 처리를 사용하여서 차선을 감지한다. 디지털 영상은 픽셀들의 2차원 배열이라 할 수 있다. 우리는 8×16 크기의 흑백 영상만을 생각하자. 각 픽셀은 검정색(0)이거나 흰색(1)이 될 수 있다.

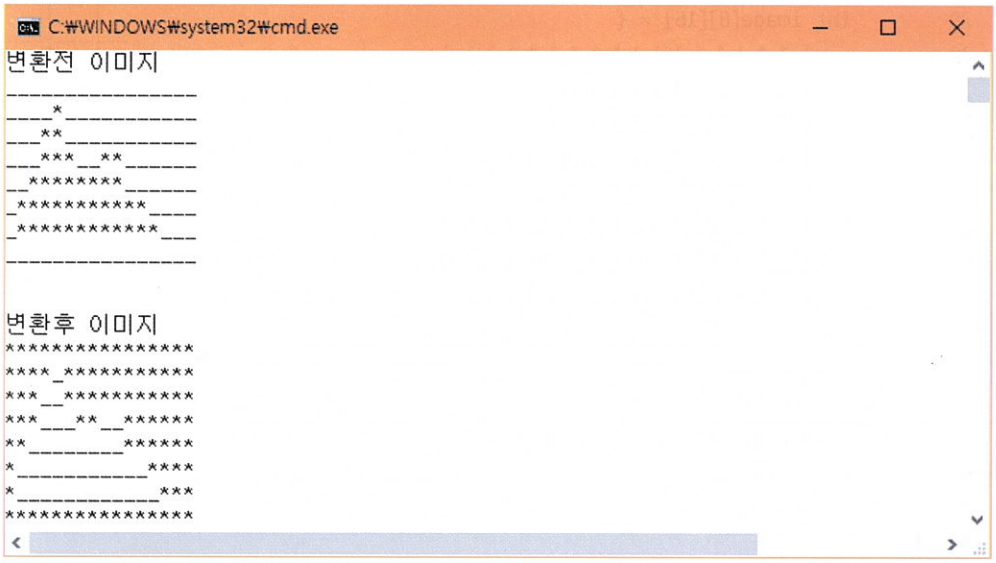
```
int image[8][16] = {
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },
        { 1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1 },
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
```



이 실습에서는 2차원 배열에 흑백 영상을 저장하고 이 배열을 받아서 영상의 색상을 반전하는 함수 inverse()를 작성해보자. inverse() 함수에서는 흰색을 검정색으로 바꾸고, 검정색은 흰색으로 변경한다.

```
void inverse(img[8][16]){
        ...
}
```

실행결과



solution

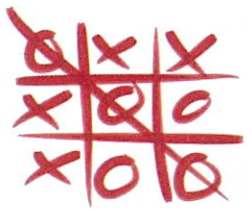
image_proc.c

```
#include <stdio.h>

void display(int image[8][16])
{
      for (int r = 0; r < 8; r++) {
         for (int c = 0; c < 16; c++) {
            if (image[r][c] == 0)
               printf("*");
            else
               printf("_");
         }
         printf("\n");
      }
}
void inverse(int img[8][16])
{
      for (int r = 0; r < 8; r++) {
         for (int c = 0; c < 16; c++) {
            if (img[r][c] == 0)
               img[r][c] = 1;
            else
               img[r][c] = 0;
         }
      }
```

```
25 }
26
27 int main(void)
28 {
29        int image[8][16] = {
30           { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
31           { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },
32           { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },
33           { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },
34           { 1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1 },
35           { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },
36           { 1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1 },
37           { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
38        printf("변환전 이미지\n");
39        display(image);
40        inverse(image);
41        printf("\n\n변환후 이미지\n");
42        display(image);
43        return 0;
44 }
```

Mini Project TIC-TAC_TOE 게임



tic-tac-toe 게임을 구현하여 보자. tic-tac-toe 게임은 2명의 경기자가 오른쪽과 같은 보드를 이용하여서 번갈아가며 O와 X를 놓는 게임이다. 같은 글자가 가로, 세로, 혹은 대각선 상에 놓이면 이기게 된다.

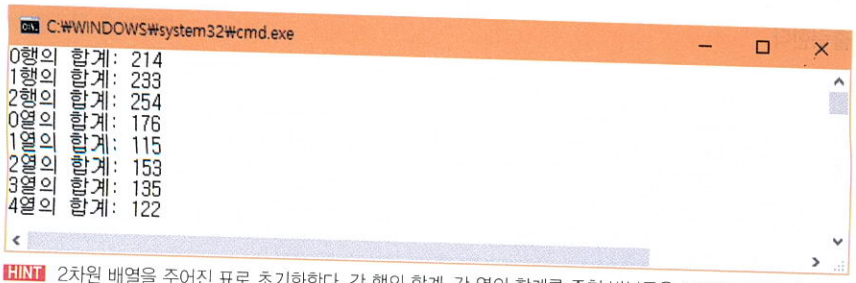
물론 최근의 "리그 오브 레전드"와 같은 게임과 비교하면 아주 고전적인 게임이지만 한번 구현해보기로 하자. 이 게임은 사람과 사람이 대결하는 게임이다. 하지만 컴퓨터와 사람이 대결하는 프로그램도 "도전 문제"로 시도하여 보자. 한 경기자씩 보드의 좌표를 입력한다.

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

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

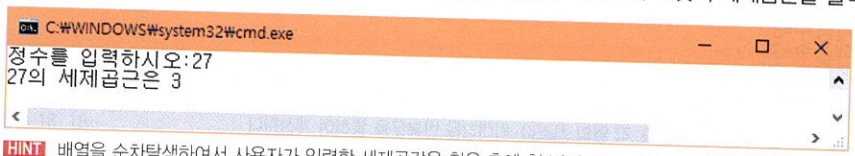
06 다음과 같은 2차원 표를 배열로 생성하고, 각 행의 합계, 각 열의 합계를 구하여 출력하는 프로그램을 작성하라.

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



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

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.

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.

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

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

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

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.

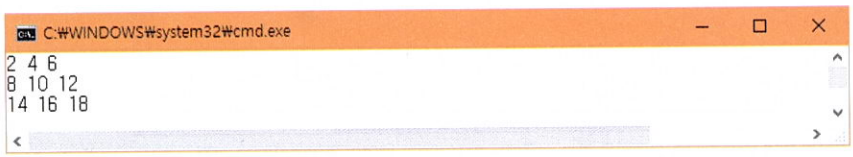
11 아주 간단한 재고 관리 시스템을 만들어보자. 상품마다 상품 번호가 붙어 있고 상품 번호를 사용자가 입력하면 물품이 어디 있는지를 알려주는 번호를 출력한다. 상품 번호는 1부터 10까지로 하고 장소를 나타내는 번호는 1부터 5까지라고 가정한다. 1차원 배열을 사용하여 미리 상품 번호마다 장소를 저장해놓고 사용자로부터 상품 번호를 받아서 장소를 출력한다.

HINT 본문에 있는 영화관 예약 시스템을 참조하여서 작성한다.

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)

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



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

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

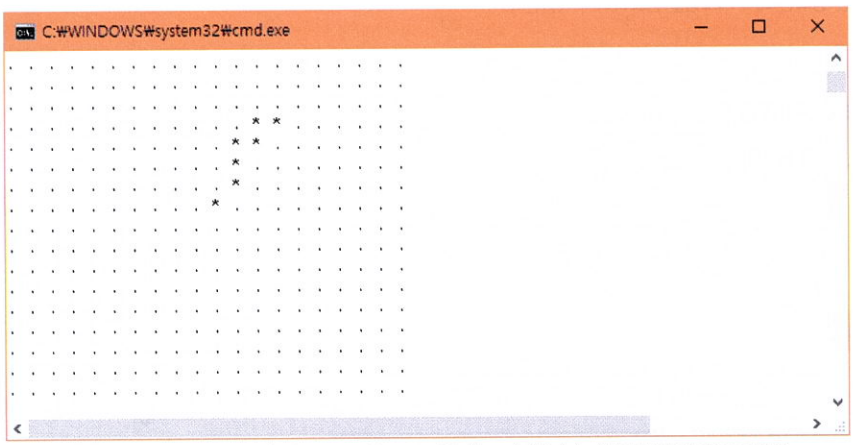
HINT 하나의 행렬을 2차원 배열로 나타내자. void transpose(int a[][3], int b[][3])로 선언하여서 a[][]를 전치하여서 b[][]로 반환한다. 배열은 원본이 함수로 전달되는 것에 유의하자.

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.

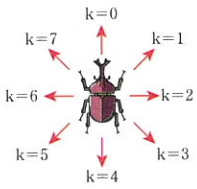
```
for(i = 0; i < 32 && n > 0; i++)
{
        binary[i] = n % 2;
        n = n / 2;
}
```

HINT binary[]에 2진수의 자리수가 저장된다.

14 수학에서의 "random walk"라 불리우는 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 걸어가는 확률은 동일하다고 가정하자. 딱정벌레가 이동하는 경로를 다음과 같이 표시하라.



HINT 방 전체를 2차원 배열 tile[n][m]로 모델링을 하고 처음에는 딱정벌레가 배열의 중앙에 있다고 가정하라. tile[][]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.



0부터 7까지의 랜덤한 숫자는 다음과 같이 생성할 수 있다. 즉 rand() 함수의 반환값을 8로 나누어 나머지를 취한다.

```
number = rand() % 8;
```