

# 1 | 2차원 배열을 이용한 선형리스트 구현

## 1 2차원 배열을 이용한 선형 리스트 구현

### ▶ 2차원 배열을 이용한 구현

- 분기와 연도를 모두 표현해야 하므로 순서가 두 종류가 필요
- 따라서 행 인덱스와 열 인덱스가 있는 2차원 배열을 사용
- 2차원 배열구조를 논리적으로 표현할 때는 행과 열의 구조로 나타내지만 실제로 메모리에 저장될 때는 1차원 구조로 저장

연도 \ 분기	1/4분기	2/4분기	3/4분기	4/4분기
2016년	63	84	140	130
2017년	157	209	251	312

[2016~2017년 분기별 노트북 판매량 리스트]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

# 1 | 2차원 배열을 이용한 선형리스트 구현

## 1 2차원 배열을 이용한 선형 리스트 구현

[2016~2017년 분기별 노트북 판매량 리스트 예]

```
int sale[2][4] = { { 63, 84, 140, 130 },
                    { 157, 209, 251, 312 } };
```

		[0]	[1]	[2]	[3]
sale	[0]	63	84	140	130
	[1]	157	209	251	312

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

## 2 2차원 배열의 물리적 저장 방법

2차원의 논리적 순서를  
1차원의 물리적 순서로 변환하는 방법 사용

### 행 우선 순서 방법(Row Major Order)

- ▶ 2차원 배열의 첫 번째 인덱스인 행 번호를 기준으로 사용하는 방법
  - $\text{sale}[0][0]=63, \text{sale}[0][1]=84, \text{sale}[0][2]=140,$   
 $\text{sale}[0][3]=130, \text{sale}[1][0]=157,$   
 $\text{sale}[1][1]=209, \text{sale}[1][2]=251,$   
 $\text{sale}[1][3]=312$

## 2 2차원 배열의 물리적 저장 방법

### 열 우선 순서 방법(Column Major Order)

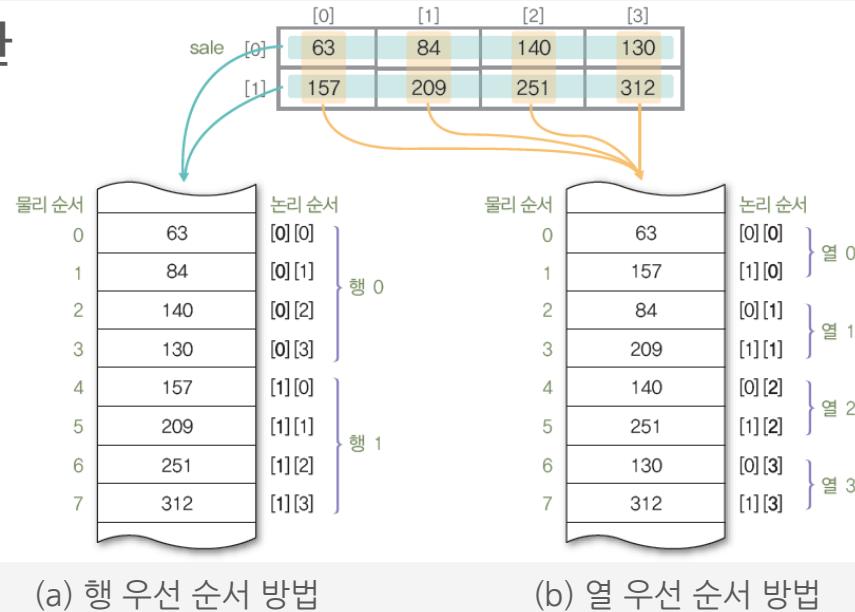
- ▶ 2차원 배열의 마지막 인덱스인 열 번호를 기준으로 사용하는 방법
  - $\text{sale}[0][0]=63, \text{sale}[1][0]=157, \text{sale}[0][1]=84,$   
 $\text{sale}[1][1]=209, \text{sale}[0][2]=140,$   
 $\text{sale}[1][2]=251, \text{sale}[0][3]=130,$   
 $\text{sale}[1][3]=312$

## 2 2차원 배열의 물리적 저장 방법

- ▶ 행의 개수가  $n_i$ 이고 열의 개수가  $n_j$ 인  
2차원 배열  $A[n_i][n_j]$ 의 시작주소가  $\alpha$ 이고,  
원소의 길이((메모리 크기))가  $\ell$  이라면,  $i$ 행  $j$ 열 원소  
즉,  $A[i][j]$ 의 원소 위치 계산
  - 행우선 :  $\alpha + (i \times n_j + j) \times \ell$
  - 열우선 :  $\alpha + (j \times n_i + i) \times \ell$

## 3 2차원 논리 순서를 1차원 물리 순서로 변환

### ▶ 변환

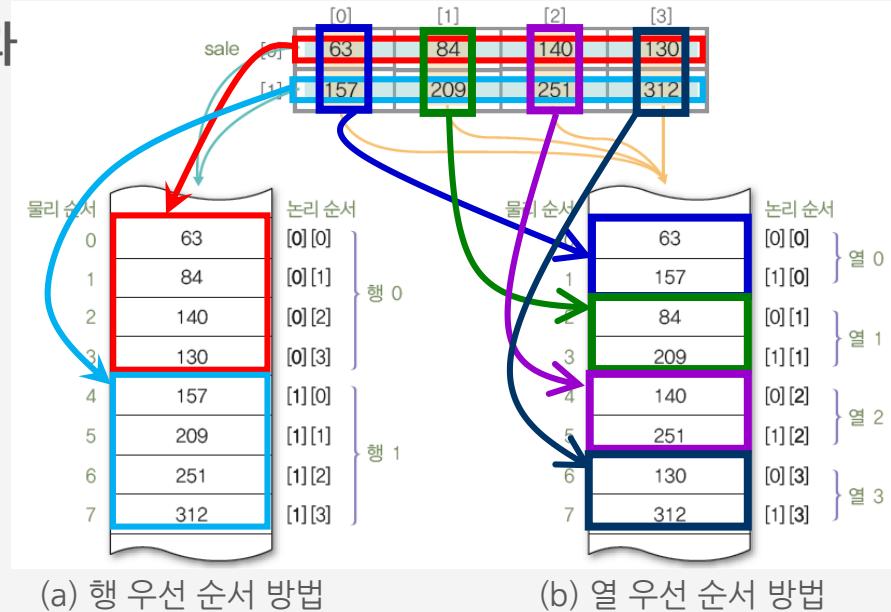


※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

# 1 | 2차원 배열을 이용한 선형리스트 구현

## 3 2차원 논리 순서를 1차원 물리 순서로 변환

### ▶ 결과



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

## 3 2차원 논리 순서를 1차원 물리 순서로 변환

### ▶ 예제

- 행우선 :  $\alpha + (i \times n_i + j) \times \ell$  (배열 크기 행  $n_i \times$  열  $n_j$ ),  
 $\alpha$ 는 시작주소
- 열우선 :  $\alpha + (j \times n_i + i) \times \ell$  (원소 위치 행  $i$ , 열  $j$ ),  
 $\ell$ 는 원소의 메모리 크기

» Self Test \* 배열 선언 : int a[4][10] ;

C언어 프로그램에서 다음과 같이 배열 a를 선언하였다. 배열 a가 할당된 시작주소는 1000이라고 가정 했을 때 ① a[2][8]:행 우선, ② a[2][8]:열 우선이 몇 번째 원소인지 구하세요.

- a[2][8]:행우선
- a[2][8]:열우선

## 4 2차원 배열의 논리적·물리적 순서 확인하기 프로그램

예제 3-2 2차원 배열의 논리적·물리적 순서 확인하기

```
01 #include <stdio.h>
02
03 void main() {
04     int i, n = 0, *ptr;
05     int sale[2][4] = {{63, 84, 140, 130},
06                         {157, 209, 251, 312}}; // 2차원 배열의 초기화
07
08     ptr = &sale[0][0];
09     for (i = 0; i<8; i++) {
10         printf("\n address : %u sale %d = %d", ptr, i, *ptr);
11         ptr++;
12     }
13     getchar();
14 }
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

## 4 실행 결과

```
address: 1374812 sale 0 = 63
address: 1374816 sale 1 = 84
address: 1374820 sale 2 = 140
address: 1374824 sale 3 = 130
address: 1374828 sale 4 = 157
address: 1374832 sale 5 = 209
address: 1374836 sale 6 = 251
address: 1374840 sale 7 = 312
```

### ▶ 실행 결과 확인

시작 주소  $\alpha=13474812$ ,  $ni=2$ ,  $nj=4$ ,  $i=1$ ,  $j=1$ ,  $\ell=4$

$sale[1][2]=251$ 의 위치 =  $\alpha + (i \times nj + j) \times \ell$

$$\begin{aligned} &= 13474812 + (1 \times 4 + 2) \times 4 \\ &= 13474812 + 24 \\ &= 13474836 \end{aligned}$$

- C 컴파일러가 행 우선 순서 방법으로 2차원 배열을 저장함을 확인!

## 2 | 3차원 배열을 이용한 선형리스트 구현

### 1 3차원 배열을 이용한 선형 리스트 구현

- ▶ 3차원 배열을 이용한 구현
  - 연도와 분기 그리고 팀 순서도 나타내야 하므로 세 종류 순서를 표현해야 함
  - 이럴 때는 면 인덱스, 행 인덱스, 열 인덱스가 있는 3차원 배열을 사용

팀	분기 연도	1/4분기	2/4분기	3/4분기	4/4분기
1팀	2016년	63	84	140	130
	2017년	157	209	251	312
2팀	2016년	59	80	130	135
	2017년	149	187	239	310

[1팀과 2팀의 분기별 노트북 판매량 리스트]

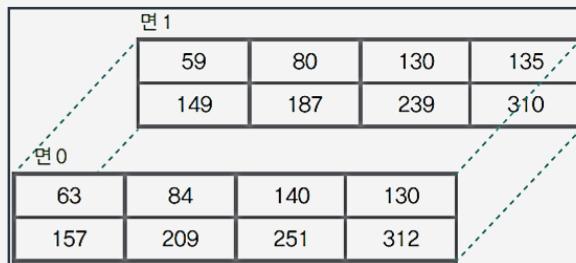
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

## 2 | 3차원 배열을 이용한 선형리스트 구현

### 1 3차원 배열을 이용한 선형 리스트 구현

[선형 리스트의 3차원 배열 예]

```
int sale[2][2][4] = { { { 63, 84, 140, 130 },
                      { 157, 209, 251, 312 } },
                      { { 59, 80, 130, 135 },
                        { 149, 187, 239, 310 } }
};
```



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

### 2 3차원 배열의 물리적 저장 방법

3차원의 논리적 순서를  
1차원의 물리적 순서로 변환하는 방법 사용

- ▶ 면의 개수가  $n_i$ 이고 행의 개수가  $n_j$ 이고  
열의 개수가  $n_k$ 인 3차원 배열  $A[n_i][n_j][n_k]$ ,  
시작주소가  $\alpha$ 이고 원소의 길이(메모리 크기)가  $\ell$  일 때,  
 $i$ 면  $j$ 행  $k$ 열 원소 즉,  $A[i][j][k]$ 의 위치(순서) 계산 방법

### 2 3차원 배열의 물리적 저장 방법

#### 면 우선 순서 방법

▶ 3차원 배열의 첫 번째 인덱스인 면 번호를 기준으로 사용하는 방법

- 원소의 위치 계산 방법  
$$\alpha + \{(i \times n_j \times n_k) + (j \times n_k) + k\} \times l$$

### 2 3차원 배열의 물리적 저장 방법

#### 열 우선 순서 방법

▶ 3차원 배열의 마지막 인덱스인 열 번호를 기준으로 사용하는 방법

- 원소의 위치 계산 방법  
$$\text{위치} = \alpha + \{(k \times n_j \times n_i) + (j \times n_i) + i\} \times l$$

### 3 3차원 논리 순서를 1차원 물리 순서로 변환

물리 순서	논리 순서
0	[0] [0] [0]
1	[0] [0] [1]
2	[0] [0] [2]
3	[0] [0] [3]
4	[0] [1] [0]
5	[0] [1] [1]
6	[0] [1] [2]
7	[0] [1] [3]
8	[1] [0] [0]
9	[1] [0] [1]
10	[1] [0] [2]
11	[1] [0] [3]
12	[1] [1] [0]
13	[1] [1] [1]
14	[1] [1] [2]
15	[1] [1] [3]

(a) 면 우선 방법

물리 순서	논리 순서
0	[0] [0] [0]
1	[1] [0] [0]
2	[0] [1] [0]
3	[1] [1] [0]
4	[0] [0] [1]
5	[1] [0] [1]
6	[0] [1] [1]
7	[1] [1] [1]
8	[0] [0] [2]
9	[1] [0] [2]
10	[0] [1] [2]
11	[1] [1] [2]
12	[0] [0] [3]
13	[1] [0] [3]
14	[0] [1] [3]
15	[1] [1] [3]

(b) 열 우선 방법

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

### 4 3차원 배열의 논리적·물리적 순서 확인하기 프로그램

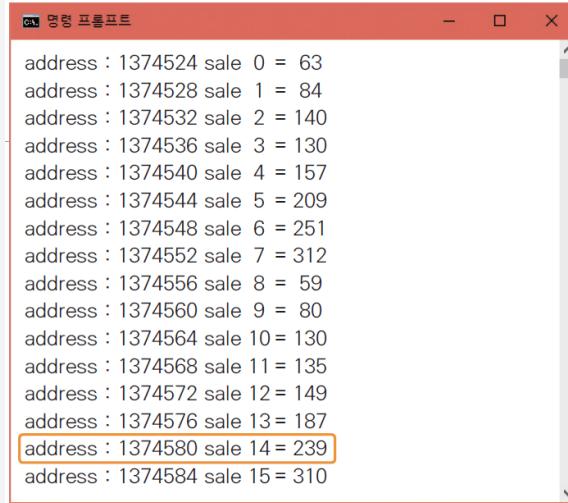
예제 3-3 3차원 배열의 논리적·물리적 순서 확인하기

```
01 #include <stdio.h>
02
03 void main() {
04     int i, n = 0, *ptr;
05     int sale[2][2][4] = {{{63, 84, 140, 130}, // 3차원 배열의 초기화
06                           {157, 209, 251, 312}},
07                           {{59, 80, 130, 135},
08                           {149, 187, 239, 310}}};
09
10     ptr = &sale[0][0][0];
11     for (i = 0; i<16; i++) {
12         printf("\n address: %u sale %2d = %3d", ptr, i, *ptr);
13         ptr++;
14     }
15     getchar();
16 }
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

## 2 | 3차원 배열을 이용한 선형리스트 구현

### 4 실행 결과



```
address : 1374524 sale 0 = 63
address : 1374528 sale 1 = 84
address : 1374532 sale 2 = 140
address : 1374536 sale 3 = 130
address : 1374540 sale 4 = 157
address : 1374544 sale 5 = 209
address : 1374548 sale 6 = 251
address : 1374552 sale 7 = 312
address : 1374556 sale 8 = 59
address : 1374560 sale 9 = 80
address : 1374564 sale 10 = 130
address : 1374568 sale 11 = 135
address : 1374572 sale 12 = 149
address : 1374576 sale 13 = 187
address : 1374580 sale 14 = 239
address : 1374584 sale 15 = 310
```

#### ▶ 실행 결과 확인

시작 주소  $\alpha=1374524$ ,  $ni=2$ ,  $nj=2$ ,  $nk=4$ ,  $i=1$ ,  $j=1$ ,  $k=2$ ,  $\ell=4$

$sale[1][1][2]=239$ 의 위치 =  $\alpha + \{(i \times nj \times nk) + (j \times nk) + k\} \times \ell$

$$= 1374524 + \{(1 \times 2 \times 4) + (1 \times 4) + 2\} \times 4$$

$$= 1374524 + 56$$

$$= 1374580$$

- C 컴파일러가 면 우선 순서 방법으로 3차원 배열을 저장함을 확인!

### 3 | 희소 행렬의 순차 자료구조 구현

## 1 행렬의 선형 리스트 표현

### 행렬(Matrix)의 개념

#### ▶ 행과 열로 구성된 자료구조

- $m \times n$  행렬 : 행 개수가  $m$ 개, 열 개수가  $n$ 개인 행렬
- 정방행렬 : 행렬 중에서  $m$ 과  $n$ 이 같은 행렬
- 전치행렬 : 행렬의 행과 열을 서로 바꿔 구성한 행렬

## 1 행렬의 선형 리스트 표현

### 행렬(Matrix)의 개념

[행렬과 전치행렬 표현과 예]

$$A = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{vmatrix} \quad A' = \begin{vmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{vmatrix}$$

(a) 행렬  $A$ ( $m \times n$  행렬)과 전치행렬  $A'$ ( $n \times m$  행렬) 표현

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

## 1 행렬의 선형 리스트 표현

### 행렬(Matrix)의 개념

[행렬과 전치행렬 표현과 예]

$$A = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{vmatrix} \xrightarrow{\text{전치행렬로 변환}} A' = \begin{vmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{vmatrix}$$

(b) 3x4 행렬 A와 전치행렬 A' 예

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

#### 1 행렬의 선형 리스트 표현

- ▶ 행렬의 각 원소는 행과 열로 표현할 수 있으므로 선형 리스트 표현  $m \times n$  행렬 A를 아래와 같이 2차원 배열 A[m][n]으로 표현할 수 있음

[행렬과 A의 2차원 배열 표현 예]

$$A = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{vmatrix} \quad \xrightarrow{\text{A}[3][4]} \quad \begin{array}{c} \text{[0]} \quad \text{[1]} \quad \text{[2]} \quad \text{[3]} \\ \hline \text{[0]} & 1 & 2 & 4 & 5 \\ \text{[1]} & 5 & 6 & 7 & 8 \\ \text{[2]} & 9 & 10 & 11 & 12 \end{array}$$

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

## 2 희소행렬(Sparse Matrix)

- ▶ 행렬의 원소 중에서 많은 항들이 0으로 구성된 행렬
- ▶ 원소 대부분이 0이라 실제로 사용하지 않는 공간이 많아 기억 공간의 활용도가 떨어짐
  - 따라서 기억공간을 좀더 효율적으로 사용하려면 0이 아닌 값이 있는 원소만 따로 배열로 구성하는 방법을 사용할 수 있음

## 2 희소행렬(Sparse Matrix)

[희소행렬 B의 2차원 배열 표현 예1]

B =

0	0	2	0	0	0	12	
0	0	0	0	7	0	0	
23	0	0	0	0	0	0	
0	0	0	31	0	0	0	
0	14	0	0	0	25	0	
0	0	0	0	0	0	6	
52	0	0	0	0	0	0	
0	0	0	0	11	0	0	

→

B[8][7]	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]	0	0	2	0	0	0	12
[1]	0	0	0	0	7	0	0
[2]	23	0	0	0	0	0	0
[3]	0	0	0	31	0	0	0
[4]	0	14	0	0	0	25	0
[5]	0	0	0	0	0	0	6
[6]	52	0	0	0	0	0	0
[7]	0	0	0	0	11	0	0

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

#### 3 희소 행렬에 대한 2차원 배열 표현

- ▶ 희소 행렬 B는 배열의 원소 56개 중 실제 사용하는 것은 0이 아닌 원소를 저장하는 10개뿐이므로 46개의 메모리 공간 낭비
- ▶ 기억 공간을 좀 더 효율적으로 사용하기 위해 0이 아닌 값이 있는 원소만 따로 배열로 구성하는 방법
  - ① 0이 아닌 원소만 추출하여 <행번호, 열번호, 원소> 쌍으로 배열에 저장
  - ② 추출한 순서쌍을 2차원 배열에 행으로 저장
  - ③ 원래의 행렬에 대한 정보를 순서쌍으로 작성하여 0번 행에 저장

# 3 | 희소 행렬의 순차 자료구조 구현

## 3 희소 행렬에 대한 2차원 배열 표현

B[8][7]

[0]	[1]	[2]	[3]	[4]	[5]	[6]	
[0]	0	0	2	0	0	0	12
[1]	0	0	0	0	7	0	0
[2]	23	0	0	0	0	0	0
[3]	0	0	0	31	0	0	0
[4]	0	14	0	0	0	25	0
[5]	0	0	0	0	0	0	6
[6]	52	0	0	0	0	0	0
[7]	0	0	0	0	11	0	0

① 희소행렬에서  
<행 번호, 열 번호, 값>  
쌍을 구함

② 2차원  
배열에 저장

③ <전체 행의 개수,  
전체 열의 개수,  
0이 아닌 원소의 개수>

[0]	[1]	[2]	
[0]	8	7	10
[1]	0	2	2
[2]	0	6	12
[3]	1	4	7
[4]	2	0	23
[5]	3	3	31
[6]	4	1	14
[7]	4	5	25
[8]	5	6	6
[9]	6	0	52
[10]	7	4	11

## 4

## 희소행렬의 전치 연산 함수 프로그램

예제 3-5

희소행렬의 전치 연산하기

```
01 #include<stdio.h>
02
03 typedef struct { // 행렬 원소를 저장하기 위한 구조체 term 정의
04     int row;
05     int col;
06     int value;
07 } term;
08
```

```
09 void smTranspose(term a[], term b[]) {
10     int m, n, v, i, j, p;
11     m = a[0].row;      // 희소 행렬 a의 행 수
12     n = a[0].col;      // 희소 행렬 a의 열 수
13     v = a[0].value;    // 희소 행렬 a에서 0이 아닌 원소 수
14     b[0].row = n;      // 전치 행렬 b의 행 수
15     b[0].col = m;      // 전치 행렬 b의 열 수
16     b[0].value = v;    // 전치 행렬 b의 원소 수
17     if (v > 0) {       // 0이 아닌 원소가 있는 경우에만 전치 연산 수행
18         p = 1;
19         for (i = 0; i < n; i++)           // 희소 행렬 a의 열별로 전치 반복 수행
20             for (j = 1; j <= v; j++)        // 0이 아닌 원소 수에 대해서만 반복 수행
21                 if (a[j].col == i) {        // 현재의 열에 속하는 원소가 있으면 b[]에 삽입
22                     b[p].row = a[j].col;
23                     b[p].col = a[j].row;
24                     b[p].value = a[j].value;
25                     p++;
26                 }
27     }
28 }
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

#### 4 희소행렬의 전치 연산 함수 프로그램

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]	0	0	2	0	0	0	12
[1]	0	0	0	0	7	0	0
[2]	23	0	0	0	0	0	0
[3]	0	0	0	31	0	0	0
[4]	0	14	0	0	0	25	0
[5]	0	0	0	0	0	0	6
[6]	52	0	0	0	0	0	0
[7]	0	0	0	0	11	0	0

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[0]	0	0	23	0	0	0	52	0
[1]	0	0	0	0	14	0	0	0
[2]	2	0	0	0	0	0	0	0
[3]	0	0	0	31	0	0	0	0
[4]	7	0	0	0	0	0	0	11
[5]	0	0	0	0	25	0	0	0
[6]	12	0	0	0	0	6	0	0

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어



## 4 희소행렬의 전치 연산 함수 프로그램

	[0]	[1]	[2]
[0]	8	7	10
[1]	0	2	2
[2]	0	6	12
[3]	1	4	7
[4]	2	0	23
[5]	3	3	31
[6]	4	1	14
[7]	4	5	25
[8]	5	6	6
[9]	6	0	52
[10]	7	4	11

전치행렬 변환



```
p = 1;
for (i = 0; i < n; i++)
    for (j = 1; j <= v; j++)
        if (a[j].col == i) {
            b[p].row = a[j].col;
            b[p].col = a[j].row;
            b[p].value = a[j].value;
            p++;
        }
```

	[0]	[1]	[2]
[0]	7	8	10
[1]	0	2	23
[2]	0	6	52
[3]	1	4	14
[4]	2	0	2
[5]	3	3	31
[6]	4	1	7
[7]	4	7	11
[8]	5	4	25
[9]	6	0	12
[10]	6	5	6

■ 전치행렬  
행렬의 행과 열을  
서로 바꿔 구성한 행렬

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어