순차 자료 구조
(선형 리스트)

남춘성

1. 선형 리스트

❖ 리스트(List)

- 자료를 나열한 목록
- [표4-1]리스트의 예

[표 4-1] 리스트 예

동창 이름 리스트	좋아하는 음식 리스트	오늘의 할 일 리스트
상원	김치찌개	운동
승희	닭볶음탕	자료구조 스터디
수영	된장찌개	과제 제출
철이	잡채	동아리 공연 연습
…	…	…

1. 선형 리스트

❖ 선형 리스트(Linear List)

- 순서 리스트(Ordered List)
- 자료들 간에 순서를 갖는 리스트
- [표4-2]선형 리스트의 예

[표 4-2] 선형 리스트 예

동창 이름 리스트		좋아하는 음식 리스트		오늘의 할 일 리스트	
1	상원	1	김치찌개	1	운동
2	승희	2	닭볶음탕	2	자료구조 스터디
3	수영	3	된장찌개	3	과제 제출
4	철이	4	잡채	4	동아리 공연 연습
…		…		…	

Data structure – 선형 자료구조(선형리스트)

- 선형 리스트(Linear List)
  - 리스트의 표현 형식

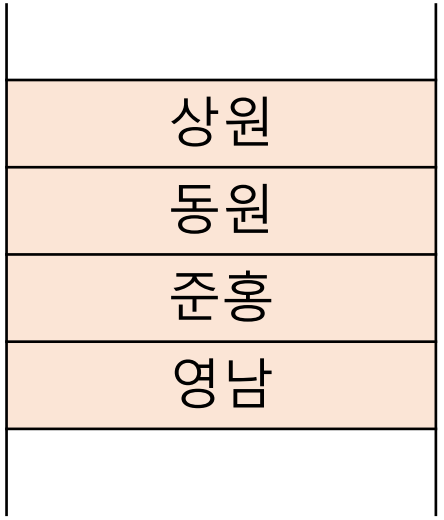
리스트 이름 = (원소1, 원소2, ..., 원소n)

  - 선형 리스트에서 원소를 나열한 순서는 원소들의 순서가 됨.
  - 이름 선형 리스트의 표현
    - 이름 = (상원, 동원, 준홍, 영남)
  - 공백 리스트
    - 원소가 하나도 없는 리스트
    - 빈 괄호를 사용하여 표현

공백리스트이름 = **( )**

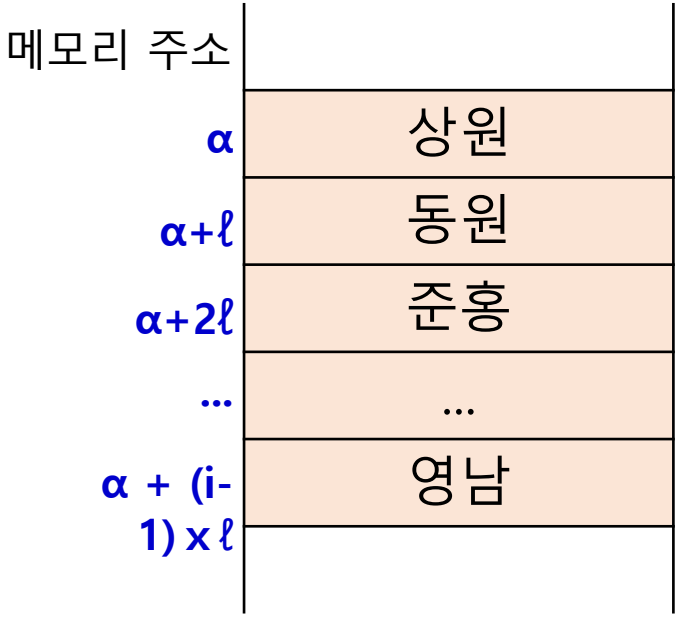
Data structure – 선형 자료구조(선형리스트)

- 원소들의 논리적 순서와 같은 순서로 메모리에 저장
- 순차 자료구조
  - 원소들의 논리적 순서 = 원소들이 저장된 물리적 순서
  - 이름 선형 리스트가 메모리에 저장된 물리적 구조



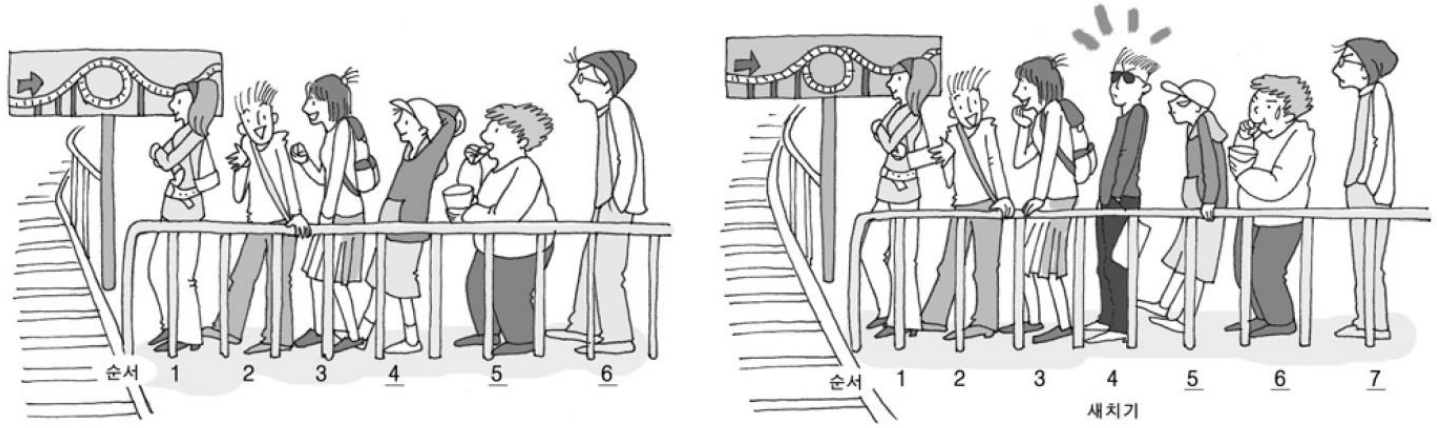
Data structure – 선형 자료구조(선형리스트)

- 선형 리스트의 저장
  - 순차 자료구조의 원소 위치 계산
    - 선형 리스트가 저장된 시작 위치 : **α**
    - 원소의 길이 : **ℓ**
    - i번째 원소의 위치 = **α + (i-1) x ℓ**



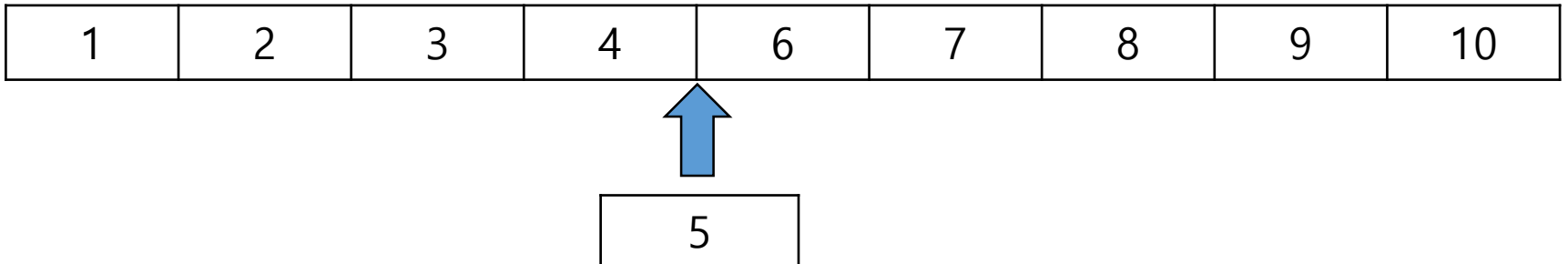
Data structure – 선형 자료구조(선형리스트)

- 선형 리스트에서 원소 삽입
  - 선형리스트 중간에 원소가 삽입되면, 그 이후의 원소들은 한자리씩 자리를 뒤로 이동하여 물리적 순서를 논리적 순서와 일치시킨다.



(a) 새치기 전 (b) 새치기 후

[그림 4-3] 새치기 전과 후의 위치와 순서 변화



Data structure – 선형 자료구조(선형리스트)

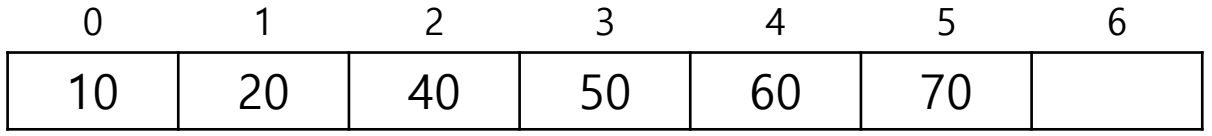
- 원소 삽입 방법

(1) 원소를 삽입할 빈 자리 만들기

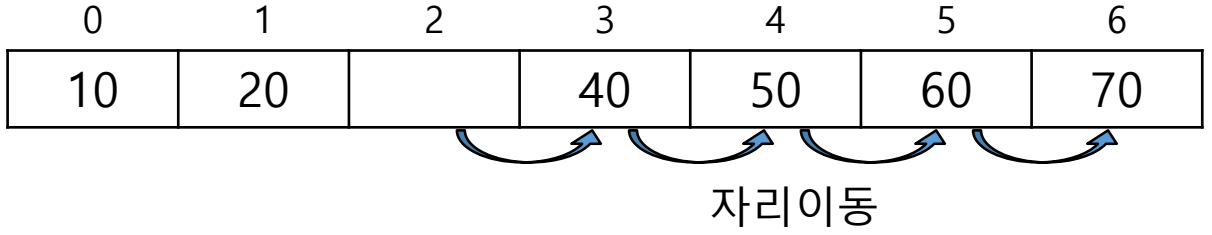
✓삽입할 자리 이후의 원소들을 한자리씩 뒤로 자리 이동

(2) 준비한 빈 자리에 원소 삽입하기

a. 원소 삽입 전

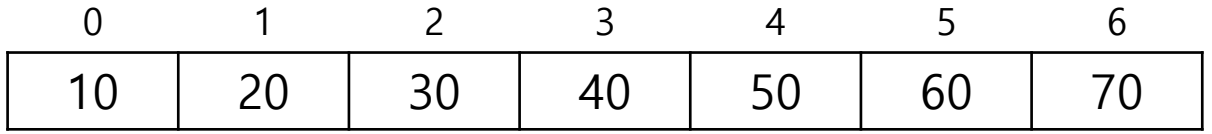


(1)



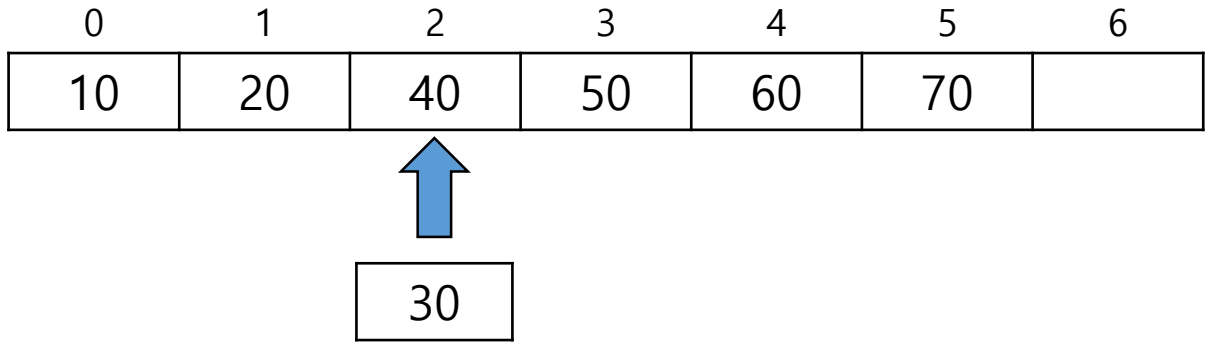
b. 원소 삽입 후

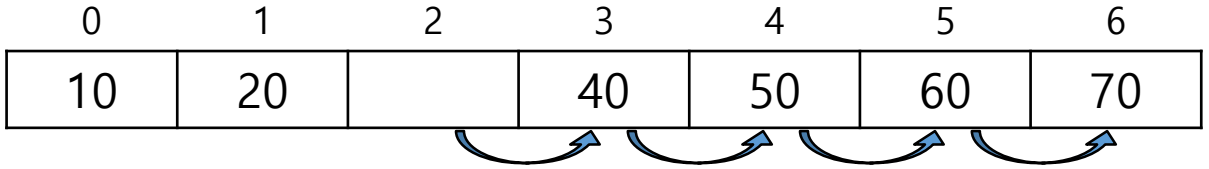
(2)



Data structure – 선형 자료구조(선형리스트)

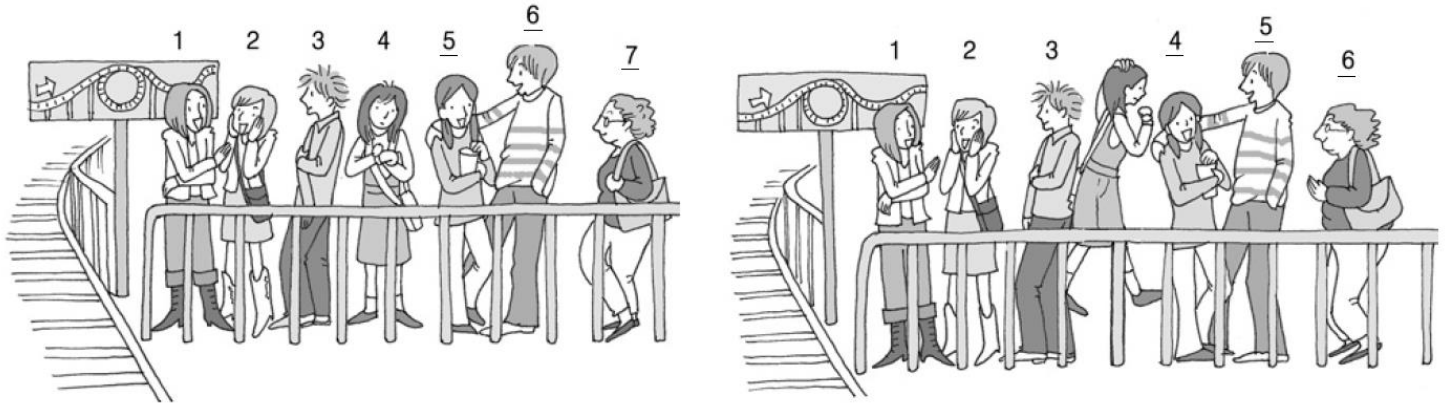
- 삽입할 자리를 만들기 위한 자리이동 횟수
  - (n+1)개의 원소로 이루어진 선형 리스트에서 k번 자리에 원소를 삽입하는 경우 :
    - k번 원소부터 마지막 인덱스 n번 원소까지 (n-k+1)개의 원소를 이동
    - **이동횟수 = n-k+1 = 마지막 원소의 인덱스 - 삽입할 자리의 인덱스 +1**





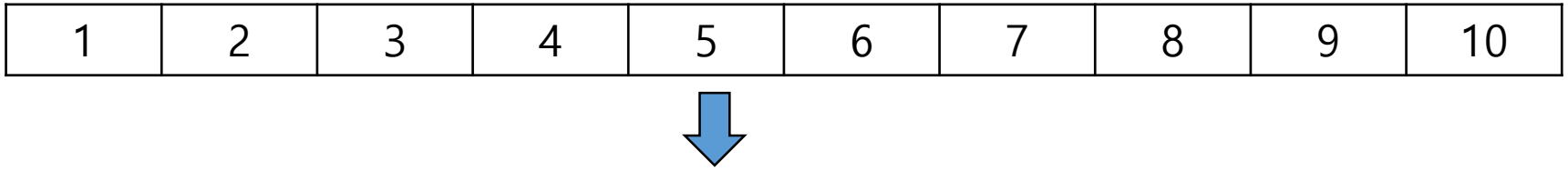
Data structure – 선형 자료구조(선형리스트)

- 선형 리스트에서 원소 삭제
  - 선형리스트 중간에서 원소가 삭제되면, 그 이후의 원소들은 한자리씩 자리를 앞으로 이동하여 물리적 순서를 논리적 순서와 일치시킨다.



(a) 나가기 전 (b) 나간 후

[그림 4-5] 줄에서 사람이 나간 후의 위치와 순서 변화



Data structure – 선형 자료구조(선형리스트)

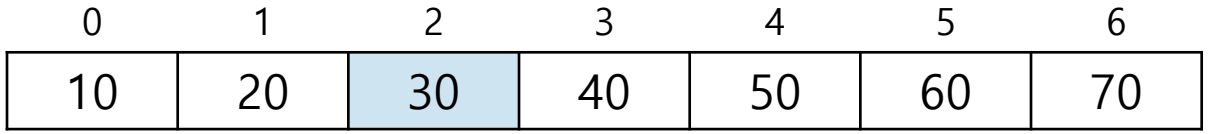
- 원소 삭제 방법

(1) 원소 삭제하기

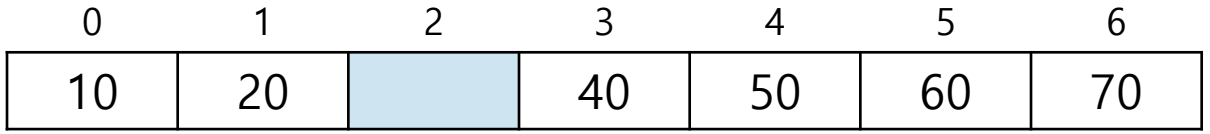
(2) 삭제한 빈 자리 채우기

☞ 삭제한 자리 이후의 원소들을 한자리씩 앞으로 자리 이동

a. 원소 삭제 전

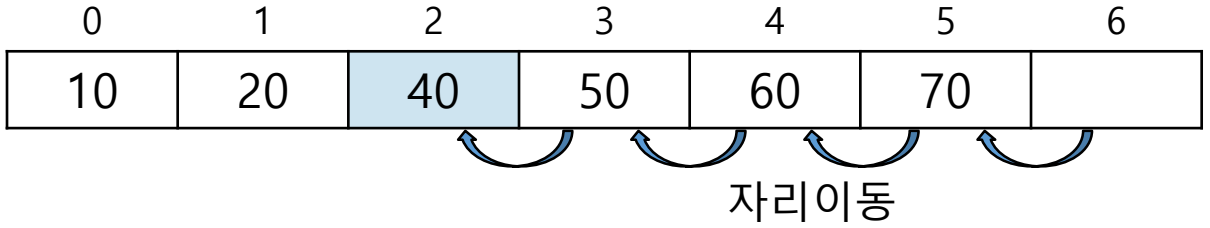


(1)



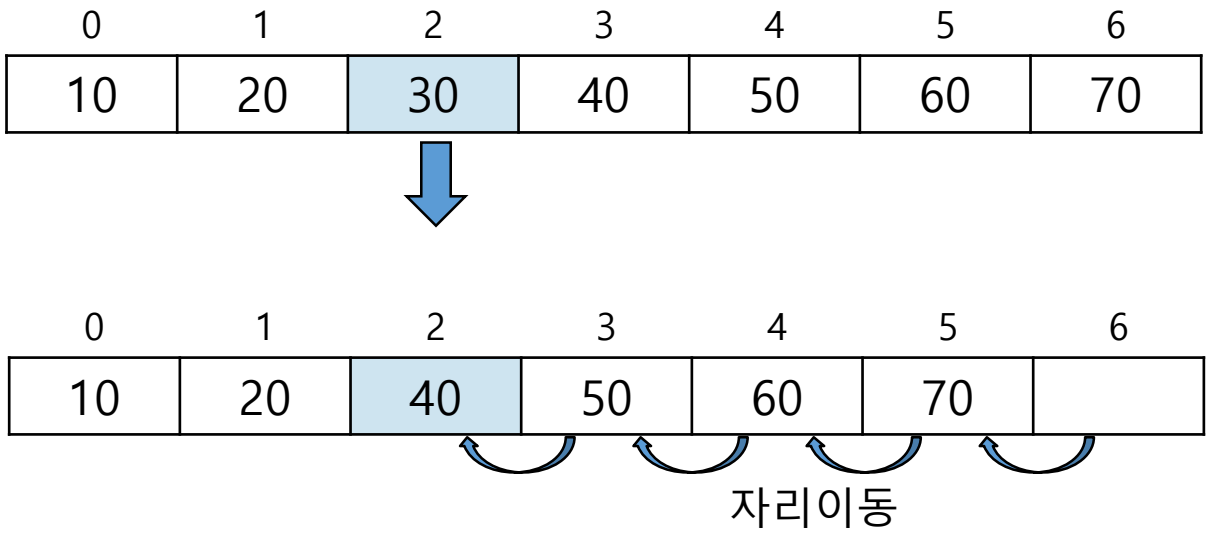
b. 원소 삭제 후

(2)



Data structure – 선형 자료구조(선형리스트)

- 삭제 후, 빈 자리를 채우기 위한 자리이동 횟수
  - (n+1)개의 원소로 이루어진 선형 리스트에서 k번 자리의 원소를 삭제한 경우 :
    - (k+1)번 원소부터 마지막 n번 원소까지 (n-(k+1)+1)개의 원소를 이동
    - **이동횟수 = n-(k+1)+1 = n-k = 마지막 원소의 인덱스-삭제한 자리의 인덱스**



Data structure – 선형 자료구조(선형리스트)

- 선형 리스트의 구현
  - 순차 구조의 리스트를 사용
    - 리스트 : <인덱스, 원소>의 순서쌍의 집합
    - 리스트의 인덱스 : 배열 원소의 순서 표현

Data structure – 선형 자료구조(선형리스트)

- Python list를 이용한 1차원 선형 리스트의 구현
  - 분기별 노트북 판매량 리스트

분기	1/4분기	2/4분기	3/4분기	4/4분기
판매량	157	209	251	312

  - python list를 이용한 구현

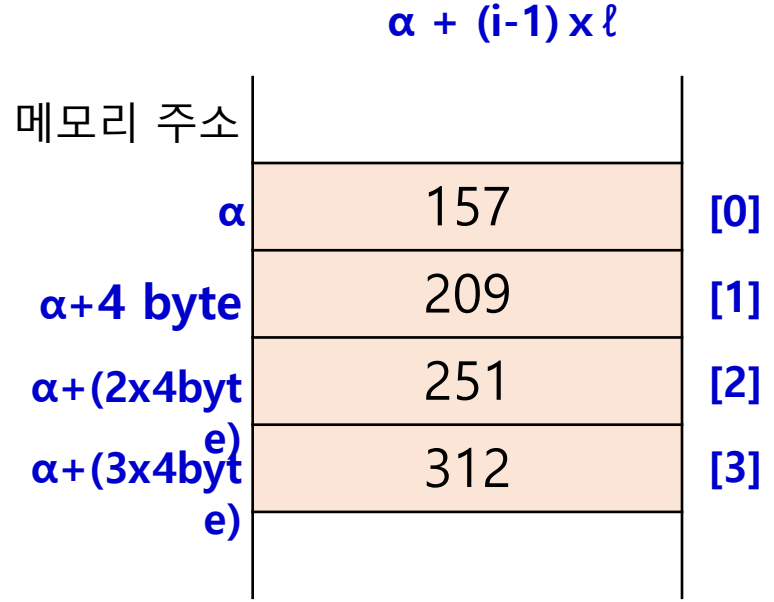
```
>>> sale=[157, 209, 251, 312]
```

  - 논리적 구조

sale

0	1	2	3
157	209	251	312

- 물리적 구조



Data structure – 선형 자료구조(선형리스트)

- 2차원 리스트를 이용한 선형 리스트의 구현
  - 분기별 노트북 판매량 리스트

분기 / 년	1/4분기	2/4분기	3/4분기	4/4분기
2009	63	84	140	130
2010	157	209	251	312

  - python list를 이용한 구현

```
>>> sale=[[63,84,140,130][157,209,251,312]]
```

  - 논리적 구조

sale

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

메모리 주소

Data structure – 선형 자료구조(선형리스트)

- 2차원 배열의 물리적 저장 방법
  - 2차원의 논리적 순서를 1차원의 물리적 순서로 변환하는 방법을 사용
  - 행 우선 순서 방법(row major order)
    - 2차원 배열의 첫 번째 인덱스인 행 번호를 기준으로 사용하는 방법
    - sale[**0**][0]=63, sale[**0**][1]=84, sale[**0**][2]=140, sale[**0**][3]=130, sale[**1**][0]=157, sale[**1**][1]=209, sale[**1**][2]=251, sale[**1**][3]=312
  - 열 우선 순서 방법(column major order)
    - 2차원 배열의 마지막 인덱스인 열 번호를 기준으로 사용하는 방법
    - sale[0][**0**]=63, sale[1][**0**]=157, sale[0][**1**]=84, sale[1][**1**]=209, sale[0][**2**]=140, sale[1][**2**]=251, sale[0][**3**]=130, sale[1][**3**]=312

Data structure – 선형 자료구조(선형리스트)

- 물리적 구조

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

(a) 행 우선 순서 방법

물리 순서		논리 순서	
0	63	[0][0]	행0
1	84	[0][1]	
2	140	[0][2]	
3	130	[0][3]	
4	157	[1][0]	행1
5	209	[1][1]	
6	251	[1][2]	
7	312	[1][3]	

(b) 열 우선 순서 방법

물리 순서		논리 순서	
0	63	[0][0]	열0
1	157	[1][0]	
2	84	[0][1]	열1
3	209	[1][1]	
4	140	[0][2]	열2
5	251	[1][2]	
6	130	[0][3]	열3
7	312	[1][3]	

Data structure – 선형 자료구조(선형리스트)

- 3차원 배열을 이용한 선형 리스트의 구현
  - 2009~20010년, A팀과 B팀의 분기별 노트북 판매량

A팀				
분기 년	1/4분기	2/4분기	3/4분기	4/4분기
2009	63	84	140	130
2010	157	209	251	312

B팀				
분기 년	1/4분기	2/4분기	3/4분기	4/4분기
2009	63	84	140	130
2010	157	209	251	312

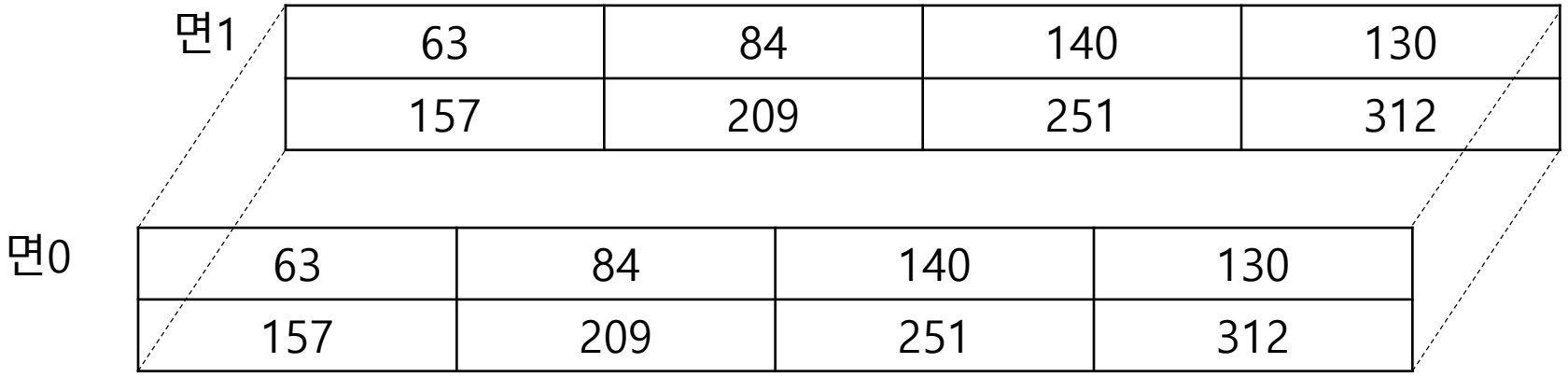
Data structure – 선형 자료구조(선형리스트)

- 3차원 리스트를 이용한 구현

```
>>> sale=[[[63,84,140,130],[157,209,251,312]],[[63,84,140,130],[157,209,251,312]]]
```

```
>>> sale=[
          [[63,84,140,130],[157,209,251,312]]
         ,[[63,84,140,130],[157,209,251,312]]
         ]
```

- 논리적 구조



Data structure – 선형 자료구조(선형리스트)

- 3차원 배열의 물리적 저장 방법
  - 3차원의 논리적 순서를 1차원의 물리적 순서로 변환하는 방법을 사용
  - 면 우선 순서 방법
    - 3차원 배열의 첫 번째 인덱스인 면 번호를 기준으로 사용하는 방법
    - 원소의 위치 계산 방법 : $\alpha + \{(i \times n_j \times n_k) + (j \times n_k) + k\} \times \ell$

    면의 개수가 n_i이고 행의 개수가 n_j이고, 열의 개수가 n_k 인 3차원 배열 $A[n_i][n_j][n_k]$, 시작주소가 α이고 원소의 길이가 ℓ 일 때, i면 j행 k열 원소 즉, A[i][j][k]의 위치

  - 열 우선 순서 방법
    - 3차원 배열의 마지막 인덱스인 열 번호를 기준으로 사용하는 방법
    - 원소의 위치 계산 방법 : $\alpha + \{(k \times n_j \times n_i) + (j \times n_i) + i\} \times \ell$

Data structure – 선형 자료구조(선형리스트)

- 물리적 구조

(a) 면 우선 순서 방법

물리순서		논리순서	
0	63	[0][0][0]	면0
1	84	[0][0][1]	
2	140	[0][0][2]	
3	130	[0][0][3]	
4	157	[0][1][0]	
5	209	[0][1][1]	
6	251	[0][1][2]	
7	312	[0][1][3]	
8	63	[1][0][0]	면1
9	84	[1][0][1]	
10	140	[1][0][2]	
11	130	[1][0][3]	
12	157	[1][1][0]	
13	209	[1][1][1]	
14	251	[1][1][2]	
15	312	[1][1][3]	

(b) 열 우선 순서 방법

물리순서		논리순서	
0	63	[0][0][0]	열0
1	63	[1][0][0]	
2		[0][1][0]	
3		[1][1][0]	
4		[0][0][1]	열1
5		[1][0][1]	
6		[0][1][1]	
7		[1][1][1]	
8		[0][0][2]	열2
9		[1][0][2]	
10		[0][1][2]	
11		[1][1][2]	
12		[0][0][3]	열3
13		[1][0][3]	
14		[0][1][3]	
15		[1][1][3]	

3. 다항식의 순차 자료구조 표현

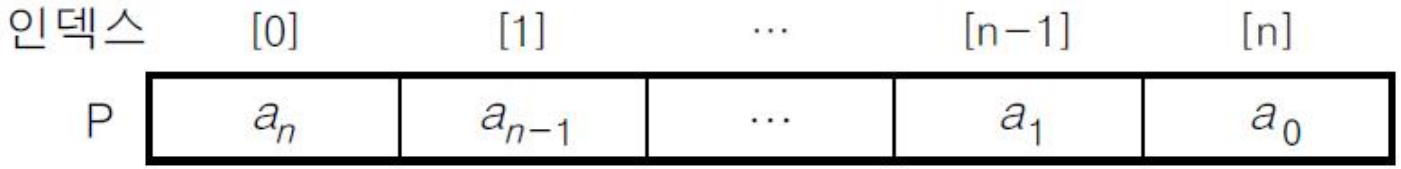
❖ **다항식의 표현**

- 각 항의 지수와 계수의 쌍에 대한 선형 리스트
  - 예) $A(x)=4x^3+3x^2+2$ ☞ p1= (3,4, 2,3, 0,2)

❖ **1차원 배열을 이용한 순차 자료구조 표현**

- 차수가 n인 다항식을 (n+1)개의 원소를 가지는 1차원 배열로 표현
- 배열 인덱스 i : 지수(n-i)을 의미
- 배열 인덱스 i의 원소 : 지수(n-i)항의 계수
  - 다항식에 포함되지 않은 지수의 항에 대한 원소에 0 저장

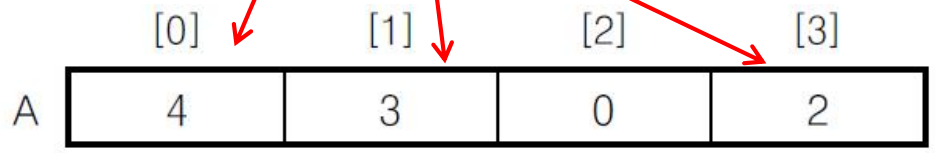
$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0 x^0$$



[그림 4-13] n차 다항식 P(x)의 순차 자료구조 표현

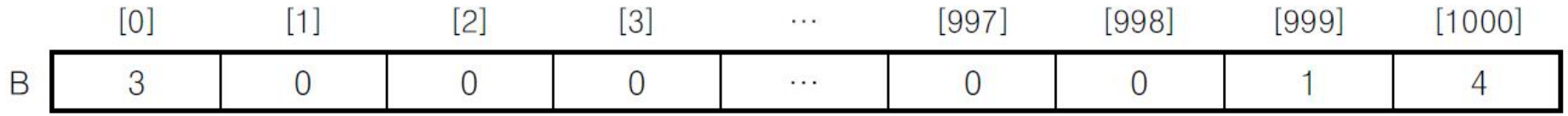
3. 다항식의 순차 자료구조 표현

- 예) $A(x)=4x^3+3x^2+2$ 의 순차 자료구조 표현



[그림 4-14] A(x)의 순차 자료구조 표현

- 희소 다항식에 대한 1차원 배열 저장
  - 예) $B(x)=3x^{1000} + x + 4$



[그림 4-15] B(x)의 순차 자료구조 표현

  - 차수가 1000이므로 크기가 1001인 배열을 사용하는데, 항이 3개 뿐이므로 배열의 원소 중에서 3개만 사용
    - ☞ 998개의 배열 원소에 대한 메모리 공간 낭비!

3. 다항식의 순차 자료구조 표현

❖ 2차원 배열을 이용한 순차 자료구조 표현

- 다항식의 각 항에 대한 **<지수, 계수>**의 쌍을 2차원 배열에 저장
  - 2차원 배열의 행의 개수 : 다항식의 항의 개수
  - 2차원 배열의 열의 개수 : 2
  - 예) $B(x)=3x^{1000} + x + 4$ 의 2차원 배열 표현
    - 1차원 배열을 사용하는 방법보다 메모리 사용 공간량 감소
      ☞ 공간 복잡도 감소 ☞ 프로그램 성능 향상!

	[0]	[1]	
[0]	1000	3	☞ $3x^{1000}$
[1]	1	1	☞ x
[2]	0	4	☞ 4

[그림 4-16] 희소 다항식 B(x)의 순차 자료구조 표현

Data structure – 선형 자료구조(선형리스트)

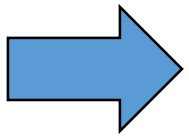
- 행렬의 선형 자료구조 표현
- 행렬(matrix)
  - m x n 행렬
    - m : 행의 개수
    - n : 열의 개수
    - 원소의 개수 : (m x n) 개

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

Data structure – 선형 자료구조(선형리스트)

- 행렬의 순차 자료구조 표현
  - 2차원 배열 사용
    - m×n행렬을 m행 n열의 2차원 배열로 표현

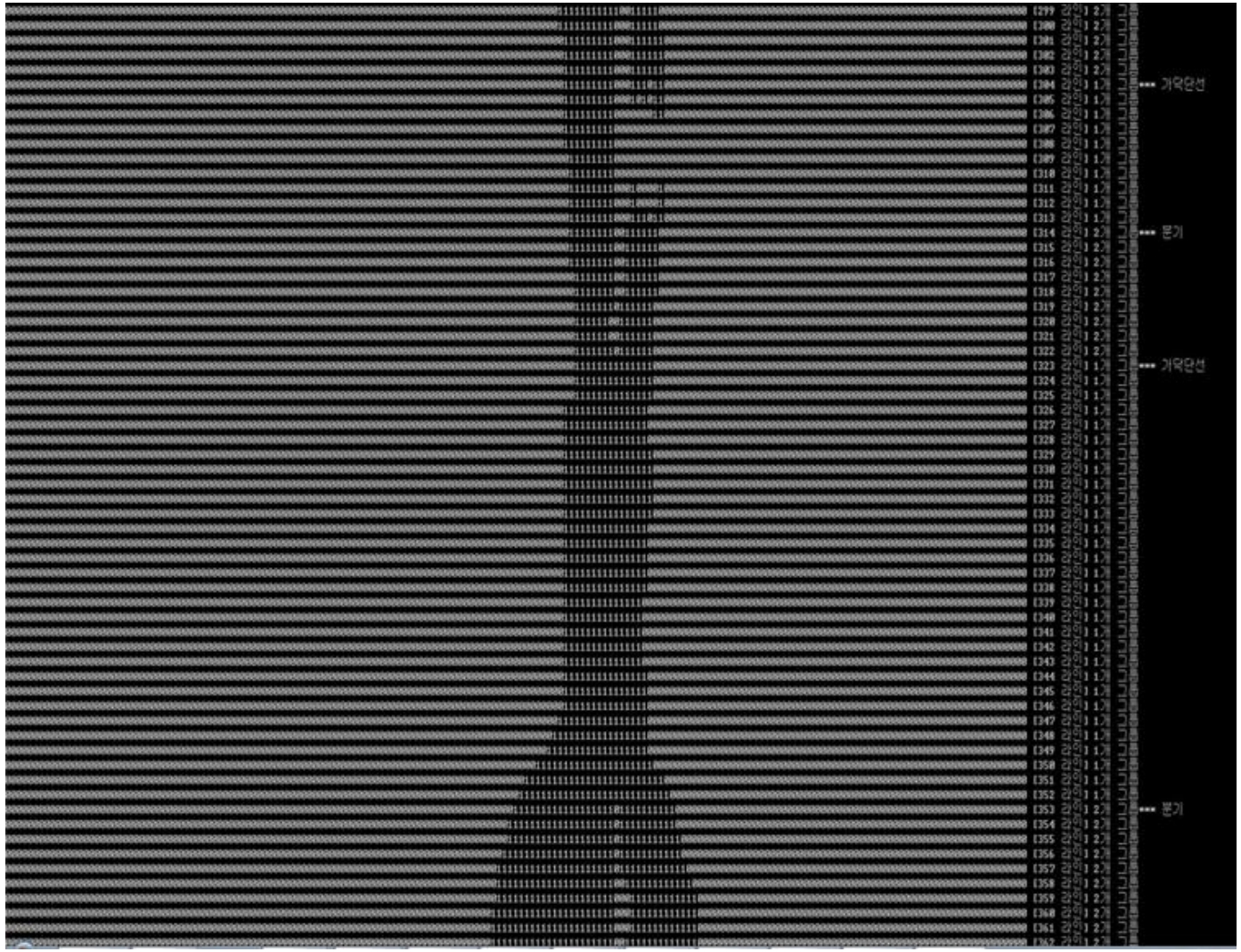
$$A = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{vmatrix}$$



A[3][4]	[0]	[0]	[0]	[0]
[0]	1	2	3	4
[0]	5	6	7	8
[0]	9	10	11	12

Data structure – 선형 자료구조(선형리스트)

- 이미지 행렬 표현



Data structure – 선형 자료구조(선형리스트)

- 희소 행렬에 대한 2차원 배열 표현
  - 희소 행렬 B는 배열의 원소 56개 중에서 실제 사용하는 것은 0이 아닌 원소를 저장하는 10개 뿐이므로 46개의메모리 공간 낭비
  - 희소 행렬인 경우에는 0이 아닌 원소만 추출하여 <행번호, 열번호, 원소>
    쌍으로 배열에 저장

B[8][7]

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

<0,2,2>
<0,6,12>
<1,4,7>
<2,0,23>
<3,3,31>
<4,1,14>
<4,5,25>
<5,6,6>
<6,0,52>
<7,4,11>

Data structure – 선형 자료구조(선형리스트)

- 순차 자료구조의 문제점
  - 삽입연산이나 삭제연산 후에 연속적인 물리 주소를 유지하기 위해서 원소들을 이동시키는 추가적인 작업과 시간 소요
    - 원소들의 이동 작업으로 인한 오버헤드는 원소의 개수가 많고 삽입 · 삭제 연산이 많이 발생하는 경우에 성능상의 문제 발생
  - 순차 자료구조는 배열을 이용하여 구현하기 때문에 배열이 갖고 있는 메모리 사용의 비효율성 문제를 그대로 가짐
  - 순차 자료구조에서의 연산 시간에 대한 문제와 저장 공간에 대한 문제를 개선한 자료 표현 방법 필요