

02

자료구조

배열

컴퓨터과학과정광식 교수



학습목차

- 1 배열의 정의
- 2 배열의 추상 자료형
- 3 배열연산의 구현
- 4 1차원 배열
- 5 배열의 확장
- 6 희소행렬의 개념

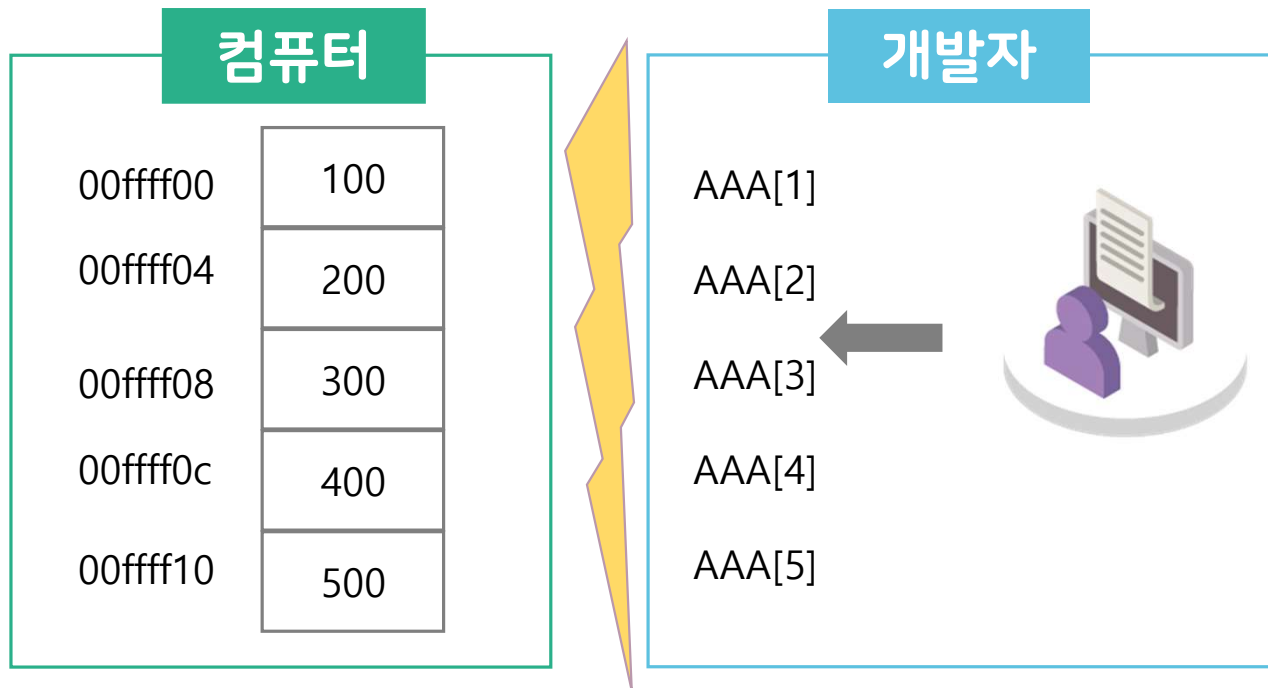
01

배열의 정의



1. 배열의 정의

배열의 정의



1. 배열의 정의

● 배열의 정의

- 일정한 차례나 간격에 따라 벌여 놓음 (사전적 정의)
- ‘차례’(순서)와 관련된 기본적인 자료구조

1. 배열의 정의

● 배열의 정의

- 원소의 메모리 공간(메인 메모리, DDR)의 물리적인 위치를 ‘순서’적으로 결정하는 특징
- 배열의 순서는 메모리 공간에서 저장되는 ‘원소값의 물리적 순서’

1. 배열의 정의

● 배열의 정의

- 인덱스와 원소값(<index, value>)의 쌍으로 구성된 집합

1. 배열의 정의

● 배열의 의미

- ‘호수’(인덱스)로 표현되는 **순서**를 갖는 ‘아파트’
⇒ 호수 : 인덱스 / 원소값 : 거주민
- 원소들이 모두 **같은 자료형**과 **같은 크기의 기억 공간**을 가짐
- 배열의 인덱스값을 이용해서 원소값에 접근하기 때문에
직접 접근이 가능함

1. 배열의 정의

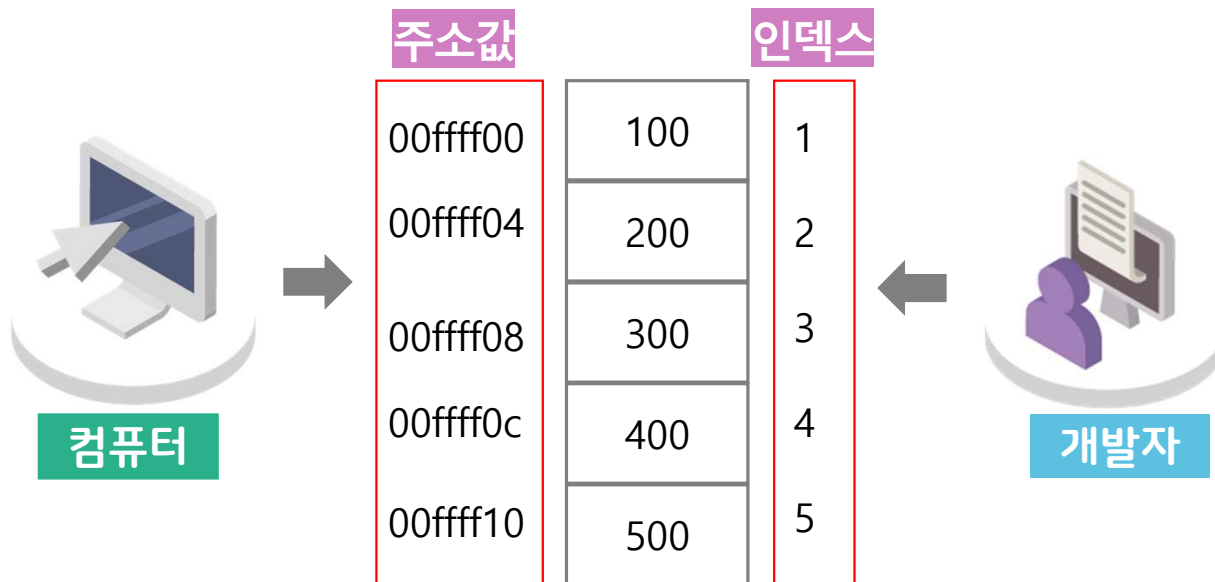
● 배열의 의미

- 배열의 인덱스값 : 추상화된 값 = 컴퓨터의 내부구조나 메모리 주소와 무관하게 개발자에게 개념적으로 정의됨
- 메모리 주소값은 실제 메모리의 물리적인 위치값
- 배열의(추상화된) 인덱스값은 프로그래밍 언어와 컴파일 과정을 통해 메모리 주소값과 연결됨

1. 배열의 정의

배열의 의미

- 인덱스와 주소값의 관계(보통 배열의 인덱스는 0부터 시작)



02

배열의 추상 자료형



KOREA NATIONAL OPEN UNIVERSITY

2. 배열의 추상 자료형

○ 추상자료형

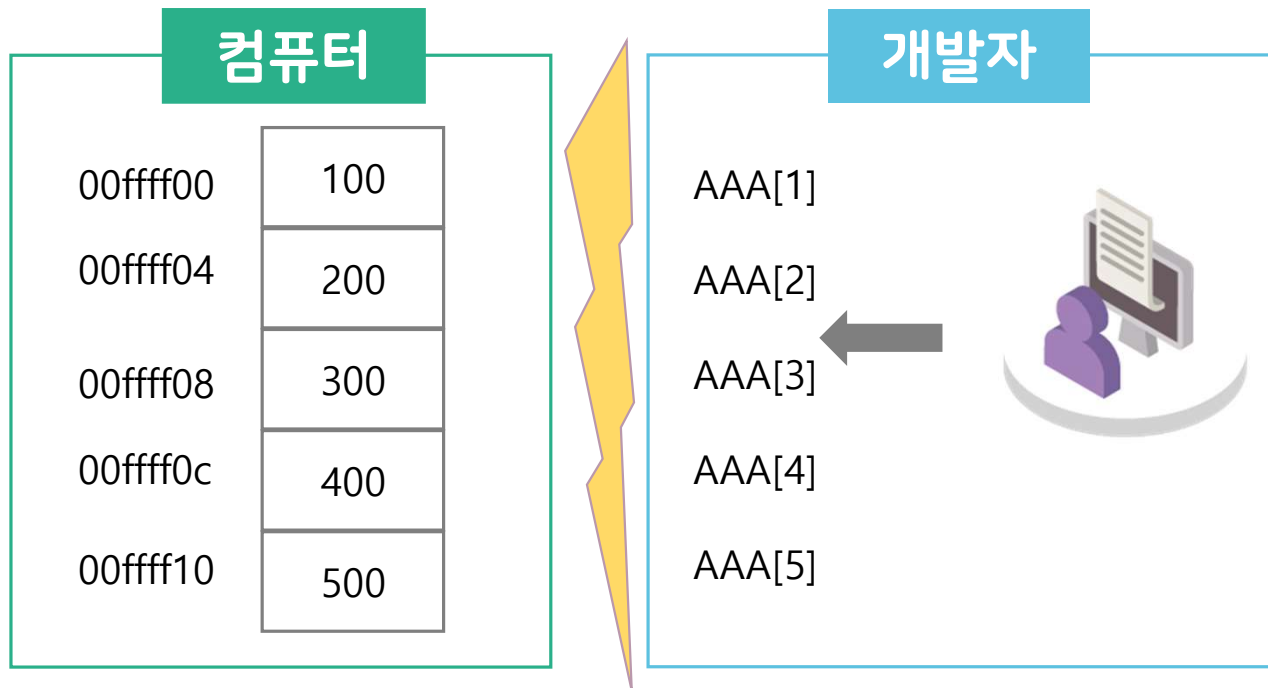
- 객체 및 관련된 연산의 정의로 구성됨
- 자료구조 구현전의 설계 단계

○ 자료형

- 메모리 저장 할당을 위한 변수 선언
- 자료구조의 구현 단계(프로그래밍 언어를 이용한 선언)

2. 배열의 추상 자료형

배열의 정의



2. 배열의 추상 자료형

- ADT Array 객체 : $\langle i \in \text{Index}, e \in \text{Element} \rangle$ 쌍들의 집합

- Index : 순서를 나타내는 원소의 유한집합
- Element : 자료형이 같은 원소의 집합

2. 배열의 추상 자료형

- 연산 : $a \in \text{Array}; i \in \text{Index}; \text{item} \in \text{Element}; n \in \text{Integer}$ 인

모든 a, item, n 에 대하여 다음과 같은 연산이 정의됨

- a : 0개 이상의 원소를 갖는 배열
- item : 배열에 저장되는 원소
- n : 배열의 최대 크기를 정의하는 정수값

2. 배열의 추상 자료형

- ① $\text{Array create}(n) ::=$ 배열의 크기가 n 인 빈 배열을 생성하고 배열을 반환한다;
- ② $\text{Element retrieve}(a, i) ::=$ if ($i \in \text{Index}$)
 then { 배열의 i 번째에 해당하는 원소값 'e'를 반환한다; }
 else { 에러 메시지를 반환한다; }
- ③ $\text{Array store}(a, i, e) ::=$ if ($i \in \text{Index}$)
 then { 배열 a 의 i 번째 위치에 원소값 'e'를 저장하고 배열 a 를 반환한다; }
 else { 인덱스 i 가 배열 a 의 크기를 벗어나면 에러 메시지를 반환한다; }

03

배열 연산의 구현



3. 배열 연산의 구현

배열의 생성

```
void create(int n) {    // n=5
    int a[n];
    int i;
    for(i=0, i<n, i++){
        a[i] = 0;
    }
}
```

	a[0]	a[1]	a[2]	a[3]	a[4]
a	0	0	0	0	0

3. 배열 연산의 구현

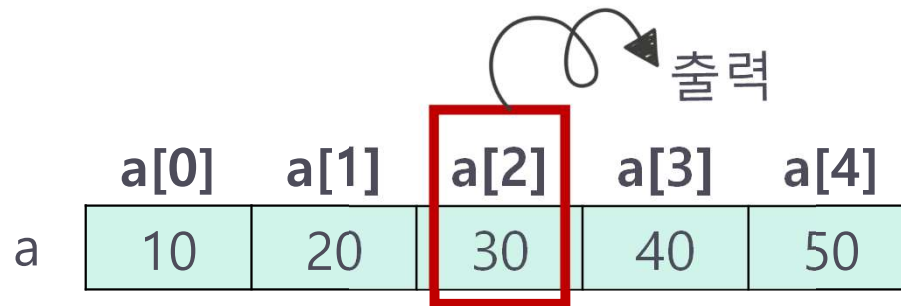
● 배열값의 검색(retrieve 연산)

```
#define array_size 5
int retrieve(int *a, int i) { // i = 2
    if( i >= 0 && i < array_size )
        return a[i];
    else { printf("Error\n");
           return(-1); }
}
```

3. 배열 연산의 구현

배열값의 검색 결과

- 다음과 같은 원소값이 저장되어 있다고 가정하며, '30'이 출력됨



	a[0]	a[1]	a[2]	a[3]	a[4]
a	10	20	30	40	50

3. 배열 연산의 구현

- 배열값의 저장(store 연산)

```
#define array_size 5
void store(int *a, int i, int e) { // i=3, e=35
    if( i >= 0 && i < array_size)
        a[i] = e;
    else printf("Error\n");
}
```

3. 배열 연산의 구현

- $a[3]$ 의 값이 35로 변경되어 저장된 모습

	$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$
a	10	20	30	40	50

↓

	$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$
a	10	20	30	35	50

04

1차원 배열



4. 1차원 배열

● 1차원 배열의 정의

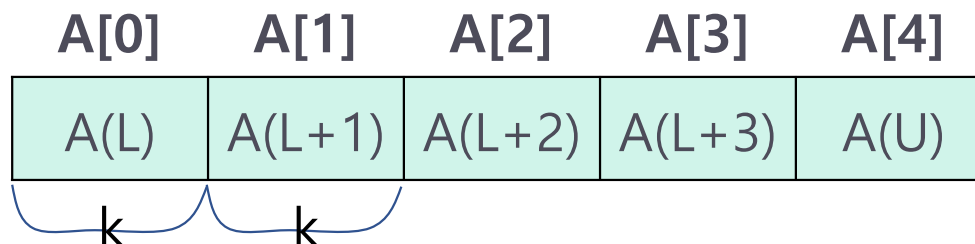
- 한 줄짜리 배열을 의미하며, 하나의 인덱스로 구분됨

$A[0]$	$A[1]$	$A[2]$	$A[3]$	$A[4]$
$A(L)$	$A(L+1)$	$A(L+2)$	$A(L+3)$	$A(U)$

4. 1차원 배열

1차원 배열의 정의

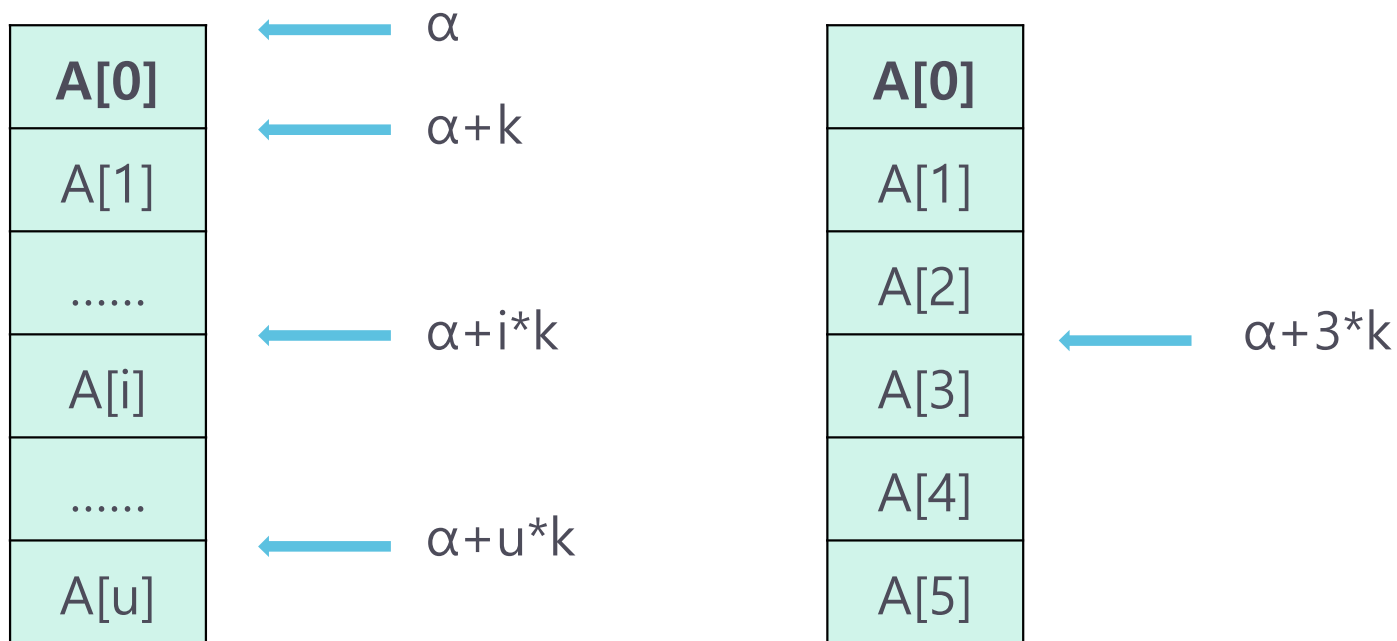
- $A[i]$ 는 배열의 첫 번째 원소 $A[0]$ 이 저장된 메모리 주소인 α 로부터 시작하여, $A[0]$ 부터 $A[i-1]$ 개까지 i 개의 배열 $A[]$ 를 지나서 저장됨
- 따라서, $A[]$ 의 메모리 시작주소를 α 라고 가정하면,
 $A[i]$ 의 메모리 저장 주소는 $[\alpha + i * k]$ 가 됨



4. 1차원 배열

1차원 배열에서의 주소 계산

- $A[0]$ 의 시작주소를 α 라고 가정하면 $A[3]$ 저장 주소는 ? $[\alpha + 3*k]$ 가 됨



05

배열의 확장



5. 배열의 확장

행렬의 배열 표현

- 행렬을 컴퓨터에서 표현하기에는 2차원 배열이 적합함

$$\begin{bmatrix} 5 & 2 & 6 & 2 \\ 7 & 2 & 0 & 0 \\ 0 & 1 & 1 & 9 \end{bmatrix}$$

5	2	6	2
7	2	0	0
0	1	1	9

5. 배열의 확장

● 행렬의 2차원 배열 표현

열(Column)



행(row)

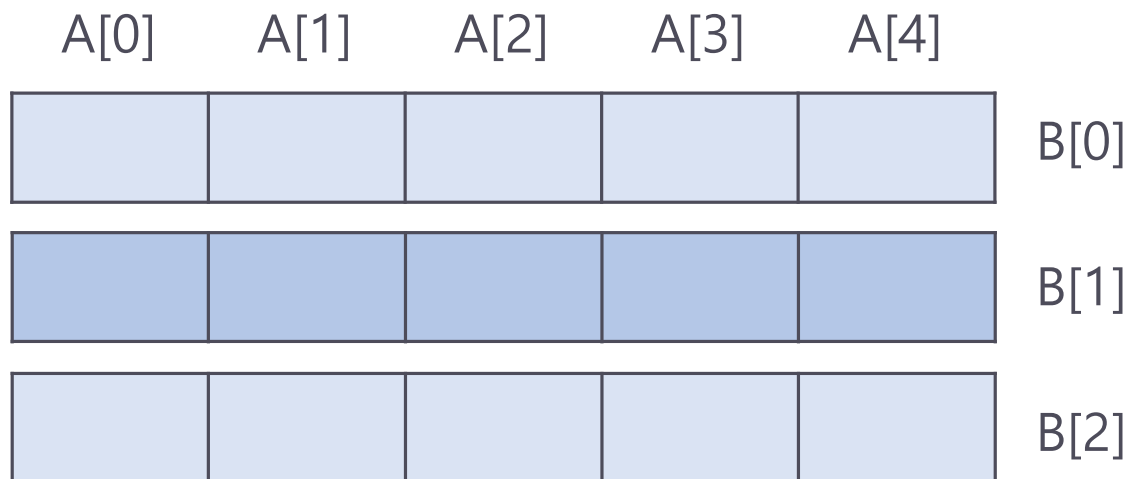


A[0][0]	A[0][1]	A[0][2]	A[0][m-1]
A[1][0]
A[2][0]
...
A[n-1][0]	A[n-1][m-1]

5. 배열의 확장

○ 행 우선 배열

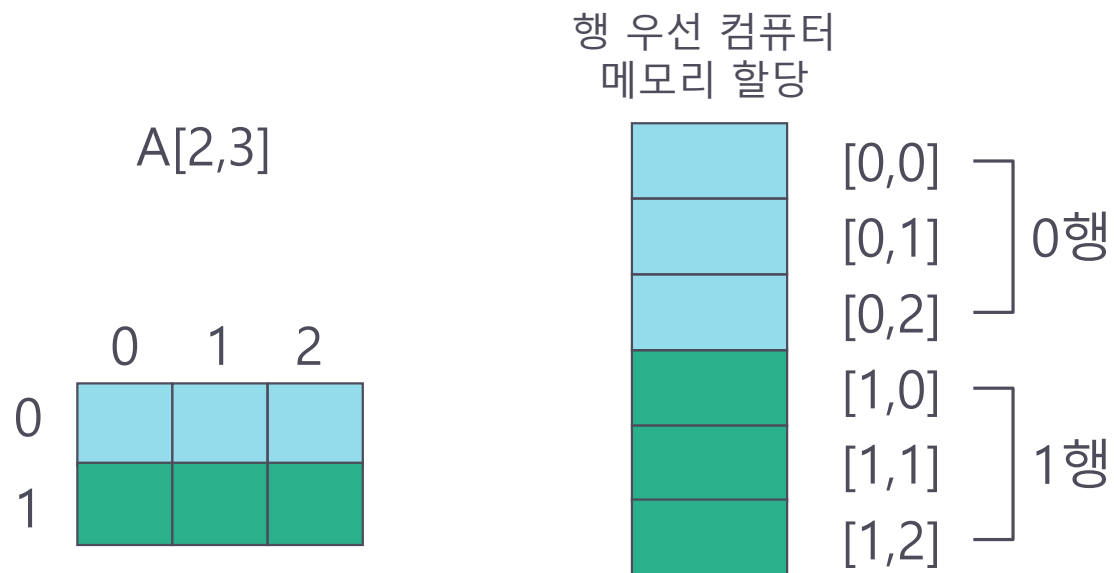
- 1차원 배열을 여러 개 쌓아 놓은 것이 2차원 배열



5. 배열의 확장

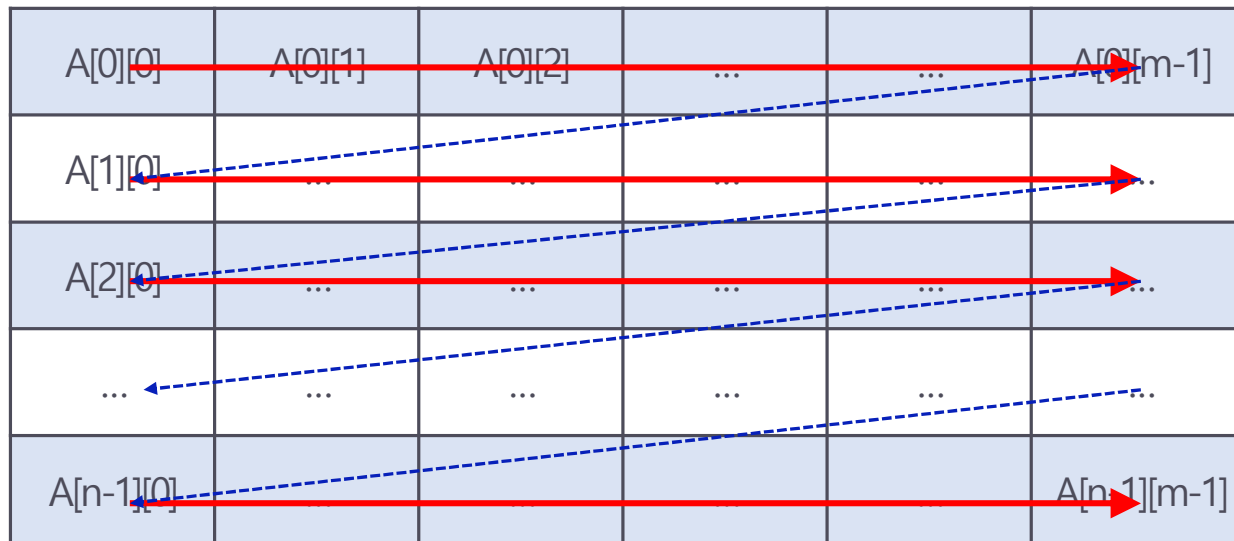
○ 행 우선 할당

- 가로 1차원 배열 단위로 메모리 영역을 우선 할당함



5. 배열의 확장

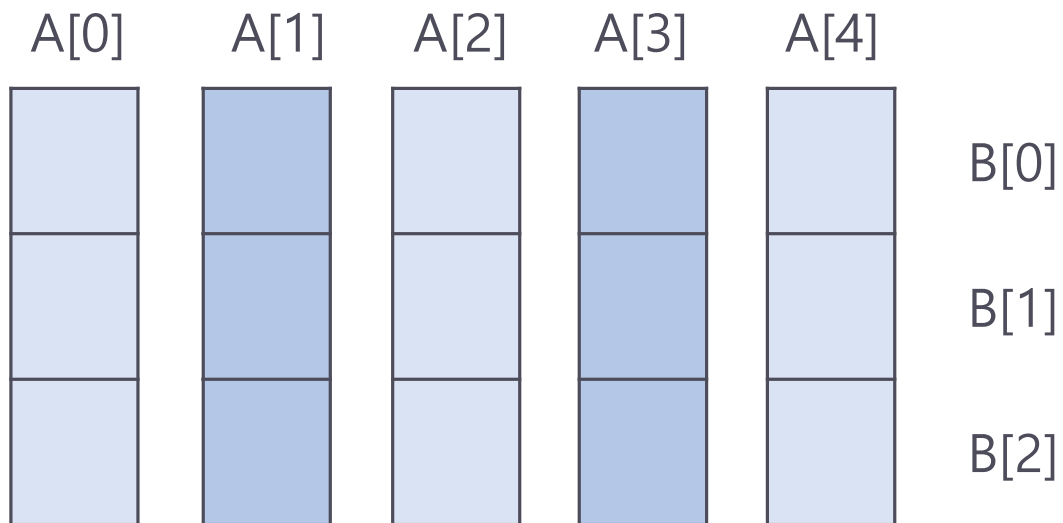
행 우선 배열



5. 배열의 확장

● 열 우선 배열

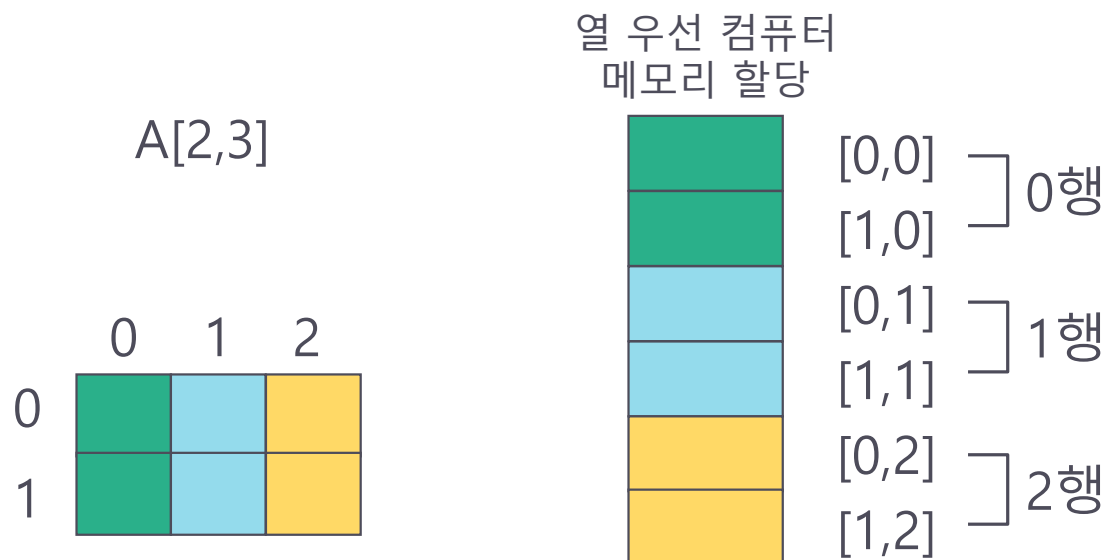
- 1차원 배열을 여러 개 세워 놓은 것이 2차원 배열



5. 배열의 확장

열 우선 할당

- 세로의 1차원 배열 단위로 메모리 영역을 우선 할당함



5. 배열의 확장

행 우선 배열

$A[0][0]$	$A[0][1]$	$A[0][2]$	$A[0][m-1]$
$A[1][0]$
$A[2][0]$
...
$A[n-1][0]$	$A[n-1][m-1]$

5. 배열의 확장

- C언어에서의 2차원 배열(행 우선 순서 저장)

- C 언어에서 `A[3][5]`을 선언하면 다음과 같은 배열이 생성됨

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

06

희소행렬의 표현



6. 희소행렬의 표현

● 희소행렬

- 원소값이 0인 원소가 그렇지 않은 원소보다 상대적으로 많음

$$A = \begin{pmatrix} 0 & 20 & 0 & 0 & 9 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 78 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 67 & 0 & 0 & 0 & 0 \\ 0 & 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 91 & 0 & 0 & 44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 19 & 0 & 0 & 27 & 0 \end{pmatrix}$$

6. 희소행렬의 표현

● 희소행렬의 일반적 배열표현

$$A = \begin{bmatrix} 0 & 20 & 0 & 0 & 9 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 78 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 67 & 0 & 0 & 0 & 0 \\ 0 & 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 91 & 0 & 0 & 44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 19 & 0 & 0 & 27 & 0 \end{bmatrix}$$

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
[0]	0	20	0	0	9	0	0	11	0
[1]	0	0	0	0	0	0	0	0	0
[2]	78	0	0	0	0	0	0	0	0
[3]	0	0	0	0	67	0	0	0	0
[4]	0	31	0	0	0	0	0	0	0
[5]	0	0	0	91	0	0	44	0	0
[6]	0	0	0	0	0	0	0	0	0
[7]	0	0	0	0	19	0	0	27	0

6. 희소행렬의 표현

● 희소행렬의 효율적 배열표현

- 0인 원소는 저장하지 않고 0이 아닌 값만을 따로 모아서 저장
- 메모리 낭비를 막고 효율성 향상

$$A = \begin{bmatrix} 0 & 20 & 0 & 0 & 9 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 78 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 67 & 0 & 0 & 0 & 0 \\ 0 & 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 91 & 0 & 0 & 44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 19 & 0 & 0 & 27 & 0 \end{bmatrix}$$

	행	열	값
0	8	9	10
1	0	1	20
2	0	4	9
3	0	7	11
4	2	0	78
5	3	4	67
6	4	1	31
7	5	3	91
8	5	6	44
9	7	4	19
10	7	7	27

6. 희소행렬의 표현

○ 희소행렬의 효율적 배열표현

	행	열	값
0	8	9	10
1	0	1	20
2	0	4	9
3	0	7	11
4	2	0	78
5	3	4	67
6	4	1	31
7	5	3	91
8	5	6	44
9	7	4	19
10	7	7	27

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
[0]	0	20	0	0	9	0	0	11	0
[1]	0	0	0	0	0	0	0	0	0
[2]	78	0	0	0	0	0	0	0	0
[3]	0	0	0	0	67	0	0	0	0
[4]	0	31	0	0	0	0	0	0	0
[5]	0	0	0	91	0	0	44	0	0
[6]	0	0	0	0	0	0	0	0	0
[7]	0	0	0	0	19	0	0	27	0

정리하기

- 배열 : 인덱스와 원소값(<index, value>)의 쌍으로 구성된 집합이며, 정의된 각 인덱스는 그 인덱스와 관련된 값을 정의함
- 2차원 배열 : 원소값을 특정하기 위해 필요한 인덱스가 두 개인 배열
- 행렬의 행우선 저장 방식 : 하나의 행을 연속적으로 메모리에 할당하고, 그 다음 행을 메모리 영역에 할당하는 방법
- 행렬의 열우선 저장 방식 : 하나의 열을 연속적으로 메모리에 할당하고, 그 다음 열을 메모리 영역에 할당하는 방법
- 희소행렬(sparse matrix) : 원소값이 0인 원소가 그렇지 않은 원소보다 상대적으로 많은 행렬

다음시간 안내

03

스택

