제 3 강의 . 배열 자료구조

제 3 강 . 배열 자료구조

학습 목차

1. 배열의 개념
2. 구조체
3. 희소(Sparce) 행렬
4. 다차원 배열의 저장

1. 배열의 개념

- 리스트는 일상 생활에서 가장 많이 쓰이는 자료 형태이다.
  예) 학생의 명단, 은행 거래 고객 명단, 월별 판매액 등

- 배열(Array)은 컴퓨터 언어에서 리스트를 저장하는 데이터 타입이다.

- 리스트와 배열은 비슷한 의미지만 약간 다르다.

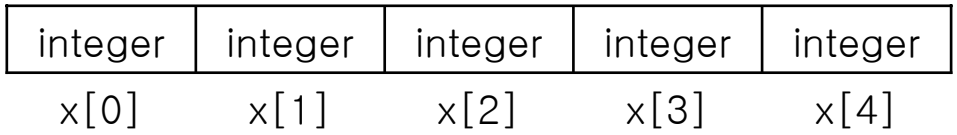
- C 언어로 표현된 리스트의 예

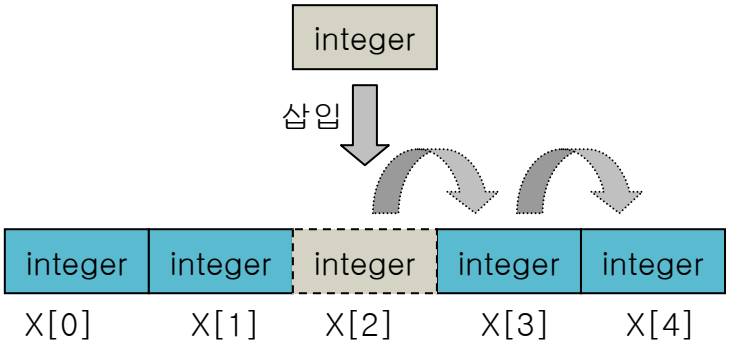
```
int student[100];  /* 학생의 번호 100 개 */
char name[100][];  /* 문자 배열의 배열 */
int sales[12];     /* 월 판매액 12개 */
```

❖ 배열에 관한 연산

- 새로운 배열의 **생성**

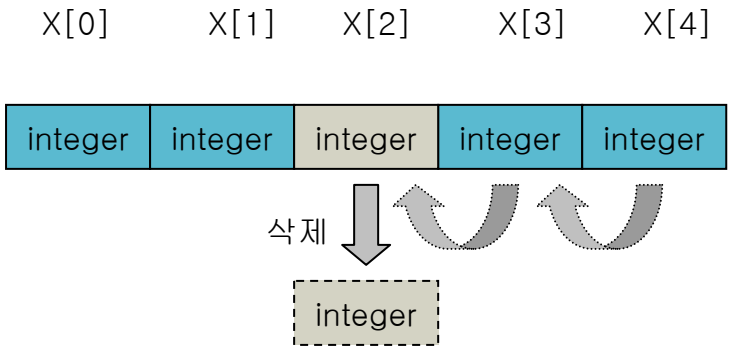
예) int x[5]; /* 언어에 따라 첨자는 0 혹은 1에서 시작 – C 언어는 0부터 시작한다. */





- 배열에 새로운 값 **삽입**
(나머지 값들을 한 칸씩 이동)

- 배열에 값을 **삭제**
(나머지 값을 한 칸씩 이동)



✓배열과 기억 장소

배열은 기억 장소에서 연속된 위치를 차지한다.

예) int list[5];

```
   /* sizeof() 함수는 데이터의 길이를 byte로 반환하는 함수 */
   /* C 언어에서 sizeof(int)의 값은 2 byte 이지만 시스템에
   따라서 4 byte 인 경우도 있다. */
```

변수	기억장소 주소
list[0]	α (<–시작 주소 값)
list[1]	α + sizeof(int)
list[2]	α + 2·sizeof(int)
list[3]	α + 3·sizeof(int)
list[4]	α + 4·sizeof(int)

❖포인터 타입이란?

포인터 타입은 데이터가 기억장소의 주소를 저장하는 타입이다.
사람이 사는 모든 집에 "주소"가 있고 주소에 가면
"거주자"들이 있는 것과 마찬가지입니다.
컴퓨터의 기억 장소도 "주소"가 있고 거주자와 같은 의미의 "내용"이 있다.

일상에서의 장소 -> 컴퓨터의 기억장소 -> 컴퓨터의 기억변수

장소(집)

"kim"

내용

"kim"

변수

서울시 하늘동 1번지

1020 번지

기억장소 주소

1020

포인터변수 –
주소를 저장하는 변수

✓포인터 타입이란?

```
int x;        /* 내용을 저장하는 변수 */
int *y;       /* 주소를 저장하는 변수 */
int z;

x = 10;       /* 내용 10을 저장 */
y = &x;       /* 변수 x의  주소를 저장 */
Z = *y;       /* 주소 y가 가리키는 곳의 내용을 저장 */
              /* z = x와 같은 의미가 된다. */

int a[20];    /* 배열 변수 a는 주소 값으로 처리한다. */
              /* 그 이유는 배열의 내용 전체를 이동할 경우
                 보다 주소 값을 알려주면 편하기 때문이다. */
```

✓ C 프로그램에서 *배열 list*[i] 의 의미

- C 언어에서 배열은 포인터 타입으로 선언된다.
- 포인터 타입은 기억 장소의 주소 값을 갖는 타입이다.

integer에 대한 포인터 선언

```
int *list1;
```

integer에 대한 포인터 선언으로 5개의 기억장소 확보

```
int list2[5];
```

(*list2*+*i*) 는 &*list2*[*i*] 와 같다.
*(*list2*+*i*) 는 *list2*[*i*] 와 같다.

- 배열에 대한 예제 프로그램

```
#define MAX_SIZE 100
float sum(float [], int);
float input[MAX_SIZE], answer;
int i;
```

전역변수 선언

```
void main(void) {
  for(i=0; i < MAX_SIZE; i++) input[i] = i;
  answer = sum(input, MAX_SIZE);
  printf("The sum is: %f\n", answer);
}
```

주 프로그램

```
float sum(float list[], int n) {
 int i;
 float tempsum = 0;
 for(i= 0; i < n; i++) tempsum += list[i];
 return tempsum;
}
```

함수 프로그램

- 배열에 대한 예제 프로그램 – 선언부분 설명

```
/* 이 부분에 선언된 변수들은 프로그램 전체에서 사용한다 */
   #define MAX_SIZE 100
/* 상수 MAX_SIZE 값을 100으로 선언한다 */
   float sum(float [], int);
/* 함수 sum()을 선언한다. 미리 선언을 해야 main() 에서 인식을
한다. */
   float input[MAX_SIZE], answer;
/* 실수형 배열변수 input[], answer 선언 */
   int i;
/* 정수형 변수 I 선언 */
```

- 배열에 대한 예제 프로그램 – main() 프로그램 설명

```
/* C 프로그램에서 main()은 반드시 1개 있어야한다. */
   int main( ) {
/* for 문은 for(초기문, 조건문, 실행문)반복문; 으로
        초기문장은 1번 실행하고 조건문이 false나 0이 될
        때까지 조건문->반복문->실행문을 반복실행한다. */
    for(i=0; i < MAX_SIZE; i++) input[i] = i;
/* for 문을 실행하여 input[I] = I 문장을 100번
     반복 실행한다. input[] 배열을 초기화시킨다.*/
    answer = sum(input, MAX_SIZE);
/* 함수 sum()을 호출한다. 인자는 배열 input과 MAX_SIZE
이다. */
     printf("The sum is: %f\n", answer);
/* 결과를 출력하는 문장이다. */
   }
```

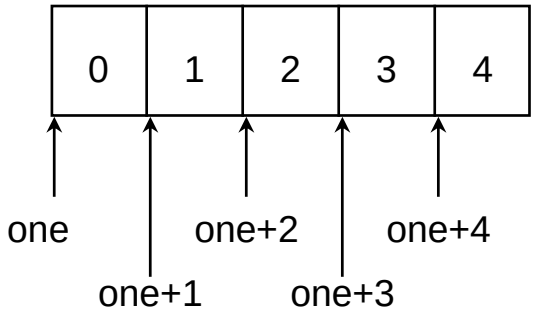
- 배열에 대한 예제 프로그램 – 함수 sum() 설명

```
 /* 함수는 인자를 받아서 결과를 반환한다. sum()의 인자는
list[]와 n이다. */
   float sum(float list[], int n) {
    int i;
/* 변수 i의 선언, 함수 안에서만 사용할 수 있다 */
    float tempsum = 0;
    for(i= 0; i < n; i++) tempsum += list[i];
/* tempsum += list[i]; 문장을 n번 실행한다 */
    return tempsum;
/* tempsum 값을 반환한다. */
   }
```

- 예제 프로그램

: 1차원 배열과 주소 값

```
int one[] = {0,1,2,3,4};
```



일차원 배열을 주소 값을 이용하여 참조하는 방법

- *i* 번째 요소의 주소

  ptr + *i*

- *i* 번째 요소의 값

  *(ptr + i)

배열의 원소의 값과 원소가 저장된 기억 장소의 값을 출력하는 함수를 작성해보자

```
void print_array(int *ptr, int rows) {
    printf("Address Contents\n");
    for(i=0;i<rows;i++)
        printf("%8u%5d\n",ptr+i,*(ptr+i));
    printf("\n");
}
```

주소 값	내 용
1228	0
1230	1
1232	2
1234	3
1236	4

2 구조체(Structures)

- **구조체는 데이터를 표현하는 단위이다.**

  예) 학생의 명단, 은행 거래 고객 명단, 월별 판매액 등

  그러나 **학생의 명단은** 학생의 학번과 이름, 주소
  고객 명단은 고객의 이름, 주소, 계좌번호 등으로
  여러 개의 작은 값(필드)들이 합해서 한 개의 데이터를 구성한다.
  이러한 데이터 타입은 복합 데이터에 해당된다. 복합 데이터를
  1개의 변수로 표현하는 방법은 구조체(structure)를 사용한다.

- **구조체는 각 데이터가 타입과 이름을 갖는다.**

❖ 구조체의 예

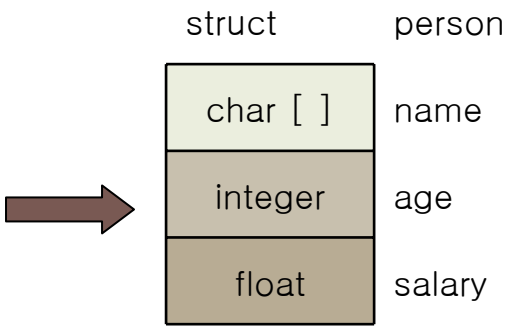
"사람(person)" 데이터 타입

어떤 사람의 신상에 관해 컴퓨터에 저장할 때 3개의 필드로 구성된다고 하자.

1) **이름**(*name*)은 **문자** 배열로 구성
2) **나이**(age)는 **정수**(integer) 변수로 구성
3) **급여**(salary)는 **실수**(float) 변수로 구성

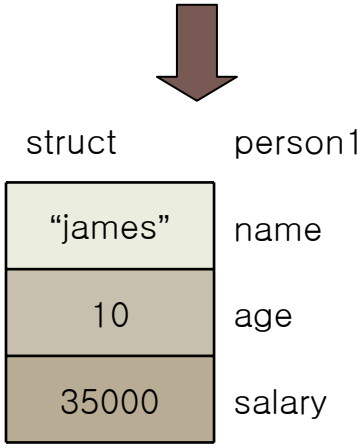
C 언어로 아래와 같이 선언이 된다.

```
struct {
    char name[10];
    int age;
    float salary;
} person;
```



✓ 구조체에 데이터 저장

```
struct person person1;
strcpy(person1.name, "james");
/* strcpy() 함수는 문자열을 복사하는 함수 */
person1.age = 10;
person1.salary = 35000;
```

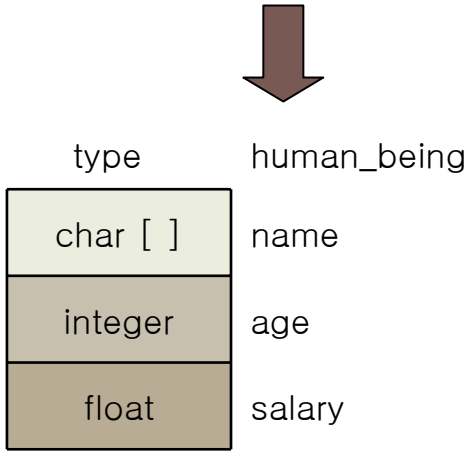


✓ typedef statement - - 구조체 타입을 선언한다.

```
typedef struct
human_being {
    char name[10];
    int age;
    float salary;
};
```

혹은

```
typedef struct {
    char name[10];
    int age;
    float salary;
} human_being;
```



❖ struct 타입

✓치환문 (assignment) - - struct 전체를 치환한다

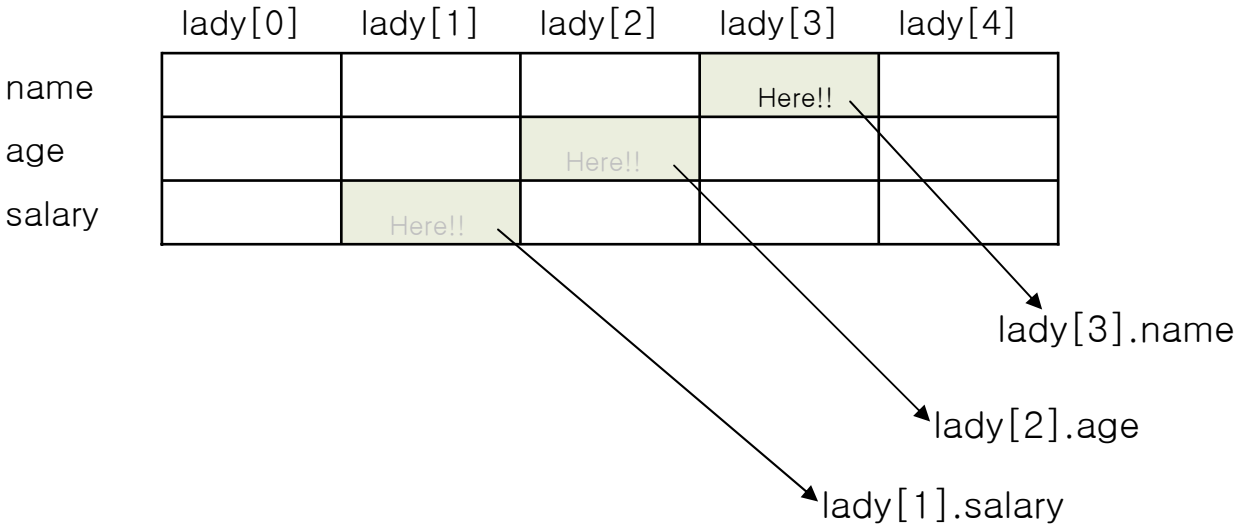
```
human_being person1, person2;
…
person1 = person2;
```

- struct 필드 단위로 치환한다.

```
strcpy(person1.name, person2.name);
person1.age=person2.age;
person1.salary=person2.salary;
```

❖ **Struct의 배열 -** 구조체의 배열은 다음과 같이 선언된다.

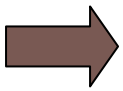
```
struct {
    char name[10];
    int age;
    float salary;
} person;
struct person lady[5];
```



❖ 자기참조 구조체(Self-referential structures)

구조체의 필드 중 하나가 구조체에 대한 포인터를 나타내는 타입이다.
앞으로 배울 연결리스트, 트리, 그래프의 구현에 쓰이는 방법이다.
다른 구조체를 연결하고 있기 때문에
동적기억장소(dynamic storage management) 구현에 쓰이는 방법이다

```
typedef struct list {
  char data;
  list *link;
};
```



type	struct list
char	data
list *	link

Link 필드의 값

- 기억장소에 대한 포인터로 *list* 타입을 가리킨다.
- 아무것도 가리키지 않으면 0(NULL)값을 갖는다.

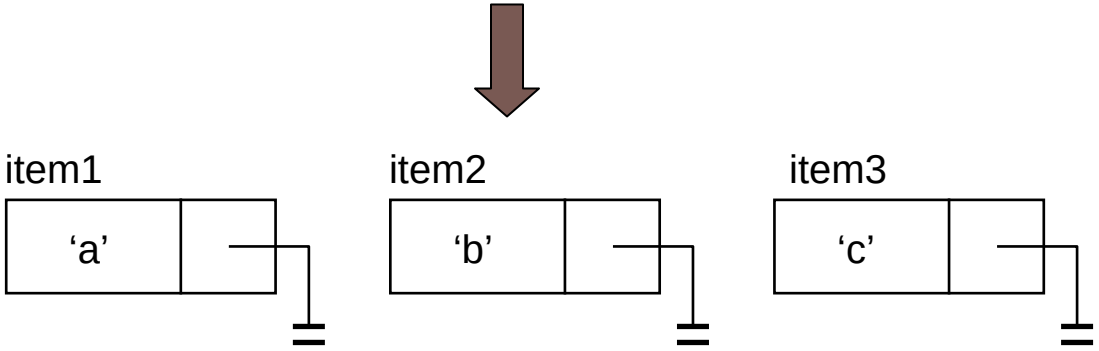
```
/* UNIX 시스템에서 실제 프로그램 코드 */
struct node {
        int data;
        struct node * link;
};
typedef struct node list_node;
typedef list_node * list_ptr;
```

< C 자료구조 입문 >

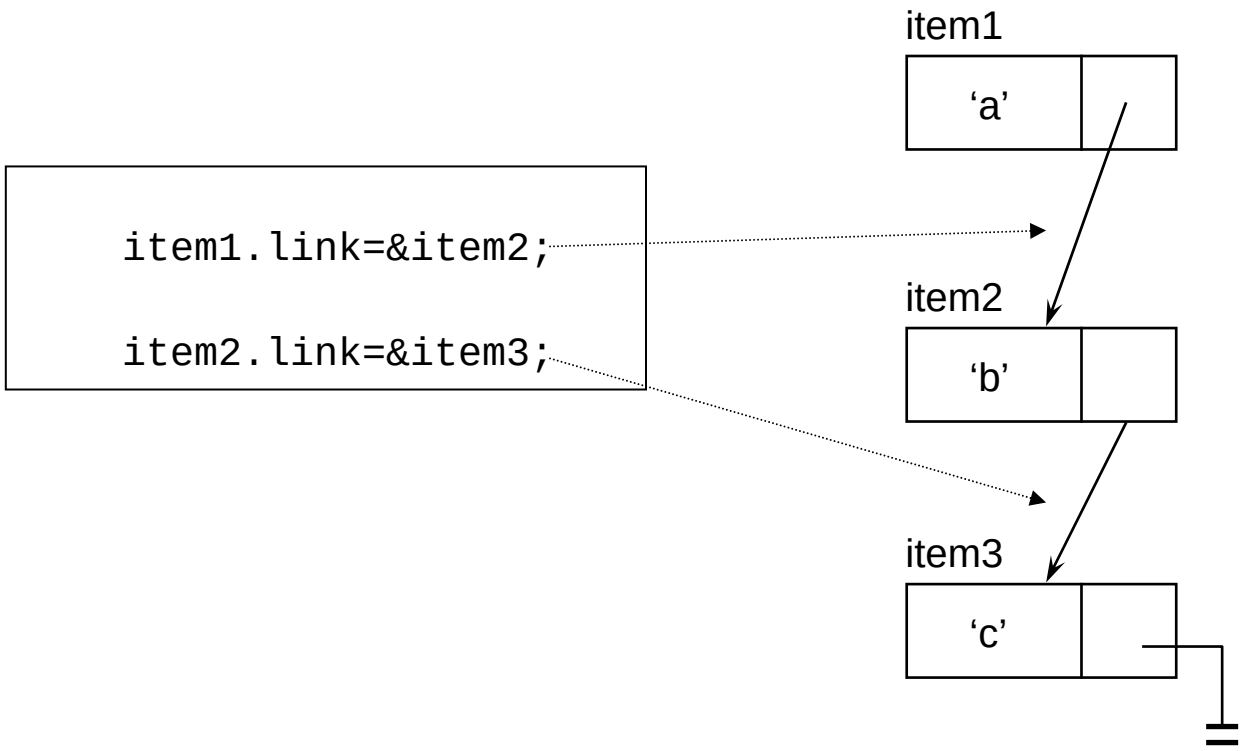
✓자기참조 구조체 노드 선언

```
List_node item1, item2, item3;

item1.data = ‘a’;
item2.data = ‘b’;
item3.data = ‘c’;
item1.link = item2.link = item3.link = NULL;
```

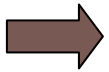


✓자기참조 구조체의 연결



3. 희소(Sparce) 행렬 – 배열의 응용 예

예) 행렬을 프로그램에 저장하려면 다음과 같이 선언을 하게 된다.

`int A[6,6]` 

	열 0	열 1	열 2	열 3	열 4	열 5
행 0	15	0	0	22	0	15
행 1	0	11	3	0	0	0
행 2	0	0	0	-6	0	0
행 3	0	0	0	0	0	0
행 4	91	0	0	0	0	0
행 5	0	0	28	0	0	0

행렬을 표현하는 데 필요한 기억 장소의 크기는

- *m* **n* (m 행, n 열)
- 공간복잡도(space complexity)

  S(*m*, *n*) = *m* * *n*

만약 m과 n의 값이 크다면(m=1000, n=10000),
행렬을 표현하는 데 필요한 기억 장소의 크기는?
 - 1000 * 10000 = 10,000,000
 - 기억 공간 활용의 비효율성

개선된 방법의 사용
 - 큰 행렬은 값이 0인 항이 많다.
  0이 아닌 항을 <행, 열, 값> 로 저장한다.
 - 공간복잡도(space complexity)
    S(m, n) = 3 * (0이 아닌 항의 수) < m * n

	col 0	col 1	col 2
row 0	-27	3	4
row 1	6	82	-2
row 2	109	-64	11
row 3	12	8	9
row 4	48	27	47

C 언어로 희소 행렬의 자료 구조를 선언하면 다음과 같다.

```
#define MAX_TERMS 101
typedef struct {
        int col;
        int row;
        int value;
} term;
term a[MAX_TERMS];
```

- *a*[0].*row*: 행의 수
- *a*[0].*col*: 열의 수
- *a*[0].*value*: 0이 아닌 항의 수

희소(sparse) 행렬의 표현된 모습

	row	col	value
a[0]	6	6	8
[1]	0	0	15
[2]	0	3	22
[3]	0	5	15
[4]	1	1	11
[5]	1	2	3
[6]	2	3	-6
[7]	4	0	91
[8]	5	2	28

- 공간 복잡도(space complexity)

  $S(n, m) = 3 * t$ where

  t: 0이 아닌 항의 수

- 공간복잡도는 행과 열의 수에 무관하며 0이 아닌 값과 관계있다.

4. 다차원 배열의 저장

- 1차원 배열은 기억장소에 순서대로 저장한다.
- N차원 배열이 어떻게 1차원의 물리적인 기억장소에 저장이 될까?
- N차원 배열의 임의의 원소는 기억장소의 어디에 저장되어 있는가?

```
int a[upper0][upper1]···[uppern-1] /* 선언 */
```

- 배열의 전체 원소의 수

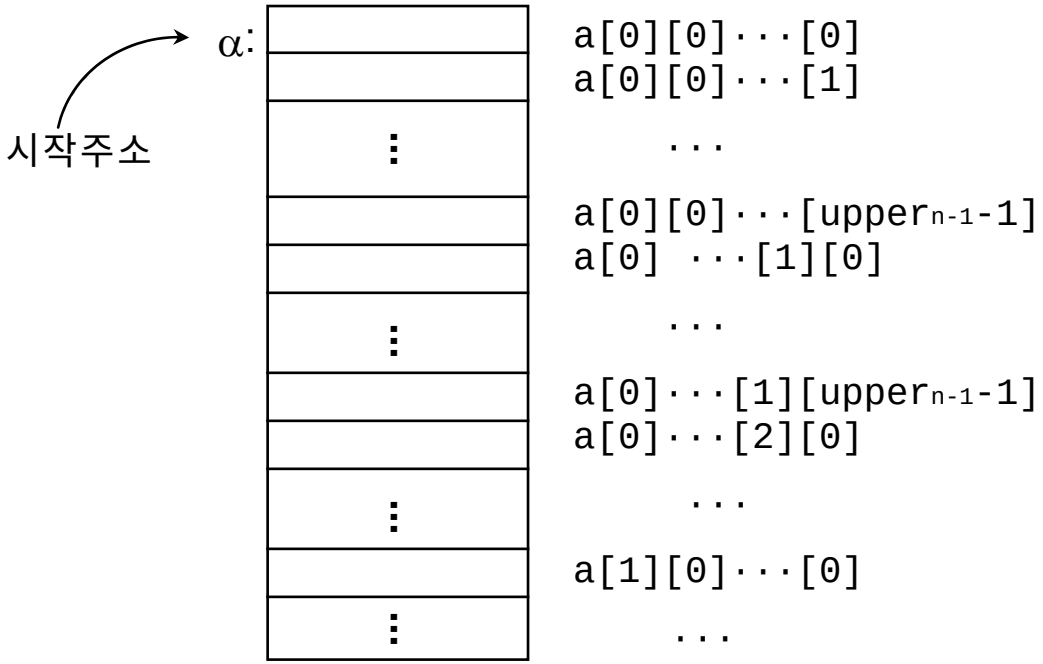
$$\prod_{i=0}^{n-1} upper_i$$

예) int a[6][7][8]으로 선언될 때
- 전체 원소의 수 = 6 * 7 * 8 = 336 (units)

❖ 다차원 배열의 저장 방법

- 행 우선저장(row major order) – 1행, 2행, 3행 … 순으로 저장
- 열 우선저장(column major order) – 1열, 2열, 3열 … 순으로 저장

예) 행 우선저장(rows major order)으로 저장했을 때



a[2][5]···[3] 항은 기억 장소의 어디에 있을까?

- 시작주소 + 시작부터의 위치
- α : 시작주소라고 가정

✓ **1 차원 배열** a[u_0]에서 a[i]은 ?

a[0]	:	α
a[1]	:	α + 1
⋮		⋮
a[u_0-1]	:	α + (u_0 - 1)

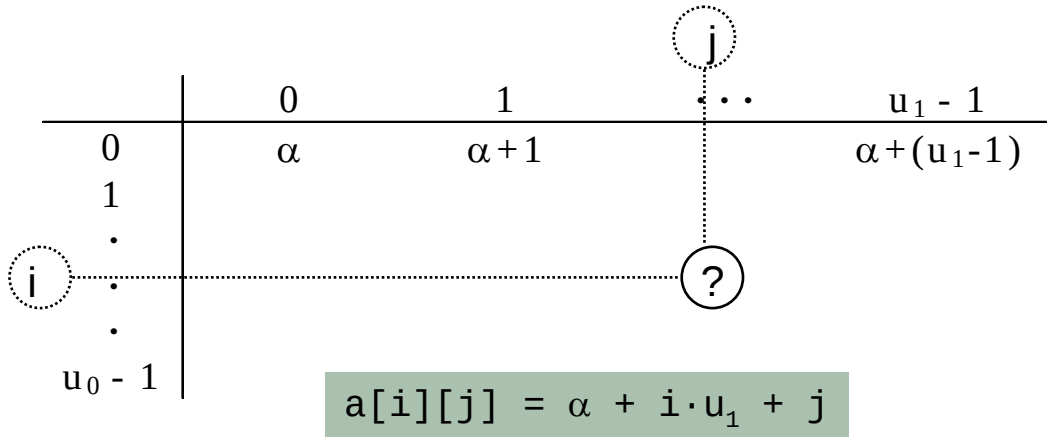
a[i] = α + i

예) int x[8]로 선언되어 있을 때, x[0]의 주소가 100번지이면,

int[5]의 주소는(integer 형이 기억장소를 2 byte 차지한다고 가정)?

-> int[5]의 주소 = 100 + 5 X 2 = 110 번지

✓ **2 차원 배열에서** a[u0][u1]는 ?



a[i][j] = α + i·u1 + j

예) int x[4][5]로 선언되어 있을 때, x[0][0]의 주소가 100번지이면, int[2][3]의 주소는?

(행우선 저장, integer 형이 기억장소를 2 byte 차지한다고 가정)

-> int[2][3]의 주소 = 100 + 2 (2 X 5 + 3) = 126 번지

✓ **3 차원 배열에서 a[u_0][u_1][u_2]는 ?**

$$a[i][j][k] = \alpha + i \cdot u_1 \cdot u_2 + j \cdot u_2 + k$$
$$= \alpha + u_2[i \cdot u_1 \cdot + j] + k$$

예) int x[3][4][5]로 선언되어 있을 때, x[0][0][0]의 주소가 100번지이면, int[2][3][4]의 주소는?
(행우선 저장, integer 형이 기억장소를 2 byte 차지한다고 가정)

-> int[2][3][4]의 주소 = 100 + 2 (2 X 4 X 5 + 3 X 5 + 4)

예) int x[3][4][5]로 선언되어 있을 때, x[0][0][0]의 주소가 100번지이면, int[2][3][4]의 주소는?
(열우선 저장, integer 형이 기억장소를 2 byte 차지한다고 가정)

-> int[2][3][4]의 주소 = 100 + 2 (2 + 3 X 3 + 4 X 3 X 4)

✓ **N 차원 배열의 경우 a[u_0][u_1]···[u_{n-1}]는 ?**

$$a[i_0][i_1]\cdots[i_{n-1}]$$
$$= \alpha + \sum i_j \cdot a_j , \quad a_j = \prod u_k, \ 0 < j < n-1$$
$$a_{n-1} = 1$$

Review

⦿ 리스트는 가장 많이 사용되는 자료의 형태이다.

배열은 프로그램 언어에서 데이터 타입이다.

리스트를 배열을 이용하면 쉽게 표현할 수 있다.

다차원 배열을 기억장소에 표현하는 방법은 행우선과 열우선 방법이 있다.

⦿ 구조체는 여러 개의 다른 데이터를 한 개의 데이터로 묶는데 사용한다.

구조체의 특별한 형태로 자기 참조 구조체가 있다.

(일상생활 데이터)	(컴퓨터 데이터)
이름, 나이, 주민등록번호	데이터들
나이의 모임, 이름의 모임	배열
한 사람에 관한 데이터(이름, 나이, ...)	구조체
여러 사람의 데이터	구조체의 배열

⦿ 포인터 타입(주소 값)

- 데이터가 저장된 기억장소 주소를 다루는 타입을 포인터 타입이라 한다.
- 선언은 *, 변수의 주소 값은 &, 주소 값의 참조는 * 연산자를 이용한다.
- 배열 변수는 자동으로 포인터 형으로 선언된다.