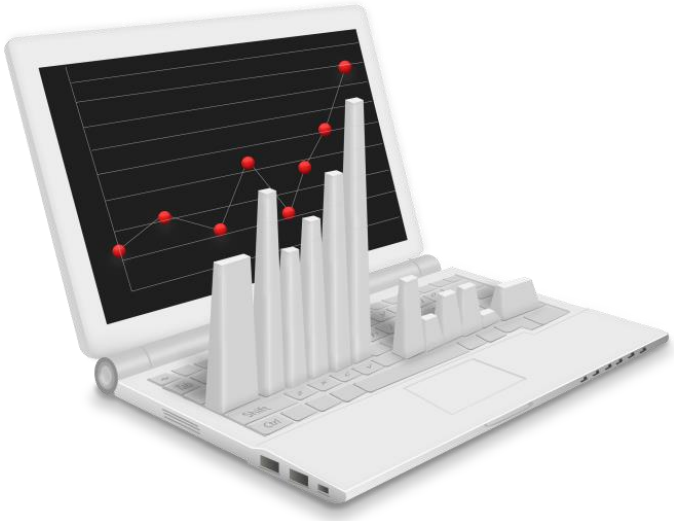
프로그래밍 언어론

배열

컴퓨터공학과

조은선



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

배열

학습목표

- 배열에 대해 전반적으로 알아본다.

학습내용

- 배열의 구성 요소
- 배열의 종류
- 배열의 연산
- 배열의 구현

목 차

- 들어 가기
- 학습하기
  - 배 열
  - 배열 index
  - 배열의 종류
  - 배열 초기화
  - 배열 연산
  - 정방형/비정방형 배열
  - 슬라이스
  - 배열의 구현
- 평가하기
- 정리하기

알고가기

데이터를 다음과 같이 저장하고자 한다.

'a'	'b'	'c'	'd'	'e'

C 프로그램 표현은 다음과 같다.

```
char a[5];
     a[0]='a'; a[1]='b'; a[2]='c'; a[3]='d'; a[4]='e';
```

이를 위한 프로그램 표현과 의미가 올바르게 설명하지 않은 것은?

① char는 각 칸에 char 타입의 변수를 저장할 수 있음을 의미한다.
② A[5]는 연속적인 5칸의 char 타입 메모리공간을 준비하는 것을 의미한다.
③ 배열이름은 배열의 첫째 칸을 나타내므로 a[0]='a'; 대신 a='a'; 과 동일하다.
④'d'를 저장할 때 그림과 같은 넷째 칸을 나타내기 위해 a[3]을 사용하였다.

| 배열

배열

- 동일한 데이터 타입을 가지는 값들의 모임
- 각 원소는 index에 의해 인식됨
  - 이 때 index는 첫 번째 원소를 기준으로 한 상대적인 위치를 의미
- 설계 시 고려할 점
  - index에 허용된 타입은?
  - index 범위 검사를 할 것인지? 범위는 언제 정해지는지?
  - 배열에 메모리를 언제 할당할 것인지?
  - 배열 개체의 초기화를 허용할 것인지?
  - 쪼개는 것을 허용할 것인지?

배열 Index

배열 index(첨자)

인덱싱 (indexing) : 배열이름과 index를 가지고 특정 원소에 대응시키는 것

- FORTRAN, PL/I, Ada 은 ( ) , 나머지는 [ ] 사용

index의 데이터 타입

- 정수 (FORTRAN, C, Java)
- `integer, boolean, char, enum` (Pascal, Ada)

index 개수 (multi-dimensional 배열)

- 3개로 한정(FORTRAN I), 7개로 한정(FORTRAN77)
- 그러나, 그 외에는 제한 없음

Static 배열, Fixed Stack-dynamic 배열

Static 배열

- FOTRAN 77
  - index의 최대 범위가 정적으로 수행 전에 결정됨
  - 배열의 메모리 할당도 정적으로

장점

실행 효율성

Fixed Stack-dynamic 배열

- C의 local 배열: `int a[10]`
  - index의 최대 범위는 정적으로 결정되나
  - 배열의 메모리 할당은 실행중에 (stack에 ...)

장점

공간 효율성

Stack-dynamic 배열, Fixed Heap-dynamic 배열

Stack-dynamic 배열

Ada의 declare block

- index 범위, 메모리 공간은 수행 중 동적으로 결정됨 (stack에)
- 그러나 시작할 때 한 번 결정되면 그 변수 생명주기가 끝날 때까지 변할 수 없음

장점

융통성
(배열이 사용되기 직전에야 index 범위, 배열 크기 등을 알 수 있을 때 유용)

```
GET (LIST_LEN);
Declare   // 실행이 이 블록에 이를 때, 배열 할당
  LIST: array (1 .. LIST_LEN) of INTEGER;
```

Fixed Heap-dynamic 배열

- Stack-dynamic 배열과 동일하나 메모리 위치가 stack이 아닌 heap임
- C의 `malloc/free`, C++의 `new/delete` 로 조작되는 배열
- Java의 모든 배열, C#도 이 유형의 배열 제공

Heap-dynamic 배열

Heap-dynamic 배열

- index 범위, 메모리 공간이 수행 중 동적으로 결정되며
- 수행하다가 변할 수 있음...

C#, Java의 ArrayList

예

```
ArrayList intList = new ArrayList();// 빈 배열 생성
intArray.Add(nextone);                // 원소 추가
```

Perl, JavaScript, PHP 배열

예

```
@list = (1,2,4,7,10);
push(@list, 13, 17); // @list 배열에 두 원소 추가
```

배열 초기화

주로 메모리에 자리잡은 순서대로 초기화됨

PASCAL은 별도 지원이 없음

C

```
int stuff [] = {2, 4, 6, 8};
char name [] = "Tom";
char * names [] = {"Tom", "Jerry"};
```

Java

```
String [] names = {"Tom", "Jerry"};
```

Ada

```
Bunch : array (1..5) of INTEGER :=
          (1 => 3, 3 => 4, others => 0);
```

배열 연산

ADA

- 배열 assignment (:=), 이어붙이기, 같은지 비교

FORTRAN

- 행렬, 벡터 연산 라이브러리가 풍부함

APL

- 가장 강력함: 배열에 대한 4칙연산, 벡터의 원소 역순,
- 행렬의 행/ 열의 역순, 전치행렬, 역행렬

```
A + B              // 벡터, 또는 행렬
A X B              // A, B의 곱
A + . X B          // A, B의 내적 합
```

Rectangle/Jagged(장방형/비장방형) 배열

Rectangle 배열 (장방형 배열)

- **모든 행이 동일 개수의 원소들을 포함, 모든 열이 동일 개수의 원소들을 포함**

참조방법

```
myArray[3,7]          // FORTRAN, Ada, C# ...
```

Jagged 배열 (비장방형 배열)

- **행, 열의 크기가 동일하지 않은 multi-dimensional 배열**

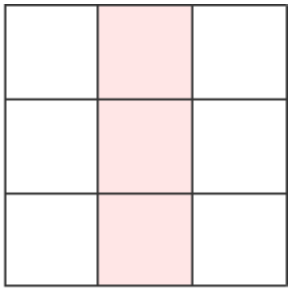
- 행렬이 3개 행을 포함하고, 첫번째 행은 5개 원소,
  두번째 행은 7개 원소, 세번째 행은 12원소를 포함할 때
- multi-dimensional 배열이 실제로 "배열들로 구성된 배열"로 보일 때 가능

참조방법

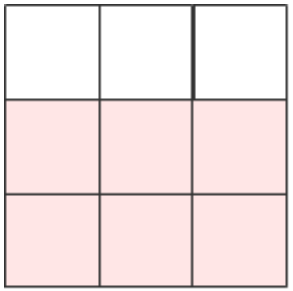
```
myArray [3][7]  // an array of arrays in C#, Java
```

Slice(슬라이스)

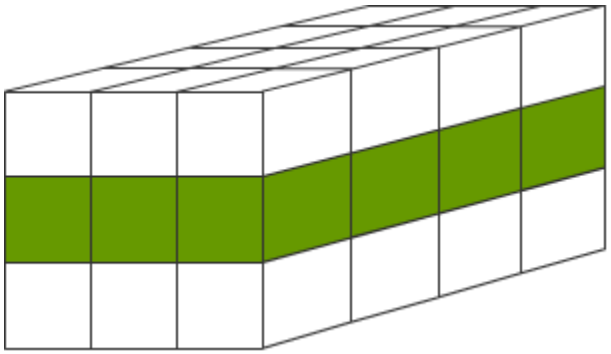
배열의 Slice = 배열의 부분을 한 단위 (또 다른 작은 배열) 로 보는 것



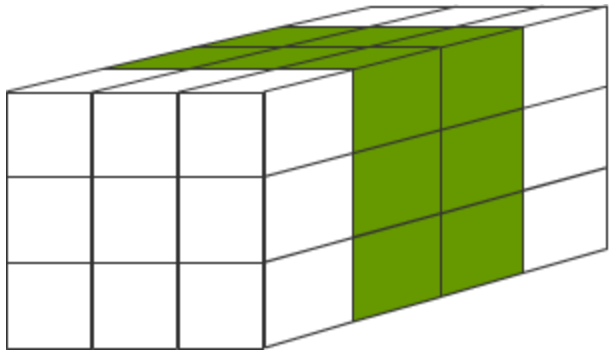
Mat(1:3, 2)



Mat(2:3, 1:3)



Cube(2, 1:3, 1:4)



Cube(1:3, 1:3, 2:3)

Slice 예

FORTRAN 95

예

```
integer vector(1:10), Mat(1:3, 1:3),
Cube(1:3, 1:3, 1:4)
…
Vector(3:6)
Mat(1:3, 2)
Mat(3, 1:3)
Cube(1:3, 1:3, 2)
Vector(2:10:2)      // 2, 4, 6, 8, 10번째 원소들로 구성
```

예

Ada: 단지 one-dimensional 배열에 대해서만 슬라이스 연산

```
LIST(4 .. 10)
```

배열의 구현 방법

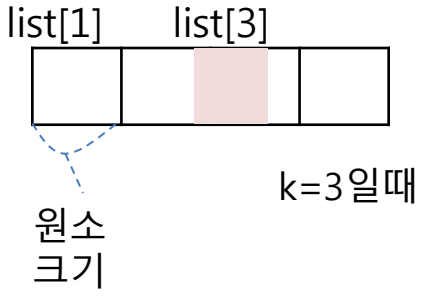
배열 원소의 접근에 대한 코드를 컴파일러가 생성

- 접근 함수를 통해서 index 식을 배열의 한 주소에 사상

주소(list[k]) = 주소(list[1]) + (k-1) * 원소크기

= (주소(list[1]) – 원소크기) + (k*원소크기)

상수 부분 변수 부분



컴파일시간 Descriptor(서술자)

- 접근 함수 구성, index 범위 검사 (index범위가 동적) 에 필요한 정보 포함

One-dimensional 배열의
컴파일시간 descriptor

Multi-dimensional 배열의
컴파일 시간 descriptor

Multi-dimensional(다차원) 배열의 저장

Multi-dimensional 배열의 저장

- **Column-major order(열-우선 순서): FORTRAN**
- **Row-major order(행-우선 순서): 다른 언어 기본 데이터 타입**

예

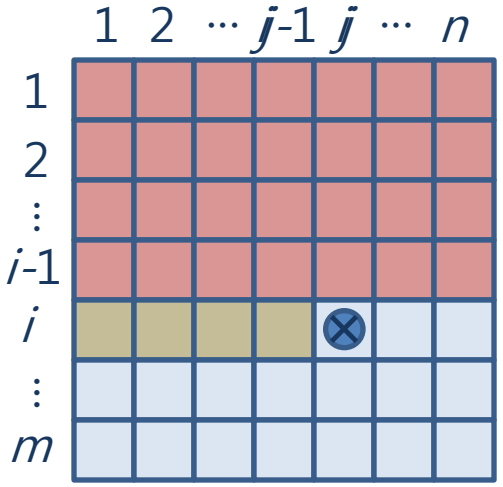
3	5	7
6	2	5
1	3	8

- 3 5 7 6 2 5 1 3 8 .. (행-우선, 즉 Row-major 순서)
- 3 6 1 5 2 3 7 5 8 .. (열-우선, 즉 Column-major 순서)

Multi-dimensional 배열의 주소

Multi-dimensional 배열주소 (행 우선, 즉 row-major 관점에서)

- a[i,j] = 주소(a[1,1]) +
  (i번째 행보다 위에 위치한 행의 개수* 행의 크기
  + j번째 열보다 왼쪽에 위치한 원소의 개수)
  * 원소_크기
  = 주소(a[1,1]) + ((i-1) *n + (j-1))*원소크기
  = 주소(a[1,1]) – (n+1)*원소크기 + (i*n+j)*원소크기
  상수부분 변수부분



Associative 배열

Associative 배열 (연상배열)

- 숫자가 아닌 index 를 가질 수가 있다.

  예) `a["Perry"], list["sing"]`

  이 때, index는 key라고 부른다.

- Perl의 예

```
%salaries = ("Gary"=>75000, "Perry"=>57000, "Mary"=>55750);
$salaries{"Perry"} = 58850;
delete $salaries{"Gary"};
```

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 3문제가 있습니다.

START

평가하기 1

1. 원소의 크기가 4byte인 배열의 첫번째 원소의 주소가 1000 번지라고 하자. 5번째 원소의 주소는?

① 1004

② 1005

③ 1016

④ 1020

확인

평가하기 2

2. index의 최대범위는 정적으로 결정되나 배열의 메모리 할당은 실행 중에 결정되는 배열을 고르면?

① Fixed Stack-dynamic 배열

② Stack-dynamic 배열

③ Fixed Heap-dynamic배열

④ Heap-dynamic 배열

확인

평가하기 3

3. 다음 배열을 row major order(행-우선)으로 저장했을 때 순서는?

1	6	8
9	2	3
5	7	2

① 168923572

② 195627832

③ 572923168

④ 832627195

확인

정리하기

배열의 구성 요소 및 개념

index, slice(배열의 부분을 한 단위 또는 작은 배열로 보는 것), multi-dimensional 배열(인덱스의 갯수가 여러 개) 등의 개념을 명확히 이해하고 사용하자

배열의 종류

배열의 생명 주기나 할당 장소, 크기 지정 시점 등에 따라 분류:

static 배열, fixed stack-dynamic 배열, stack-dynamic, fixed heap-dynamic 배열, heap-dynamic 배열

배열의 구현

- Descriptor : 접근 함수 구성, index 범위 검사 등에 필요한 정보
- Column-major order는 한열씩 저장하고, row-major order는 한 행씩 저장