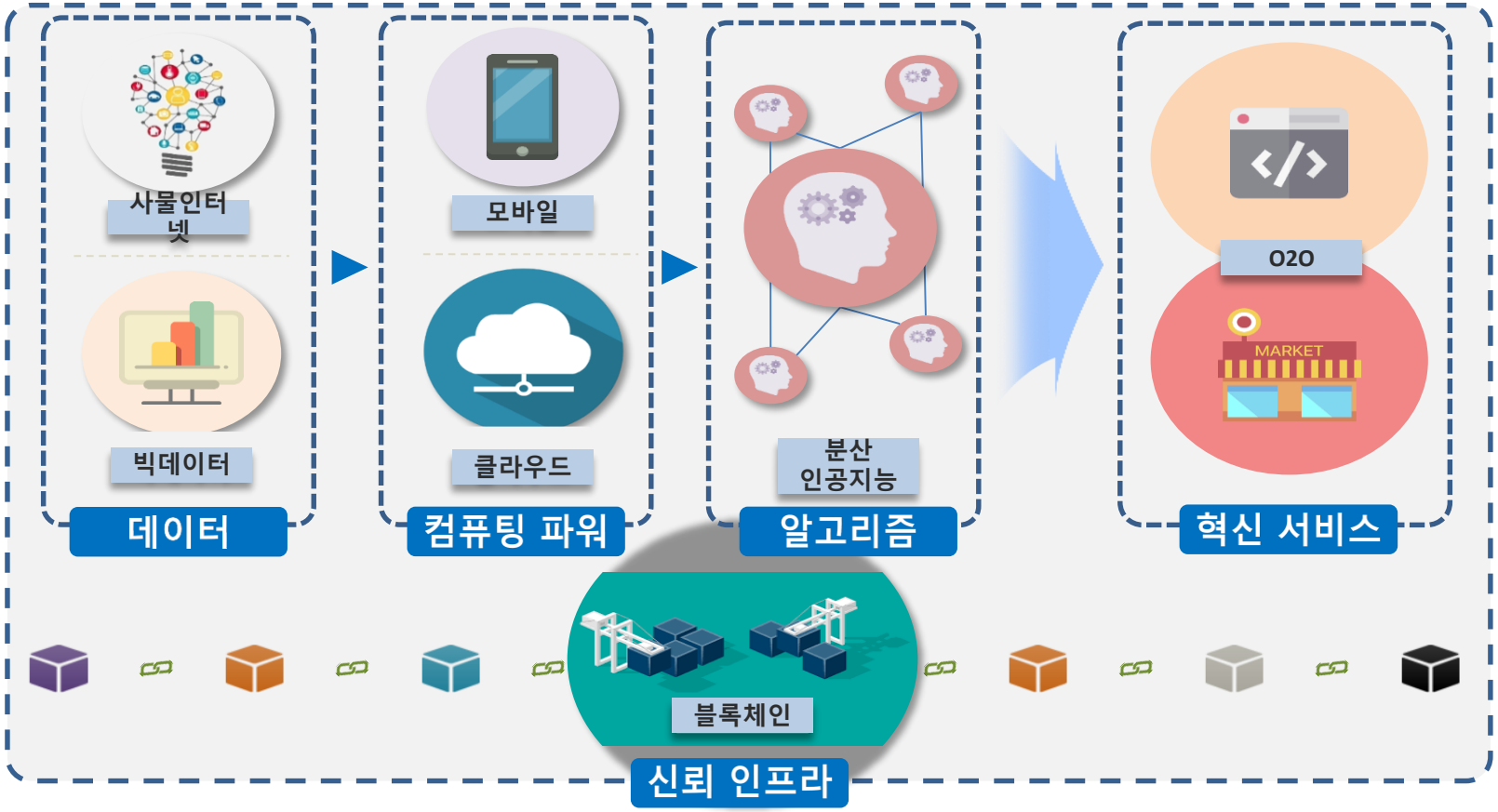
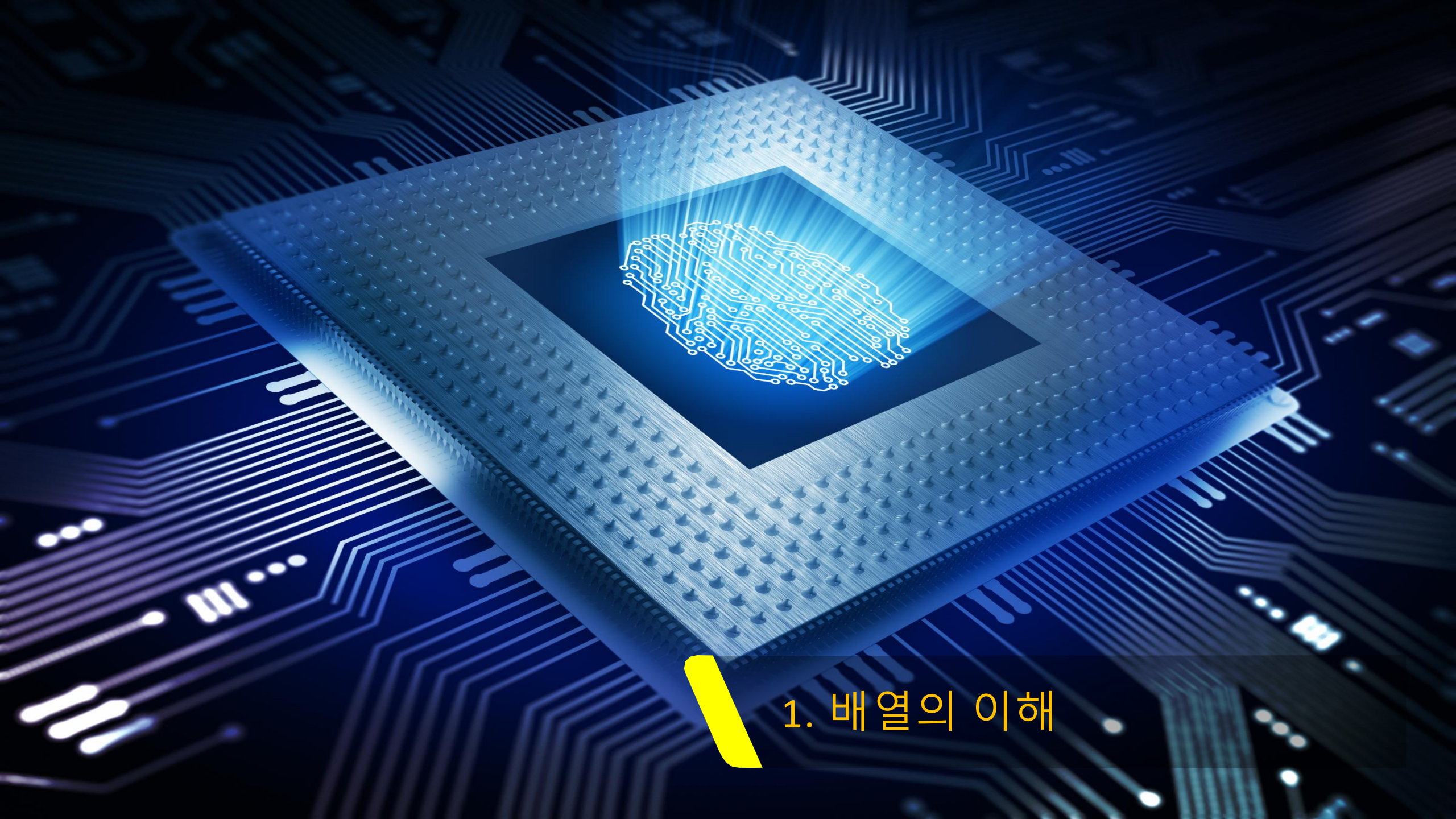


Hanyang Univ.

자료구조론 4주차

강 진 영

CONTENTS

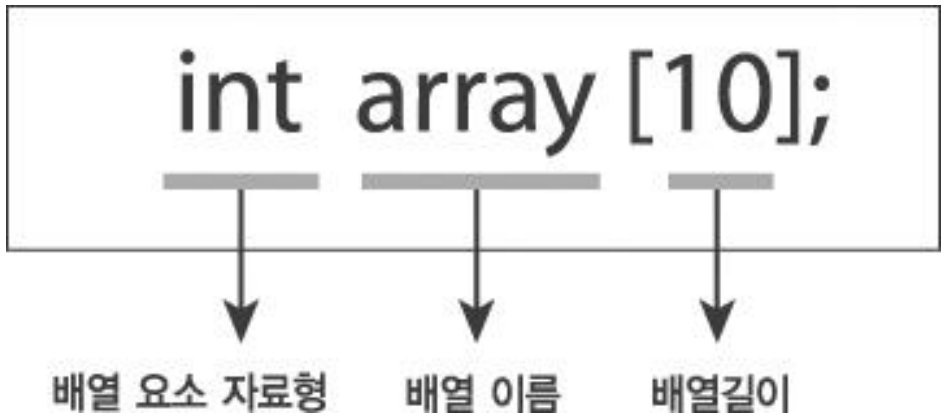


1. 배열의 이해

Contents

배열이란

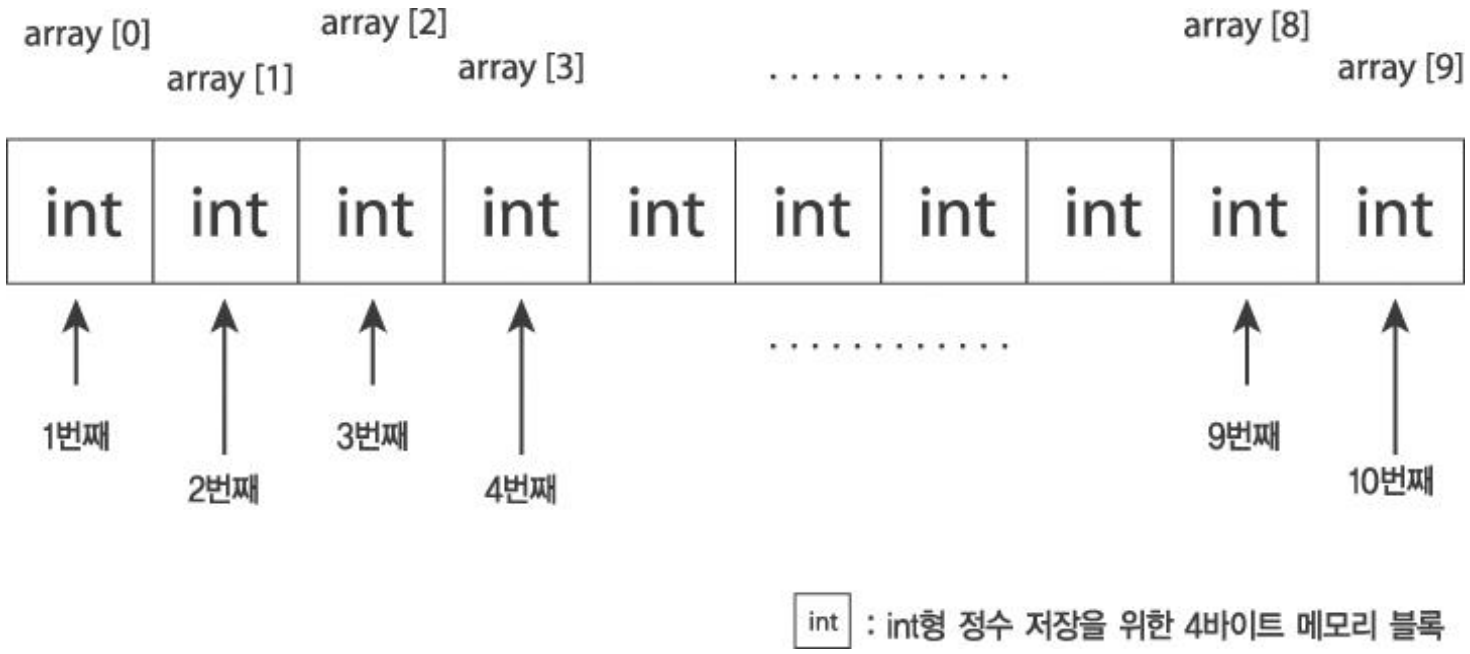
- **배열의 3요소**
  - 배열 길이 : 배열을 구성하는 변수의 개수(반드시 상수를 사용)
  - 배열 요소 자료형 : 배열을 구성하는 변수의 자료형
  - 배열에 접근할 때 사용되는 이름



Contents

배열이란

- **1차원 배열의 접근**
  - 배열 요소의 위치를 표현 : 인덱스(index)
  - 인덱스는 0에서부터 시작

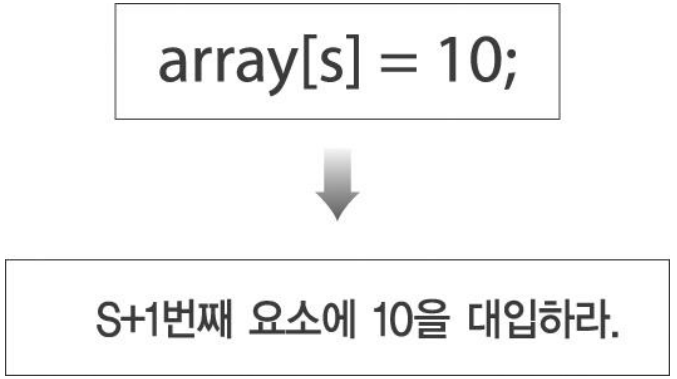


Contents

배열이란

- 배열 선언과 접근의 예

```
int main(void)
{
    int array[10];      // 배열 선언
    array[0]=10;        // 첫 번째 요소 접근
    array[1]=20;        // 두 번째 요소 접근
    array[2]=30;        // 세 번째 요소 접근
    . . . . .
    return 0;
}
```

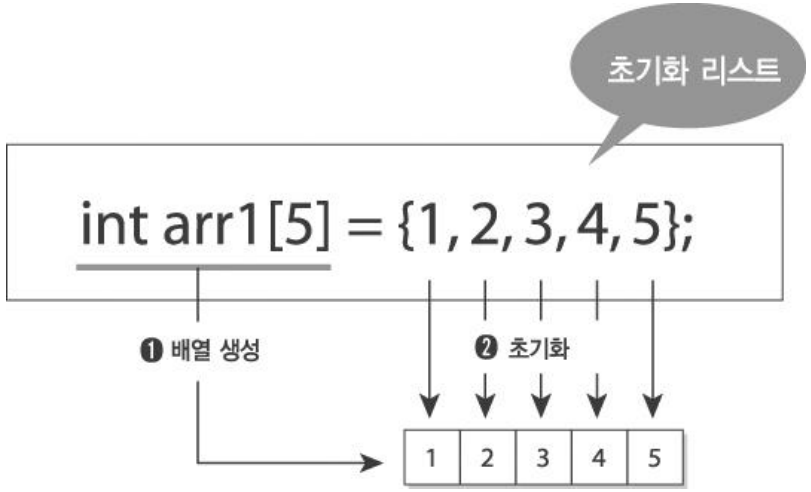


Contents

배열이란

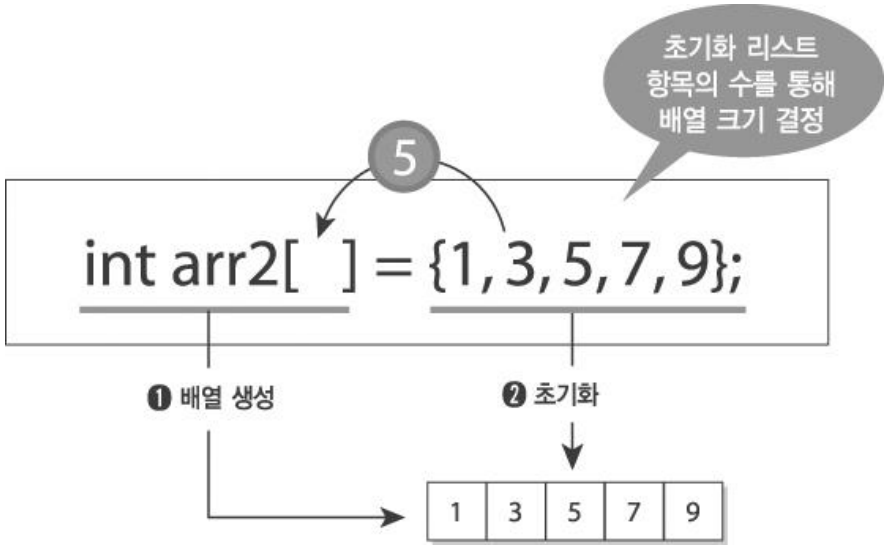
- 선언과 동시에 초기화

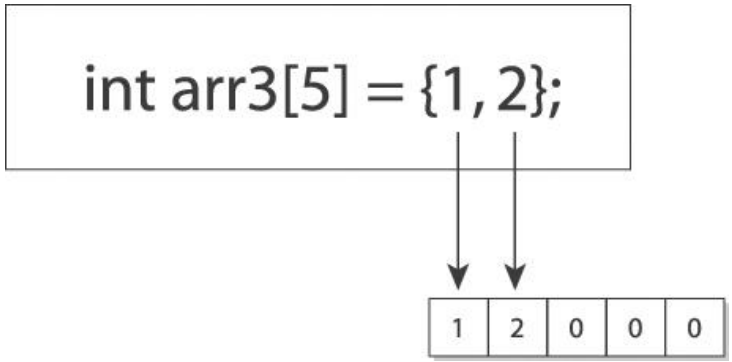
```
int main(void)
{
   int arr1[5]={1, 2, 3, 4, 5};
   int arr2[ ]={1, 3, 5, 7, 9};
   int arr3[5]={1, 2}
}
```



Contents

배열이란





Contents

배열 기반 문자열 변수

- **문자열 상수**
  - ➢ 문자열이면서 상수적인 특징

```
printf("Hello World! ₩n");
```

- **문자열 변수**
  - ➢ 문자열이면서 변수적인 특징

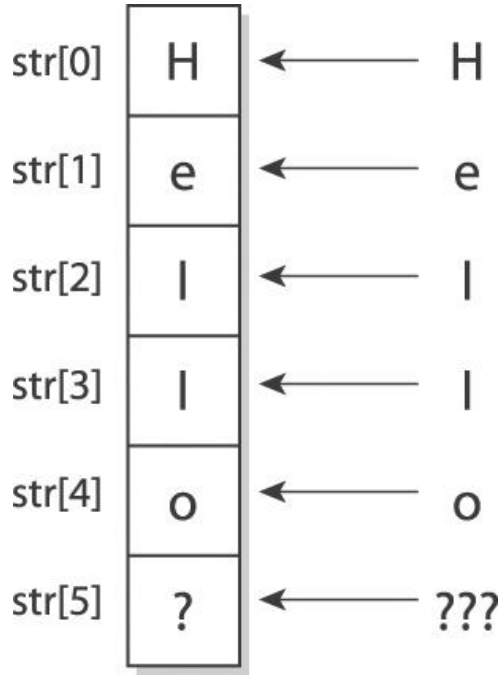
```
char str1[5]="Good";
char str2[]="morning";
```

Contents

배열 기반 문자열 변수

- 문자열의 특징
  - 문자열은 널(null)문자를 끝에 반드시 포함
  - 널(null)문자 : '₩0'(아스키 코드 값 : 0)

```
int main(void)
{
    char str[6]="Hello";
    printf("Hello");
    . . . . .
```

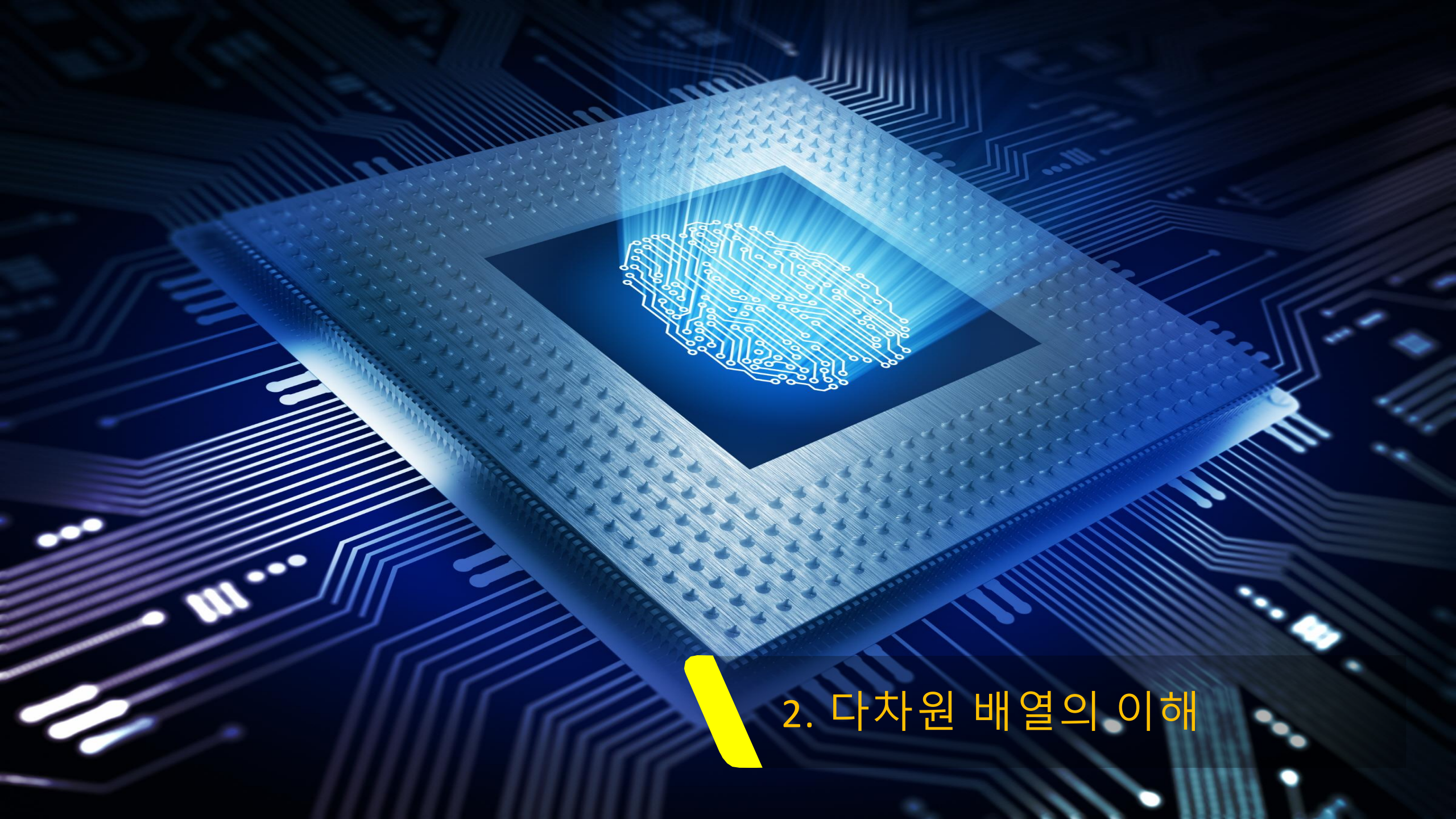


Contents

배열 기반 문자열 변수

- 널(null) 문자를 포함하는 이유
  - 문자열의 끝을 표현하기 위해
  - 쓰레기 값과 실제 문자열의 경계를 나타내기 위해
  - printf 함수는 널 문자를 통해서 출력의 범위를 결정

```
int main(void)
{
    char str[100]=”Hello World!”;
    printf(“%s ₩n”, str);
    . . . . .
```



2. 다차원 배열의 이해

Contents

다차원 배열이란

- 다차원 배열의 의미
  - ➢ 2차원 이상의 배열을 의미

- 다차원 배열의 선언

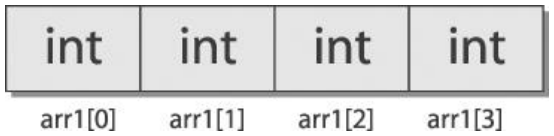
배열 선언 예	몇 차원 배열인가?
int arr[100]	1차원 배열
int arr[10][10]	10×10, 2차원배열
int arr[5][5][5]	5×5×5, 3차원 배열

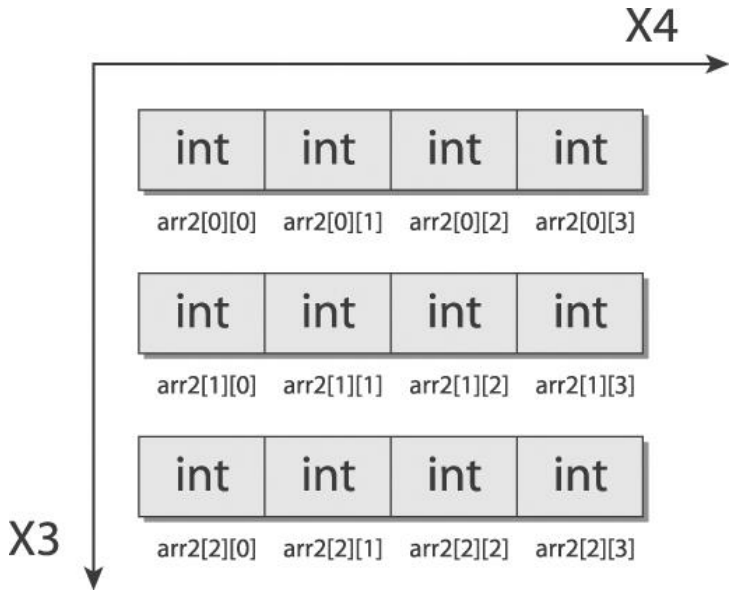
Contents

다차원 배열이란

- 2차원 배열의 선언
  - 2차원적 메모리 구조를 구성

```
int main(void)
{
   int arr1[4];
   int arr2[3][4];
   . . . . .
```



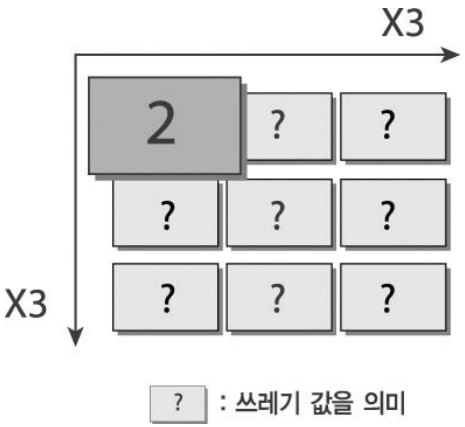


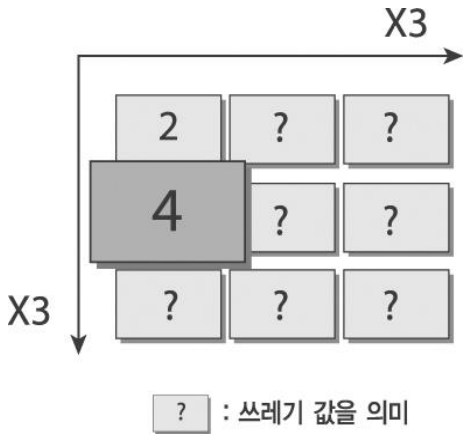
Contents

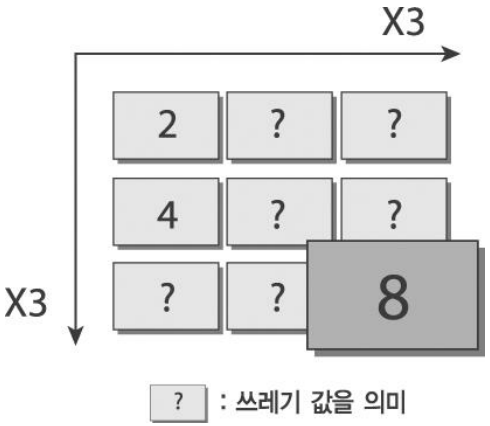
다차원 배열이란

- 2차원 배열 요소의 접근 방법

```
int main(void)
{
   int arr[3][3];
   arr[0][0]=2;
   arr[1][0]=4;
   arr[2][2]=8;
   . . . . .
```





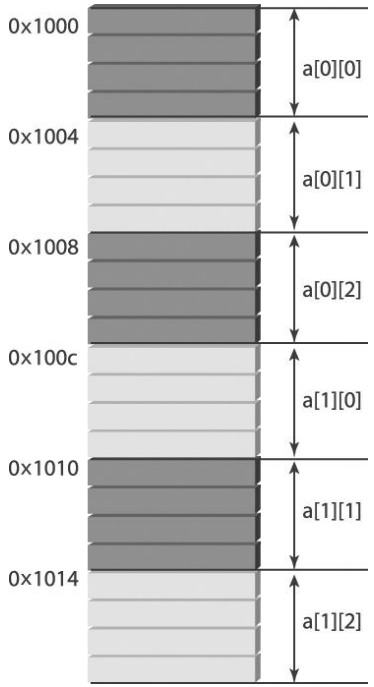


Contents

다차원 배열이란

- 다차원 배열의 실제 메모리 구성
  - 1차원 배열과 동일한 할당
  - 접근 방법을 2차원을 “해석”할 뿐
  - 2차원적으로 이해하는 것이 좋은 습관!

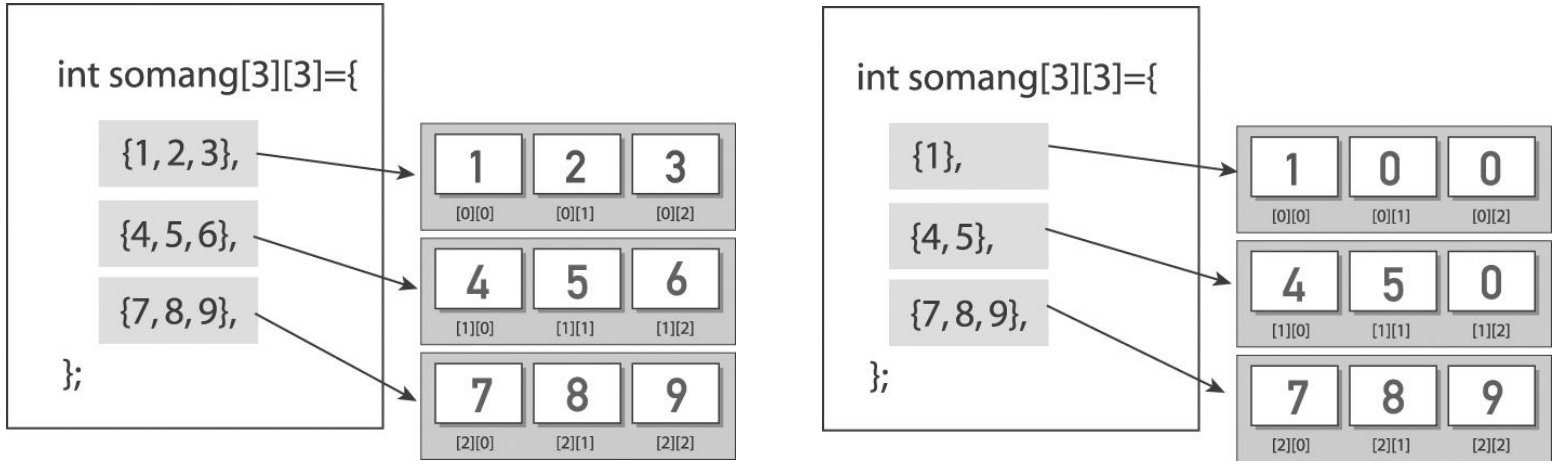
```
int a[2][3]
```



Contents

다차원 배열이란

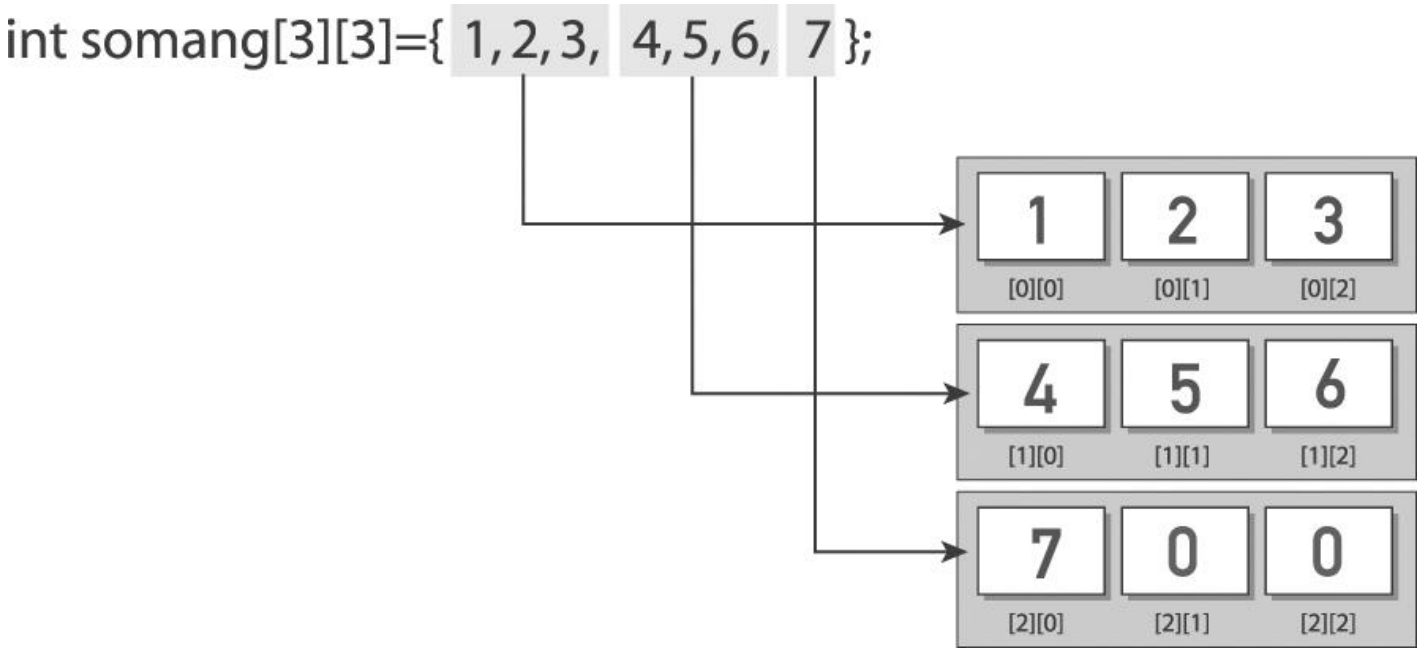
- 2차원 배열의 선언과 동시에 초기화
  - Case 1 : 행 단위로 모든 요소들을 초기화
  - Case 2 : 행 단위로 일부 요소들만 초기화



Contents

다차원 배열이란

- 2차원 배열의 선언과 동시에 초기화
  - ➢ Case 3 : 1차원 배열 형태의 초기화

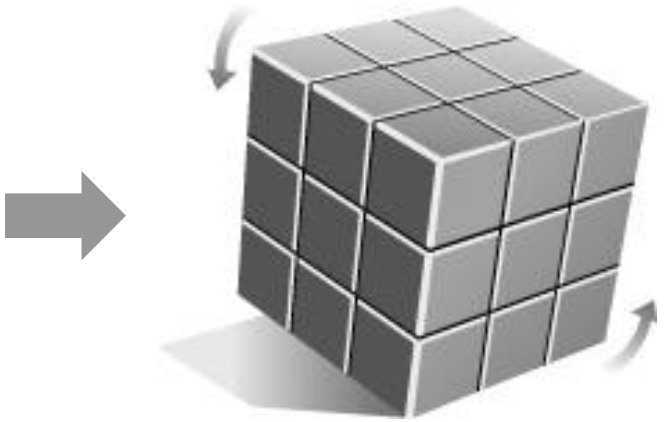


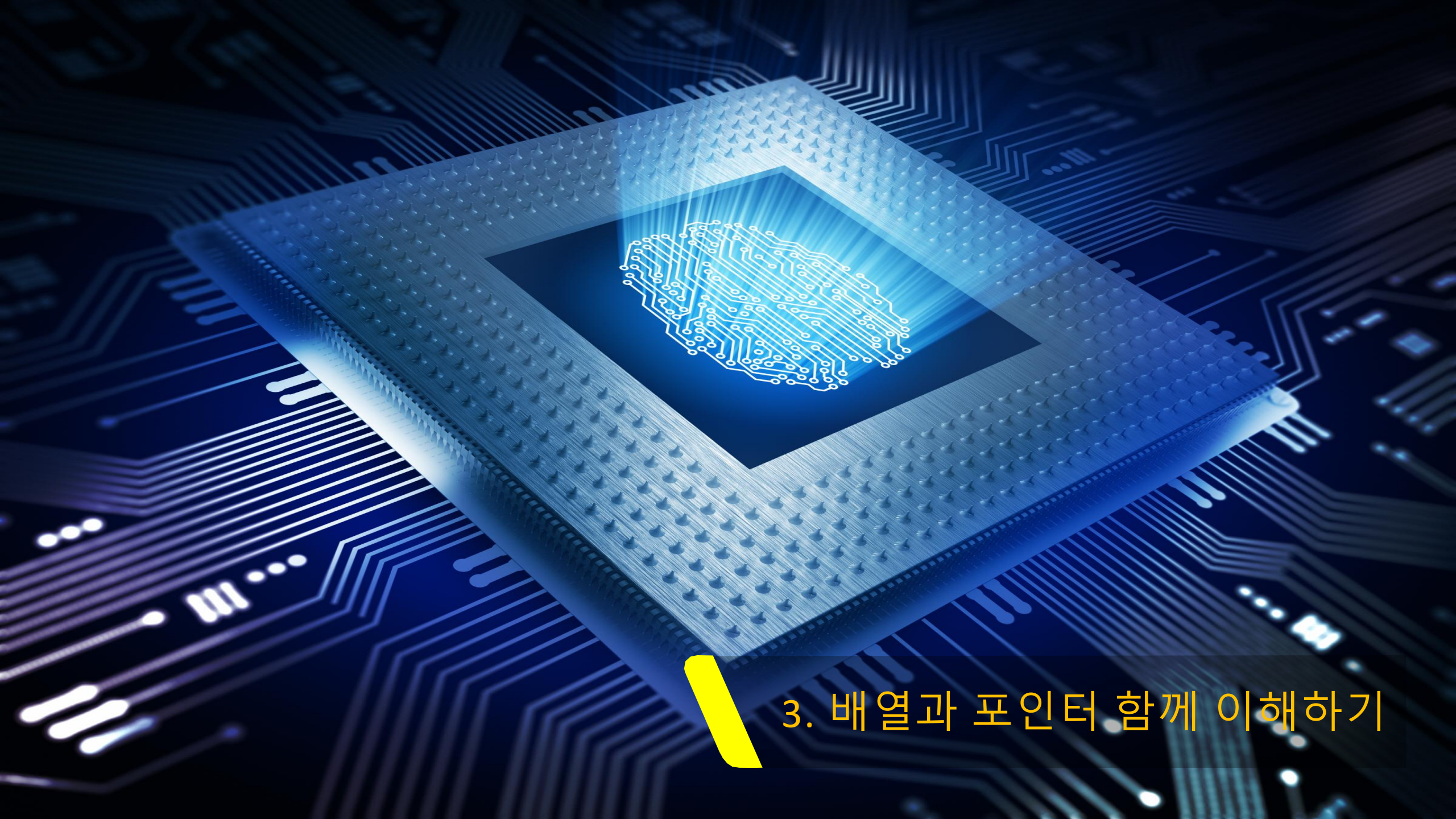
Contents

다차원 배열이란

- **3차원 배열의 선언과 의미**
  - 3차원적 메모리 구조를 의미
  - 개념만 이해하면 충분하며, 일반적으로 미사용
  - 4차원 이상의 배열은 4차원의 형태가 되므로 구조적인 이해가 불가능

```
int a[3][3][3]
```



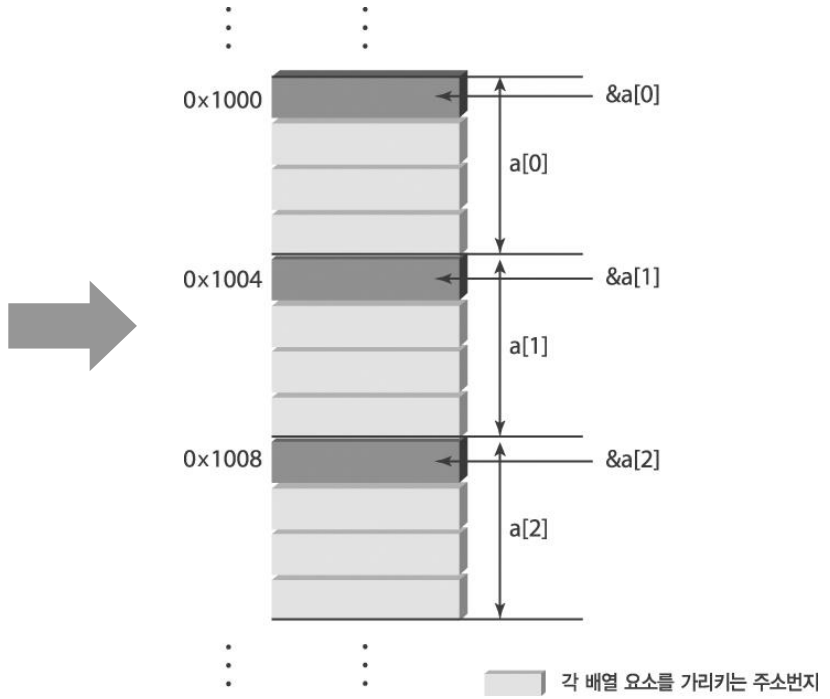


3. 배열과 포인터 함께 이해하기

포인터와 배열의 관계

- 배열 이름이 갖는 의미
  - ➢ 배열 이름은 첫 번째 요소의 주소 값과 동일

```
int a[5]={0, 1, 2, 3, 4}
```



포인터와 배열의 관계

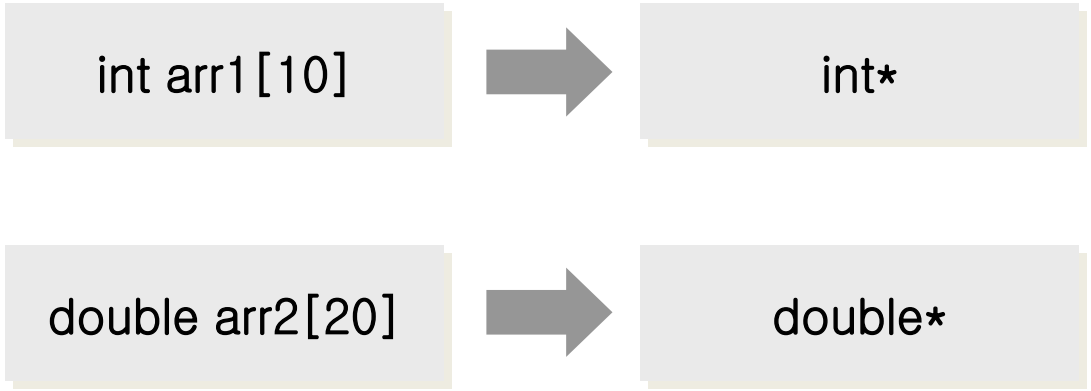
- 배열과 포인터의 비교

비교 대상 / 비교 조건	포인터	배열 이름
이름이 존재하는가	물론 있다.	당연히 있다.
무엇을 나타내는가	메모리의 주소	메모리의 주소
변수인가 상수인가	변수	상수

```
int main(void)
{
    int a[5]={0, 1, 2, 3, 4};
    int b=10;
    a=&b;   //a는 상수이므로 오류, a가 변수였다면 OK!
}
```

포인터와 배열의 관계

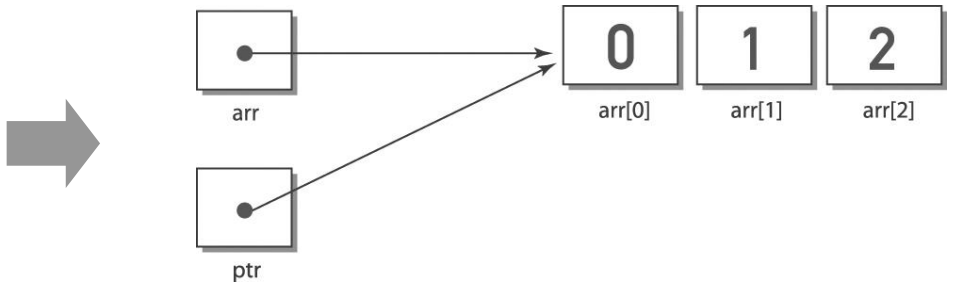
- 배열 이름의 타입
  - 배열 이름도 포인터이므로 타입이 존재
  - 배열 이름이 가리키는 배열 요소에 의해 결정



포인터와 배열의 관계

- **배열 이름의 활용**
  - 배열 이름을 포인터처럼, 포인터를 배열 이름처럼 활용하는 것이 가능

```
#include <stdio.h>

int main(void)
{
    int arr[3]={0, 1, 2};
    int *ptr;

    ptr=arr;

    printf("%d, %d, %d ₩n", ptr[0], ptr[1], ptr[2]);
    return 0;
}
```



포인터와 배열의 관계

- 포인터 연산
  - 포인터가 지니는 값을 증가 혹은 감소시키는 연산을 의미
  - 일반 변수의 증가, 감소와는 다르게 타입의 크기 만큼 증가되는 연산

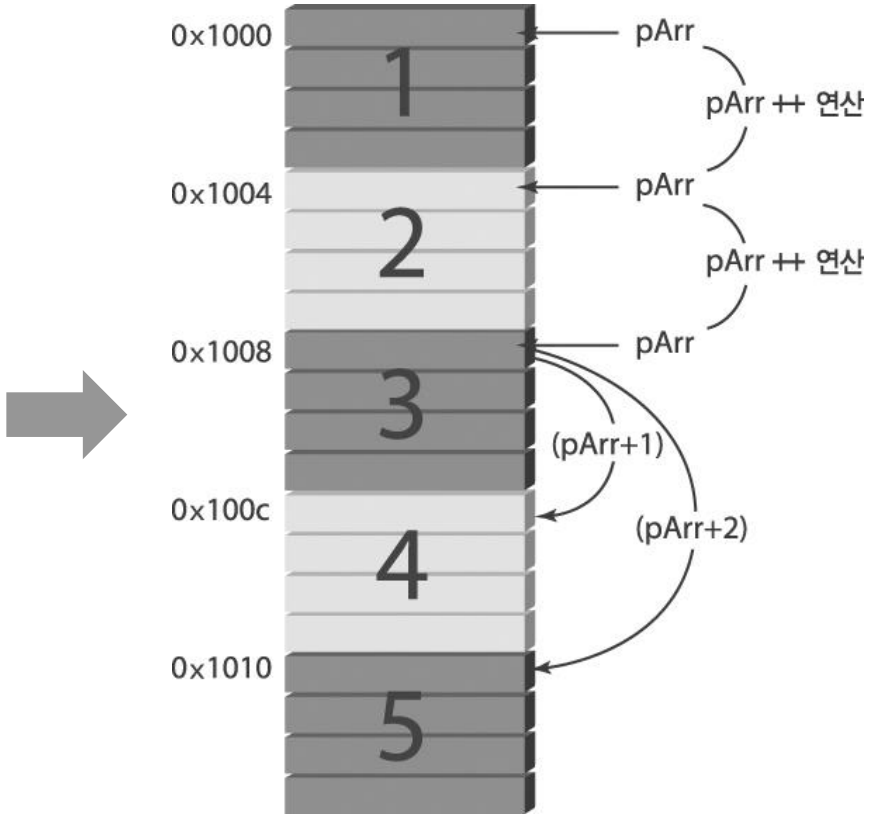
```
/* pointer_op.c */
#include <stdio.h>

int main(void)
{
    int* ptr1=0;                    // int* ptr1=NULL; 과 같은 문장
    char* ptr2=0;                   // char* ptr2=NULL; 과 같은 문장
    double* ptr3=0;                 // double* ptr3=NULL; 과 같은 문장

    printf("%d 번지, %d 번지, %d 번지 ₩n", ptr1++, ptr2++, ptr3++);
    printf("%d 번지, %d 번지, %d 번지 ₩n", ptr1, ptr2, ptr3);

    return 0;
}
```

Contents

포인터와 배열의 관계

- 포인터 연산을 통한 배열 요소의 접근

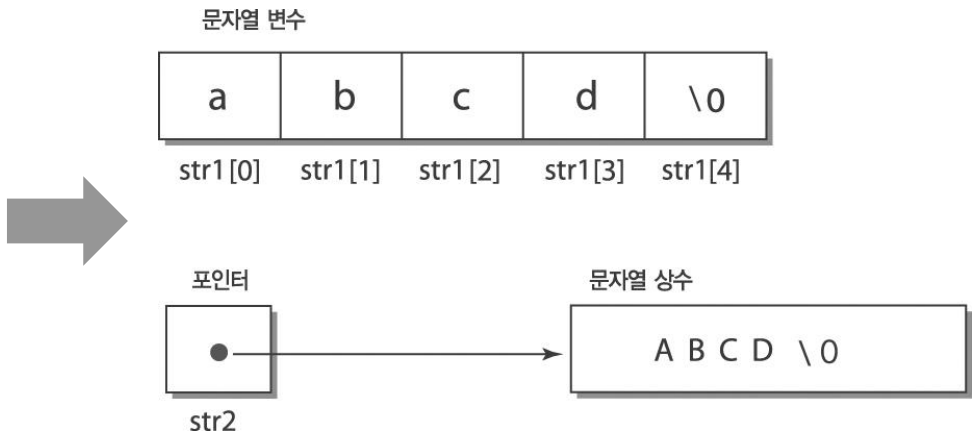
```
/* pointer_array3.c */
#include <stdio.h>

int main(void)
{
   int arr[5]={1, 2, 3, 4, 5};

   int* pArr=arr;
   printf("%d ₩n", *pArr);

   printf("%d ₩n", *(++pArr));
   printf("%d ₩n", *(++pArr));

   printf("%d ₩n", *(pArr+1));
   printf("%d ₩n", *(pArr+2));

   return 0;
}
```



포인터와 배열의 관계

- 문자열 표현 방식의 이해
  - 배열 기반의 문자열 변수
  - 포인터 기반의 문자열 상수

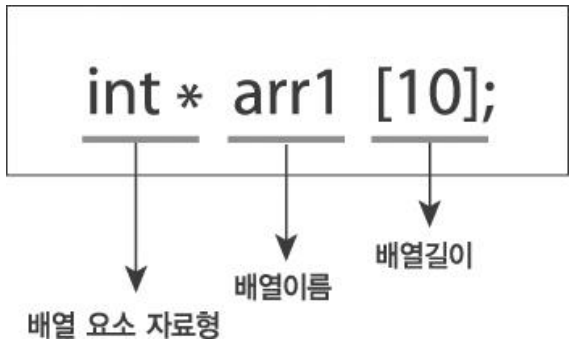
```
char str1[5]="abcd";
char *str2="ABCD";
```



포인터와 배열의 관계

- 포인터 배열
  - ➢ 배열의 요소로 포인터를 지니는 배열

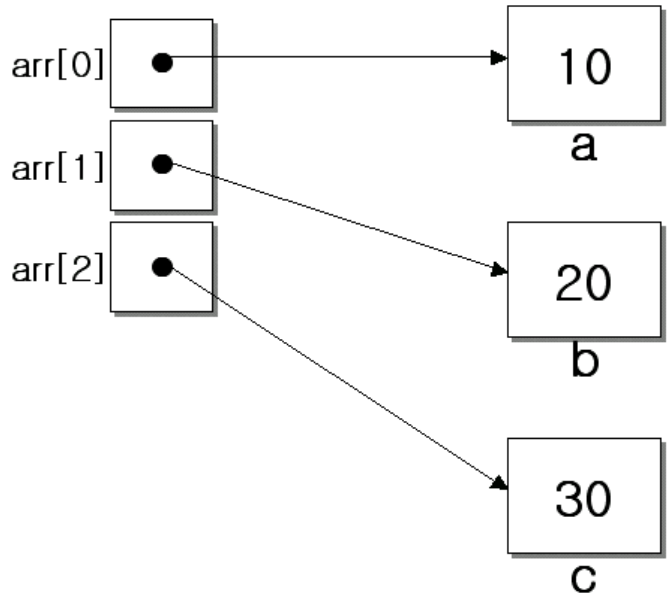
```
int* arr1[10];
double* arr2[20];
char* arr3[30];
```



포인터와 배열의 관계

- 포인터 배열 예제 1

```
#include <stdio.h>

int main(void)
{
   int a=10, b=20, c=30;
   int* arr[3]={&a, &b, &c};

   printf("%d ₩n", *arr[0]);
   printf("%d ₩n", *arr[1]);
   printf("%d ₩n", *arr[2]);

   return 0;
}
```



Contents

포인터와 배열의 관계

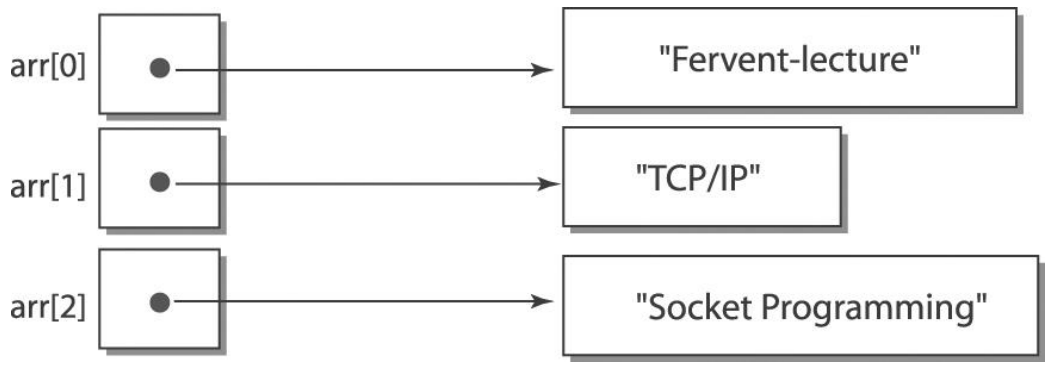
- 포인터 배열 예제 2

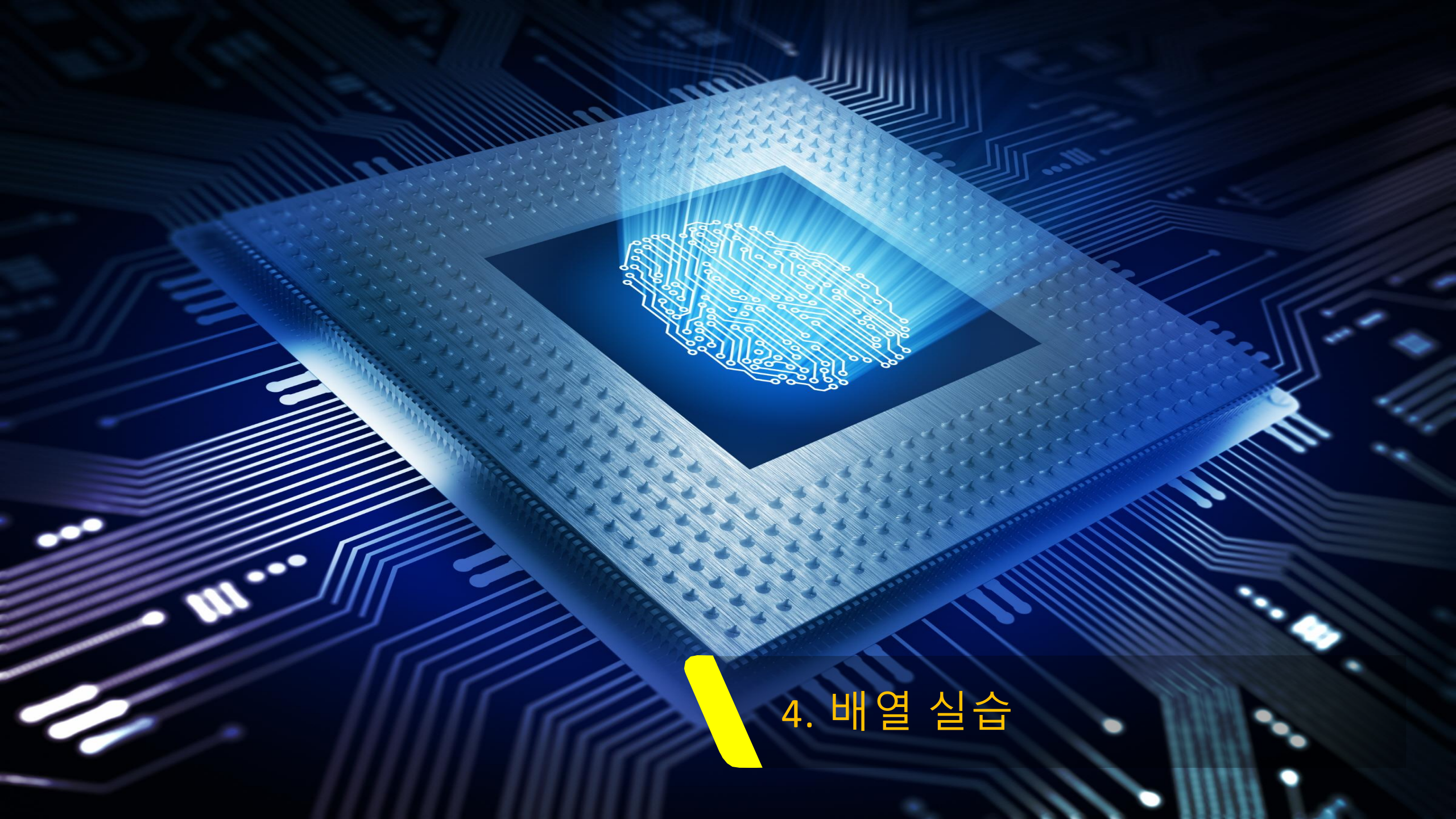
```
/* str_arr.c */
#include <stdio.h>

int main(void)
{
    char* arr[3]={
            "Fervent-lecture",
            "TCP/IP",
            "Socket Programming"
    };
     printf("%s ₩n", arr[0]);
    printf("%s ₩n", arr[1]);
    printf("%s ₩n", arr[2]);

    return 0;
}
```

char * arr[3] = {"Fervent-lectur","TCP/IP","Socket Programming"};

↓

char * arr[3] = {0×1000, 0×2000, 0×3000};



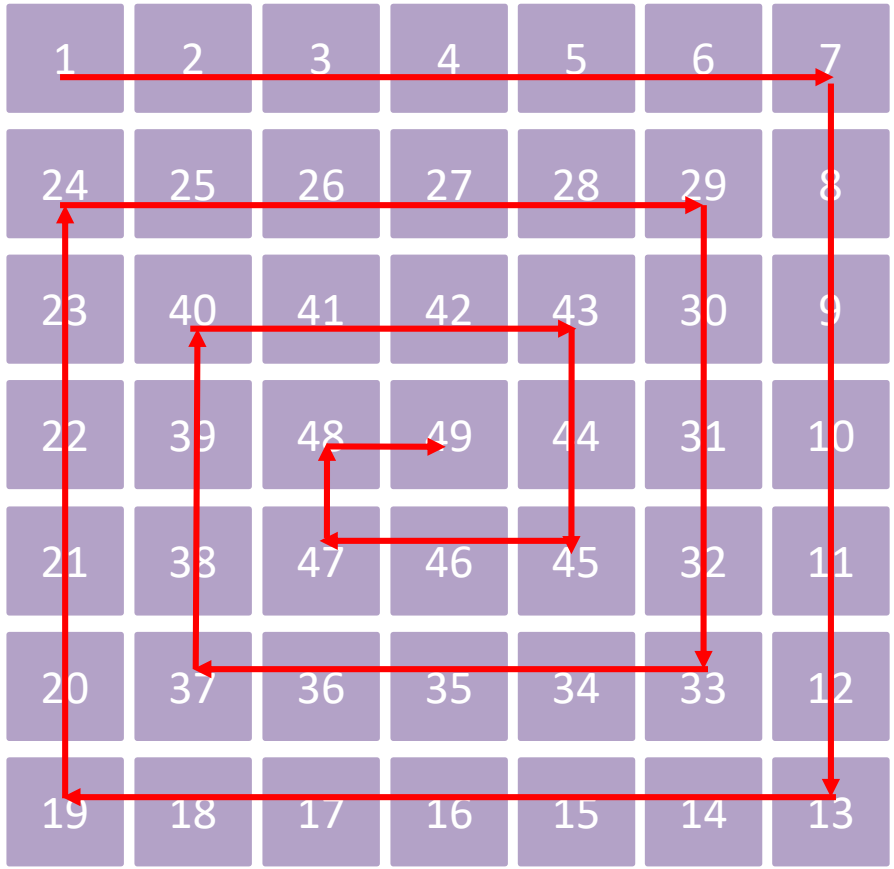


4. 배열 실습

Contents

달팽이 배열

- 입력이 n 일 때, n*n 배열 생성
  - 아래 예제는 입력이 7인 경우



Q & A