



C프로그래밍

Lecture 13. 포인터와 배열! 함께 이해하기

동덕여자대학교
데이터사이언스 전공
권 범

목차

- ❖ 01. 포인터와 배열의 관계
- ❖ 02. 포인터 연산
- ❖ 03. 상수 형태의 문자열을 가리키는 포인터
- ❖ 04. 포인터 변수로 이뤄진 배열: 포인터 배열
- ❖ 05. 연습 문제

01. 포인터와 배열의 관계

- 02. 포인터 연산
- 03. 상수 형태의 문자열을 가리키는 포인터
- 04. 포인터 변수로 이뤄진 배열: 포인터 배열
- 05. 연습 문제

01. 포인터와 배열의 관계

❖ 배열의 이름은 무엇을 의미하는가? (1/2)

- 배열의 이름은 배열의 시작 주소 값을 의미하는(배열의 첫 번째 요소를 가리키는) 포인터입니다.
- 단순히 주소 값이 아닌 포인터인 이유는 메모리 접근에 사용되는 * 연산이 가능하기 때문입니다.

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int arr[3] = {0, 1, 2};
5      printf("배열의 이름: %p\n", arr);
6      printf("첫 번째 요소: %p\n", &arr[0]);
7      printf("두 번째 요소: %p\n", &arr[1]);
8      printf("세 번째 요소: %p\n", &arr[2]);
9      return 0;
10 }
```

배열의 이름: 0012FF50
첫 번째 요소: 0012FF50
두 번째 요소: 0012FF54
세 번째 요소: 0012FF58



배열 요소간 주소 값의 크기는 4바이트임을 알 수 있습니다.

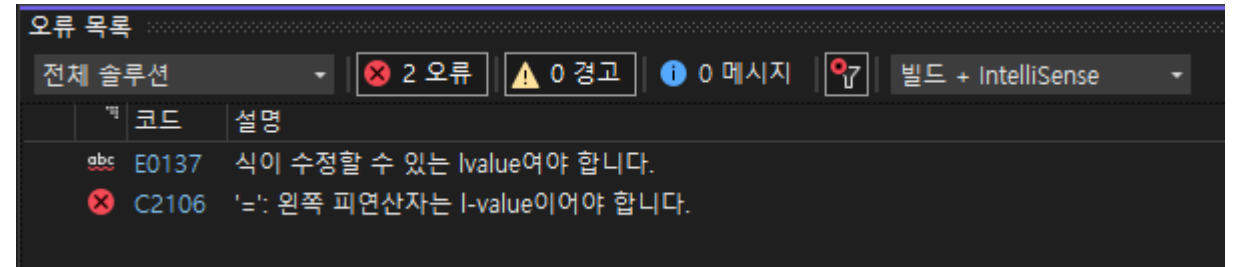
01. 포인터와 배열의 관계

❖ 배열의 이름은 무엇을 의미하는가? (2/2)

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int arr[3] = {0, 1, 2};
5      printf("배열의 이름: %p\n", arr);
6      printf("첫 번째 요소: %p\n", &arr[0]);
7      printf("두 번째 요소: %p\n", &arr[1]);
8      printf("세 번째 요소: %p\n", &arr[2]);
9      arr = &arr[0];
10     return 0;
11 }
```

이 문장은 컴파일 에러를 일으킵니다.

배열의 이름은 변수가 아닌 상수 형태의 포인터이기에 대입연산이 불가능합니다.



배열 이름과 포인터 변수 비교

비교조건 \ 비교대상	포인터 변수	배열의 이름
이름이 존재하는가?	존재한다	존재한다
무엇을 나타내거나 저장하는가?	메모리의 주소 값	메모리의 주소 값
주소 값의 변경이 가능한가?	가능하다	불가능하다.

01. 포인터와 배열의 관계

❖ 1차원 배열 이름의 포인터 형(Type)

1차원 배열 이름의 포인터 형을 결정하는 방법

- ✓ 배열의 이름이 가리키는 변수의 자료형을 근거로 판단합니다.
- ✓ int형 변수를 가리키면 int* 형(예: int arr1[5]; 에서 arr1은 int* 형)
- ✓ double형 변수를 가리키면 double* 형(예: double arr2[7]; 에서 arr2은 double* 형)

```
1  #include <stdio.h>
2  int main(void)
3  {
4      int arr1[3] = {1, 2, 3};
5      double arr2[3] = {1.1, 2.2, 3.3};
6
7      printf("%d %g\n", *arr1, *arr2);
8      *arr1 += 100;
9      *arr2 += 120.5;
10     printf("%d %g\n", arr1[0], arr2[0]);
11     return 0;
12 }
```

배열 이름을 대상으로
포인터 연산을 하고 있음에
주목하세요!

1 1.1
101 121.6

- ✓ arr1이 int형 포인터이므로 *연산의 결과로 4바이트 메모리 공간에 정수를 저장
- ✓ arr2는 double형 포인터이므로 *연산의 결과로 8바이트 메모리 공간에 실수를 저장

01. 포인터와 배열의 관계

❖ 포인터를 배열의 이름처럼 사용할 수도 있다 (1/2)


- arr은 int형 포인터이니 int형 포인터를 대상으로 배열접근을 위한 [idx] 연산을 진행한 셈입니다.
- 포인터 변수 ptr을 대상으로 ptr[0], ptr[1], ptr[2]와 같은 방식으로 메모리 공간에 접근이 가능합니다.

```
int main(void)
{
    int arr[3] = {0, 1, 2};
    arr[0] += 5;
    arr[1] += 7;
    arr[2] += 9;
    ...
}
```

01. 포인터와 배열의 관계

❖ 포인터를 배열의 이름처럼 사용할 수도 있다 (2/2)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int arr[3] = {15, 20, 25};
6      int* ptr = &arr[0];      // int* ptr = arr; 와 동일한 문장
7
8      printf("%d %d\n", ptr[0], arr[0]);
9      printf("%d %d\n", ptr[1], arr[1]);
10     printf("%d %d\n", ptr[2], arr[2]);
11     printf("%d %d\n", *ptr, *arr);
12     return 0;
13 }
```



15	15
20	20
25	25
15	15

포인터 변수를 이용해서 배열의 형태로
메모리 공간에 접근하고 있음에 주목하세요!

02. 포인터 연산

- 01. 포인터와 배열의 관계
- 03. 상수 형태의 문자열을 가리키는 포인터
- 04. 포인터 변수로 이뤄진 배열: 포인터 배열
- 05. 연습 문제

02. 포인터 연산

❖ 포인터를 대상으로 하는 증가 및 감소 연산 (1/3)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int* ptr1 = 0x0010;
6      double* ptr2 = 0x0010;
7
8      printf("%p %p\n", ptr1 + 1, ptr1 + 2);
9      printf("%p %p\n", ptr2 + 1, ptr2 + 2);
10     printf("%p %p\n", ptr1, ptr2);
11     ptr1++;
12     ptr2++;
13     printf("%p %p\n", ptr1, ptr2);
14     return 0;
15 }
```

원래는 적절치 않은 초기화입니다.

00000014 00000018
00000018 00000020
00000010 00000010
00000014 00000018

- ✓ 포인터 변수에 저장된 값을 대상으로 하는 증가 및 감소 연산을 수행할 수 있습니다. (곱셈, 나눗셈 등은 불가능합니다.)
- ✓ 포인터 연산의 일종입니다.

02. 포인터 연산

❖ 포인터를 대상으로 하는 증가 및 감소 연산 (2/3)

- 예제의 실행 결과를 통해서 다음 사실을 알 수 있습니다.

int형 포인터 변수 대상의 증가, 감소 연산 시
sizeof(int)의 크기만큼 값이 증가 및 감소합니다.

double형 포인터 변수 대상의 증가, 감소 연산 시
sizeof(double)의 크기만큼 값이 증가 및 감소합니다.

일반화

type형 포인터 변수 대상의 증가, 감소 연산 시
sizeof(type)의 크기만큼 값이 증가 및 감소합니다.

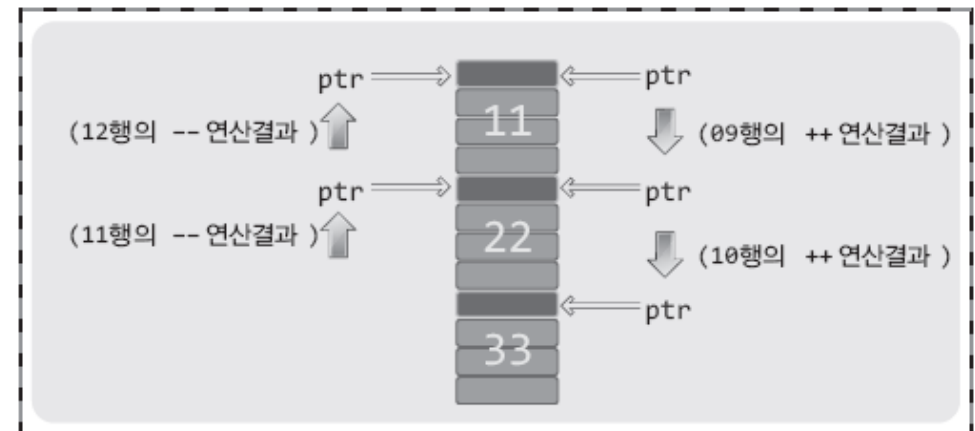
02. 포인터 연산

❖ 포인터를 대상으로 하는 증가 및 감소 연산 (3/3)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int arr[3] = {11, 22, 33};
6      int* ptr = arr;          // int* ptr = &arr[0]; 과 같은 문장
7
8      printf("%d %d %d\n", *ptr, *(ptr+1), *(ptr+2));
9      printf("%d ", *ptr); ptr++; // printf 함수 호출 후, ptr++ 실행
10     printf("%d ", *ptr); ptr++;
11     printf("%d ", *ptr); ptr--;  // printf 함수 호출 후, ptr-- 실행
12     printf("%d ", *ptr); ptr--;
13     printf("%d ", *ptr); printf("\n");
14     return 0;
15 }
```



11 22 33
11 22 33 22 11



02. 포인터 연산

❖ 중요한 결론! $arr[i] == *(arr+i)$

```
#include <stdio.h>

int main(void)
{
    int arr[3] = {11, 22, 33};
    int* ptr = arr;           // int* ptr = &arr[0]; 과 같은 문장

    printf("%d %d %d\n", *ptr, *(ptr+1), *(ptr+2));
    ...
}
```

- ✓ 배열의 이름도 포인터이니, 포인터 변수를 이용한 배열의 접근방식을 배열의 이름에도 사용할 수 있습니다.
- ✓ 그리고 배열의 이름을 이용한 접근방식도 포인터 변수를 대상으로 사용할 수 있습니다.

[결론] arr이 포인터 변수의 이름이건 배열의 이름이건
 $arr[i] == *(arr+i)$

↓

```
printf("%d %d %d\n", *(ptr+0), *(ptr+1), *(ptr+2)); // *(ptr+0)은 *ptr과 같습니다.
printf("%d %d %d\n", ptr[0], ptr[1], ptr[2]);
printf("%d %d %d\n", *(arr+0), *(arr+1), *(arr+2)); // *(arr+0)은 *arr과 같습니다.
printf("%d %d %d\n", arr[0], arr[1], arr[2]);
```

03. 상수 형태의 문자열을 가리키는 포인터

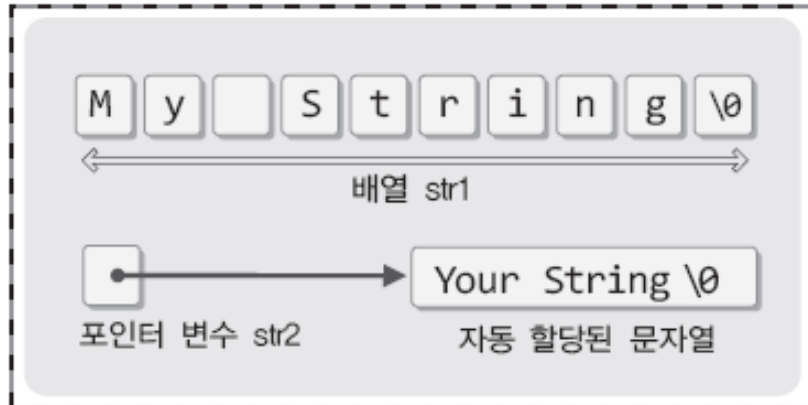
- 01. 포인터와 배열의 관계
- 02. 포인터 연산
- 04. 포인터 변수로 이뤄진 배열: 포인터 배열
- 05. 연습 문제

03. 상수 형태의 문자열을 가리키는 포인터

❖ 두 가지 형태의 문자열 표현

```
char str1[] = "My String";  
char* str2 = "Your String";
```

문자열의 저장 방식



- ✓ str1은 문자열이 저장된 배열입니다.
즉, 문자 배열이기 때문에 변수 성향의 문자열입니다.
- ✓ str2는 문자열의 주소 값을 저장합니다.
즉, 자동 할당된 문자열의 주소 값을 저장하기 때문에 상수 성향의 문자열입니다.

```
int main(void)  
{  
    char* str = "Your team";  
    str = "Our team";    // 의미 있는 문장  
    ...  
}
```

```
int main(void)  
{  
    char str[] = "Your team";  
    str = "Our team";    // 의미 없는 문장  
    ...  
}
```

03. 상수 형태의 문자열을 가리키는 포인터

❖ 두 가지 형태의 문자열 표현의 예

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char str1[] = "My String";
6      char* str2 = "Your String";
7      printf("%s %s\n", str1, str2);
8
9      str2 = "Our String";
10     printf("%s %s\n", str1, str2);
11
12     str1[0] = 'X';
13     printf("%s\n", str1);
14     str2[0] = 'X';
15     printf("%s\n", str2);
16     return 0;
17 }
```

My String Your String
My String Our String
Xy String

- ✓ 변수 성향의 str1에 저장된 문자열은 변경이 가능합니다!
- ✓ 반면 상수 성향의 str2에 저장된 문자열은 변경이 불가능합니다!
- ✓ 간혹 상수 성향의 문자열 변경도 허용하는 컴파일러가 있으나, 이러한 형태의 변경은 바람직하지 못합니다!

03. 상수 형태의 문자열을 가리키는 포인터

❖ 어디서든 선언할 수 있는 상수 형태의 문자열

```
char* str = "Const String";  
                                     문자열 저장 후 주소 값 반환  
char* str = 0x1234;
```

- ✓ 문자열이 먼저 할당된 이후에
그 때 반환되는 주소 값이 저장되는 방식입니다.

```
printf("Show your String");  
                                     문자열 저장 후 주소 값 반환  
printf(0x1234);
```

- ✓ 위와 동일합니다.
- ✓ 문자열은 선언 된 위치로 주소 값이 반환됩니다.

```
WhoAreYou("Hong");  
                                     문자열을 전달받은 함수의 선언  
void WhoAreYou(char* str){...}
```

- ✓ 문자열의 전달만 보더라도
함수의 매개변수 형 (Type)을 짐작할 수 있습니다.

04. 포인터 변수로 이뤄진 배열: 포인터 배열

- 01. 포인터와 배열의 관계
- 02. 포인터 연산
- 03. 상수 형태의 문자열을 가리키는 포인터
- 05. 연습 문제

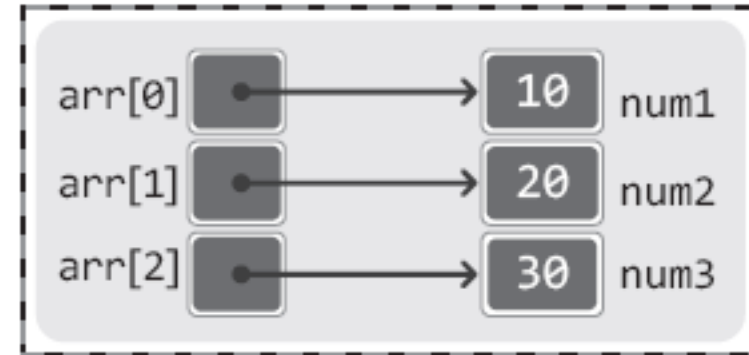
04. 포인터 변수로 이뤄진 배열: 포인터 배열

❖ 포인터 배열의 이해

```
int* arr1[20];    // 길이가 20인 int형 포인터 배열 arr1
double* arr2[30]; // 길이가 30인 double형 포인터 배열 arr2
```

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num1 = 10, num2 = 20, num3 = 30;
6      int* arr[3] = {&num1, &num2, &num3};
7
8      printf("%d\n", *arr[0]);
9      printf("%d\n", *arr[1]);
10     printf("%d\n", *arr[2]);
11     return 0;
12 }
```

10
20
30



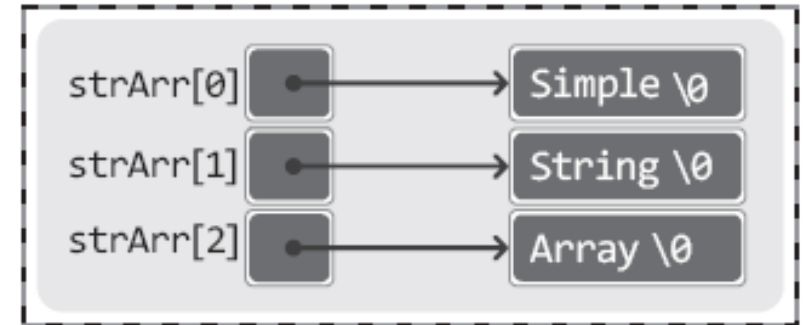
- ✓ 포인터 배열이라고 해서 일반 배열의 선언과 차이가 있지는 않습니다.
- ✓ 변수의 자료형을 표시하는 위치에 int나 double을 대신해서 int* 나 double*를 적을 뿐이다.

04. 포인터 변수로 이뤄진 배열: 포인터 배열

❖ 문자열을 저장하는 포인터 배열

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char* strArr[3] = {"Simple", "String", "Array"};
6      printf("%s\n", strArr[0]);
7      printf("%s\n", strArr[1]);
8      printf("%s\n", strArr[2]);
9      return 0;
10 }
```

Simple
String
Array



```
char* strArr[3] = {"Simple", "String", "Array"};
```

```
char* strArr[3] = {0x1004, 0x1048, 0x2012}; // 반환된 주소 값은 임의로 결정한 것입니다.
```

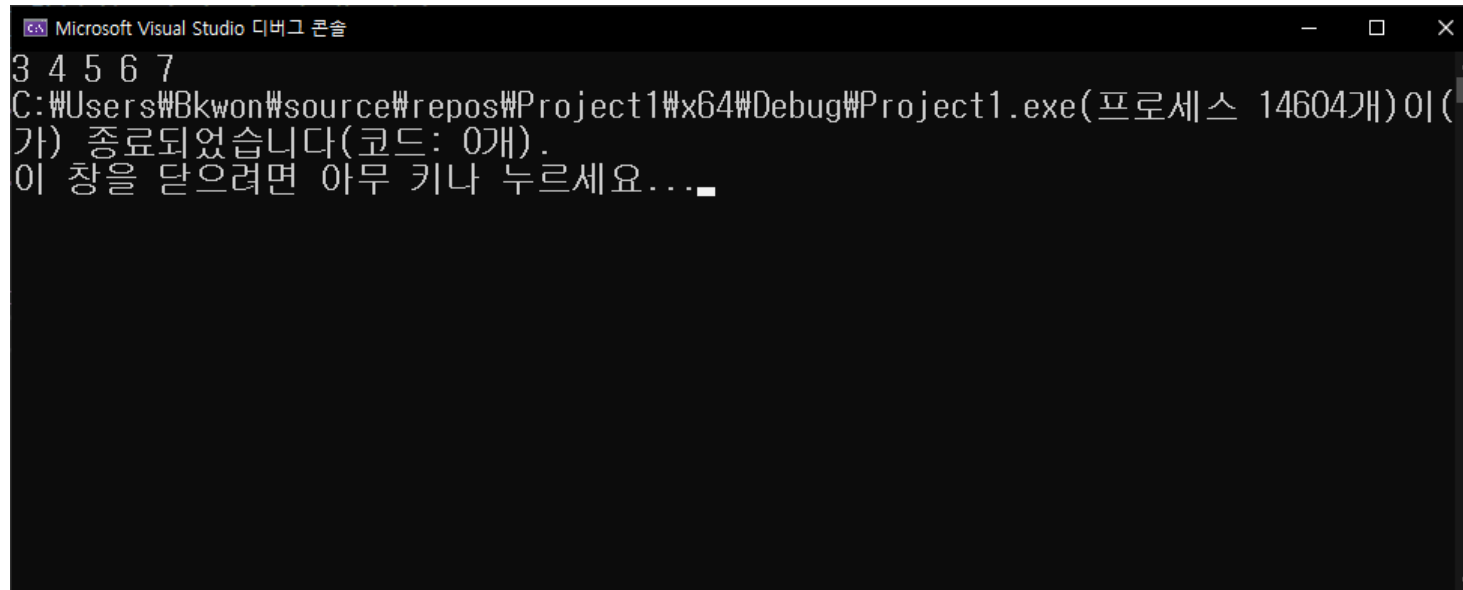
05. 연습 문제

- 01. 포인터와 배열의 관계
- 02. 포인터 연산
- 03. 상수 형태의 문자열을 가리키는 포인터
- 04. 포인터 변수로 이뤄진 배열: 포인터 배열

05. 연습 문제

❖ 연습 문제 1.

- 크기가 5인 int형 배열 arr을 선언하고 1, 2, 3, 4, 5로 초기화한 다음, 포인터 p를 선언해서 배열 arr의 첫 번째 요소를 가리키게 하세요. 그 다음 포인터 p를 조작(포인터 연산을 의미함)해서 배열 요소의 값을 2씩 증가시킨 후, 전체 배열 요소를 출력하는 프로그램을 작성하세요.



```
Microsoft Visual Studio 디버그 콘솔
3 4 5 6 7
C:\Users\Bkwon\source\repos\Project1\x64\Debug\Project1.exe(프로세스 14604개)이(
가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...■
```

05. 연습 문제

❖ 연습 문제 1. 정답 및 해설

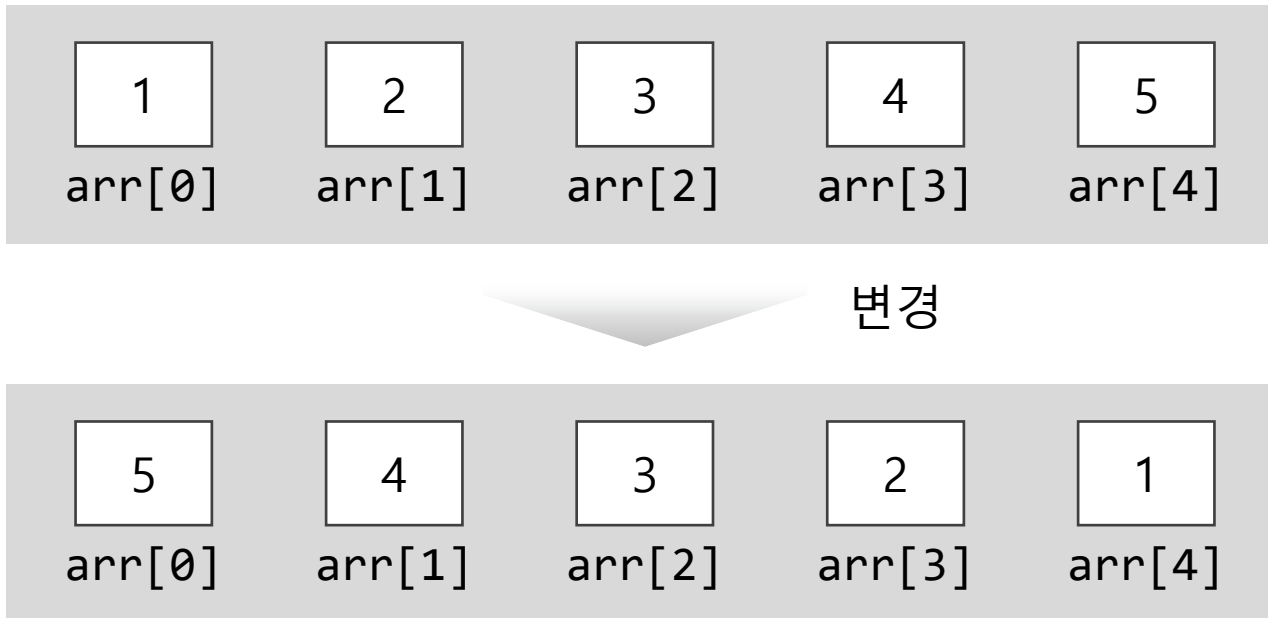
```
1  /* example1.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int arr[5] = {1, 2, 3, 4, 5};
7      int* p = arr;
8
9      for (int j = 0; j < 5; j++)
10         p[j] += 2;          // *(p+j) += 2; 와 같습니다.
11
12     for (int j = 0; j < 5; j++)
13         printf("%d ", p[j]);
14
15     return 0;
16 }
```

3 4 5 6 7

05. 연습 문제

❖ 연습 문제 2.


- 크기가 5인 int형 배열 arr을 선언하고 1, 2, 3, 4, 5로 초기화한 다음, 포인터 p를 선언해서 배열 arr의 첫 번째 요소를 가리키게 하세요. 그 다음에는 포인터 p를 이용해서 배열 요소의 순서를 뒤바꾼 후, 그 결과를 출력하는 프로그램을 작성하세요.



05. 연습 문제

❖ 연습 문제 2. 정답 및 해설

```
1  /* example2.c */
2  #include <stdio.h>
3  int main(void)
4  {
5      int arr[5] = {1, 2, 3, 4, 5};
6      int* p = arr;
7      int temp;
8
9      for (int j = 0; j < 4 - j; j++)
10     {
11         temp = p[j];
12         p[j] = p[4-j];
13         p[4-j] = temp;
14     }
15     for (int j = 0; j < 5; j++)
16         printf("%d ", p[j]);
17
18     return 0;
19 }
```



5 4 3 2 1

- ❖ 01. 포인터와 배열의 관계
- ❖ 02. 포인터 연산
- ❖ 03. 상수 형태의 문자열을 가리키는 포인터
- ❖ 04. 포인터 변수로 이뤄진 배열: 포인터 배열
- ❖ 05. 연습 문제

THANK YOU!

Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr