

■ 간접 지정 연산자(간접 참조 연산자)

포인터가 단순히 메모리 주소를 저장만 한다면 쓸 데가 없다. 하지만 포인터가 유용한 이유는 포인터가 가르키는 변수의 값을 읽어오거나 변경할 수 있기 때문이다.

- 간접 참조(지정) 연산자 : 주소연산과 정반대되는 개념의 연산자 (간접 참조 연산자 *) <-> (주소 연산자 &)
- * (간접 참조(지정) 연산자) : 포인터가 가르키는 위치의 내용(값)을 반환하는 연산자
‘에스터리스크’라고 읽음
- 포인터 변수 예제(5)

```
#include<stdio.h>
int main()
{
    int i = 10;
    int* p = &i;
    printf("i 값: %d\n", i); // 직접 참조
    printf("*p 값: %d\n", *p); // 간접 참조
    return 0;
}
```

실행결과

- 포인터 변수 예제(6)

```
#include<stdio.h>
int main()
{
    int i = 10;
    int* p = &i;
    printf("i 값: %d\n", i);
    *i = 20;
    printf("i 값: %d\n", i);
    return 0;
}
```

실행결과

- `int i = 10;` 변수를 선언하여 변수에 직접 값을 지정해주는 것이 직접 지정 방식
- `*i = 20;` 포인터가 가르키는 변수의 값을 지정해주는 것이 간접 지정 방식

● 포인터 변수 예제(7)

```
#include<stdio.h>
int main()
{
    int num1 = 100, num2 = 100;
    int* pnum;

    pnum = &num1;    // 포인터 pnum이 num1 을 가리킴
    printf("num 1 : %d, &num1 : %p, pnum : %p, *pnum : %d\n", num1,&num1, pnum, *pnum);
    (*pnum) += 30;    // num1 += 30; 과 동일, 즉 *pnum은 num1과 같음
    printf("num 1 : %d, &num1 : %p, pnum : %p, *pnum : %d\n", num1,&num1, pnum, *pnum);

    pnum = &num2;    // 포인터 pnum이 num2 를 가리킴, 즉 *pnum은 num2와 같음
    printf("num 2 : %d, &num2 : %p, pnum : %p, *pnum : %d\n", num2,&num2, pnum, *pnum);
    (*pnum) -= 30;    // num2 -= 30; 과 동일
    printf("num 2 : %d, &num2 : %p, pnum : %p, *pnum : %d\n", num2,&num2, pnum, *pnum);

    return 0;
}
```

실행결과

● 포인터 변수 예제(8) - 포인터 사용 시 주의할 점

```
#include<stdio.h>
int main()
{
    int* p;
    *p = 100;
    printf("%d\n", *p);
    return 0;
}
```

```
#include<stdio.h>
int main()
{
    int i;
    double *p;
    p = &i;
    *p = 36.5;
    printf("%d\n", *p);
    return 0;
}
```

퀴즈) 왼쪽 두 코드는 오류를 범하고 있다. 각각 어떤 오류를 범한 것일까?

1. 포인터 타입과 변수의 타입이 일치하지 않는다.
2. 포인터를 초기화하지 않고 사용했다.

■ 포인터와 배열

- 배열의 이름은 배열의 시작 주소 값을 의미하며, 그 형태는 값이 저장 불가능한 상수이다. 다시 말하면 배열 이름 자체가 포인터이고, 첫 번째 배열 원소의 주소와 같다.

```
#include<stdio.h>
int main(){
    int arr[4] = { 10, 20, 30, 40 };
    printf("배열의 이름 : %p \n", arr); // &arr은 뭘까요?
    printf("arr[0]의 주소 : %p \n", &arr[0]);
    printf("arr[1]의 주소 : %p \n", &arr[1]);
    printf("arr[2]의 주소 : %p \n", &arr[2]);
    printf("arr[3]의 주소 : %p \n", &arr[3]);
    return 0;
}
```

실행결과

```
배열의 이름 : 00A2FAB8
arr[0]의 주소 : 00A2FAB8
arr[1]의 주소 : 00A2FABC
arr[2]의 주소 : 00A2FAC0
```

- 위에서 배열은 원소들이 메모리에서 연속된 공간을 차지하고 있음을 알 수 있다.

```
#include<stdio.h>
int main(){
    int arr[3] = { 0 };
    int arr2[3] = { 0,1,2 };

    arr = arr2; // Error 발생!! arr 은 변경이 불가능함.
}
```

- 배열의 이름이 포인터라고 한다면, 이 배열의 이름으로 간접 참조 연산(*)을 적용해볼 수 있지 않을까? YES!
- 포인터 변수 예제(6)

```
#include<stdio.h>
int main()
{
    int arr1[3] = { 1, 2, 3 };
    double arr2[3] = { 1.1, 2.2, 3.3 };

    printf("arr[0] : %d, arr2[0] : %lf\n", arr1[0], arr2[0]);
    printf("*arr1 : %d, *arr2 : %lf\n", *arr1, *arr2);

    *arr1 += 100;
    *arr2 += 100.0;

    printf("arr[0] : %d, arr2[0] : %lf\n", arr1[0], arr2[0]);
    printf("*arr1 : %d, *arr2 : %lf\n", *arr1, *arr2);
    return 0;
}
```

실행결과

```

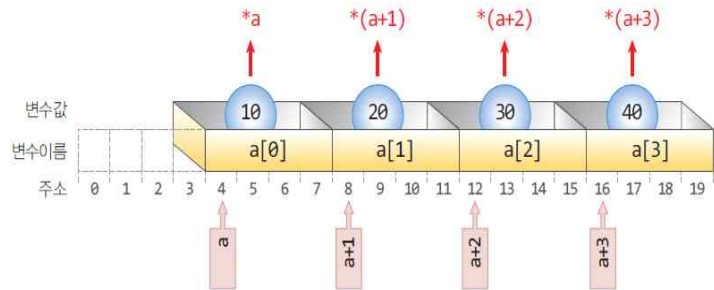
arr[0] : 1,   arr2[0] : 1.100000
*arr1 : 1,   *arr2 : 1.100000
예제6번 실행 결과 -> arr[0] : 101, arr2[0] : 101.100000
                     *arr1 : 101, *arr2 : 101.100000

```

```

#include<stdio.h>
int main()
{
    int a[] = { 10, 20, 30, 40, 50 };
    int* p = a; // a는 &a[0]과 동일
    printf("a = %p = %p\n", a, p);
    printf("a+1 = %p = %p\n", a+1, p+1);
    printf("*a = %d = %d\n", *a, *p);
    printf("*(a+1) = %d = %d", *(a+1), *(p+1));
    return 0;
}

```



- 배열의 이름이 주소이므로 포인터 변수에 저장할 수 있다.
- 배열 이름이 포인터라면 역으로 포인터도 배열 이름처럼 사용할 수 있을까? YES!
- 포인터 변수 예제(7)

```

#include<stdio.h>
int main()
{
    int arr[3] = { 15, 25, 35 };
    int* ptr = arr; // int *ptr = &arr[0];과 동일한 문장

```

```

    printf(" ptr : %p, arr : %p, &arr[0] : %p\n", ptr, arr, &arr[0]);
    printf(" ptr+1 : %p, arr+1 : %p, &arr[1] : %p\n", ptr+1, arr+1, &arr[1]);
    printf(" ptr+2 : %p, arr+2 : %p, &arr[2] : %p\n", ptr+2, arr+2, &arr[2]);

```

```

    printf("\n");

```

```

    *ptr = 20;

```

실행결과 (arr[0]의 주소는 100이라고 가정하고 실행결과를 적어보자.)

```

    printf("arr[0] : %d, ptr[0] : %d, *(ptr+0) : %d\n", arr[0], ptr[0], *(ptr+0));
    printf("arr[1] : %d, ptr[1] : %d, *(ptr+1) : %d\n", arr[1], ptr[1], *(ptr+1));
    printf("arr[2] : %d, ptr[2] : %d, *(ptr+2) : %d\n", arr[2], ptr[2], *(ptr+2));

```

```

    return 0;
}

```

```

ptr : 006FFE9C, arr : 006FFE9C, &arr[0] : 006FFE9C
ptr+1 : 006FFEA0, arr+1 : 006FFEA0, &arr[1] : 006FFEA0
ptr+2 : 006FFEA4, arr+2 : 006FFEA4, &arr[2] : 006FFEA4

예제7번 실행결과 -> arr[0] : 20, ptr[0] : 20, *(ptr+0) : 20
                    arr[1] : 25, ptr[1] : 25, *(ptr+1) : 25
                    arr[2] : 35, ptr[2] : 35, *(ptr+2) : 35

```

- **포인터 변수로 사칙연산이 가능할까?** 곱셈, 나눗셈은 안되지만 덧셈, 뺄셈은 가능!
- 만약 포인터 변수 p가 1000 번지를 가리키고 있다. p를 하나 증가시키면(p++) p의 값은 어떻게 될까? 일반적으로 1001이 될 것 같지만 p가 어떤 자료형을 가리키는 포인터인가에 따라 다르다. 따라서 char형 포인터를 증가시키면 char형의 크기인 1바이트만큼 증가한다.

포인터 타입	++ 연산 후 증가되는 값
char	1
int	4
float	4
double	8

- 포인터 변수 예제(포인터 사칙연산)

```
#include<stdio.h>
int main(){
    char* pc = (char*)100; // pc, pi, pd에 메모리 주소 100번지 강제 대입(절대주소 대입)
    int* pi = (int*)100; // int* pi = 100; 으로 해도 자동 형변환됨
    double* pd = (double*)100;
    printf("증가 전 pc= %p, pi= %p, pd= %p\n", pc, pi, pd);
    pc++;
    pi++;
    pd++;
    printf("증가 후 pc= %p, pi= %p, pd= %p\n", pc, pi, pd);
    printf("pc+2= %p, pi+2= %p, pd+2= %p\n", pc+2, pi+2, pd+2);
    return 0;
}
```

포인터 사칙연산 실행 결과->

```
증가 전 pc= 0000000000000064, pi= 0000000000000064, pd= 0000000000000064
증가 후 pc= 0000000000000065, pi= 0000000000000068, pd= 000000000000006C
pc+2= 0000000000000067, pi+2= 0000000000000070, pd+2= 000000000000007C
```

- 간접 참조 연산자와 증감 연산자

수식	의미
v = *p++	p가 가리키는 값을 v에 대입한 후 p를 증가한다.
v = (*p)++	p가 가리키는 값을 v에 대입한 후 가리키는 값을 증가한다.
v = ++*p	p를 증가시킨 후에 p가 가리키는 값을 v에 대입한다.
v = ++*p	p가 가리키는 값을 가져온 후 그 값을 증가하여 v에 대입한다.

- *p++; 이 문장은 p가 가리키는 위치에서 값을 가져온 후에 p를 증가한다. ++의 우선 순위가 *보다 높기 때문이다. 만약 포인터가 가리키는 대상의 값을 증가하려고 했으면 (*p)++;과 같이 하여야 한다. 이 문장에서 괄호가 *연산자를 먼저 수행하게 만든다.

● 포인터 변수 예제- 증감연산자 종합1

```
#include <stdio.h>
int main()
{
    int i[2] = { 10,20 }, v=0;
    int* pi = i;
    printf("i[0] = %d, i[1]= %d, v= %d, pi = %p\n", i[0], i[1], v, pi);
    v = ++*pi; // pi가 가리키는 대상을 증가
    printf("i[0] = %d, i[1]= %d, v= %d, pi = %p\n", i[0], i[1], v, pi);
    v = **pi; // pi를 증가
    printf("i[0] = %d, i[1]= %d, v= %d, pi = %p\n", i[0], i[1], v, pi);
    v = (*pi)++; // pi가 가리키는 대상을 증가
    printf("i[0] = %d, i[1]= %d, v= %d, pi = %p\n", i[0], i[1], v, pi);
    v = *pi++; // pi를 증가
    printf("i[0] = %d, i[1]= %d, v= %d, pi = %p\n", i[0], i[1], v, pi);

    return 0;
}
```

실행결과 i배열의 주소는 100이라고 가정

예제 증감연산자 실행 결과 - >

```
Microsoft Visual Studio 디버그 콘솔
i[0] = 10, i[1] = 20, v = 0, pi = 00000040ED59F9A8
i[0] = 11, i[1] = 20, v = 11, pi = 00000040ED59F9A8
i[0] = 11, i[1] = 20, v = 20, pi = 00000040ED59F9AC
i[0] = 11, i[1] = 21, v = 20, pi = 00000040ED59F9AC
i[0] = 11, i[1] = 21, v = 21, pi = 00000040ED59F9B0
```

● 포인터 변수 예제- 증감연산자 종합2

```
#include <stdio.h>
int main()
{
    int i = 10;
    int* pi = &i;
    printf("i = %d, pi = %p\n", i, pi);
    (*pi)++; // pi가 가리키는 대상을 증가
    printf("i = %d, pi = %p\n", i, pi);

    printf("i = %d, pi = %p\n", i, pi);
    *pi++; // pi를 증가
    printf("i = %d, pi = %p\n", i, pi);

    return 0;
}
```

실행결과

(i의 주소는 100이라고 가정하고 실행결과를 적어 보자.)

예제 증감연산자 실행 결과 - >

```
Microsoft Visual Studio 디버그 콘솔
i = 10, pi = 00000086AFFBF624
i = 11, pi = 00000086AFFBF624
i = 11, pi = 00000086AFFBF624
i = 11, pi = 00000086AFFBF628
```

● 포인터 변수 예제(8)

```
#include<stdio.h>
int main(){
    int i;
    int arr[5] = { 1,2,3,4,5 };
    int* ptr = arr; // int *ptr = &arr[0];과 동일한 문장
    printf("ptr : %p, arr : %p\n", ptr, arr);
    for (i = 0; i < 5; i++)
        printf("%d ", arr[i]);

    printf("\n");
    for (i = 0; i < 5; i++)
        printf("%d ", *(ptr+i));
    printf("\n");
    for (i = 0; i < 5; i++)
        printf("%d ", *(ptr++));
    printf("\n");
    printf("*ptr : %d\n", *ptr); // 무엇을 가르키고 있을까?
    printf("ptr : %p, arr : %p\n", ptr, arr);
    return 0;
}
```

arr[0]의 주소가 100이라고 가정하고 실행결과

```
ptr : 006FFA2C, arr : 006FFA2C
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
예제8 실행 결과 - > *ptr : -858993460
ptr : 006FFA40, arr : 006FFA2C
```

● 포인터 변수 예제(9)

```
#include <stdio.h>
int main(){
    int i, arr[] = { 1,2,3,4,5,6,7,8,9,10 };
    int* ptr = arr;
    int len = sizeof(arr) / sizeof(int);

    for (i = 0; i < len; i++) {
        printf("ptr : %p, *ptr : %d, arr : %p\n", ptr, *ptr, arr);
        *ptr *= 2;
        ptr++;
    }
    printf("ptr : %p, *ptr : %d, arr : %p\n", ptr, *ptr, arr);
    for (i = 0; i < len; i++){
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

arr[0]의 주소가 100이라고 가정하고 실행결과

예제 9 실행결과 - >

```
ptr : 00EFF880, *ptr : 1, arr : 00EFF880
ptr : 00EFF884, *ptr : 2, arr : 00EFF880
ptr : 00EFF888, *ptr : 3, arr : 00EFF880
ptr : 00EFF88C, *ptr : 4, arr : 00EFF880
ptr : 00EFF890, *ptr : 5, arr : 00EFF880
ptr : 00EFF894, *ptr : 6, arr : 00EFF880
ptr : 00EFF898, *ptr : 7, arr : 00EFF880
ptr : 00EFF89C, *ptr : 8, arr : 00EFF880
ptr : 00EFF8A0, *ptr : 9, arr : 00EFF880
ptr : 00EFF8A4, *ptr : 10, arr : 00EFF880
ptr : 00EFF8A8, *ptr : -858993460, arr : 00EFF880
2 4 6 8 10 12 14 16 18 20
```

● 포인터 변수 예제(10)

```
#include <stdio.h>
int main(){
    int i, arr[] = { 10,9,8,7,6,5,4,3,2,1};
    int* ptr = arr;
    int len = sizeof(arr) / sizeof(int);

    for (i = 0; i < len; i++) {
        printf("ptr+i : %p, *(ptr+i) : %d, arr : %p\n", ptr+i, *(ptr + i), arr);
        *(ptr + i) *= 2;
    }
    printf("ptr+i : %p, *(ptr+i) : %d, arr : %p\n", ptr + i, *(ptr + i), arr);
    for (i = 0; i < len; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

실행결과

예제10 실행결과->

```
ptr+i : 0079F750, *(ptr+i) : 10, arr : 0079F750
ptr+i : 0079F754, *(ptr+i) : 9, arr : 0079F750
ptr+i : 0079F758, *(ptr+i) : 8, arr : 0079F750
ptr+i : 0079F75C, *(ptr+i) : 7, arr : 0079F750
ptr+i : 0079F760, *(ptr+i) : 6, arr : 0079F750
ptr+i : 0079F764, *(ptr+i) : 5, arr : 0079F750
ptr+i : 0079F768, *(ptr+i) : 4, arr : 0079F750
ptr+i : 0079F76C, *(ptr+i) : 3, arr : 0079F750
ptr+i : 0079F770, *(ptr+i) : 2, arr : 0079F750
ptr+i : 0079F774, *(ptr+i) : 1, arr : 0079F750
ptr+i : 0079F778, *(ptr+i) : -858993460, arr : 0079F750
20 18 16 14 12 10 8 6 4 2
```

● 포인터 변수 예제(11)

```
#include <stdio.h>
int main(){
    int i, arr[] = { 10,9,8,7,6,5,4,3,2,1 };
    int* ptr = arr;

    int total = *ptr, len = sizeof(arr) / sizeof(int);
    printf("ptr : %p, *ptr : %d, total : %d\n", ptr, *ptr, total);
    for (i = 0; i < len-1; i++) {
        total += *(++ptr);
        printf("ptr : %p, *ptr : %d, total : %d\n", ptr, *ptr, total);
    }
    printf("total : %d\n", total);
    return 0;
}
```

실행결과 (arr[0]의 주소가 100번지라고 가정)

예제 11 실행결과 - >

```
ptr : 00FAFCCC, *ptr : 10, total : 10
ptr : 00FAFCDD, *ptr : 9, total : 19
ptr : 00FAFCDE, *ptr : 8, total : 27
ptr : 00FAFCDF, *ptr : 7, total : 34
ptr : 00FAFCDC, *ptr : 6, total : 40
ptr : 00FAFCE0, *ptr : 5, total : 45
ptr : 00FAFCE4, *ptr : 4, total : 49
ptr : 00FAFCE8, *ptr : 3, total : 52
ptr : 00FAFCEC, *ptr : 2, total : 54
ptr : 00FAFCF0, *ptr : 1, total : 55
total : 55
```


● 포인터 변수 예제(12)

```
#include <stdio.h>
int main(){
    int i, j, arr[] = { 10,9,8,7,6,5,4,3,2,1 };
    int len = sizeof(arr) / sizeof(int), tmp;
    int* fptr = arr, *bptr = arr+len-1;

    for (i = 0; i < len/2; i++) {
        printf("*ftpr :%2d, *bptr :%2d [ ", *fptr, *bptr);
        tmp = *fptr;
        *fptr = *bptr;
        *bptr = tmp;
        fptr++; bptr--;
        for (j = 0; j < len; j++) {
            printf("%d ", *(arr + j));
        }
        printf("]\n");
    }
    printf("\n");
    for (i = 0; i < len; i++) {
        printf("%d ", *(arr+i));
    }
    return 0;
}
```

실행결과

```
*ftpr :10, *bptr : 1 [ 1 9 8 7 6 5 4 3 2 10 ]
*ftpr : 9, *bptr : 2 [ 1 2 8 7 6 5 4 3 9 10 ]
*ftpr : 8, *bptr : 3 [ 1 2 3 7 6 5 4 8 9 10 ]
*ftpr : 7, *bptr : 4 [ 1 2 3 4 6 5 7 8 9 10 ]
*ftpr : 6, *bptr : 5 [ 1 2 3 4 5 6 7 8 9 10 ]

예제12 실행결과 - > 1 2 3 4 5 6 7 8 9 10
```

■ 상수 형태의 문자열을 가리키는 포인터


● 문자열 선언의 두 가지 형태

```
char str1[] = "I am student!"; // 변수(배열) 형태의 문자열
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
str1	I		a	m		s	t	u	d	e	n	t	!	\0

배열 str1

```
char* str2 = "I dreamed a dream"; // 상수 형태의 문자열
```

str2  "I dreamed a dream"

포인터변수 str2 자동 할당된 문자열

● 문자열 예제(1)

```
#include <stdio.h>
int main(){
    char str1[] = "I am student!"; // 변수 형태의 문자열
    char* str2 = "I dreamed a dream"; /* 상수 형태의 문자열, 이 문자열은 읽기 전용 메모리에
                                         문자열 상수로 저장됨 */

    printf("str1: %s, str2: %s\n", str1, str2);

    // str1 = "Change String!!!"; 변경 불가!
    str2 = "Change String!!!"; // 변경 가능
    printf("str1: %s, str2: %s\n", str1, str2);

    str1[0] = '*'; // 변경 가능
    // str2[0] = '*'; // 변경 불가!
    printf("str1: %s, str2: %s\n", str1, str2);
    return 0;
}
```

실행결과

```
str1: I am student!, str2: I dreamed a dream
str1: I am student!, str2: Change String!!!
str1: * am student!, str2: Change String!!!
```

● 문자열 예제(2)

```
#include <stdio.h>
int main(){
    char str1[] = "I am student!\n"; // "... " <- 이 문자열의 주소가 str1 배열에 초기화 됨
    char* str2 = "I dreamed a dream\n"; // "... " <- 이 문자열의 주소가 str2 포인터에 초기화 됨

    printf("I am student!\n");
    printf(str1);

    printf("I dreamed a dream\n");
    printf(str2);
    return 0;
}
```

● 문자열 예제(3)

<pre>// char 배열을 사용 #include <stdio.h> int main() { char buf[] = "abcdefghi"; *(buf) = 'A'; *(buf+1) = 'B'; printf("%s", buf); return 0; }</pre>	<pre>// char* 사용 #include <stdio.h> int main() { char* buf = "abcdefghi"; *(buf) = 'A'; *(buf+1) = 'B'; printf("%s", buf); return 0; }</pre>
<p>● 문자열 배열을 선언하고 그 배열의 값을 "abcdefghi"라는 값으로 순차적으로 '대입'한 것</p>	<ul style="list-style-type: none"> ● "abcdefghi" 라는 상수(문자열 상수)를 만들고 이를 문자형 포인터 변수로 가르키고 있는 것! ● 문자열 상수값은 변경 불가하므로 문자열 일부를 바꾸려 하면 <u>에러</u> 발생함!

연습 문제 (기본수학1)

- 5 **브론즈 V** <https://www.acmicpc.net/problem/10757> 큰 수 A+B
- 4 **브론즈 IV** <https://www.acmicpc.net/problem/1712> 손익분기점
- 3 **브론즈 III** <https://www.acmicpc.net/problem/10250> ACM 호텔
- 2 **브론즈 II** <https://www.acmicpc.net/problem/2292> 벌집
- 2 **브론즈 II** <https://www.acmicpc.net/problem/1193> 분수찾기
- 2 **브론즈 II** <https://www.acmicpc.net/problem/2775> 부녀회장이 됨여야
- 1 **브론즈 I** <https://www.acmicpc.net/problem/2869> 달팽이는 올라가고 싶다
- 1 **브론즈 I** <https://www.acmicpc.net/problem/2839> 설탕 배달

연습 문제 (기본수학2)

- 3 **브론즈 III** <https://www.acmicpc.net/problem/1085> 직사각형에서 탈출
- 3 **브론즈 III** <https://www.acmicpc.net/problem/3009> 네 번째 점
- 3 **브론즈 III** <https://www.acmicpc.net/problem/3053> 택시 기하학
- 3 **브론즈 III** <https://www.acmicpc.net/problem/4153> 직각삼각형
- 5 **실버 V** <https://www.acmicpc.net/problem/2581> 소수
- 4 **실버 IV** <https://www.acmicpc.net/problem/1978> 소수 찾기
- 4 **실버 IV** <https://www.acmicpc.net/problem/11653> 소인수분해
- 2 **실버 II** <https://www.acmicpc.net/problem/1929> 소수 구하기
- 2 **실버 II** <https://www.acmicpc.net/problem/4948> 베르트랑 공준

■ Call-by-value, Call-by-reference

- 함수를 호출할 때 단순히 값을 전달하는 형태의 함수호출을 가리켜 Call-by-value
- 메모리의 접근에 사용되는 주소값을 전달하는 형태의 함수호출을 가리켜 Call-by-reference
- Call-by-value 예제

```
#include <stdio.h>
void swap(int n1, int n2);
int main()
{
    int num1 = 10, num2 = 20;
    printf(" num1 : %d, num2 : %d\n", num1, num2);
    swap(num1, num2);
    printf(" num1 : %d, num2 : %d\n", num1, num2);
    return 0;
}
void swap(int n1, int n2)
{
    int tmp;
    tmp = n1;
    n1 = n2;
    n2 = tmp;
}
```

실행결과

- Call-by-reference 예제(1)

```
#include <stdio.h>
void swap(int *p1, int *p2);
int main()
{
    int num1 = 30, num2 = 40;
    printf(" num1 : %d, num2 : %d\n", num1, num2);
    swap(&num1, &num2);
    printf(" num1 : %d, num2 : %d\n", num1, num2);
    return 0;
}
void swap(int* p1, int* p2)
{
    int tmp;
    tmp = *p1;
    *p1 = *p2;
    *p2 = tmp;
}
```

실행결과

num1 : 30, num2 : 40
call by reference 실행 결과-> num1 : 40, num2 : 30

■ 함수의 인자(매개변수)로 배열 전달하기

- 함수의 인자로 배열을 전달하기
- 주소를 통해 호출자 메모리에 접근할 수 있는 방법을 제시(Call-by-reference)
- 배열인자 예제(1)

```
#include <stdio.h>
void PrintList(int* pList, int nSize);
int main(){
    // 배열을 지역변수로 선언했기 때문에
    // 다른 함수에서는 직접 접근할 수 없다.
    int aList1[5] = { 3,2,4,5,1 };
    int aList2[10] = { 8,5,6,9,1,4,2,3,7,0 };
    PrintList(aList1, 5);
    PrintList(aList2, 10);
    return 0;
}
// 함수 호출 시 배열을 실매개변수(실인수)로 보낼 때 '포인터' 형식매개변수로 받는다
// 포인터에는 요소의 개수 정보가 없으므로 int 매개변수가 더 필요하다.
void PrintList(int* pList, int nSize){
    int i;
    printf("PrintList()\n");
    for (i = 0; i < nSize; i++)
        printf(" %d", pList[i]);
    printf("\n");
}
```

실행결과

```
PrintList()
3 2 4 5 1
PrintList()
8 5 6 9 1 4 2 3 7 0
```

- 배열을 함수의 인자로 전달받는 함수의 선언문

int 형 배열의 주소 값을 인자로 전달받을 수 있도록 int 형 포인터 변수가 선언되어 있음.
void PrintList(int* pList, int nSize);

다음과 같이 선언하는 것도 가능

```
void PrintList(int pList[], int Len);
```

즉 int* pList 와 int pList[]는 동일한 선언이다. 후자의 선언이 배열이 인자로 전달된다는 느낌을 더 강하게 주는 선언임.

일반적으로 배열의 주소값이 인자(실매개변수)로 전달될 때에는 형식매개변수는 int pList[]형태의 선언을 주로 사용한다.

아래처럼 일반적으로 대입은 불가능! (매개변수로만 가능)

```
int arr[3] = { 1,2,3 };
int pList[] = arr; // 불가능
```

● Call-by-reference 예제(2)

```
#include <stdio.h>
// 아래 프로그램은 치명적인 버그가 존재함.
void SetName(char* pszName);
int main(){
    char szName[32] = { 0 };
    SetName(szName);
    printf(" 당신의 이름은 %s입니다.\n", szName);
    return 0;
}
void SetName(char* pszName)
{
    printf(" 이름을 입력하세요 : ");
    scanf("%s",pszName);
}
```

실행결과

이름을 입력하세요 : 홍길동
당신의 이름은 홍길동입니다.

배열인자 예제(2)

```
#include <stdio.h>
void PrintList(int* pList, int Len);
void AddList(int* pList, int Len, int add);
int main(){
    int arr[3] = { 1,2,3 };
    AddList(arr, sizeof(arr) / sizeof(arr[0]), 1);
    PrintList(arr, sizeof(arr) / sizeof(arr[0]));

    AddList(arr, sizeof(arr) / sizeof(*arr), 2);
    PrintList(arr, sizeof(arr) / sizeof(*arr));

    AddList(arr, sizeof(arr) / sizeof(int), 3);
    PrintList(arr, sizeof(arr) / sizeof(int));
    return 0;
}
void AddList(int pList[], int Len, int add)
{
    int i;
    for (i = 0; i < Len; i++)
        pList[i] += add;
}
void PrintList(int pList[], int Len){
    int i;
    printf(" PrintList()\n");
    for (i = 0; i < Len; i++)
        printf(" %d", pList[i]);

    printf("\n");
}
```

실행결과

```
PrintList()
2 3 4
PrintList()
4 5 6
PrintList()
7 8 9
```

● 함수 연습(1)

다음은 프로그램 사용자가 입력하는 문자열에서 영문자를 모두 대문자로 출력하는 프로그램이다.

`void Input(char* buffer)` 함수와 `void Upper(char* buffer)` 함수를 작성해보세요!

<pre>#include <stdio.h> void Input(char*); void Upper(char*); int main() { char buffer[1025] = { 0 }; Input(buffer); Upper(buffer); printf("%s\n", buffer); return 0; }</pre>	<pre>void Input(char buffer[]) { // 작성하세요! } void Upper(char buffer[]) { // 작성하세요! }</pre>
---	--

입력의 길이는 1024 바이트를 넘지 않음

입력은 영문자, 숫자, 특수문자로 만들어진 문자열이 공백없이 입력된다.

입력 : ASDFQWTEADFasdfqlwketjlaksdfjqo23059u2035 입력 : 23424abcdlad
출력 : ASDFQWTEADFASDFQLWKETJLAKSDFJQO23059U2035 출력 : 23424ABCDLAD

입력 : 111111adsfkljletja@@%\$#!@#!asdfasdf
출력 : 111111ADSFLKJLETJA@@%\$#!@#!ASDFASDF

● 함수 연습(2) - <https://www.acmicpc.net/problem/9613>

양의 정수 n 개가 주어졌을 때, 가능한 모든 쌍의 GCD의 합을 구하는 프로그램을 작성하시오.

`void Input(int arr[], int len)` : arr 배열에 len개 입력받는 함수 ($1 < len \leq 100$)

`long long SumGCD(int arr[], int len)` : arr 배열의 모든 쌍의 GCD 합을 구하는 함수

`int GCD(int a, int b)` : 입력 a, b 의 최대공약수를 구하는 함수

<pre>#include <stdio.h> int GCD(int a, int b); void Input(int arr[], int len); long long SumGCD(int arr[], int len); int main() { int TC, num; long long int result; int arr[101] = { 0 }; scanf("%d", &TC); while (TC--) { scanf("%d", &num); Input(arr, num); result = SumGCD(arr, num); printf("%lld\n", result); } return 0; }</pre>	<pre>void Input(int arr[], int len) { // 작성하세요! } long long SumGCD(int arr[], int len) { // 작성하세요! } int GCD(int a, int b) { // 작성하세요! }</pre>
--	--

예제 입력 1 복사

```
3
4 10 20 30 40
3 7 5 12
3 125 15 25
```

예제 출력 1 복사

```
70
3
35
```


● 함수 연습(3) - <https://www.acmicpc.net/problem/11005>

10진법 수 N이 주어진다. 이 수를 B진법으로 바꿔 출력하는 프로그램을 작성하시오.

10진법을 넘어가는 진법은 숫자로 표시할 수 없는 자리가 있다. 이런 경우에는 다음과 같이 알파벳 대문자를 사용한다. A: 10, B: 11, ..., F: 15, ..., Y: 34, Z: 35

`void Trans(char res[], int N, int B)` : 10진법 수 N을 B진법으로 바꾸어 arr배열에 저장하는 함수

```
#include <stdio.h>
void Trans(char res[], int N, int B);
int main()
{
    char res[35] = { 0 };
    int num, B;
    scanf("%d %d", &num, &B);
    Trans(res, num, B);
    printf("%s\n", res);
    return 0;
}
```

```
void Trans(char res[], int N, int B)
{
    // 작성하세요!
}
```

예제 입력 1 복사

60466175 36

예제 출력 1 복사

ZZZZZ

● 함수 연습(4) - <https://www.acmicpc.net/problem/2745>

B진법 수 N이 주어진다. 이 수를 10진법으로 바꿔 출력하는 프로그램을 작성하시오.

10진법을 넘어가는 진법은 숫자로 표시할 수 없는 자리가 있다. 이런 경우에는 다음과 같이 알파벳 대문자를 사용한다. A: 10, B: 11, ..., F: 15, ..., Y: 34, Z: 35

`int Trans(char arrN[], int B)` : B진법 수 N을 10진법으로 반환하는 함수

```
#include <stdio.h>
int Trans(char arrN[], int B);
int main()
{
    char arrN[35] = { 0 };
    int num, B;
    scanf("%s %d", res, &B);
    num = Trans(arrN, B);
    printf("%d\n", num);
    return 0;
}
```

```
int Trans(char arrN[], int B)
{
    // 작성하세요!
}
```

예제 입력 1 복사

ZZZZZ 36

예제 출력 1 복사

60466175