C 프로그래밍 프로젝트

Chap 17. 포인터의 포인터

2013.11.03.

오 병 우

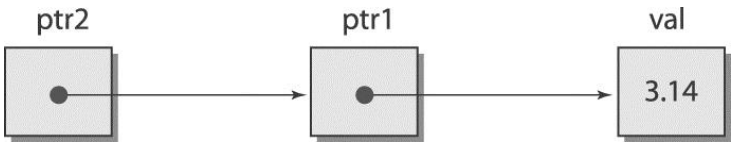
컴퓨터공학과

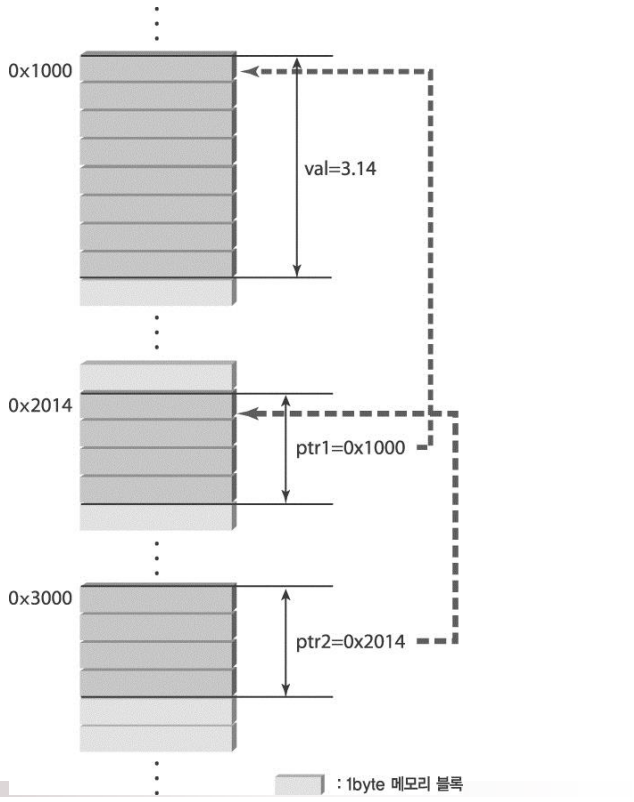
kit 국립금오공과대학교
Kumoh National Institute of Technology

17-1 포인터의 포인터

- 포인터의 포인터
  - 더블 포인터라고 불린다.
  - 싱글 포인터의 주소 값을 저장하는 용도의 포인터

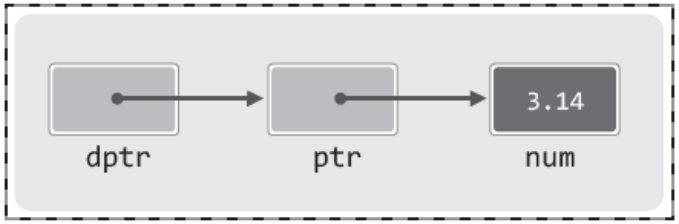
```
int main(void)
{
   double val=3.14;
   double *ptr1 = &val;     // 싱글 포인터
   double **ptr2 = &ptr1; // 더블 포인터
   . . .
```





포인터의 주소를 저장하는 변수

```
int main(void)
{
    double num=3.14;
    double * ptr=&num;
    double ** dptr =&ptr;
    .....
}
```

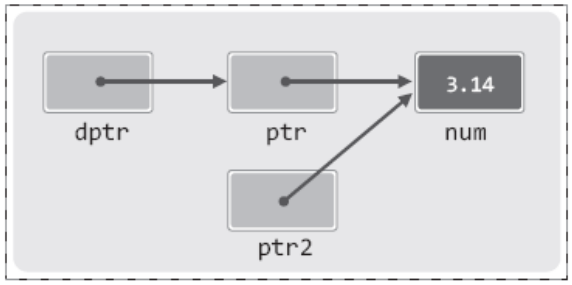


포인터 변수의 주소 값을 저장하는 것이 이중 포인터 변수(더블 포인터 변수)이다.

위의 상황에서 ***dptr**은 포인터 변수 **ptr**을...

***(*dptr)**은 변수 **num**을 의미하게 된다.

```
int main(void)
{
    double num = 3.14;
    double *ptr = &num;
    double **dptr = &ptr;
    double *ptr2;

    printf("%9p %9p \n", ptr, *dptr);
    printf("%9g %9g \n", num, **dptr);
    ptr2 = *dptr;   // ptr2 = ptr 과 같은 문장
    *ptr2 = 10.99;
    printf("%9g %9g \n", num, **dptr);
    return 0;
}
```



이 상황에서 변수 **num**에 접근하는 네 가지 방법은?

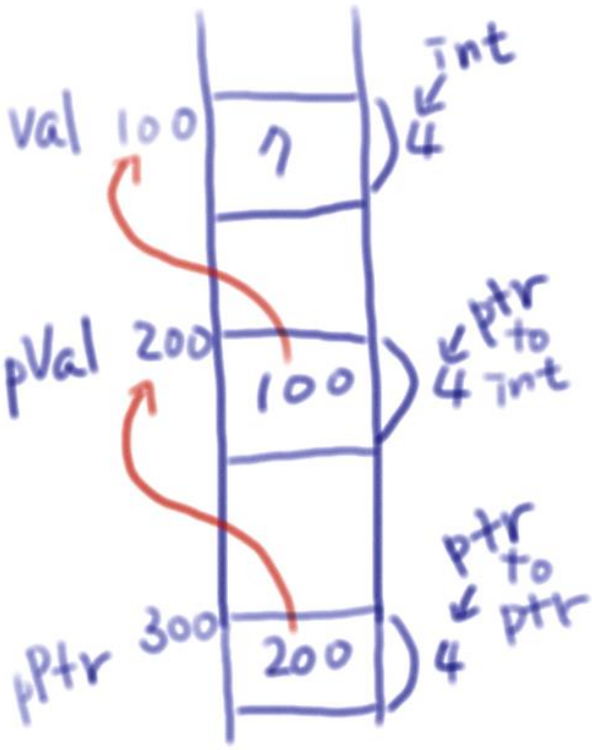
```
0032FD00    0032FD00
    3.14        3.14
   10.99       10.99
```

실행결과

포인터의 포인터 예제

- 활용 예제
  - val을 double 타입으로 바꾸고 변화 확인할 것

```
int val      = 7;
int *pVal    = &val;
int **pPtr   = &pVal;

printf("val = %d, pVal = %x, pPtr = %x₩n", val, pVal, pPtr);

(*pVal)++;
printf("val = %d, pVal = %x, pPtr = %x₩n", val, pVal, pPtr);

(**pPtr)++;
printf("val = %d, pVal = %x, pPtr = %x₩n", val, pVal, pPtr);

(*pPtr)++;
printf("val = %d, pVal = %x, pPtr = %x₩n", val, pVal, pPtr);

pPtr++;
printf("val = %d, pVal = %x, pPtr = %x₩n", val, pVal, pPtr);
```



17-1 포인터의 포인터?

- 더블 포인터의 사용

```
int main(void)
{
   double val[2] = {3.14, 7.31};
   double *ptr1 = val;     // 싱글 포인터
   double **ptr2 = &ptr1; // 더블 포인터

   printf("val = %d, ptr1 = %d, ptr2 = %d\n", val, ptr1, ptr2);
   printf("val[0] = %f\n", val[0]);
   printf("*++ptr1 = %f\n", *++ptr1);
   printf("**ptr2 = %f\n", **ptr2);
   return 0;
}
```

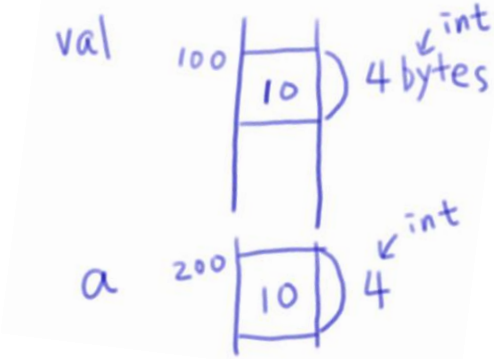
복습: 14장 Summary

- 함수 내에서 값을 바꾸고 그 값을 호출한 놈이 써야 한다면 포인터를 써라.
  - 유식하게는 Call-By-Reference라고 한다.

Call-By-Value

```
{
    int val = 10;
    fct(val);
}
void fct(int a)
{
}
```

바꿀 수 없음

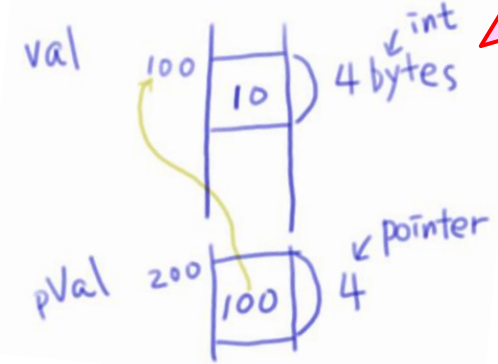


Call-By-Reference

사고치면 "아버지 모시고 와"

```
{
    int val = 10;
    fct(&val);
}
void fct(int *pVal)
{
}
```

val을 바꾸고 싶으면 포인터 사용



포인터에 대한 Call-By-Reference

- 포인터를 함수로 전달하여 그 함수 내에서 포인터 변수의 값(주소)을 바꾸고 그 값을 호출한 놈이 써야 한다면 포인터의 포인터 (더블 포인터)를 써라.

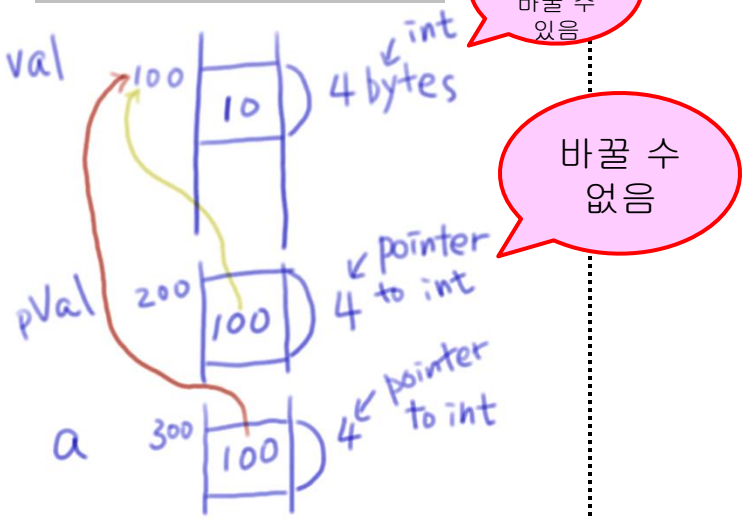
Call-By-Value

```
{
    int val = 10;
    int *pVal = &val;
    fct(pVal);
}
void fct(int *a)
{
}
```

아들이 사고치면 "아버지 모시고 와"

a를 통해서 바꿀 수 있음

바꿀 수 없음

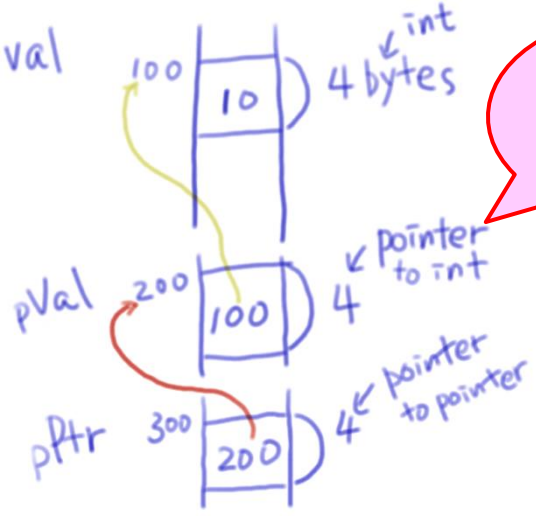


Call-By-Reference

```
{
    int val = 10;
    int *pVal = &val;
    fct(&pVal);
}
void fct(int **pPtr)
{
}
```

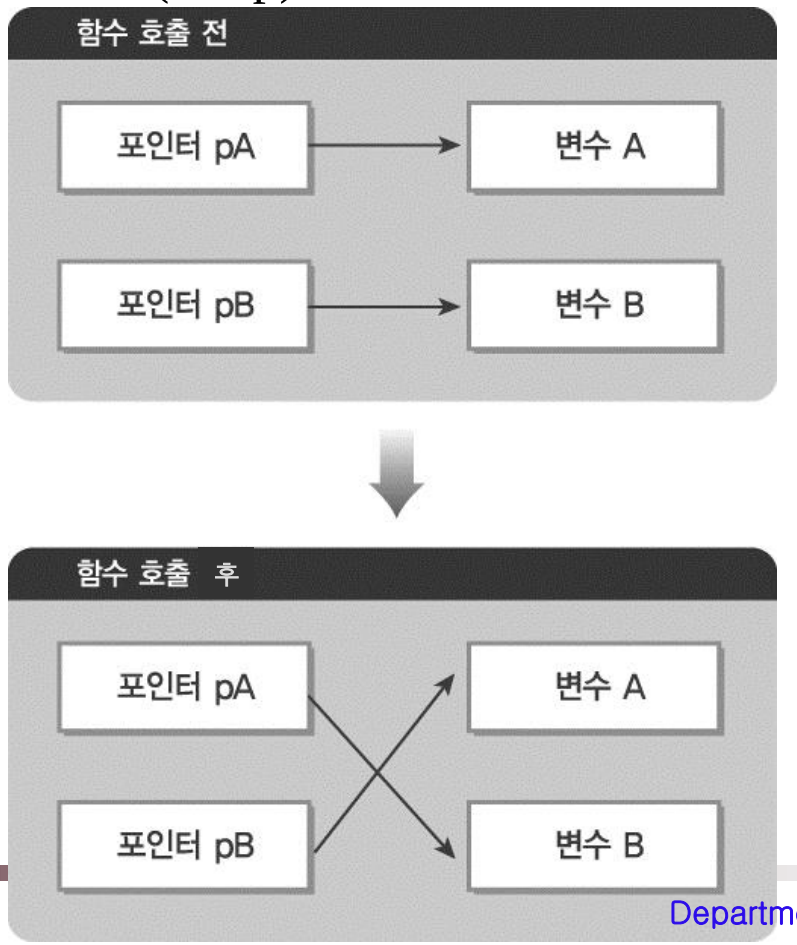
아버지가 사고치면 "할아버지 모시고 와"

pVal을 바꾸고 싶으면 더블 포인터 사용



17-1 포인터의 포인터?

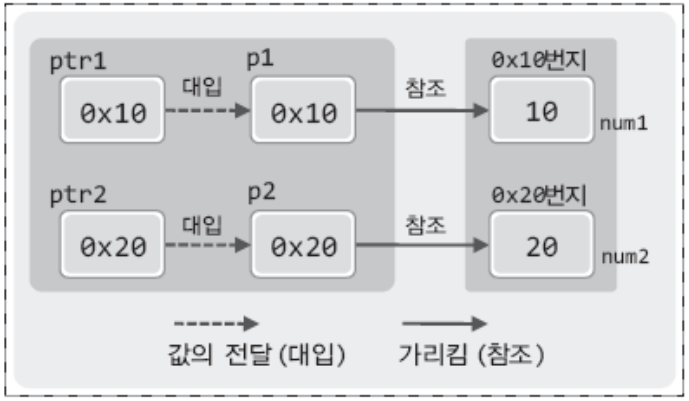
- 더블 포인터의 의한 Call-By-Reference
  - 포인터 바꿔치기 (swap) 예제

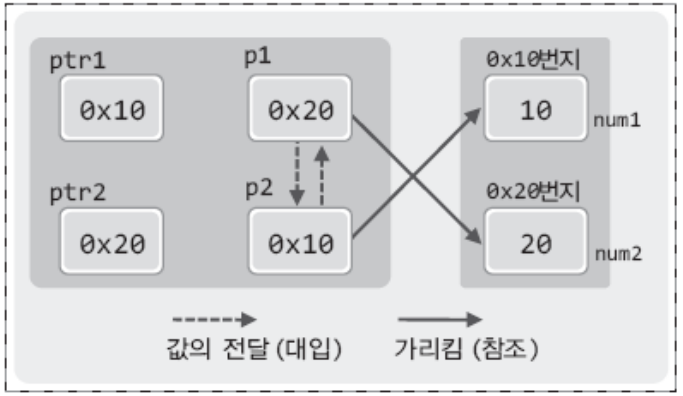


포인터 Swap (잘못된 예제)

```
void SwapIntPtr(int *p1, int *p2)
{
    int * temp=p1;
    p1=p2;
    p2=temp;
}

int main(void)
{
    int num1=10, num2=20;
    int *ptr1, *ptr2;
    ptr1=&num1, ptr2=&num2;
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);

    SwapIntPtr(ptr1, ptr2);
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);
    return 0;
}
```

ptr1과 ptr2의 swap은 성공하는가? 문제점은?






```
*ptr1, *ptr2: 10 20
*ptr1, *ptr2: 10 20
```

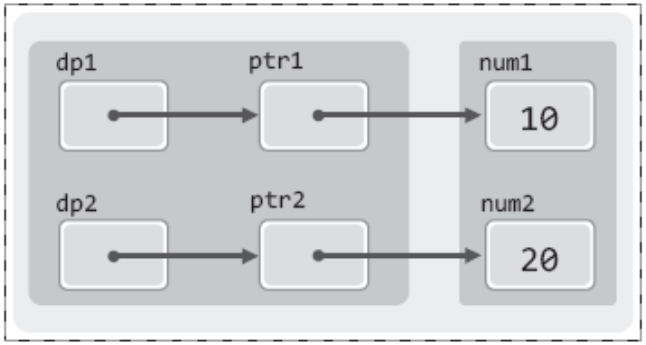
실행결과

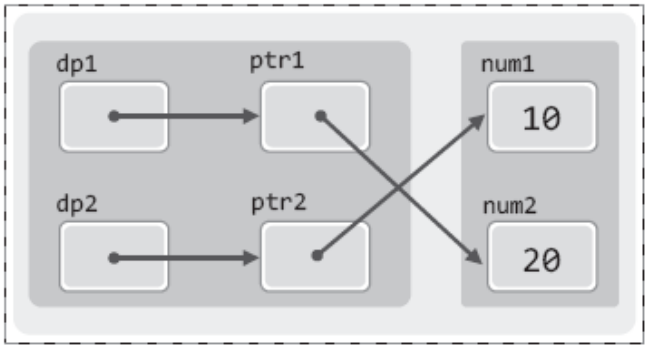
윗편 예제의 실행결과! 결과적으로 ptr1과 ptr2에 저장된 값은 서로 바뀌지 않는다.

포인터 Swap

```
void SwapIntPtr(int **dp1, int **dp2)
{
    int *temp = *dp1;
    *dp1 = *dp2;
    *dp2 = temp;
}

int main(void)
{
    int num1=10, num2=20;
    int *ptr1, *ptr2;
    ptr1=&num1, ptr2=&num2;
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);

    SwapIntPtr(&ptr1, &ptr2);  // ptr1과 ptr2의 주소 값 전달!
    printf("*ptr1, *ptr2: %d %d \n", *ptr1, *ptr2);
    return 0;
}
```

포인터 변수에 저장된 값의 변경이 목적이므로 포인터 변수의 주소 값을 함수에 전달해야 한다.





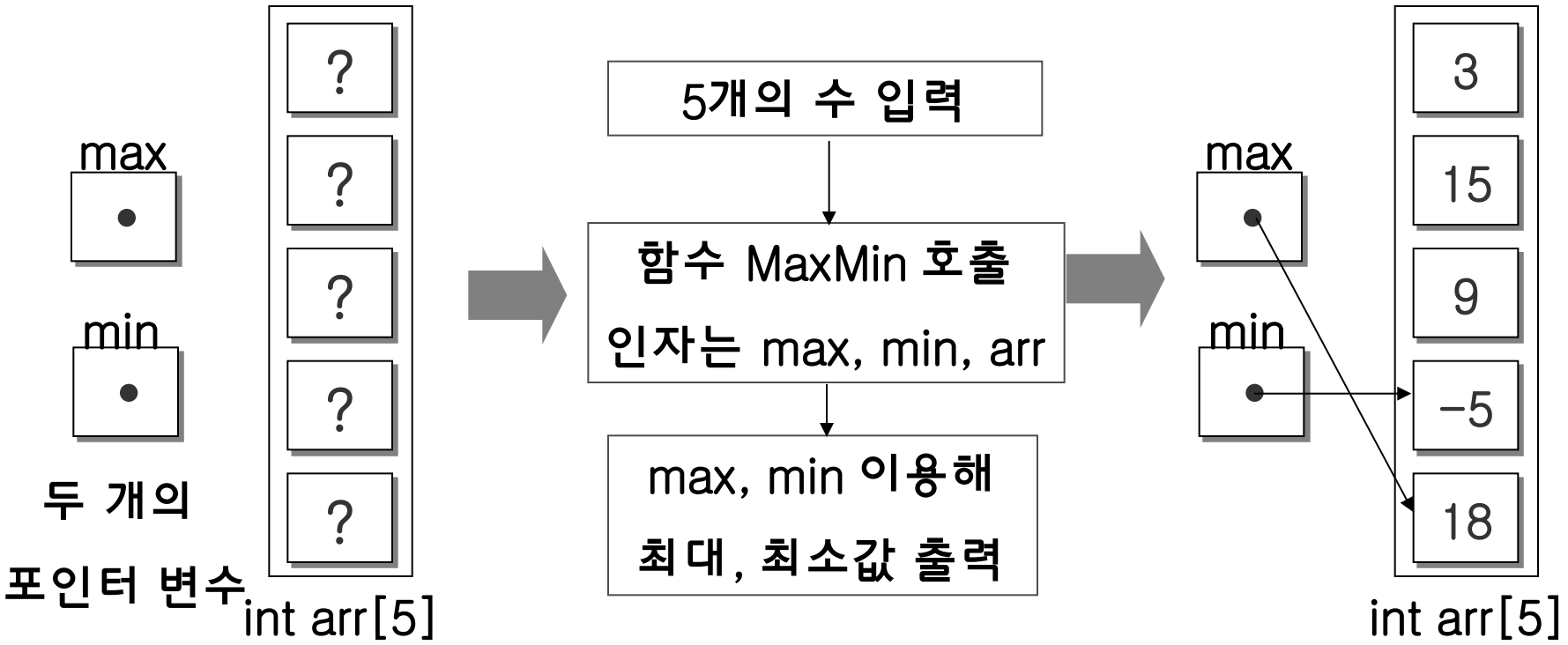
이중 포인터를 이용해서 두 포인터 변수의 swap에 성공한다.

```
*ptr1, *ptr2: 10 20
*ptr1, *ptr2: 20 10
```

실행결과

더블 포인터 연습문제

- 함수 MaxMin



더블 포인터 연습문제

```
void main(void)
{   …
    int *max = NULL, *min = NULL;
    MaxMin (arr, sizeof(arr)/sizeof(int), &max, &min);
    printf("Max: %d, Min: %d\n", *max, *min);
}
void MaxMin (int *arr, int size, int** mx, int** mn)
{
    int i, *max, *min;

    max = min = arr;
    for (i = 0; i<size; ++i) {
        if (*max < arr[i])
            max = &arr[i];
        if (*min > arr[i])
            min = &arr[i];
    }
    *mx = max; *mn = min;
 }
```

포인터들을 저장하는 배열

- 포인터 배열과 포인터 타입
  - 배열로 포인터들 저장

```
int* arr1[10];

double* arr2[20];

char* arr3[30];
```

Summary

- 14장 복습: 함수 내에서 변수의 값을 바꾸고 그 값을 호출한 놈이 써야 한다면 포인터를 써라.
  - Call-by-Reference라고 한다.
- 포인터를 함수로 전달하여 그 함수 내에서 포인터 변수의 값(주소)을 바꾸고 그 값을 호출한 놈이 써야 한다면 포인터의 포인터를 써라.
  - 포인터에 대한 Call-by-Reference
    - Double Pointer
- 포인터의 필요성
  - 메모리의 동적 할당
  - 자료 구조
  - Call-by-Reference