시스템프로그래밍기초 실습

Introduction to System Programming

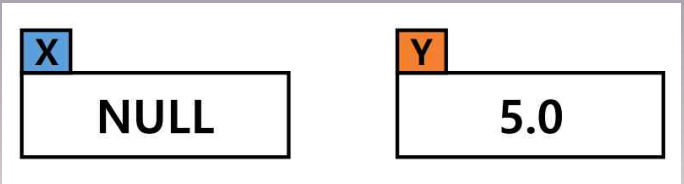
Pointer

call_by_value.c

```c
#include <stdio.h>

int compute_sum(int n);

int main(void)
{
        int n = 3, sum;
        printf("%d\n", n);
        sum = compute_sum(n);
        printf("%d\n", n);
        printf("%d\n", sum);
        return 0;
}

int compute_sum(int n)
{
        int sum = 0;
        for(;n > 0; --n) sum += n;
        return sum;
}
```

Declaration of pointers

```
double * x = NULL;
double y = 5;
```

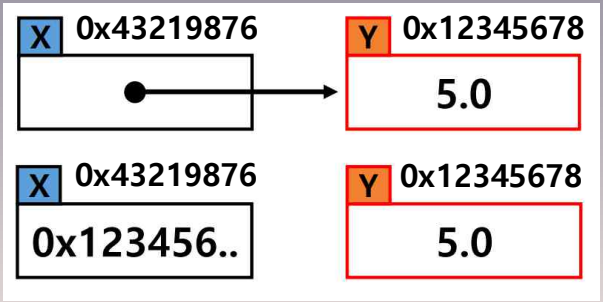


Declaration of pointers

```
double *x = NULL;
double y = 5;
x = &y;
```

```
double *x = &y; <-> double *x;
                    x = &y;
```

'선언할 때' 쓰는 '*'은 포인터 변수임을 알려주기 위한 목적 (포인터 연산자 *)
간접 참조 연산자 '*' 와 구분할 것



call_by_reference.c

```c
#include <stdio.h>

void swap(int *p, int *q);

int main(void)
{
        int  I = 3, j = 5;
        swap(&I, &j);
        printf("%d %d\n", I, j);
        return 0;
}

void swap(int *p, int *q)
{
        int tmp;
        tmp = *p;
        *p = *q;
        *q = tmp;
}
```

Multiple Dimensional Array & Pointer ptr1.c

```c
#include <stdio.h>
#include <stdlib.h>

void print_triple_array(const char *title, double p[3][4][3], int x, int y, int z){
        int i,j,k;
        printf("\nPrinting '%s' array\n", title);
        for(i = 0; i < x; i++){
                printf("[ ");
                for(j = 0; j < y; j++){
                        printf("{");
                        for(k = 0; k < z; k++){
                                printf("%3.0f",p[i][j][k]);
                        }
                        printf("}");
                        if(j!=y-1) printf(", ");
                }
                printf(" ]\n");
        }
        printf("\n");
}
```

Multiple Dimensional Array & Pointer ptr1.c

```c
int main()
{
        double a[3][4][3] = {
                {{ 1,  2,  3},{ 4,  5,  6},{ 7,  8,  9},{10, 11, 12}},
                {{13, 14, 15},{16, 17, 18},{19, 20, 21},{22, 23, 24}},
                {{25, 26, 27},{28, 29, 30},{31, 32, 33},{34, 35, 36}}
        };
        print_triple_array("a", a, 3, 4, 3);
        double (*b)[4][3];
        double *(*c)[4];
        int i, j, k;

        b = (double(*)[4][3])malloc(sizeof(double[3][4][3]));
        // Initialize b by a.

        print_triple_array("b",b,3,4,3);

        c = (double*(*)[4])malloc(sizeof(double*[4][3]));
        // Initialize c by b. Use double loops.

        printf("\nAssigned c by b.\n");
        printf("a[2][3] = %p\n", a[2][3]);
        printf("b[2][3] = %p\n", b[2][3]);
        printf("c[2][3] = %p\n", c[2][3]);
        printf("*c[2][3] = %3.0f\n", *c[2][3]);
        return 0;
}
```

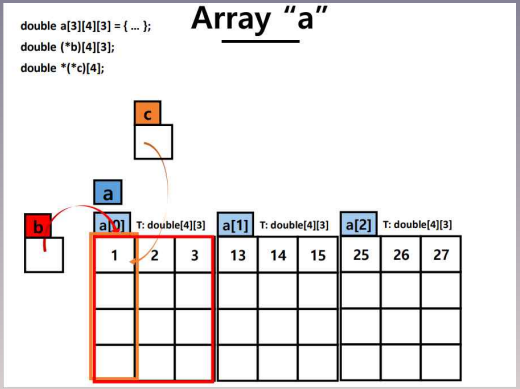
* 다차원 배열
  - type name[row][column] = { {...} , ... , {...} };
  - type name[layer][row][column] = { { {...} , ... {...} } , ... , { {...} , ... , {...} } };

  - 기억장소 사상 함수

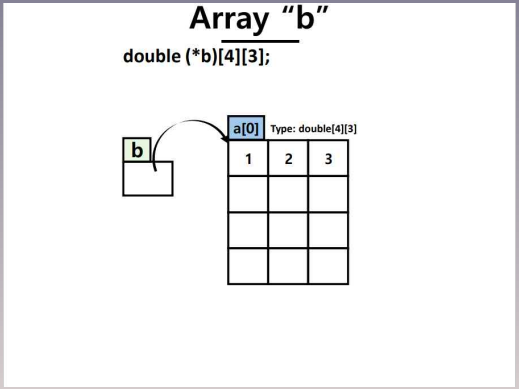
    int a[3][5];
    → 배열 a의 a[i][j]에 대한 기억장소 사상 함수
    *(&a[0][0] + **5** * i + j)
    int a[][5] : j값은 명시가 되어야함

    int a[7][9][2];
    → a[i][j][k]를 위한 기억장소 사상 함수;
    *(&a[0][0][0] + **9** * **2** * i + **2** * j + k)
    int a[][9][2] : j, k값은 명시가 되어야함

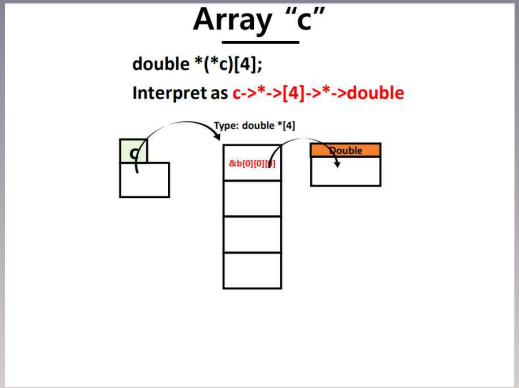
Multiple Dimensional Array & Pointer array “a”



Multiple Dimensional Array & Pointer array “b”



Multiple Dimensional Array & Pointer array “c”



Multiple Dimensional Array & Pointer Result

```
cpslab@www:~/workspace/cprog$ vim ptr1.c
cpslab@www:~/workspace/cprog$ gcc -o ptr1 ptr1.c
cpslab@www:~/workspace/cprog$ ./ptr1

Printing 'a' array
[ {  1  2  3}, {  4  5  6}, {  7  8  9}, { 10 11 12} ]
[ { 13 14 15}, { 16 17 18}, { 19 20 21}, { 22 23 24} ]
[ { 25 26 27}, { 28 29 30}, { 31 32 33}, { 34 35 36} ]


Printing 'b' array
[ {  1  2  3}, {  4  5  6}, {  7  8  9}, { 10 11 12} ]
[ { 13 14 15}, { 16 17 18}, { 19 20 21}, { 22 23 24} ]
[ { 25 26 27}, { 28 29 30}, { 31 32 33}, { 34 35 36} ]


Assigned c by b.
a[2][3] = 0x7ffc7f9f10f8
b[2][3] = 0x7ffc7f9f10f8
c[2][3] = 0x7ffc7f9f10f8
*c[2][3] =  34
cpslab@www:~/workspace/cprog$
```

끝