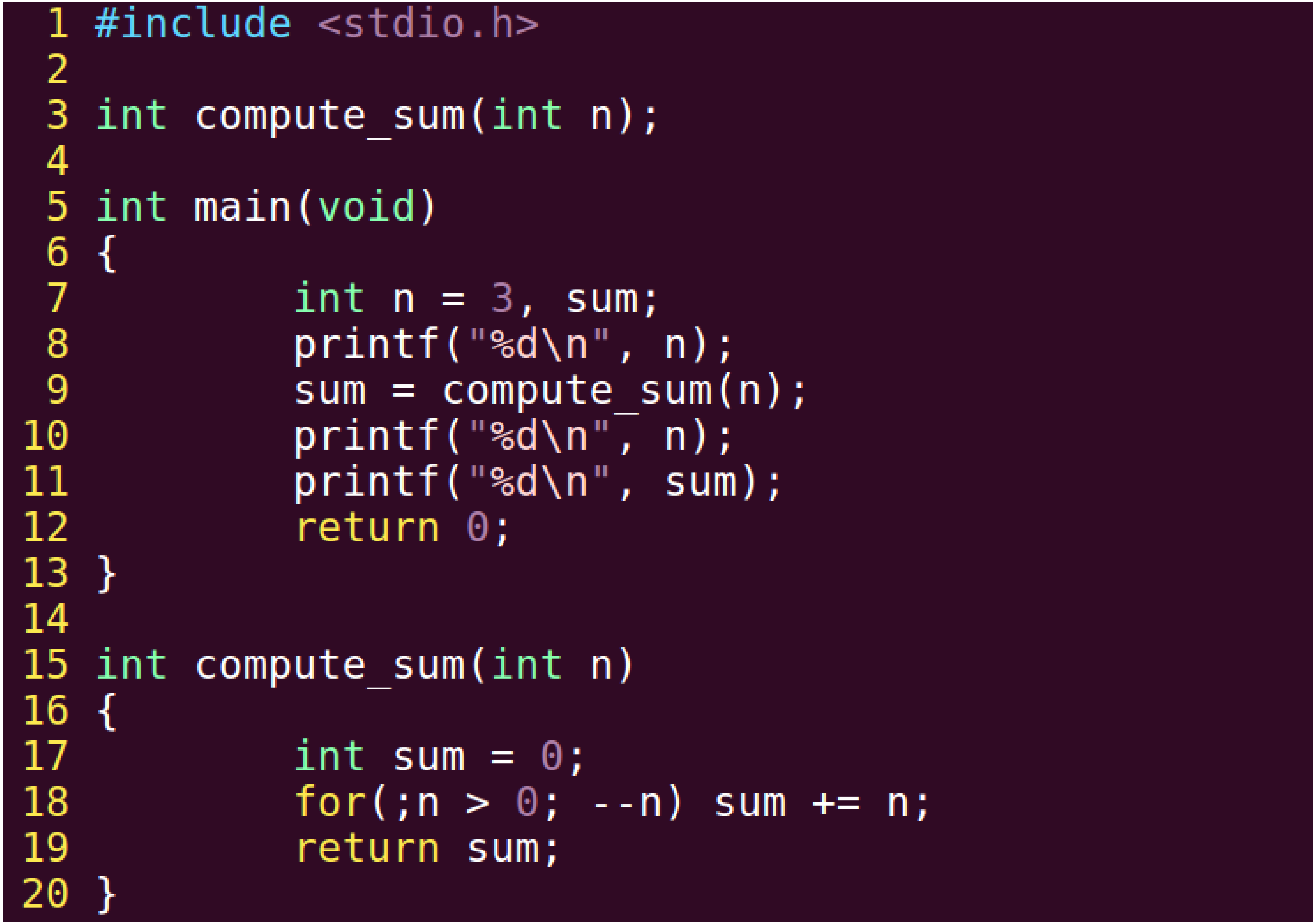
시스템프로그래밍기초 실습

Ch6. Arrays, Pointers, and Strings

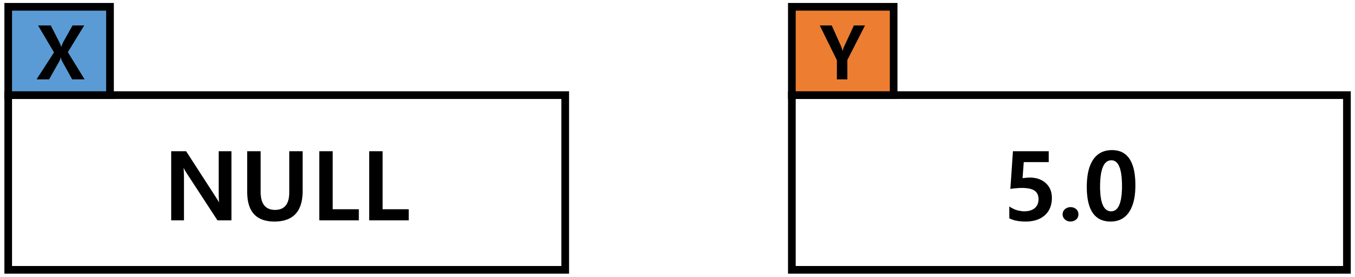
예제 1) call-by-value.c



```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;
}
```

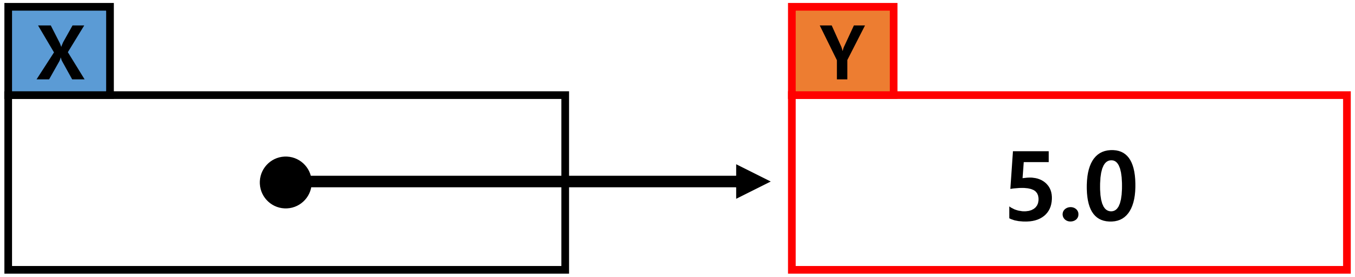
Declaration of pointers

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



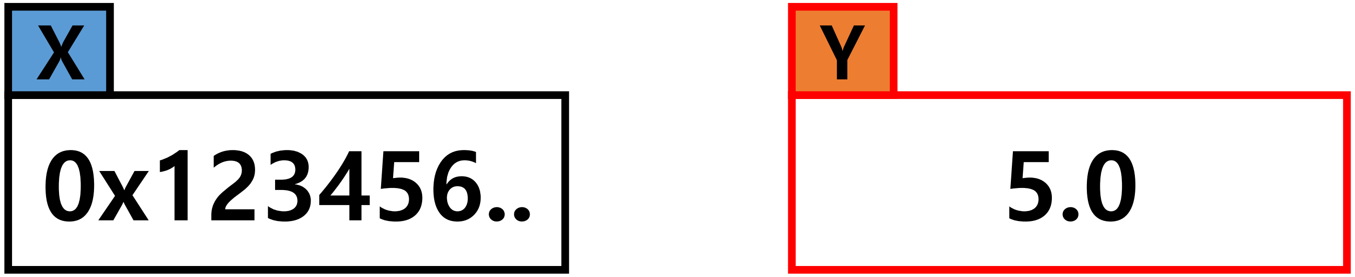
Declaration of pointers

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



Declaration of pointers

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



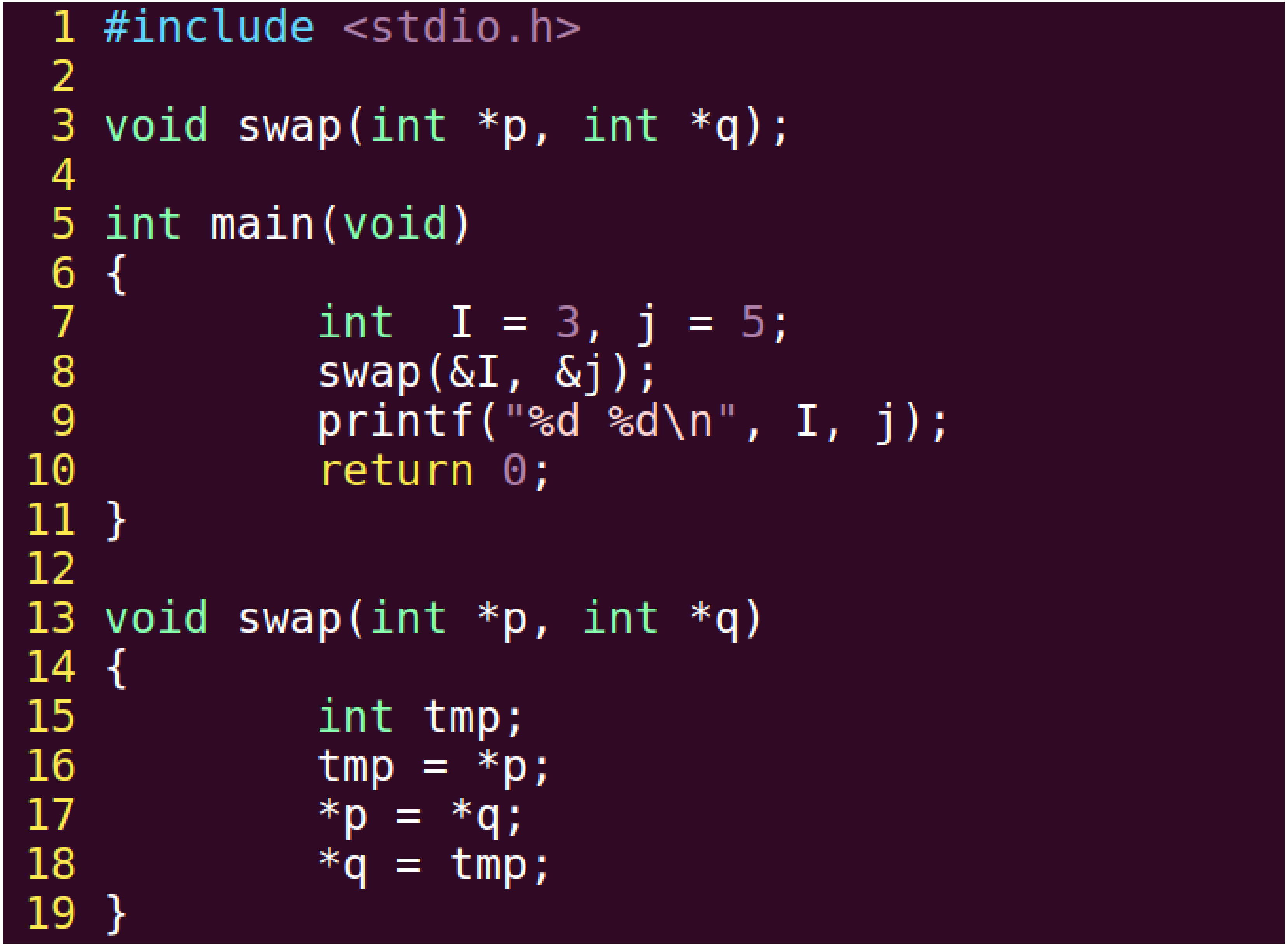
X

0x123456..

Y

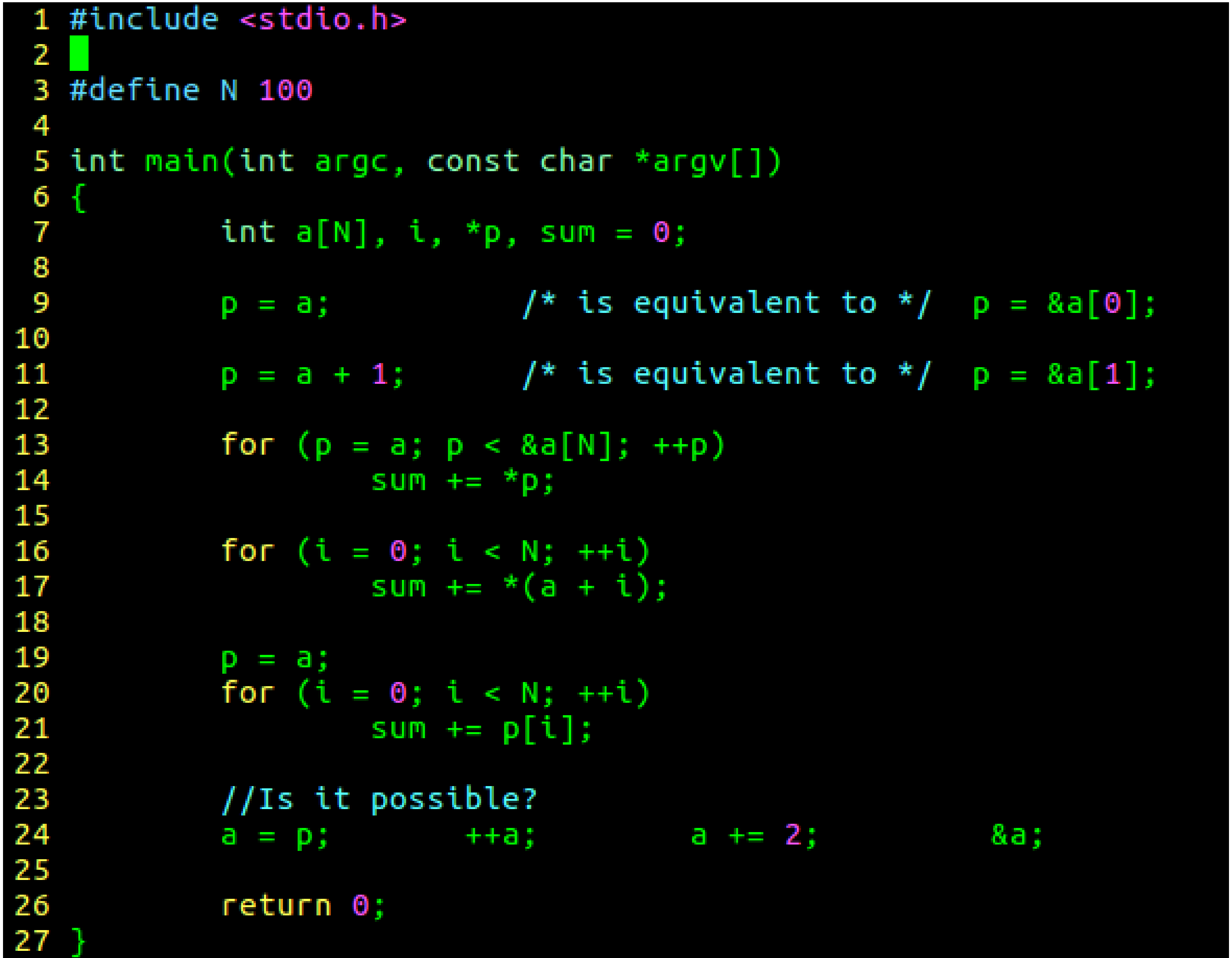
5.0

예제 2) 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;
}
```

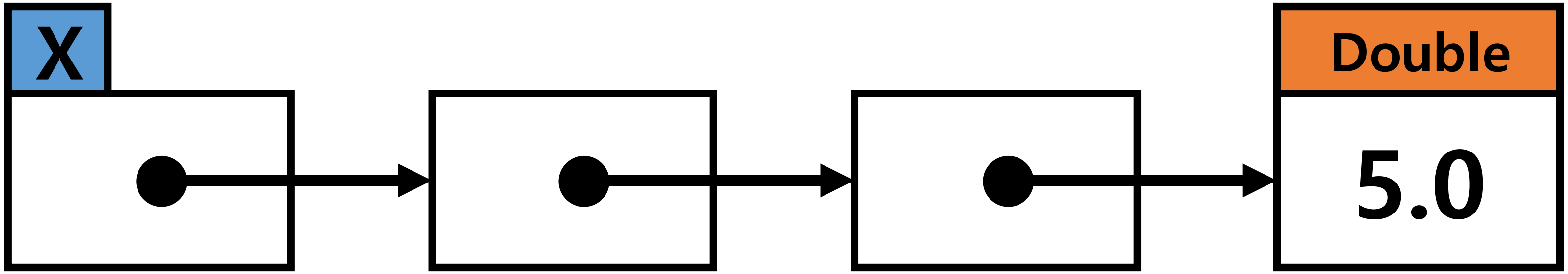
Equivalent expressions of Pointers and Arrays



```
#include <stdio.h>

#define N 100

int main(int argc, const char *argv[])
{
        int a[N], i, *p, sum = 0;

        p = a;                  /* is equivalent to */   p = &a[0];

        p = a + 1;              /* is equivalent to */   p = &a[1];

        for (p = a; p < &a[N]; ++p)
                sum += *p;

        for (i = 0; i < N; ++i)
                sum += *(a + i);

        p = a;
        for (i = 0; i < N; ++i)
                sum += p[i];

        //Is it possible?
        a = p;       ++a;        a += 2;          &a;

        return 0;
}
```

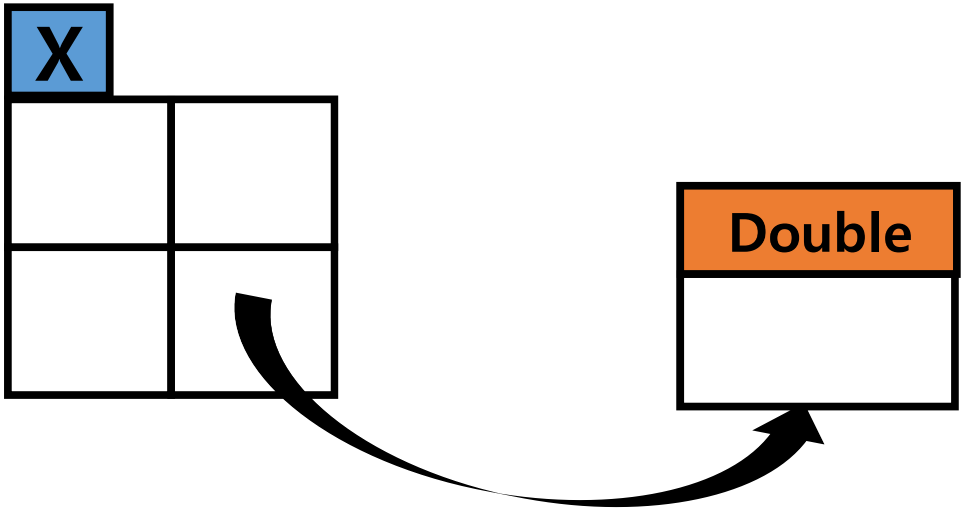
Declaration of Multiple Dimensional Arrays & Pointers

```
double *** x;
```



Declaration of Multiple Dimensional Arrays & Pointers

```
   ❺      ❹
double *x[2][2];
         ❶ ❷  ❸
```



How to draw the order of this pointer?

Declaration of Multiple Dimensional Arrays & Pointers

```
    ❺   ❹
double *(*x)[2][2];
          ❶   ❷  ❸
```

Casting :

```
    ❺    ❹❸
(double **[2][2])
            ❶  ❷
```

```
    ❺    ❹
(double *(*)[2][2])
           ❶  ❷  ❸
```

Arrays & Pointers를 이해하기

1. Interpreting the order of declarations

2. Visual structure of Arrays and Pointers

3. Assign arrays to appropriate pointers

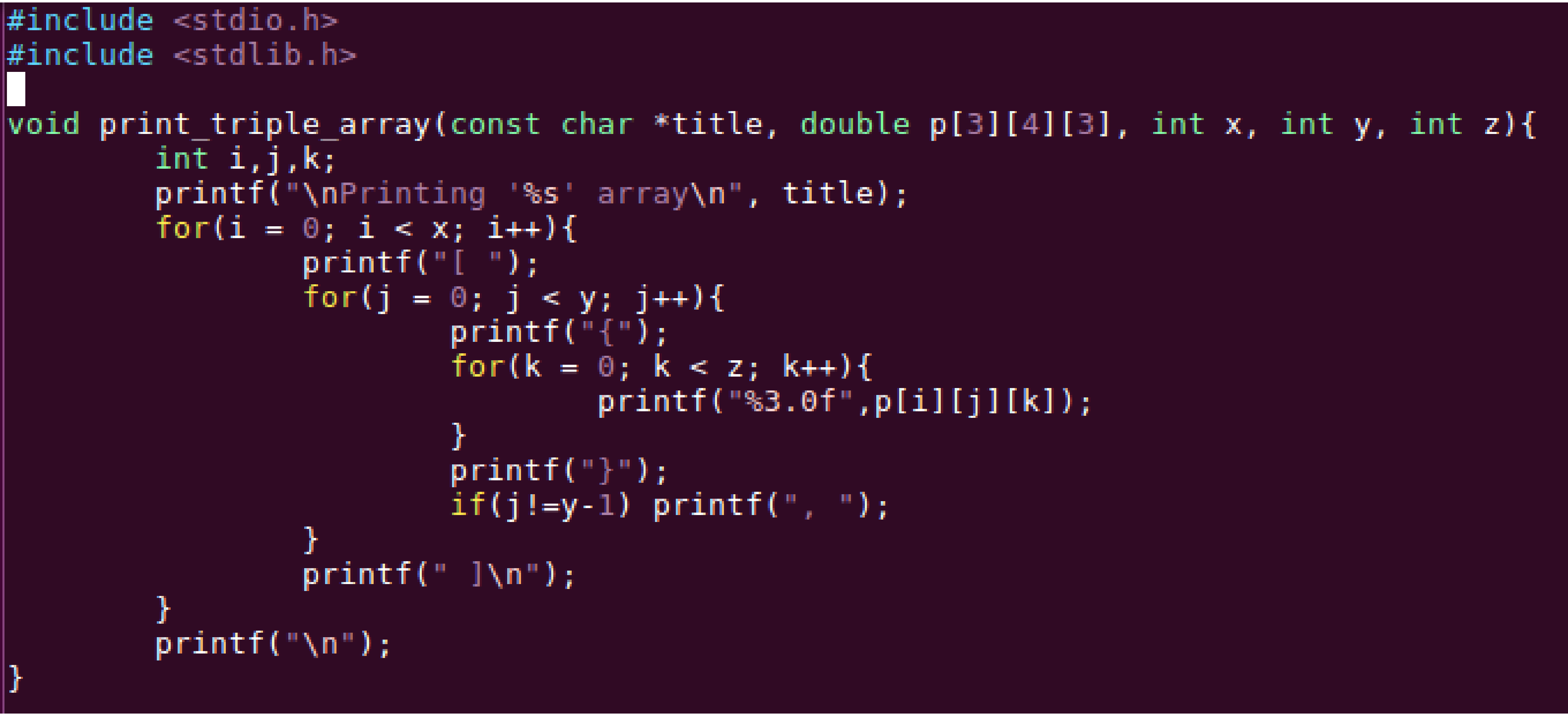
   (type match, assign address or value)

   Ex) Pointer = Pointer = Array (or Value)

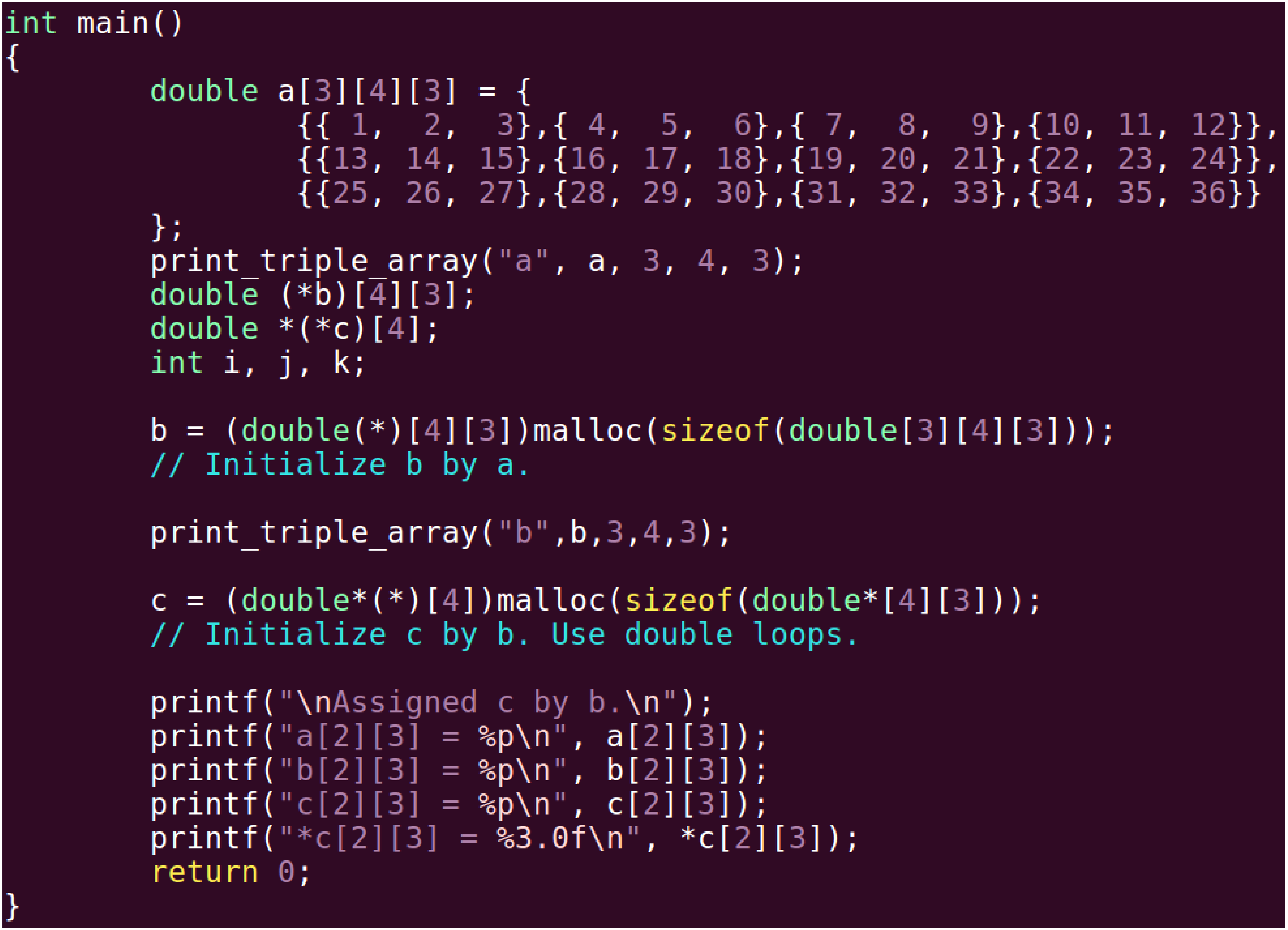
   WARNING is not allowed!

과제 1) Multiple Dimensional Array & Pointer

ptr1.c



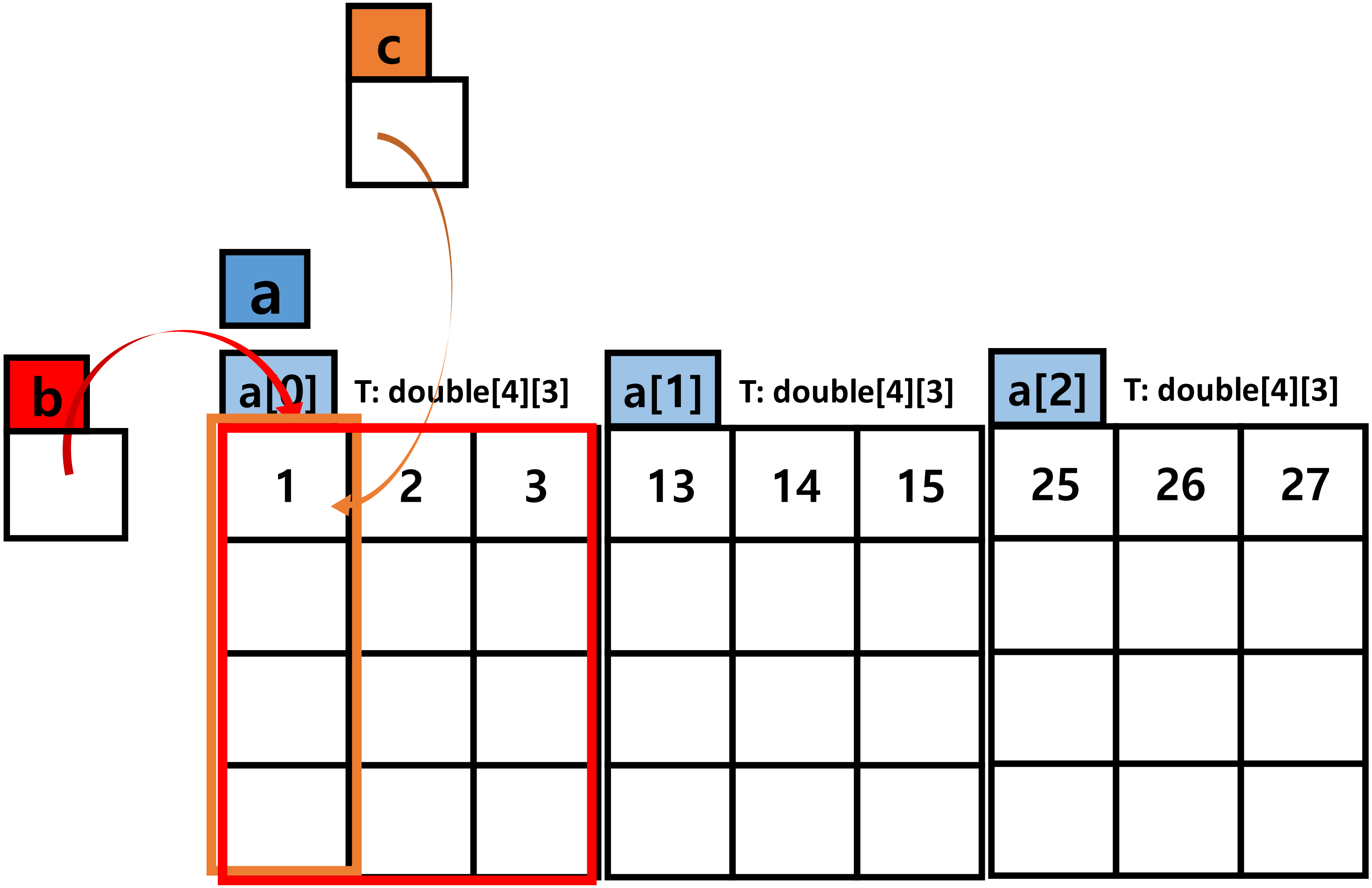
```c
#include <stdio.h>
#include <stdlib.h>

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");
}
```



```
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;
}
```

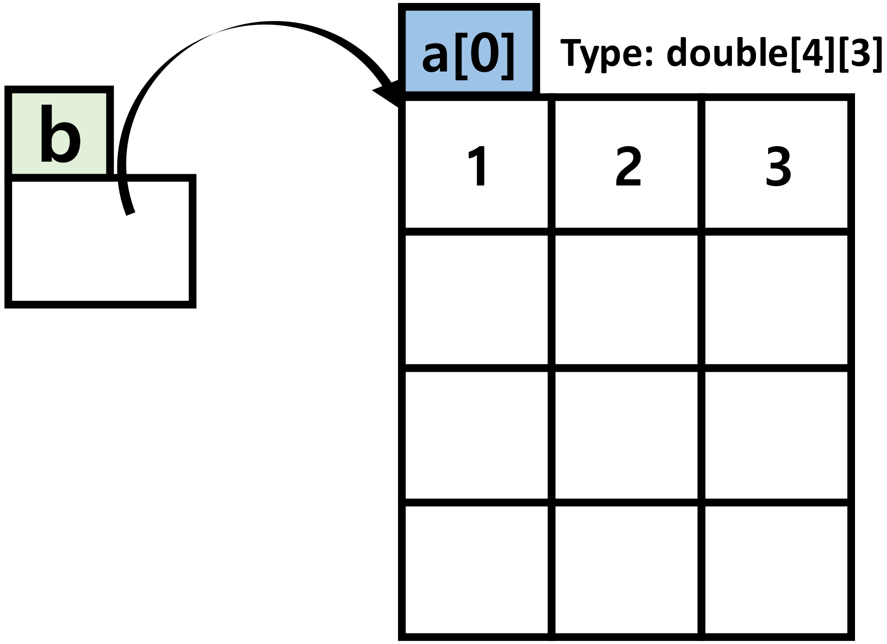
Array “a”

```
double a[3][4][3] = { … };
double (*b)[4][3];
double *(*c)[4];
```



Array “b”

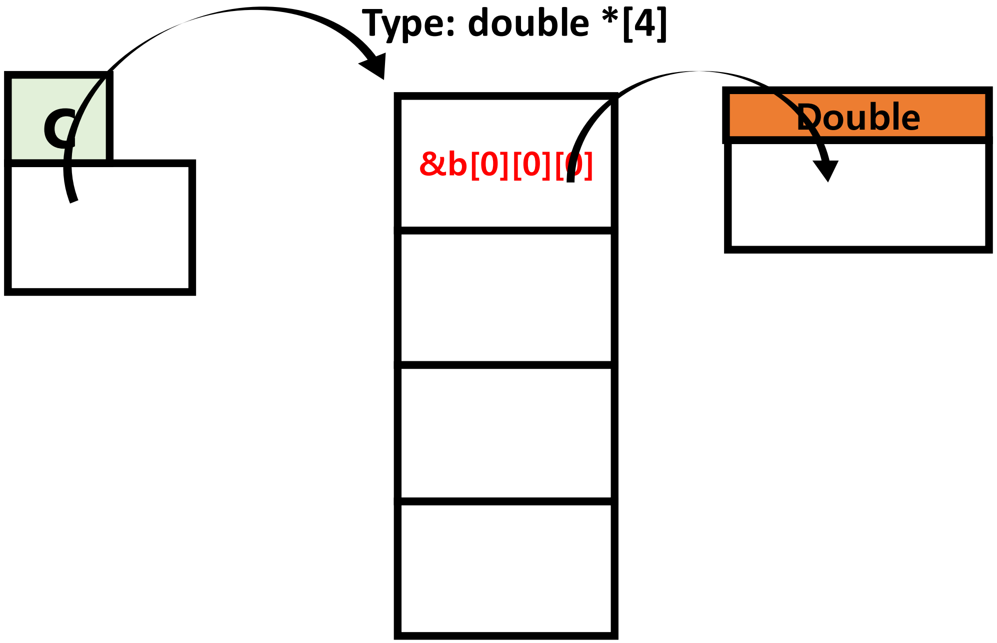
double (*b)[4][3];



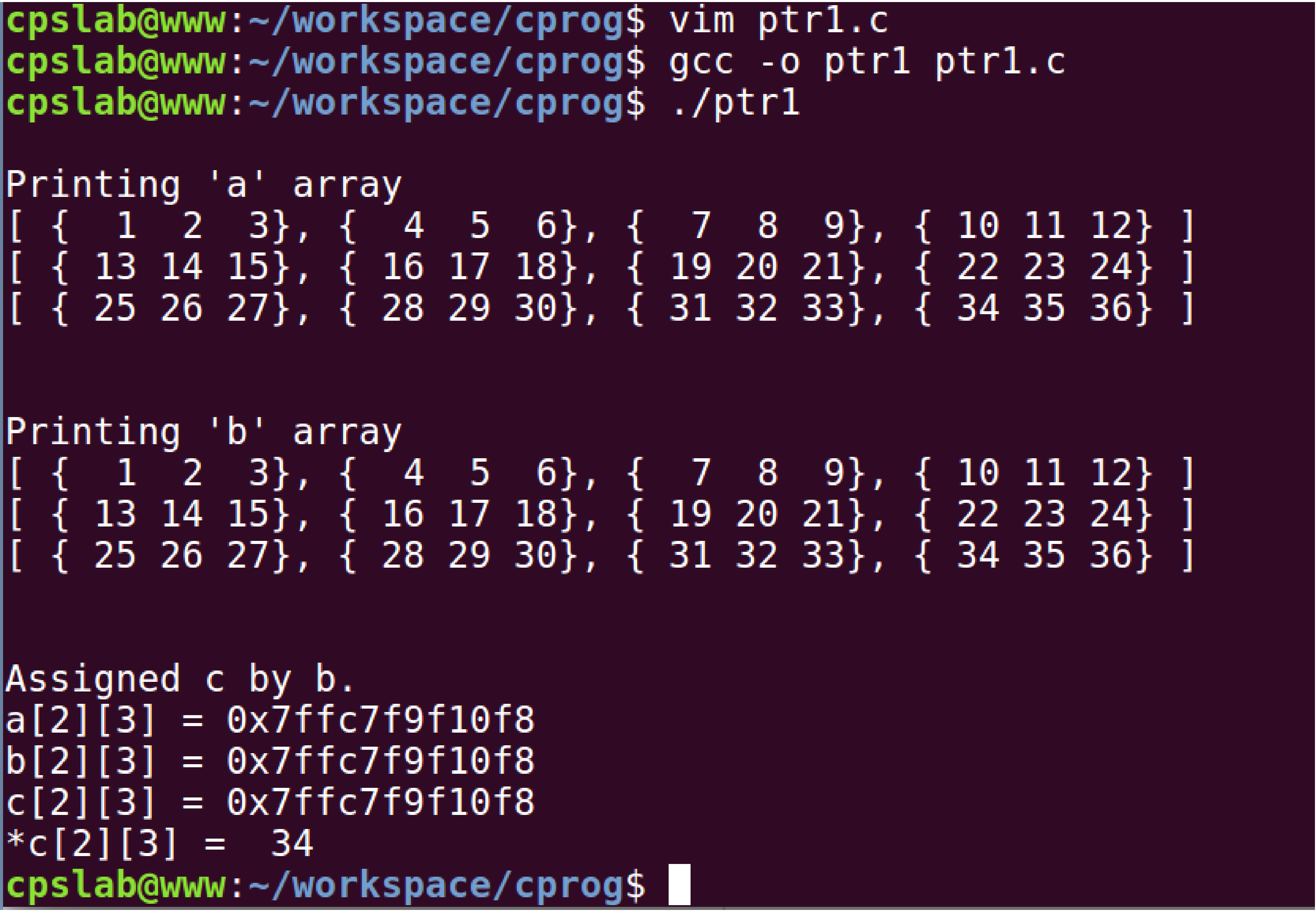
Array "c"

double *(*c)[4];

Interpret as c->*->[4]->*->double

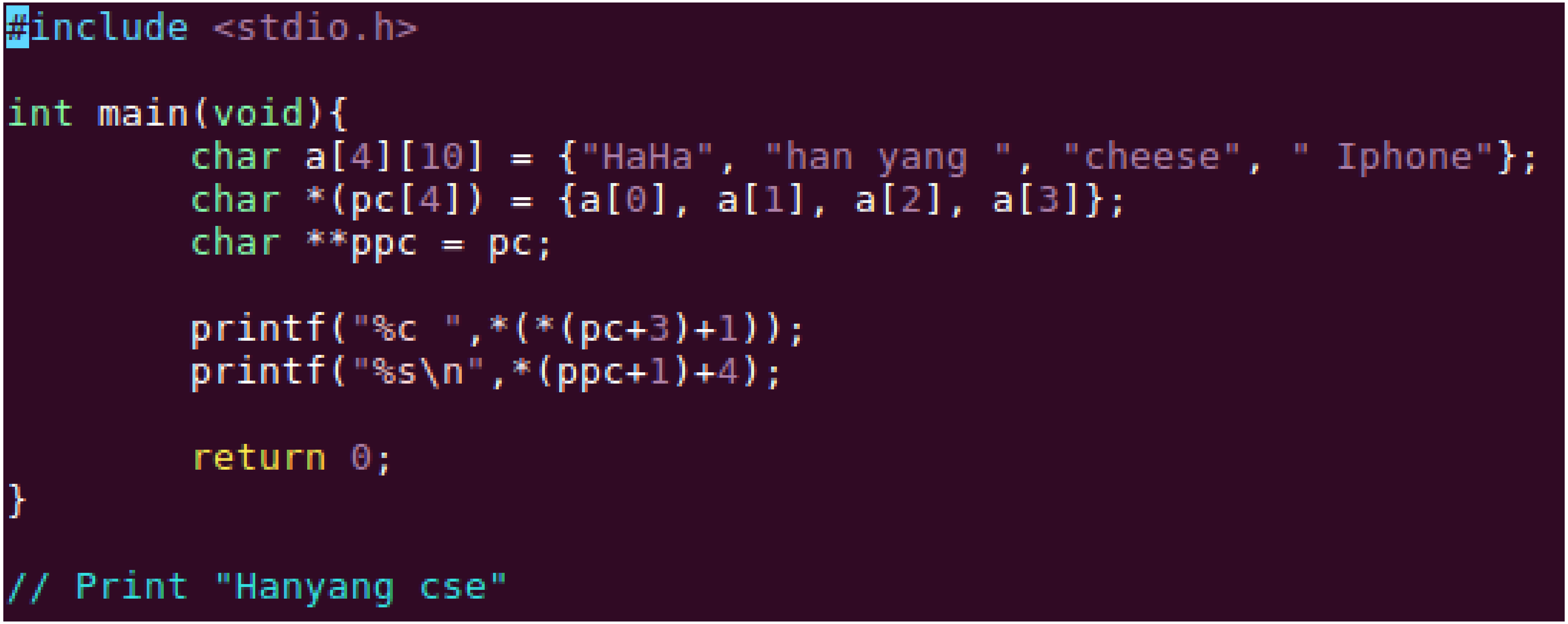


Do you see the difference between C and B?



```
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$ 
```

과제 2) String Pointer (ptr2.c)



```c
#include <stdio.h>

int main(void){
        char a[4][10] = {"HaHa", "han yang ", "cheese", " Iphone"};
        char *(pc[4]) = {a[0], a[1], a[2], a[3]};
        char **ppc = pc;

        printf("%c ",*(*(pc+3)+1));
        printf("%s\n",*(ppc+1)+4);

        return 0;
}

// Print "Hanyang cse"
```

{ ppc, pc, *, +, -, 정수 }만을 사용하시요.

“Hanyang cse”를 출력하세요.

과제 검사방법

과제 1) Multiple Dimensional Array & Pointer

1. 코드 빈칸 작성하기
2. 슬라이드와 똑같은 결과값 출력하기

과제 2) String pointer

1. 제공된 소스를 통해 출력하기