Basic and Practice in Programming

Week 9

Functions

- Two types of function call
  - Call by value
  - Call by reference

```
void func1(int a)
{
    …
}
```

Call by value

- Call by value
  - Argument is just “value”

```
void func2(int *a)
{
    …
}
```

Call by reference

- Call by reference
  - Argument is given by “variable’s address”

Call by Value

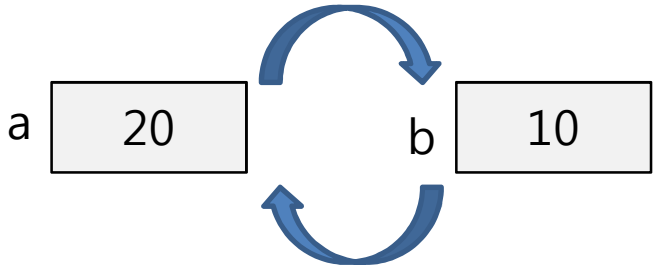
```
/* Practice 1 : Call by value */
#include <stdio.h>

int add (int x, int y)
{
   return x + y;
}


int main(void)
{
   printf("%d + %d = %d₩n", 10, 20, add(10, 20));

   return 0;
}
```

Call by Reference (1/3)

- Swap function
  - Exchange two variable's contents

```
int a = 10, b = 20;
```


```
swap(a, b);
```



Call by Reference (2/3)

```
/* Practice 2 :
    Call by reference example1*/
#include <stdio.h>

void swap (int x, int y)
{
   int temp;
   temp = y;
   y = x;
   x = temp;
}
```

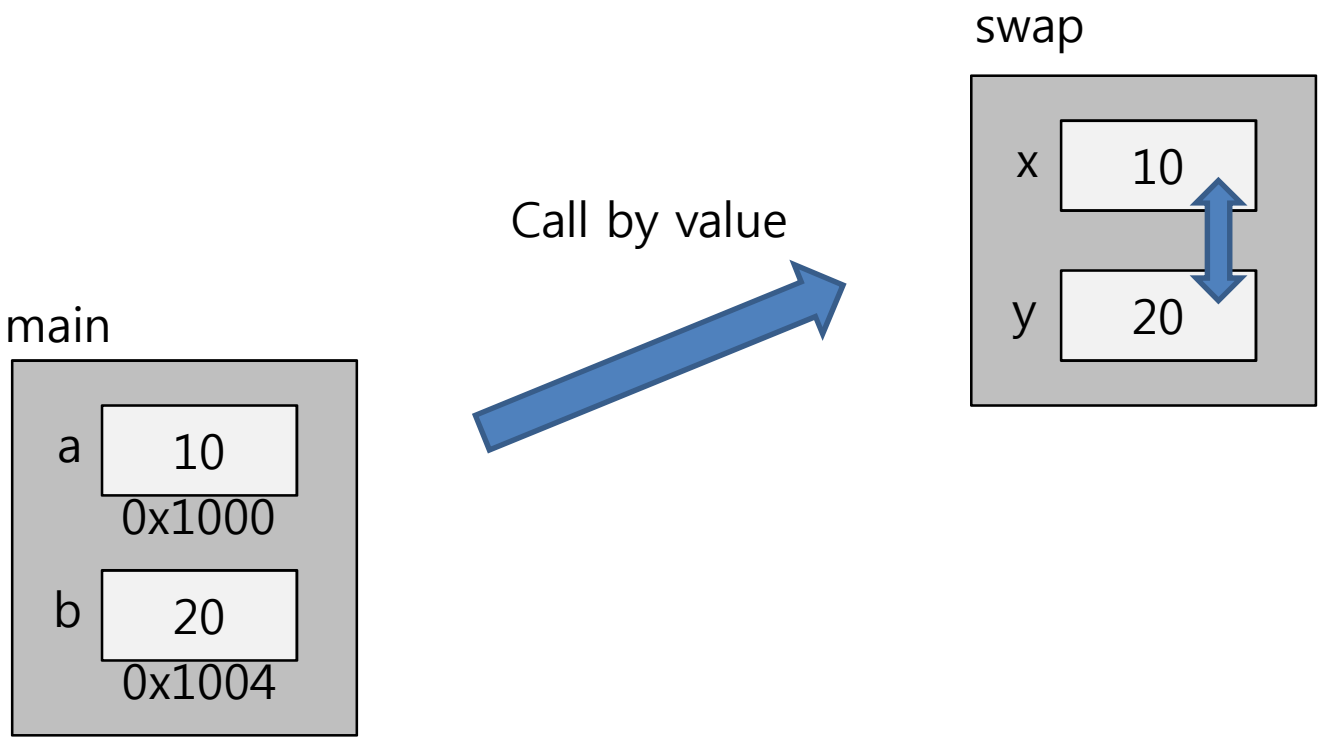
```
int main(void)
{
   int a = 10, b = 20;

   printf("a : %d b : %d", a, b);
   swap(a, b);
   printf("a : %d b : %d", a, b);

   return 0;
}
```

Call by Reference (3/3)

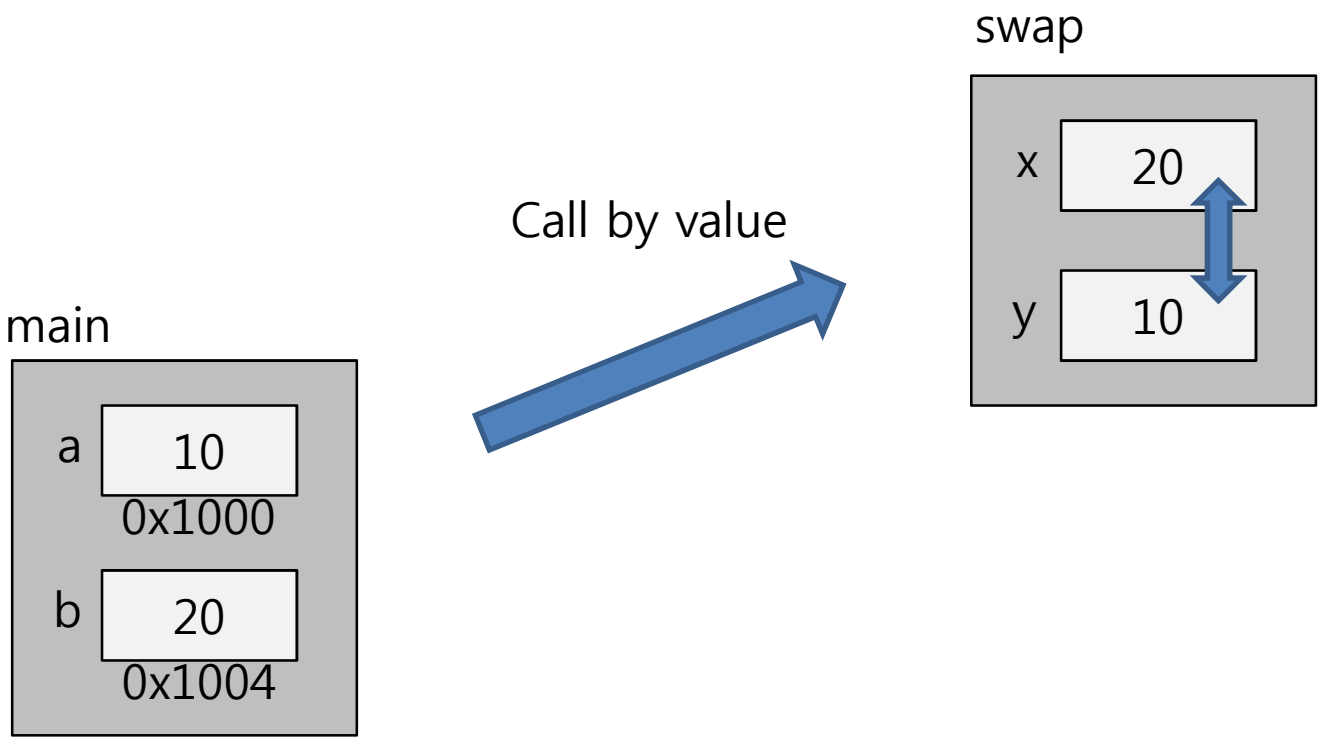
```
/* Practice 3 :
    Call by reference example2*/
#include <stdio.h>

void swap (int *x, int *y)
{
   int temp;
   temp = *y;
   *y = *x;
   *x = temp;
}
```

```
int main(void)
{
   int a = 10, b = 20;

   printf("a : %d b : %d", a, b);
   swap(&a, &b);
   printf("a : %d b : %d", a, b);

   return 0;
}
```

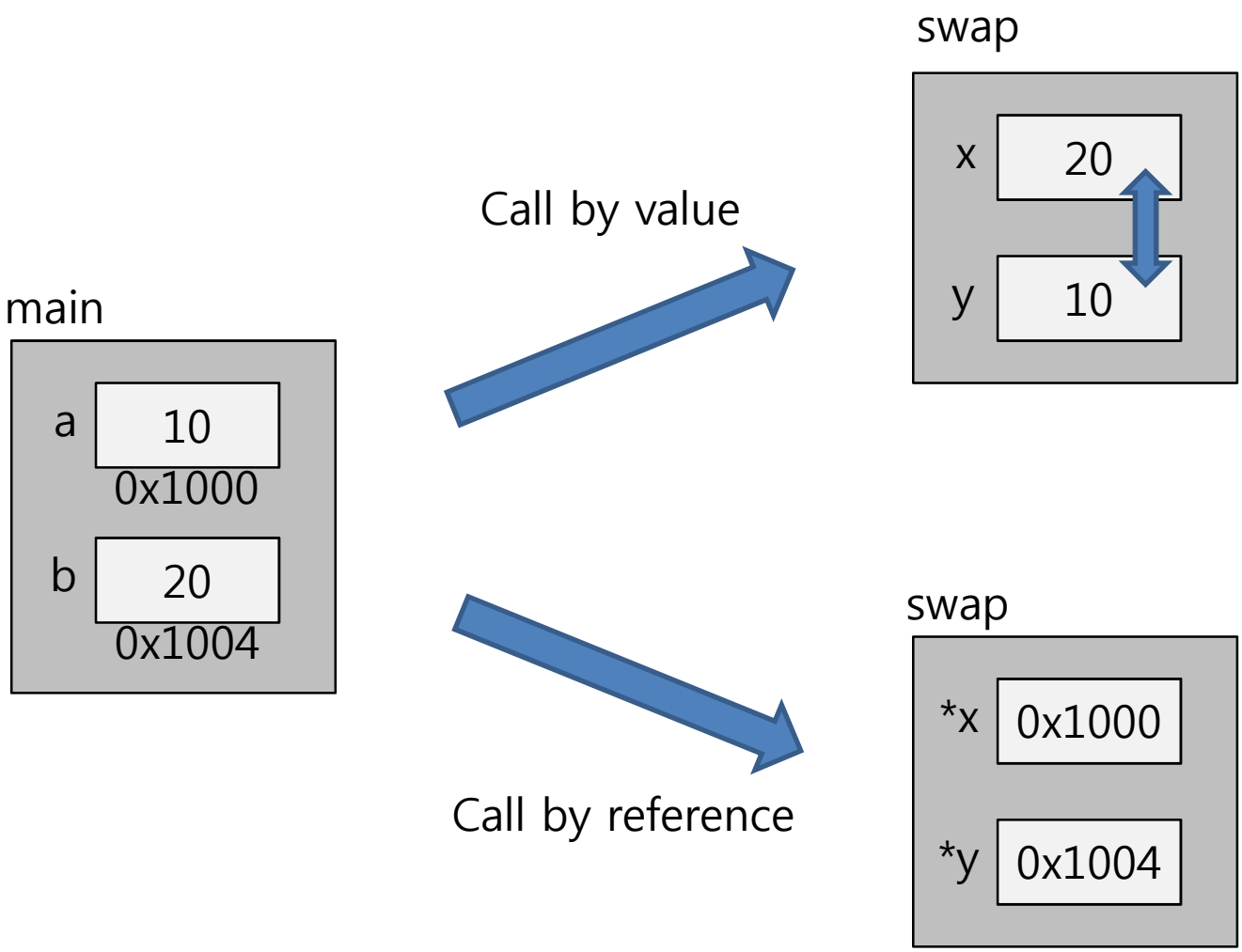
Reference?



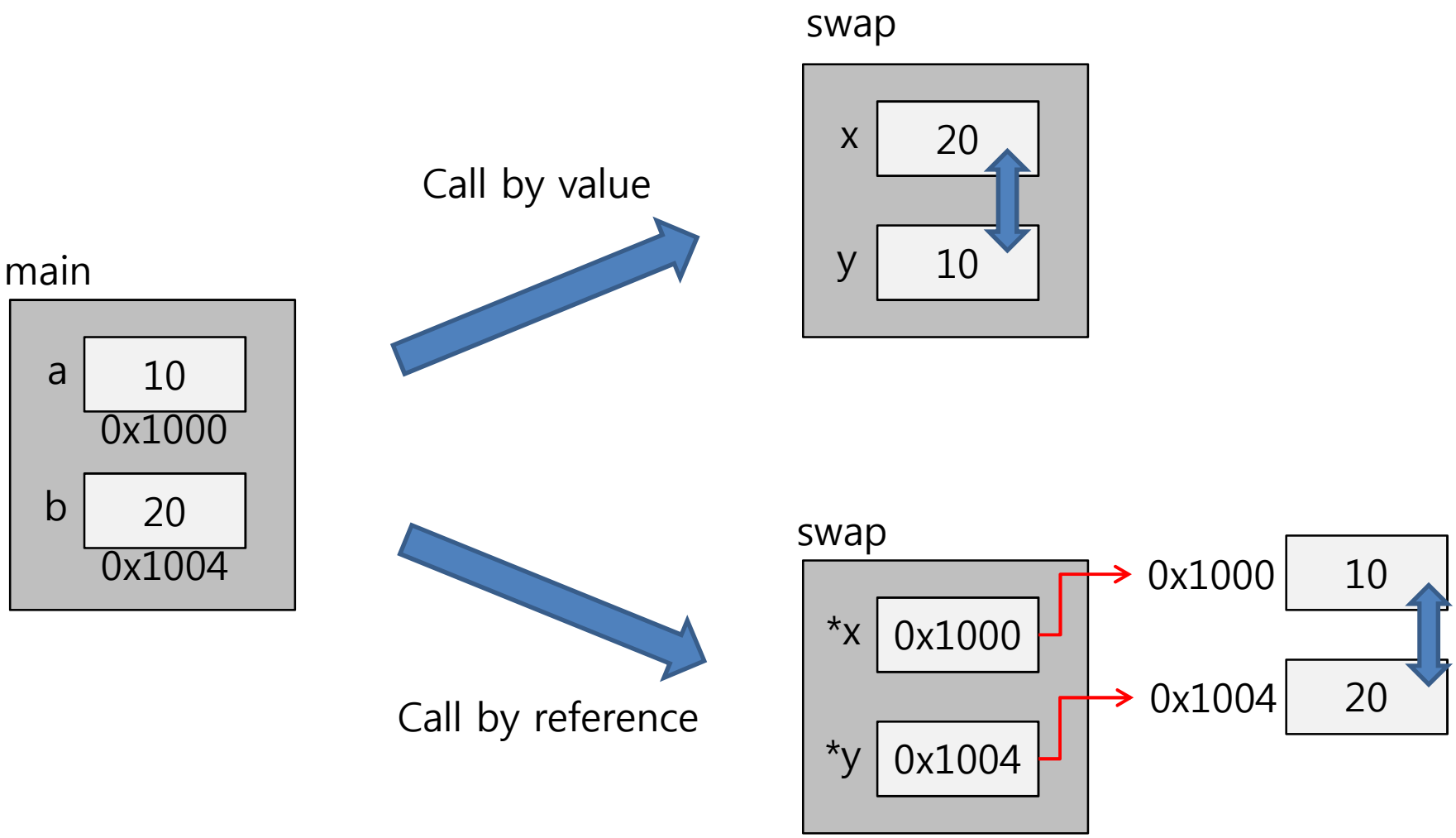
Reference?



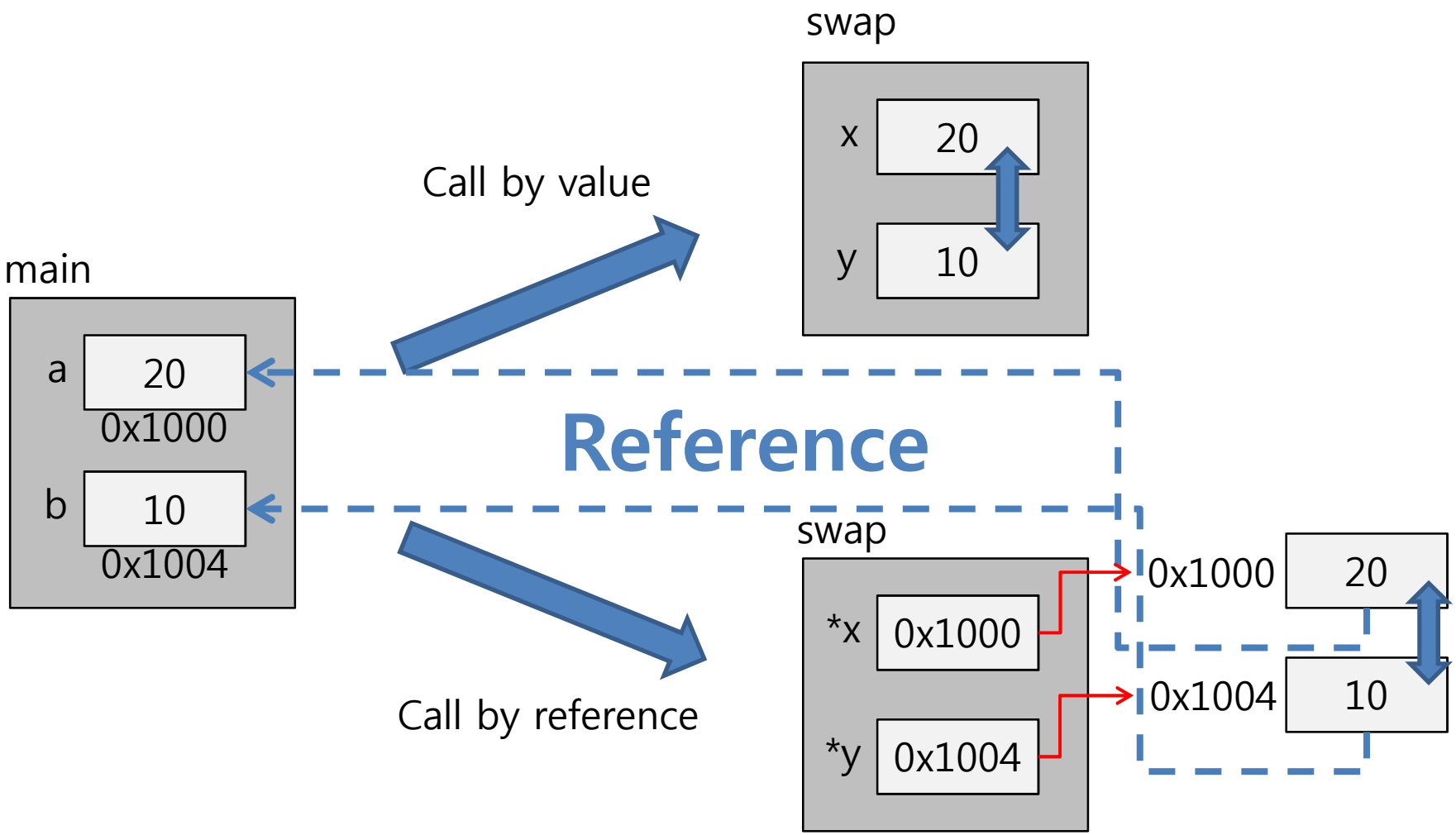
Reference?


swap
x
20
y
10
Call by value
main
a
10
0x1000
b
20
0x1004
swap
*x
0x1000
*y
0x1004
Call by reference

Reference?


swap
x
20
y
10
Call by value
main
a
10
0x1000
b
20
0x1004
swap
*x
0x1000
*y
0x1004
0x1000
10
0x1004
20
Call by reference

Reference?


swap
x
20
y
10
Call by value
main
a
20
0x1000
b
10
0x1004
Reference
swap
*x
0x1000
*y
0x1004
Call by reference
0x1000
20
0x1004
10

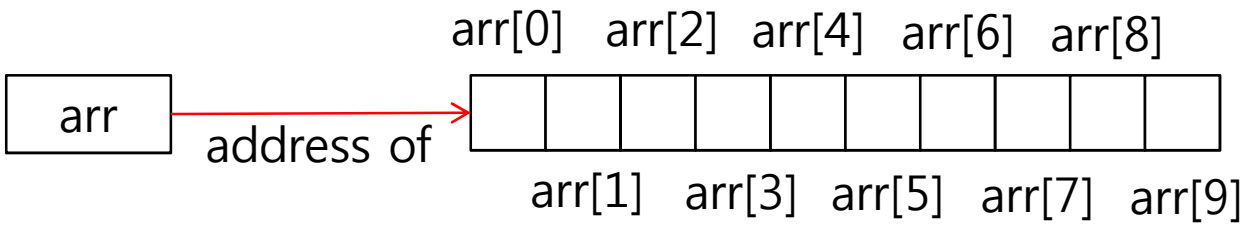
Array (1/4)

- Pointer
  - Container of memory address
  - Can access variable's value via '*'

```
int *p;
p = &a;
```

- Array
  - Reference of continuous memory addresses
  - '*' operator?

```
int arr[10]
```




```
arr == &arr[0]
*arr?
```

Array (2/4)

```
/* Practice 4 : Array 1*/
#include <stdio.h>

int main (void)
{
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

    printf("%p %p₩n", &arr[i], arr);
    printf("%d %d₩n", arr[i], *arr);
    //arr = 10; wrong statement

    return 0;
}
```

Array (3/4)

```
/* Practice 5 : Array 2*/
#include <stdio.h>

int main (void)
{
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int i;

    for (i=0;i<10;i++)
        printf("%p %p₩n", &arr[i], arr+i);

    for (i=0;i<10;i++)
        printf("%d %d₩n", arr[i], *(arr+i));

    return 0;
}
```

arr[i]

==

arr+i

==

&arr[0] + i*sizeof(int)

Array (4/4)

```
/* Practice 6 : Array 3*/
#include <stdio.h>

int main (void)
{
    char arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int i;

    for (i=0;i<10;i++)
        printf("%p %p₩n", &arr[i], arr+i);

    for (i=0;i<10;i++)
        printf("%d %d₩n", arr[i], *(arr+i));

    return 0;
}
```

arr[i]

==

arr+i

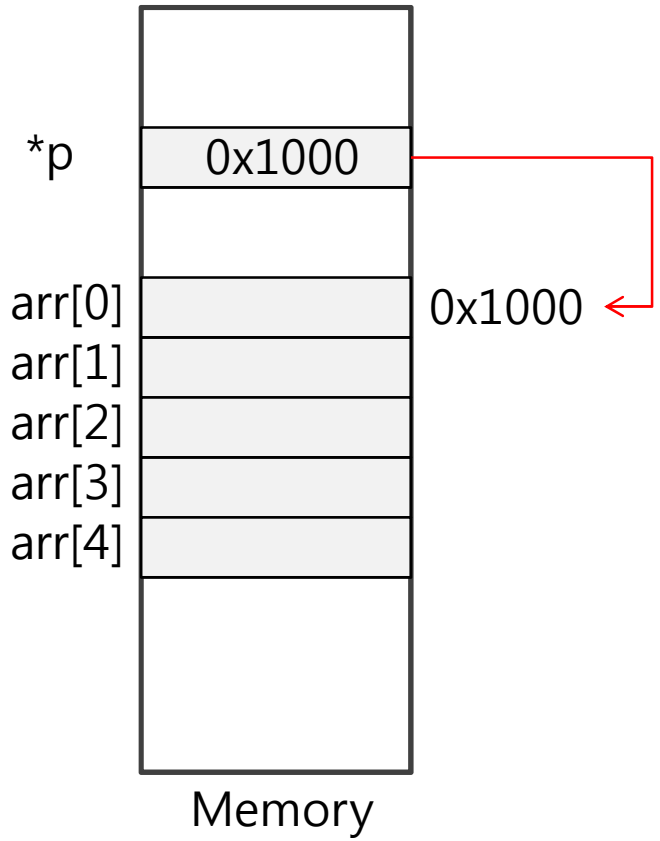
==

&arr[0] + i*sizeof(**char**)

Pointer (1/6)

- Pointer can be used as array

```
int arr[5];
int *p;
p = &arr[0]; //same as p = arr;
```



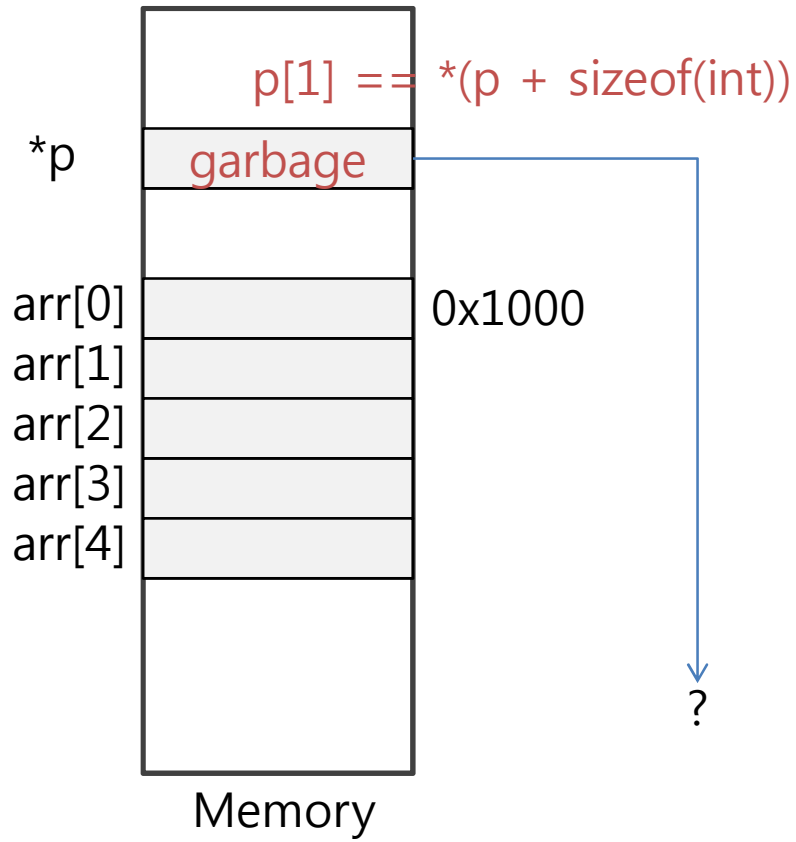
Pointer (2/6)

```
/* Practice 7 : Pointer 1*/
#include <stdio.h>

int main (void)
{
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int i;
    int *p;
    p = &arr[0]; //same as p = arr

    for (i=0;i<10;i++)
        printf("%p %p %p₩n", &p[i], p+i, &arr[i]);

    for (i=0;i<10;i++)
        printf("%d %d %d₩n", p[i], *(p+i), arr[i]);

    return 0;
}
```

Pointer (3/6)

- Pointer must be carefully used

```
int arr[5];
int *p;
printf("%d₩n", p[1]);
```



Pointer (4/6)

```
/* Practice 8 : Pointer misuse*/
#include <stdio.h>

int main (void)
{
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int i;
    int *p;
    //p = &arr[0];

    for (i=0;i<10;i++)
        printf("%p %p %p₩n", &p[i], p+i, &arr[i]);

    for (i=0;i<10;i++)
        printf("%d %d %d₩n", p[i], *(p+i), arr[i]);

    return 0;
}
```

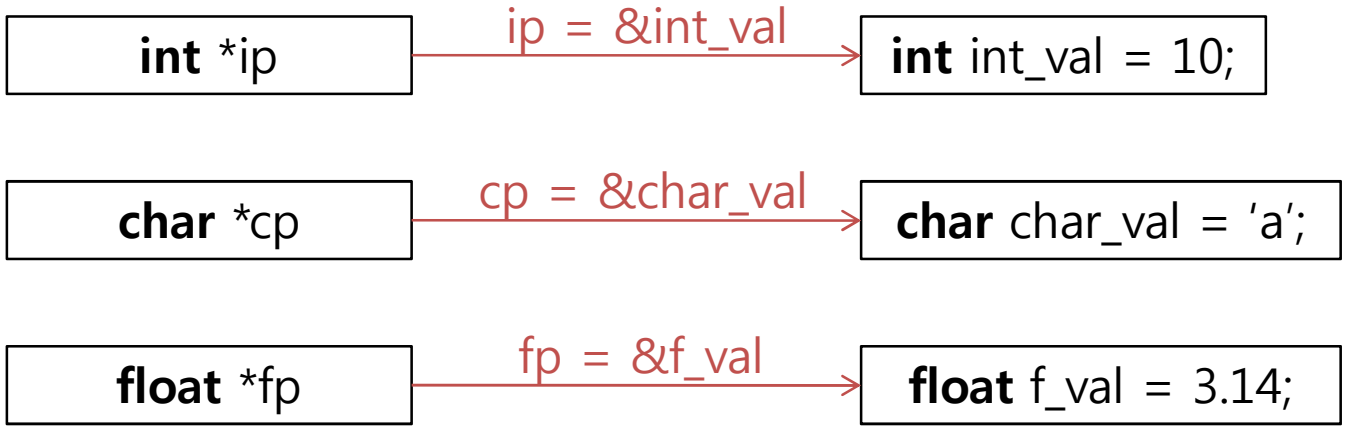
Pointer (5/6)

```
/* Practice 9 : Pointer 2*/
#include <stdio.h>

int string_length(char *s) {
    int n;
    for (n = 0;*s != '₩0';s++)
        n++;
    return n;
}
```

```
int main (void)
{
    char str[10];

    scanf("%s", &str[0]);

    printf("%s₩nLength : %d₩n", str, string_lenth(str));

    return 0;
}
```

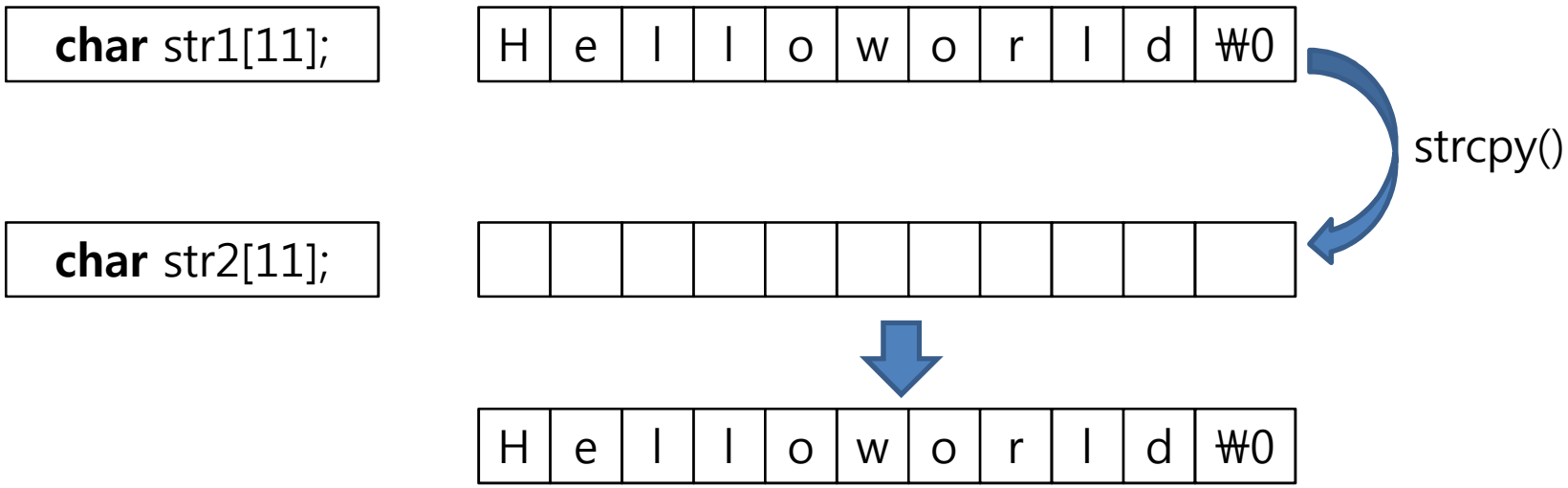
Pointer (6/6)

- Pointer has a ‘type’
  - Integer, character, floating point…
  - Pointer must be used to point to same type variable



Pointer Example (1/3)

- String copy
  - "strcpy()" is defined in standard C library (string.h)



Pointer Example (2/3)

```
/* Practice 11 : String copy – array*/
#define STRMAX 100
void string_copy(char *s, char *d) {
    int i=0;
    while(s[i] != '₩0') {
        d[i] = s[i];
        i++;
    }
    d[i] = s[i];
}
```

```
int main (void)
{
    char str1[STRMAX], str2[STRMAX];

    scanf("%s", str1); //str1 == &str1[0]
    string_copy(str1, str2);

    printf("str1 : %s₩nstr2 : %s₩n", str1, str2);

    return 0;
}
```

Pointer Example (3/3)

```
/* Practice 12 : String copy – array*/
#define STRMAX 100
void string_copy(char *s, char *d) {
    while(*s != '₩0') {
        *d = *s; s++; d++;
    }
    *d = *s;
}
```

```
int main (void)
{
    char str1[STRMAX], str2[STRMAX];

    scanf("%s", str1); //str1 == &str1[0]
    string_copy(str1, str2);

    printf("str1 : %s₩nstr2 : %s₩n", str1, str2);

    return 0;
}
```

Exercise

- String inverter (Due date : Today's 11:59 PM)
  - Input : One string (length <= 100)
  - Output : inverted input string
  - Input string do not have any white space