Basic and Practice in Programming

Lab7

Variable and Its Address (1/3)

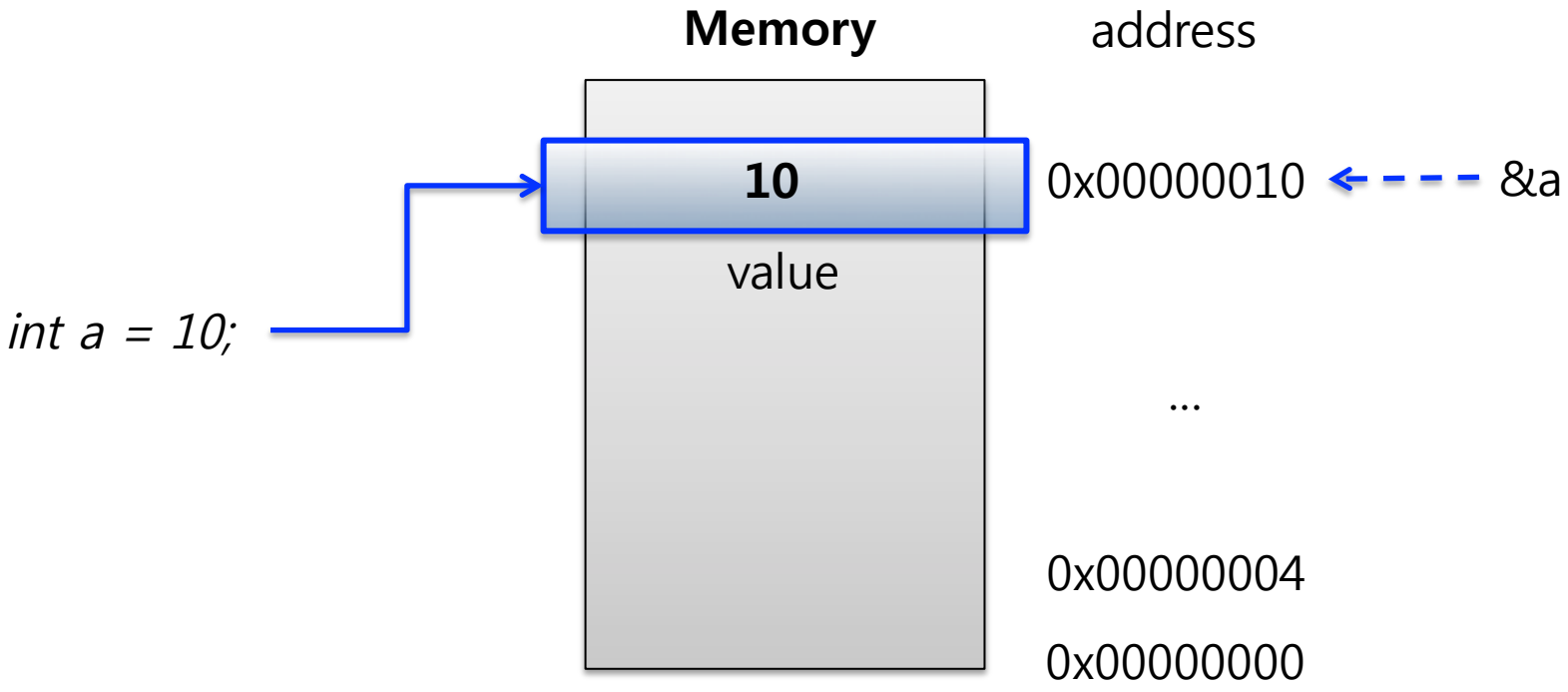
int a = 10;

variable value

How to understand this statement at machine's perspective?

Variable and Its Address (2/3)

- What is the variable?
  - Abstracted representation of allocated memory
  - Having address & value



Variable and Its Address (3/3)

```
 /* Practice 1 : Variable and Address */
 #include <stdio.h>

int main(void)
{
   int a = 10;

   printf("value: %d address: %p₩n", a, &a);

   return 0;
}
```

A Consideration

- Let's consider follow function form

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

A Consideration

- Let's consider follow function form

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

Realistic executable function form

```
int add (_x, _y)
{
    int x, y;
    x = _x;
    y = _y;
    x += y;
    return x;
}
```

The function uses arguments as form of local variables

A Consideration

- Let's consider follow function form

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

```
int main (void)
{
    int x = 10, y = 10, sum;
    sum = add(x, y);
    return 0;
}
```

What is the value of x?

Realistic executable function form

```
int add (_x, _y)
{
    int x, y;
    x = _x;
    y = _y;
    x += y;
    return x;
}
```

The function uses arguments as form of local variables

Call by Value (1/2)

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

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

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

    return 0;
}
```

Call by Value (2/2)

```
/* Practice 3 : Call by value 2 */
#include <stdio.h>

int add (int x, int y)
{
    x += y;
    printf ("x: %p y: %p", &x, &y);
    return x;
}

int main(void)
{
    int x = 10, y = 20;
    printf("x: %p y: %p", &x, &y);
    printf("%d + %d = %d₩n", x, y, add(x, y));

    return 0;
}
```

A Question

- How to use same “variable”
  - Across two functions

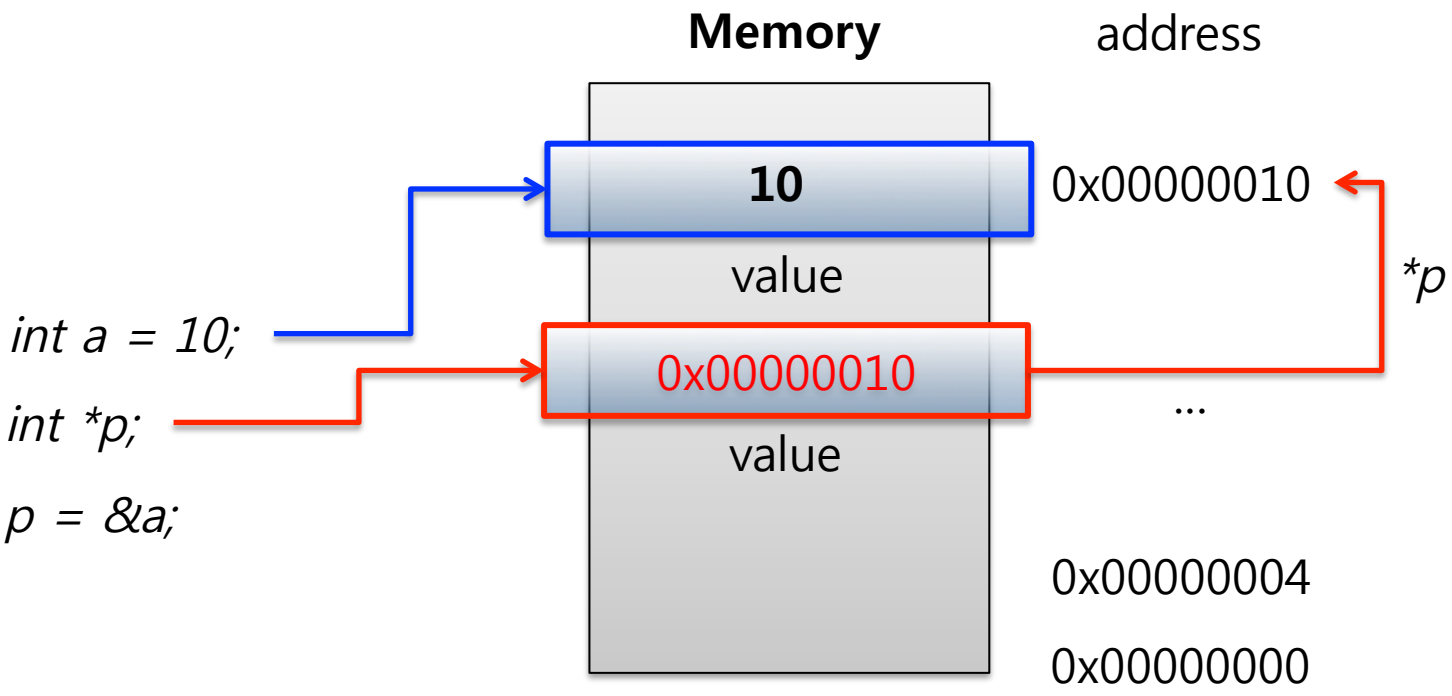
- The answers are
  - Using global variable but,
    - Using global variable is **NOT** recommended
    - Security, consistency, reliability problems
  - Using variable’s address instead of variable itself
    - **Pointer**

Pointer

- It is not a concept
  - Just a method
  - There is no serious “philosophy” in the pointer
  - Pointer is just variable for containing address

```
int a = 10;
int *p = &a;  //declaration form of pointer

//p == &a, *p == a
```

Pointer


Memory
address
10
0x00000010
value
*p
int a = 10;
0x00000010
int *p;
...
p = &a;
value
0x00000004
0x00000000

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

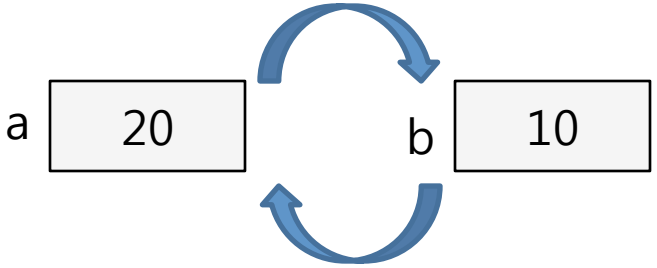
- Call by reference
  - Argument is given by “variable’s address”

Call by Reference (1/3)

- Swap function
  - Exchange two variable's contents

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


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



Call by Reference (2/3)

```
/* Practice 4 :
    Call by reference example1*/
#include <stdio.h>

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

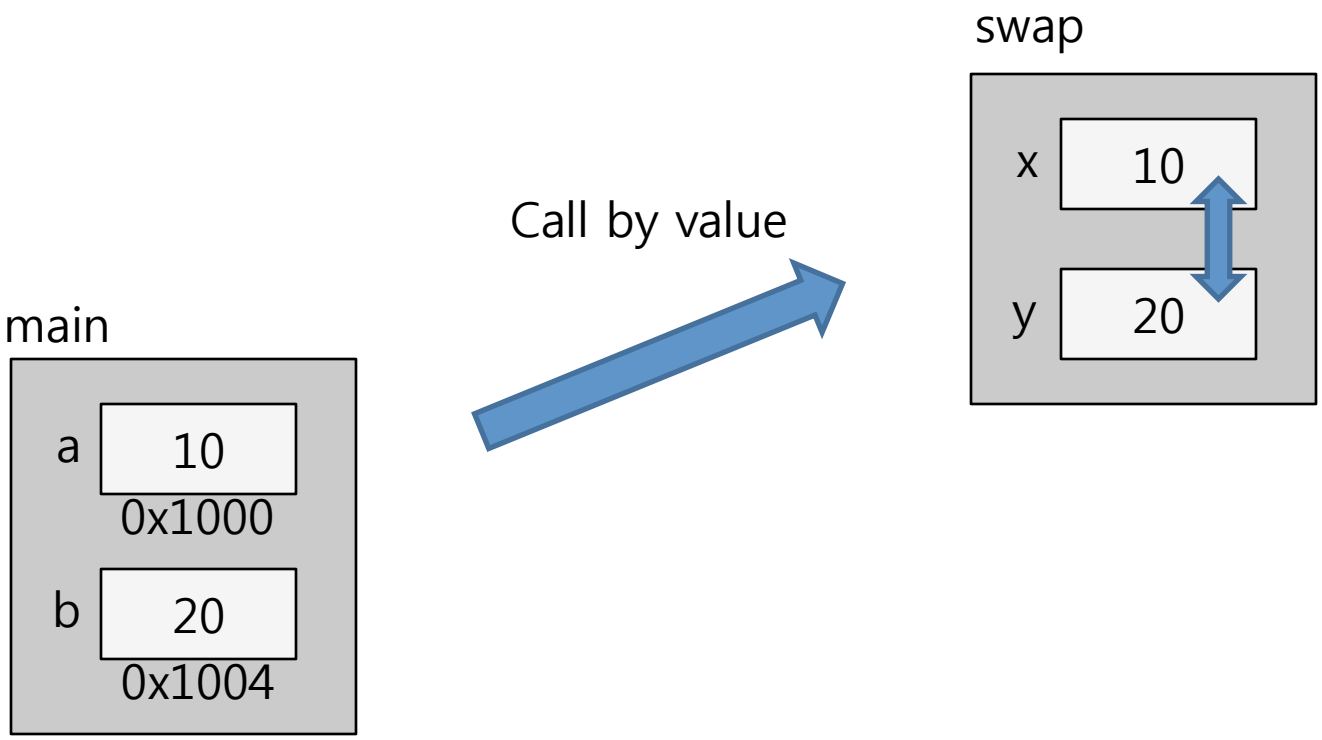
```
int main(void)
{
   int a = 10, b = 20;

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

   return 0;
}
```

Call by Reference (3/3)

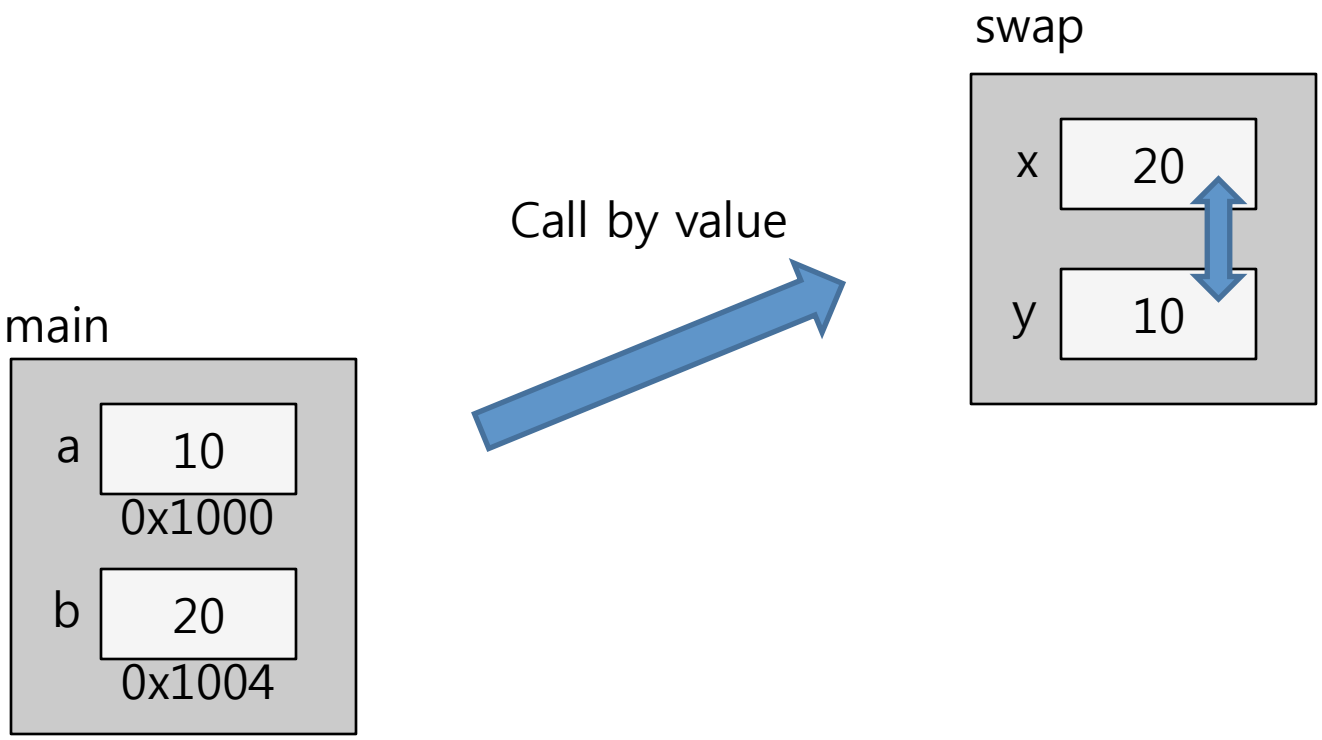
```
/* Practice 5 :
    Call by reference example2*/
#include <stdio.h>

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;
}
```

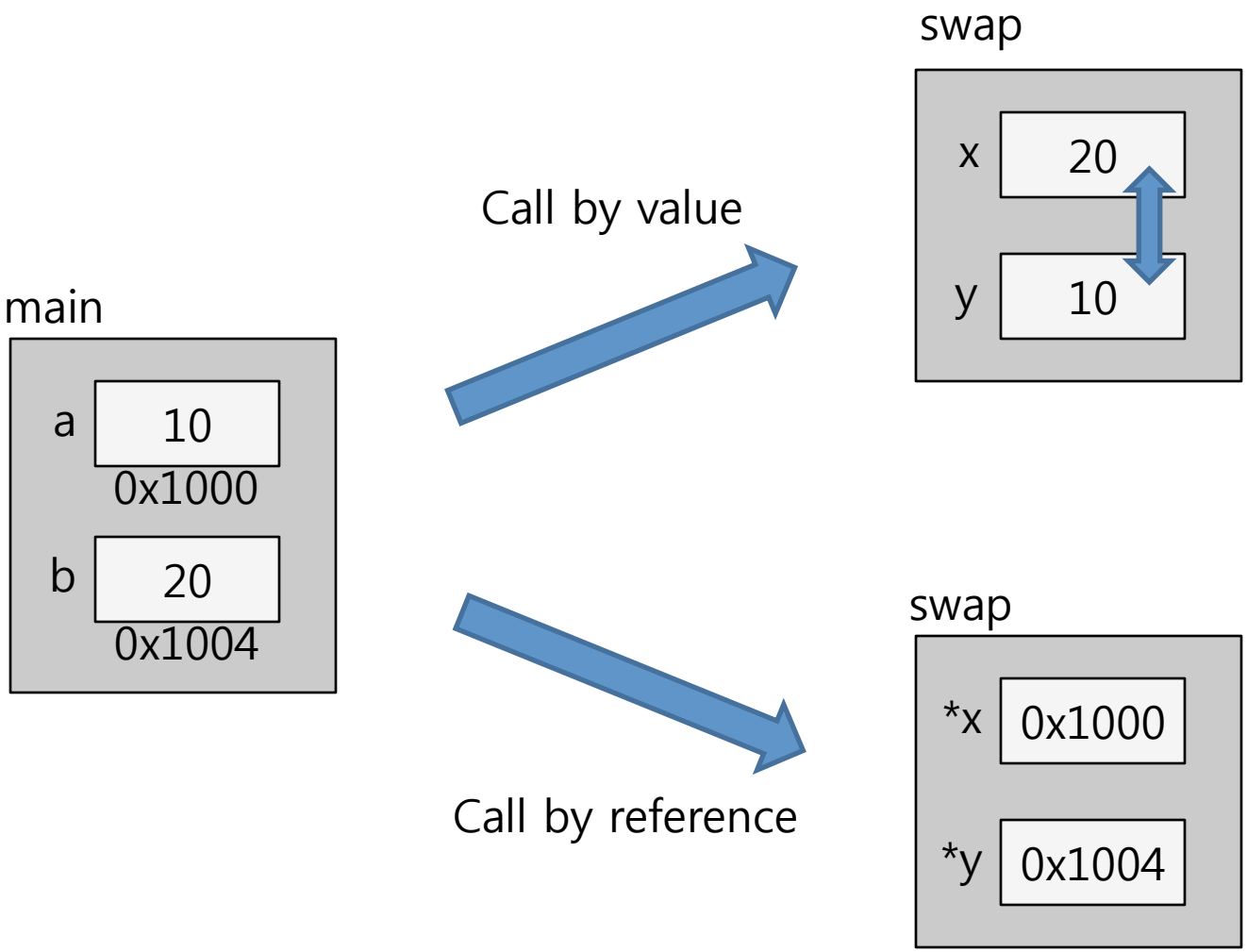
Reference?



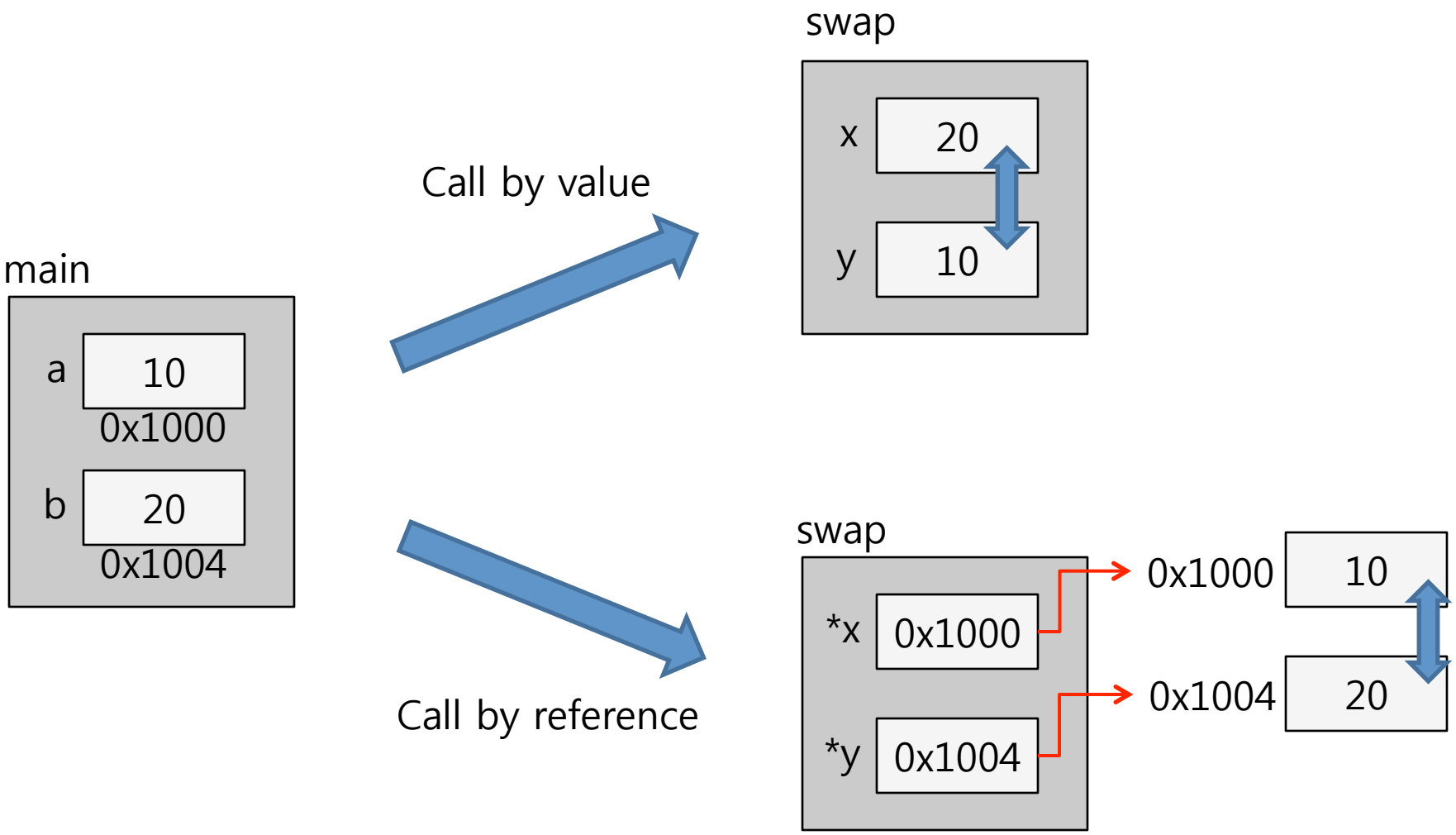
Reference?



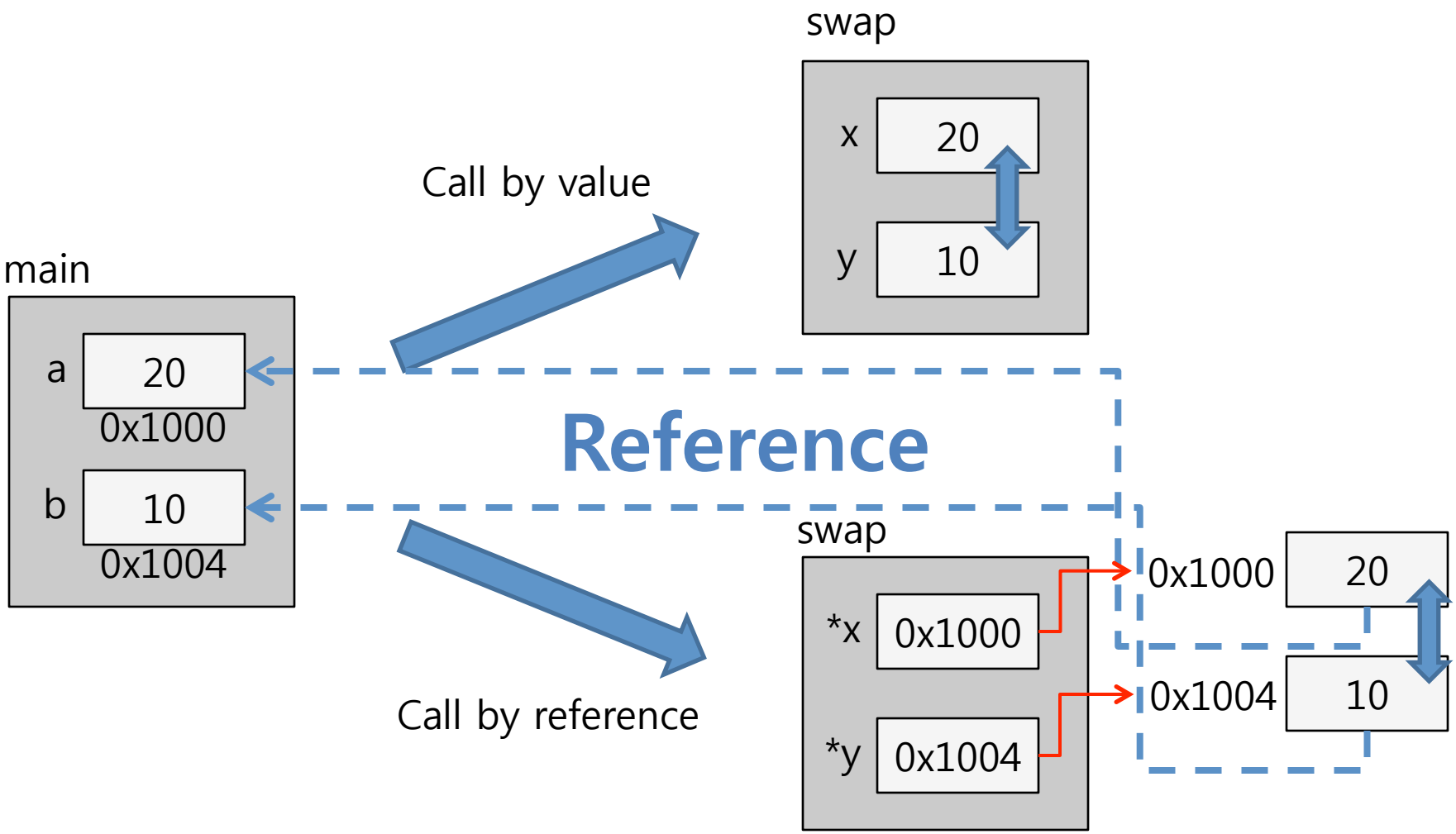
Reference?


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

Reference?

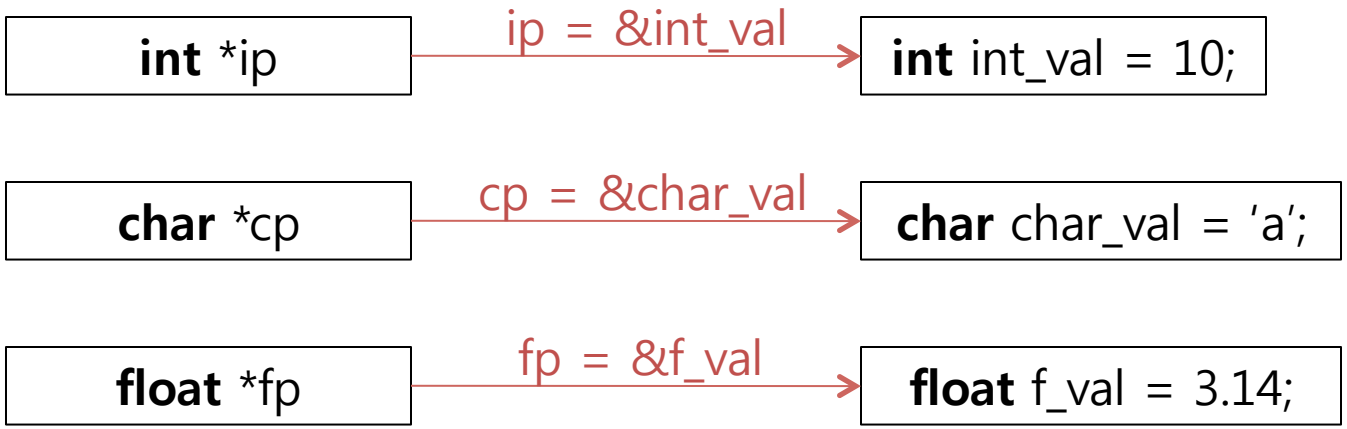

swap
x
20
y
10
Call by value
main
a
10
0x1000
b
20
0x1004
swap
*x
0x1000
*y
0x1004
0x1000
10
0x1004
20
Call by reference

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

Type

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



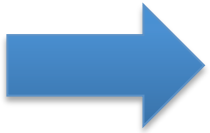
Pointer of Pointer

- Pointer is just a variable
  - Of course, it has its address

```
/* Practice 5 :
   Pointer of pointer */
#include <stdio.h>

int main (void)
{
   int x = 10, *p, **pp;
   p = &x;
   pp = &p;
   printf ("%d %d %d₩n", x, *p, **pp);
   printf ("%p %p %p₩n", &x, p, *pp);
   printf ("%p %p₩n", &p, pp);

   return 0;
}
```

Exercise 1

- Left shifter
  - Input is given as form of sequence of 5 distinct char acters
  - Last input is “the number of shift”
  - Skeleton code is given
    - Check the homepage
  - You must not print last input

A B C D E 3  D E A B C