



# Chapter 05. Passing parameter of a pointer variable and function

# 목차

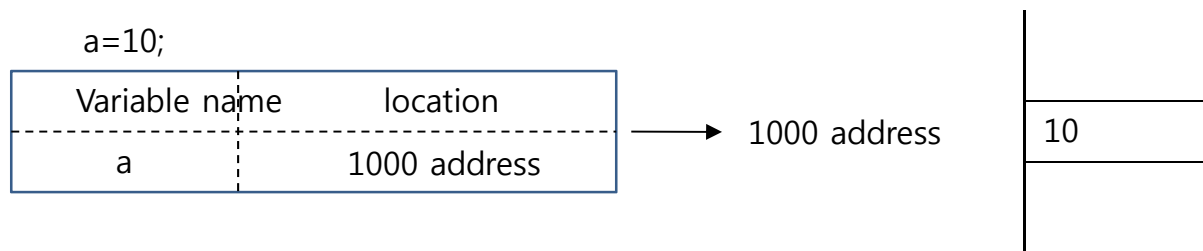
1. Studying Pointer
2. How to pass parameters in the function

# 학습목표

- Learn about how to declare and use pointers to pointers and program.
- Find out about pointer arithmetic.
- Learn the passing scheme by a passing method according to the pointer value from the passing factor of the system function.

# 01 Studying Pointer

- There attached a serial number, which is taken as the address starting from address 0 trying to separate the location memory
- The address is an integer in bytes. To the computer to process the data first have to transfer the data to the memory (RAM).



[Picture 5-1] Relationship between the memory address and the variable for the data

# 01 Studying Pointer

## ■ Pointer operators

- The pointer tells the computer that the memory address (the address of a variable), where the data is stored.. In C ++ provides a pointer operator (&) to enable the pointers directly.

& General Variable Name

포인터 연산자 & 기본 형식

- That attaching the & operator in front of the variable name is a variable location anywhere in the memory indicates that address.

```
int a;  
cout<<&a<<endl;
```

- \* **Address informs the operator preceded the values of variables located at the address.**

\* Pointer variable name

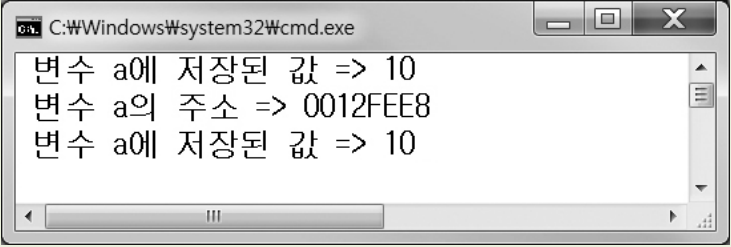
포인터 연산자 \* 기본 형식

[Table 5-1] Pointer types of operators and means

division	meaning
&	It extracts the address of a variable.
*	A pointer to extract the values in the memory address.

## Example 5-1. Output value of the variable address (05\_01.cpp)

```
01 #include<iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10;
06     cout<<" 변수 a에 저장된 값 => "<< a <<"\n";
07     cout<<" 변수 a의 주소 => "<< &a <<"\n";
08     cout<<" 변수 a에 저장된 값 => "<< *(&a) <<"\n";
09 }
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the C++ program in Korean. The output consists of three lines: "변수 a에 저장된 값 => 10", "변수 a의 주소 => 0012FEE8", and "변수 a에 저장된 값 => 10". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Windows\system32\cmd.exe
변수 a에 저장된 값 => 10
변수 a의 주소 => 0012FEE8
변수 a에 저장된 값 => 10
```

# 01 Studying Pointer

## ■ Pointer Variables

- In C ++, a pointer variable to store only the addresses are available separately. When you declare a pointer variable, the variable must try to distinguish general should add the \* symbol.

```
Data type * pointer variable name; // Pointer Variable Declaration
```

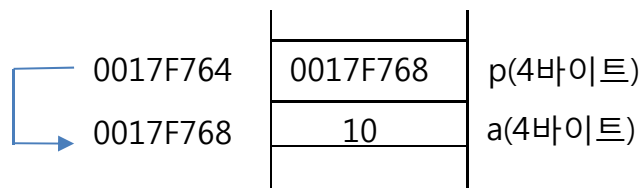
포인터 변수 기본 형식

ex) `cout<<" Value stored in the variable x => "<< *(&a) <<"\n";`

\* Operators visit the address by prefixing the address (& a) indicates the value stored there. Therefore, use for the \* operator, the pointer variable p can be the value of the variable a.

```
cout<<" Value stored in the variable x => "<< *p <<"\n";
```

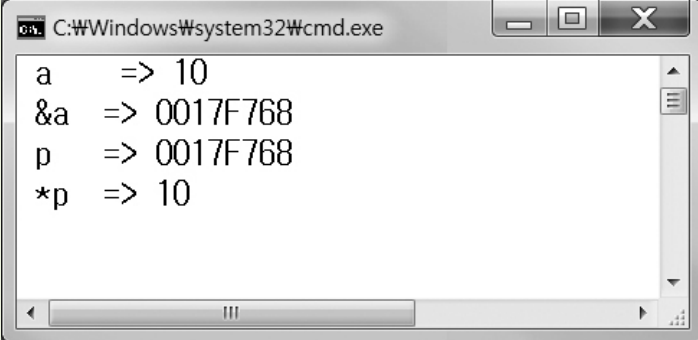
It is called a pointer variable p in isolation means the pointer value (address), but in addition to the pointer variable \* p \* is not a longer address means the value stored in the corresponding address. So, because the address of the variable p has a stored \* p is the value of the variable a.



[Picture 5-2] A pointer variable in the memory allocation structure

## Example 5-2. Structure (05\_02.cpp) assigned to the pointer variable in the memory

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10;
06     int *p;
07     p=&a;
08     cout<<" a => "<< a <<"\n";
09     cout<<" &a => "<< &a <<"\n";
10     cout<<" p => "<< p <<"\n";
11     cout<<" *p => "<< *p <<"\n";
12 }
```



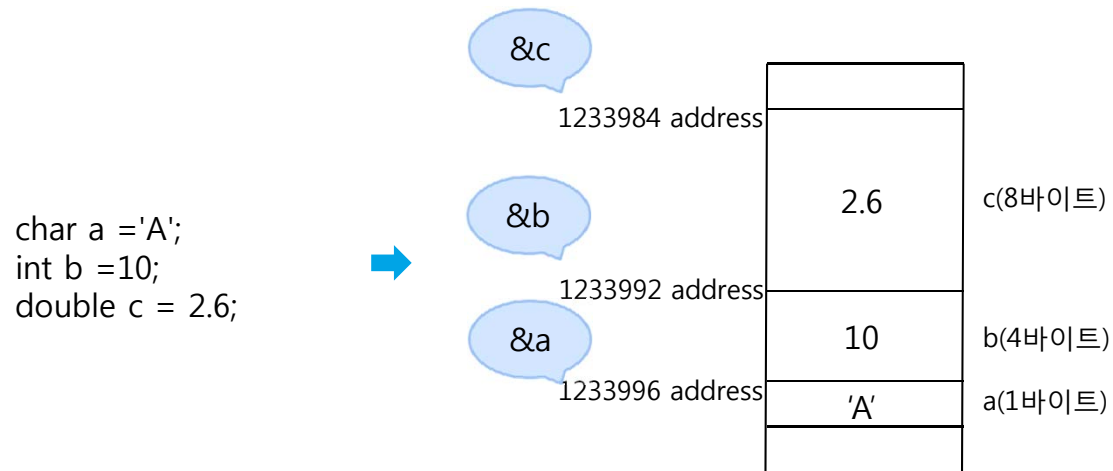
The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
a    => 10
&a  => 0017F768
p    => 0017F768
*p   => 10
```



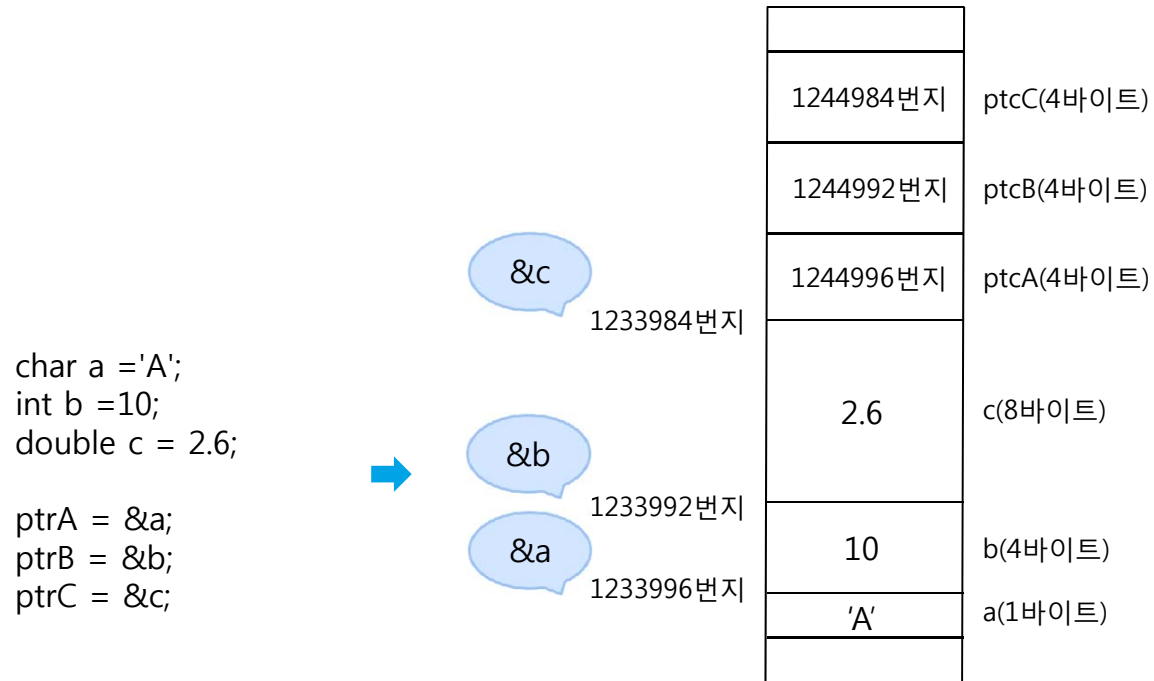
# 01 Studying Pointer

- When the pointer variable declaration types role is determined by the data type stored in the variable is a pointer type. Therefore, you should also determine the type of a pointer variable depending on the type of a variable to store.



[Picture 5-3] Assigned to various types of memory

# 01 Studying Pointer



[Picture 5-4] Memory status when stored as a pointer variable

```
char * ptrA;    // * The operator reads a byte from the address stored in the ptrA.
int * ptrB;     // * The operator reads a byte from the address stored in the ptrB
double * ptrC;  // * The operator reads a byte from the address stored in the ptrC
```

※ **The size of the memory allocated to all pointer variable is a four-byte.**

**Pointer variable stores only the starting address of the memory space where the data is stored.**

# 01 Studying Pointer

## ■ Initialization of a pointer variable

- At the same time, let's declare a pointer variable and assign the address.

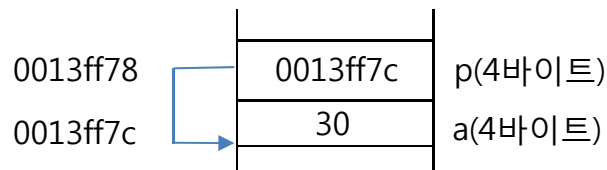
<code>int *p = &amp;a;</code>	<code>==</code>	<code>int *p;</code> <code>p = &amp;a</code>
-------------------------------	-----------------	---

- Let's assign a pointer variable in general variable.
  - ⓐ B is in a compile-time error occurs failed to save the address that the pointer variable, the variable stored in general.
  - ⓑ Because of the p and stores the address of a p \* has come to find a value that is stored to find the address of a a and stores it in a b.

ⓐ <code>int b;</code> <code>b=p; // Compile error</code>	ⓑ <code>b=*p;</code>
---	----------------------

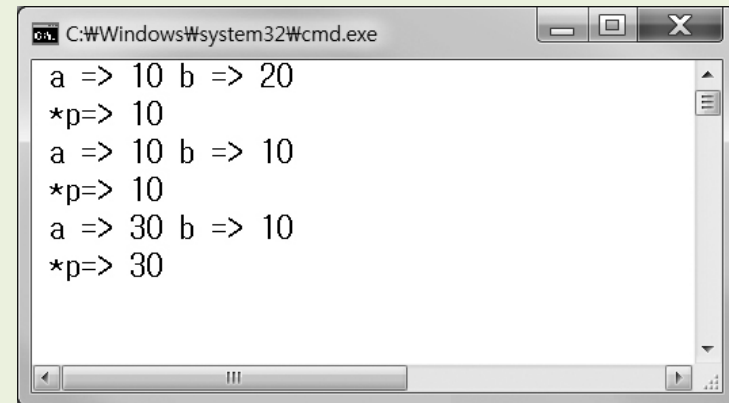
- Let's save the pointer variable, constant and regular.
  - p is constant only because you want to save the store only address assignment must attach the front p \* operator.

ⓒ <code>p=30; // Compile error</code>	ⓓ <code>*p=30;</code>
---------------------------------------	-----------------------



## Example 5-3. Structure (05\_03.cpp) assigned to the pointer variable in the memory

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10, b=20;
06     int *p=&a;
07     cout<<" a => "<< a <<" b => "<< b <<"\n";
08     cout<<" *p=> "<< *p <<"\n";
09     b=*p;
10     cout<<" a => "<< a <<" b => "<< b <<"\n";
11     cout<<" *p=> "<< *p <<"\n";
12     *p=30;
13     cout<<" a => "<< a <<" b => "<< b <<"\n";
14     cout<<" *p=> "<< *p <<"\n";
15 }
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

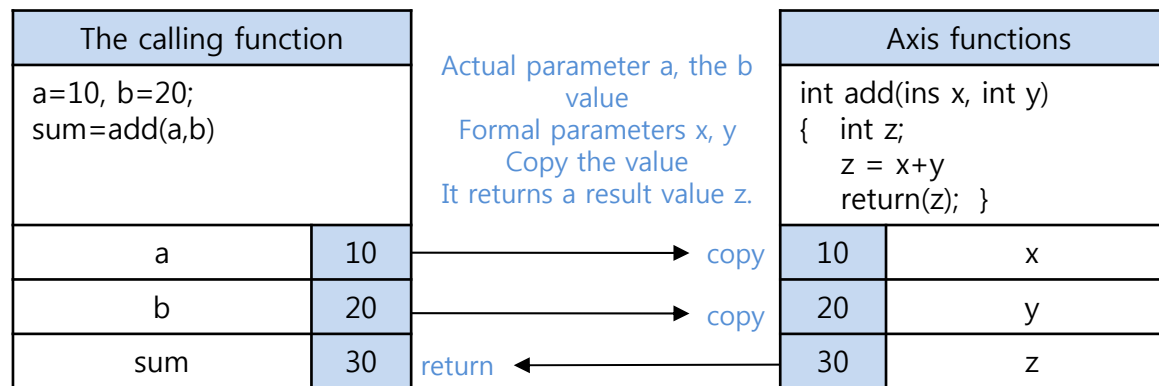
```
a => 10 b => 20
*p=> 10
a => 10 b => 10
*p=> 10
a => 30 b => 10
*p=> 30
```

## 02 How to pass parameters in the function

### ■ How to pass parameters in the function

- ① Call by Value
- ② Call by Address
- ③ Call by Reference

#### ① Transfer scheme according to the value

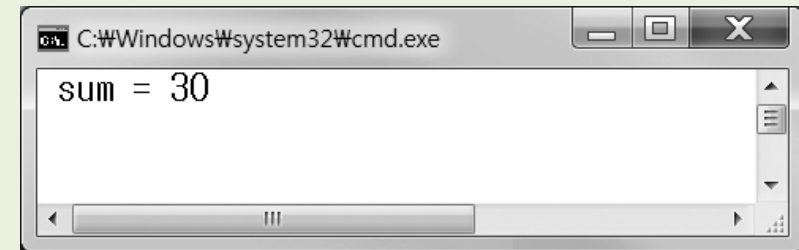


[Picture 5-5Parameters and return values

- If you want to pass a value from one side to the calling function, the function must return a return statement..
- Copy the value type parameter is assigned a separate storage space and take actual parameters, so also does the function to change the value of the type parameter in the defined area not change the value of the actual parameter.

## Example 5-4. Learn the way of the transfer function by value (05\_04.cpp)

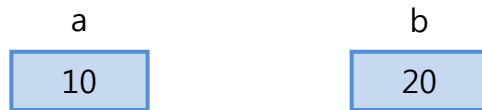
```
01 #include <iostream>
02 using namespace std;
03 int add(int x, int y);
04 void main()
05 {
06     int a=10, b=20, sum;
07     sum=add(a, b);      // Call by value
08     cout<<" sum = "<< sum <<"\n";
09 }
10 int add(int x, int y)
11 {
12     int z;
13     z=x+y;
14     return(z);
15 }
```



## 02 How to pass parameters in the function

### ② Transfer scheme according to the address

❶ a=10, b=20;



❷ a=b;

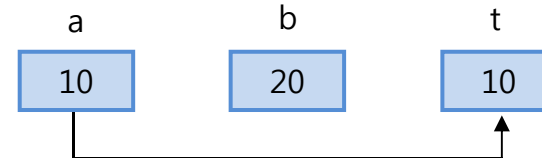


❸ b=a;

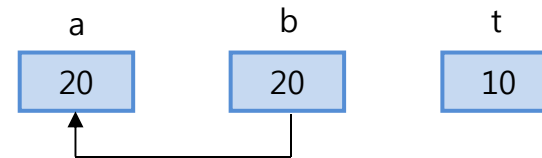


[Picture 5-7] Invalid exchange algorithm

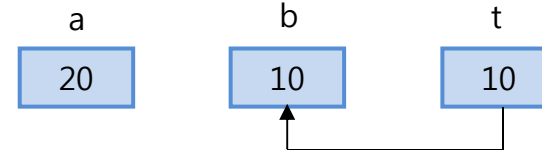
❹ t=a;



❺ a=b;



❻ b=t;

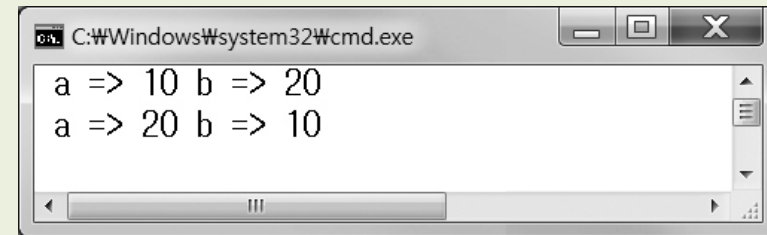


[Picture 5-8] Exchange algorithm

- ❹ It stores a value of a variable in the temporary storage of t.
- ❺ Even if the value of the variable b stored in a temporary variable to save the left, because a value to a variable t.
- ❻ Storing the values of the variables t, the value of the variables are exchanged.

## Example 5-6. To exchange value is stored in two variables (05\_06.cpp)

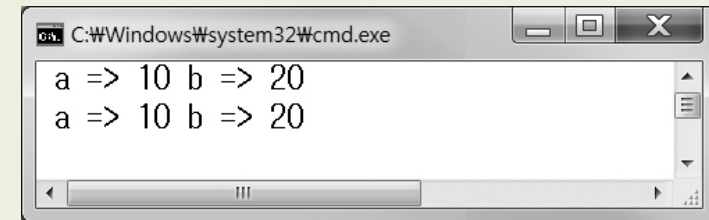
```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10, b=20;
06     cout<<" a => " << a <<" b => " << b <<"\n";
07     int t;
08     t=a;
09     a=b;
10     b=t;
11     cout<<" a => " << a <<" b => " << b <<"\n";
12 }
```





## Example 5-7. A transmission method according to the value of a function to exchange the two variable values (05\_07.cpp)

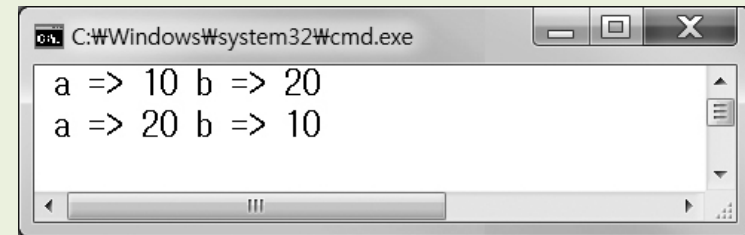
```
01 #include <iostream>
02 using namespace std;
03 void swap(int a, int b);
04 void main()
05 {
06     int a=10, b=20;
07     cout<<" a => " << a <<" b => " << b <<"\n";
08     swap(a, b);
09     cout<<" a => " << a <<" b => " << b <<"\n";
10 }
11 void swap(int a, int b)
12 {
13     int t;
14     t=a;
15     a=b;
16     b=t;
17 }
```



// Call by value case

## Example 5-8. Creating a function for exchanging the two parameter values to the transmission method according to the address (05\_08.cpp)

```
01 #include <iostream>
02 using namespace std;
03 void swap(int *pa, int *pb);
04 void main()
05 {
06     int a=10, b=20;
07     cout<<" a => " << a <<" b => " << b <<"\n";
08     swap(&a, &b);                // Call by address
09     cout<<" a => " << a <<" b => " << b <<"\n";
10 }
11 void swap(int *pa, int *pb)
12 {
13     int t;
14     t=*pa;
15     *pa=*pb;
16     *pb=t;
17 }
```



## 02 How to pass parameters in the function

### ③ Transfer scheme according to the reference

- The reference variable refers to the alias (sort of a different name). Reference variable does not separate memory space is allocated because it is used as an alias for the given reference variable names when declaring variables that have already been declared. Reference variable is available by accessing the only variables that exist only in the memory as many names.

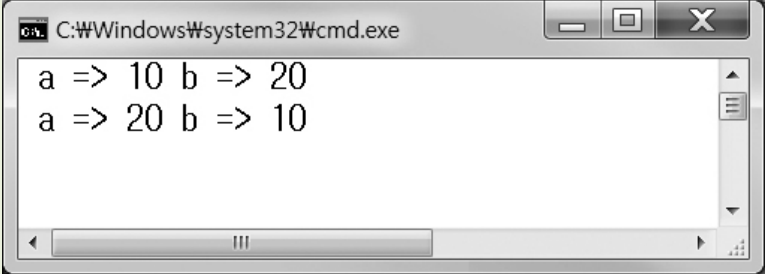
Reference types and variable names = The previously declared variable name

참조 변수 선언 기본 형식

- How to declare a reference variable and adds the symbol before the variable name used as alias. At this time, using the & symbol is called the reference operator.

**Example 5-10. A transmission method by reference to create a function that exchanges the two variable values (05\_10.cpp)**

```
01 #include <iostream>
02 using namespace std;
03 void swap(int &x, int &y);
04 void main()
05 {
06     int a=10, b=20;
07     cout<<" a => " << a <<" b => " << b <<"\n";
08     swap(a, b);    // Call by reference
09     cout<<" a => " << a <<" b => " << b <<"\n";
10 }
11 void swap(int &x, int &y)    // 참조변수 선언
12 {
13     int t;
14     t=x;
15     x=y;
16     y=t;
17 }
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
a => 10 b => 20
a => 20 b => 10
```

# Homework

---

- Chapter 5 Exercise: 14, 15, 16