



Chapter 05. 포인터와 함수의 매개변수 전달

목차

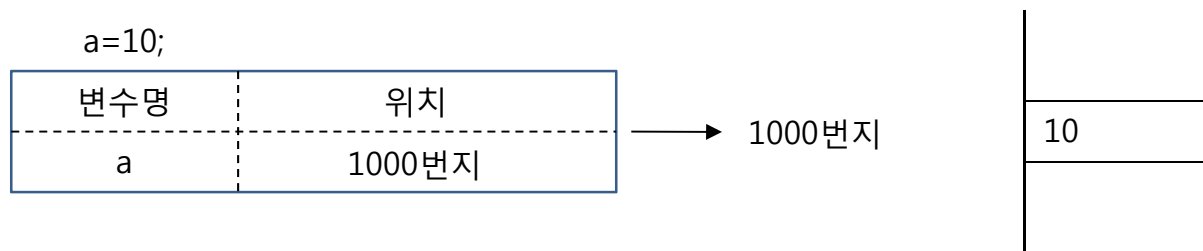
1. 포인터 살피기
2. 함수에서 매개변수를 전달하는 방법

학습목표

- 포인터에 대해 알아보고 포인터 선언 방법과 사용법을 익힌다.
- 포인터 연산에 대해 알아본다.
- 함수의 인자 전달 방식 중에서 값에 의한 전달 방식과 포인터에 의한 전달 방식을 학습한다.

01 포인터 살펴기

- 메모리에는 위치를 구분하려고 0번지부터 시작해서 일련번호가 붙여져 있는데, 이것을 주소(address)라 한다. 주소는 정수 형태이며 단위는 바이트다. 컴퓨터가 데이터를 처리하려면 먼저 데이터를 메모리(램)로 옮겨야 한다.



[그림 5-1] 데이터에 대한 메모리 주소와 변수와의 관계

01 포인터 살피기

■ 포인터 연산자

- 포인터는 컴퓨터의 메모리 번지(변수의 주소)로, 데이터가 어디에 저장되어 있는지를 알려준다. C++에서는 이 포인터를 직접 사용할 수 있도록 포인터 연산자(&)를 제공한다.

&일반변수명

포인터 연산자 & 기본 형식

- 변수명 앞에 & 연산자를 붙이면 변수 a가 메모리의 어느 곳에 위치하는지 그 주소값을 알려준다.

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

- * 연산자는 주소 앞에 붙어서 해당 주소에 위치한 변수값을 알려준다.

*포인터변수명

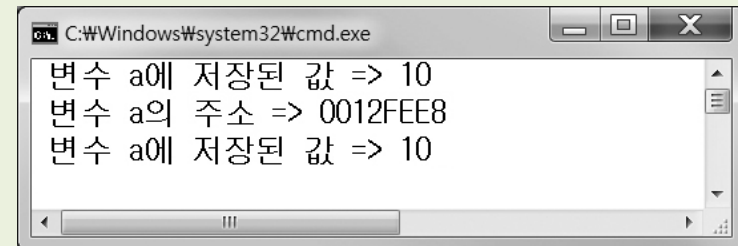
포인터 연산자 * 기본 형식

[표 5-1] 포인터 연산자의 종류와 의미

구분	의미
&	변수의 주소를 추출한다.
*	포인터가 가리키는 메모리 주소에 있는 값을 추출한다.

예제 5-1. 변수의 주소값 출력하기(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 }
```



01 포인터 살피기

■ 포인터 변수

- C++에서는 주소만을 저장하는 포인터 변수가 별도로 제공된다. 일반적인 변수와 구분하려고 포인터 변수를 선언할 때 반드시 * 기호를 덧붙여야 한다.

자료형 *포인터변수명; // 포인터 변수 선언

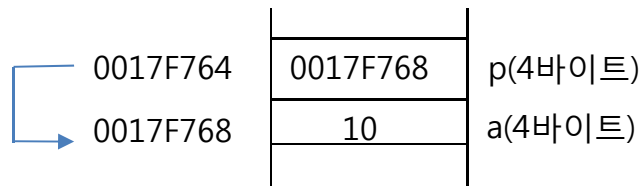
포인터 변수 기본 형식

ex) cout<<" 변수 a에 저장된 값 => "<< *(&a) <<"\n";

* 연산자는 주소(&a) 앞에 붙여서 해당 주소를 찾아가 그 곳에 저장되어 있는 값을 알려준다. 따라서 * 연산자를 포인터 변수 p에 대해 사용하여 변수 a의 값을 출력할 수 있다.

cout<<" 변수 a에 저장된 값 => "<< *p <<"\n";

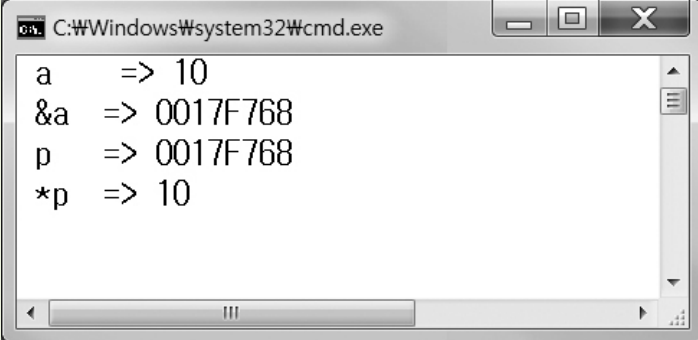
포인터 변수를 단독적으로 p라고 하는 것은 포인터값(주소)을 의미하지만 포인터 변수에 *을 덧붙여 *p는 더 이상 주소가 아니라 해당 주소에 저장된 값을 의미한다. 그래서 p에는 변수 a의 주소가 저장되어 있으므로 *p는 변수 a의 값을 출력한다.



[그림 5-2] 포인터 변수를 메모리에 할당한 구조

예제 5-2. 포인터 변수를 메모리에 할당한 구조(05_02.cpp)

```
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 }
```

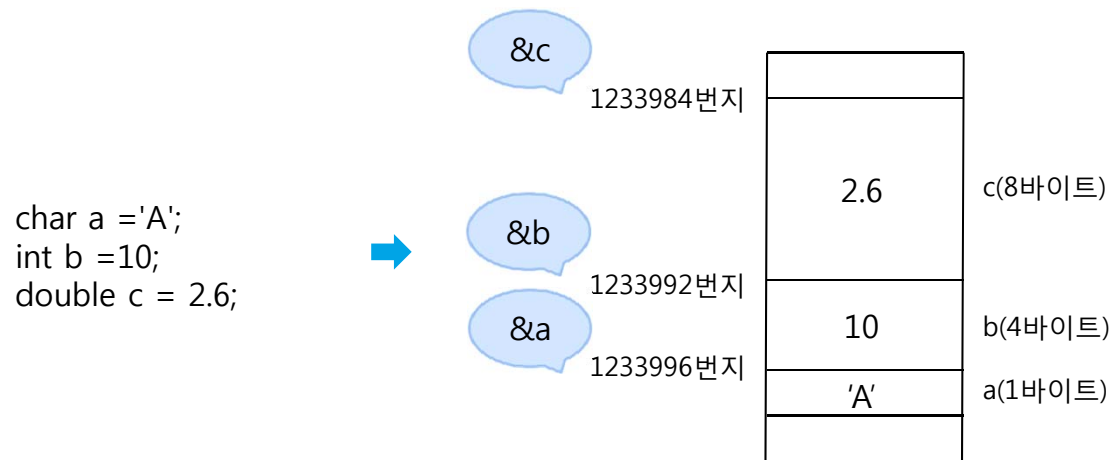


The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

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

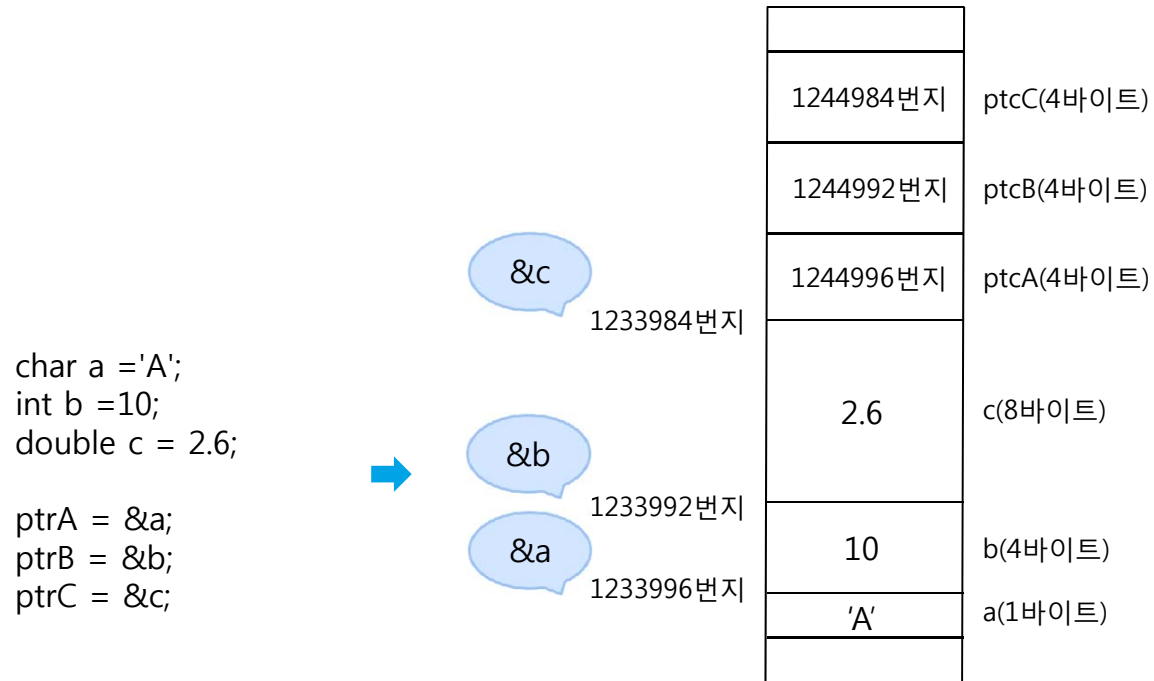

01 포인터 살피기

- 포인터 변수 선언시 자료형의 역할은 포인터가 가리키는 변수에 저장된 자료형에 의해서 결정된다. 따라서 저장할 변수의 자료형에 따라 포인터 변수의 자료형도 결정해야 한다.



[그림 5-3] 다양한 자료형의 메모리 할당

01 포인터 살피기



[그림 5-4] 포인터 변수로 저장했을 때의 메모리 상태

```
char * ptrA;    // * 연산자는 ptrA에 저장된 주소로부터 1바이트를 읽어온다.
int * ptrB;     // * 연산자는 ptrB에 저장된 주소로부터 4바이트를 읽어온다.
double * ptrC; // * 연산자는 ptrC에 저장된 주소로부터 8바이트를 읽어온다.
```

※ 모든 포인터 변수에 할당된 메모리의 크기는 4바이트이다.

포인터 변수는 데이터가 저장된 기억공간의 시작주소만 저장한다.

01 포인터 살피기

■ 포인터 변수의 초기화

- 포인터 변수 선언과 동시에 주소값을 대입해 보자.

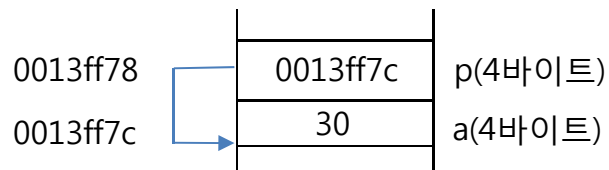
<code>int *p = &a;</code>	<code>==</code>	<code>int *p;</code> <code>p = &a</code>
-------------------------------	-----------------	---

- 일반 변수에 포인터 변수를 대입해 보자.
 - ㉠에서 b는 일반 변수여서 포인터 변수가 저장하고 있는 주소값을 저장하지 못해 컴파일 에러가 발생한다.
 - ㉡의 p가 a의 주소값을 저장하고 있으므로 *p는 a의 주소값을 찾아 a가 저장하고 있는 값을 찾아와서 이를 b에 저장한다.

㉠ <code>int b;</code>	㉡ <code>b = *p;</code>
<code>b = p; // 컴파일 에러</code>	

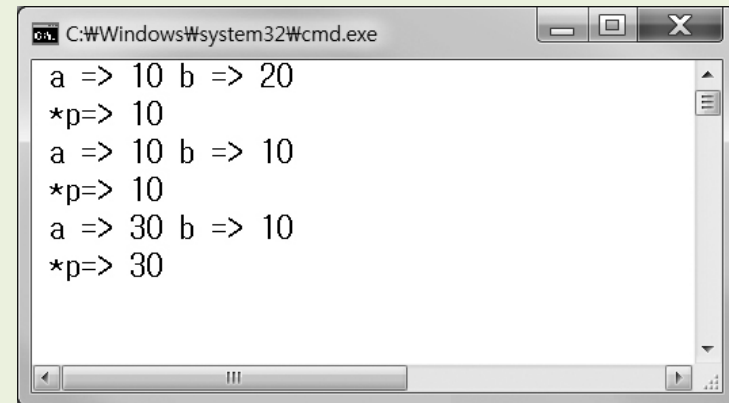
- 포인터 변수에 일반 상수값을 저장해 보자.
 - p는 주소만 저장할 뿐이므로 상수값을 저장하려면 p앞에 * 연산자를 붙여서 대입해야 한다.

㉢ <code>p = 30; // 컴파일 에러</code>	㉣ <code>*p = 30;</code>
----------------------------------	-------------------------



예제 5-3. 포인터 변수를 메모리에 할당한 구조(05_03.cpp)

```
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:

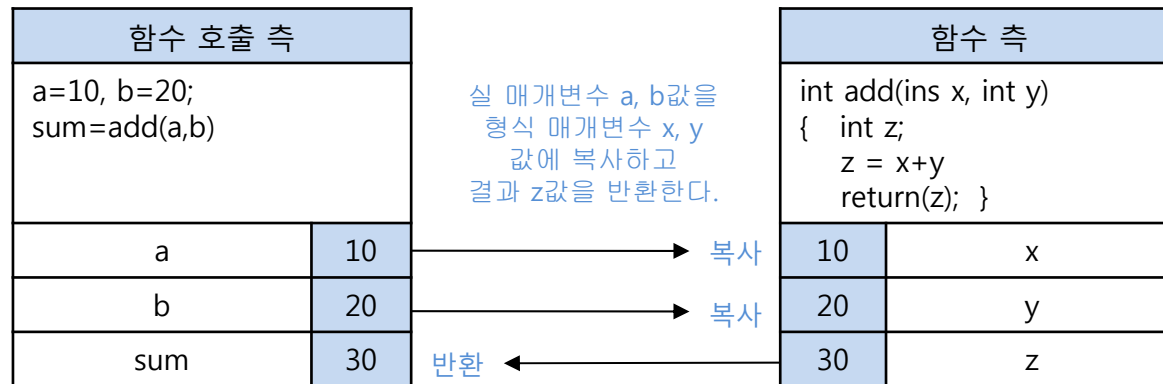
```
a => 10 b => 20
*p=> 10
a => 10 b => 10
*p=> 10
a => 30 b => 10
*p=> 30
```

02 함수에서 매개변수를 전달하는 방법

■ 함수에서 매개변수를 전달하는 방법

- ① 값에 의한 전달 방식(Call by Value)
- ② 주소에 의한 전달 방식(Call by Address)
- ③ 참조에 의한 전달 방식(Call by Reference)

① 값에 의한 전달 방식

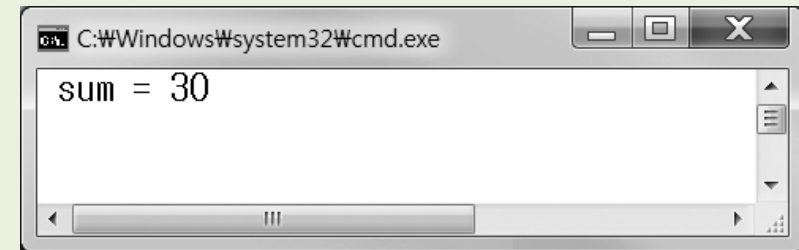


[그림 5-5] 매개변수와 반환값

- 만일 함수 측에서 함수를 호출한 측으로 값을 전달하려면 return 문으로 반환해야 한다.
- 형식 매개변수가 실 매개변수와 별개의 기억공간을 할당 받아 값을 복사하므로 함수를 정의한 곳에서 형식 매개변수의 값을 변경해도 실 매개변수의 값은 변경되지 않는다.

예제 5-4. 값에 의한 전달 방식의 함수 익히기(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 함수에서 매개변수를 전달하는 방법

② 주소에 의한 전달 방식

❶ a=10, b=20;



❷ a=b;

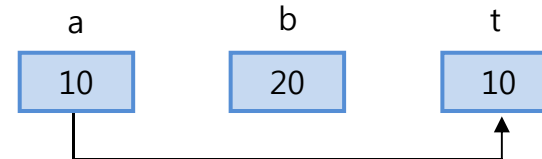


❸ b=a;

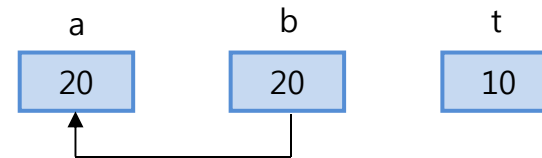


[그림 5-7] 잘못된 교환 알고리즘

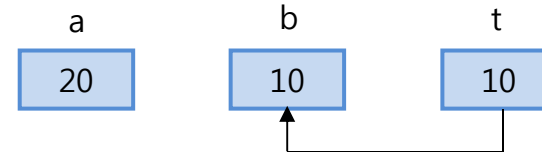
❹ t=a;



❺ a=b;



❻ b=t;

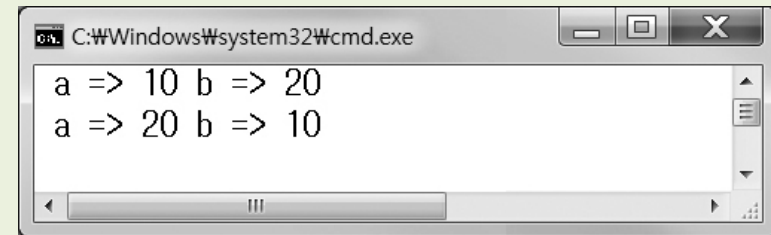


[그림 5-8] 교환 알고리즘

- ❹ 변수 a의 값을 임시 기억장소인 t에 저장한다.
- ❺ 변수 b의 값을 임시 변수 a에 저장하더라도 변수 t에 a값을 저장해 두었으므로
- ❻ t의 값을 변수 b에 저장하면 두 변수의 값이 교환된다.

예제 5-6. 두 변수에 저장된 값 교환하기(05_06.cpp)

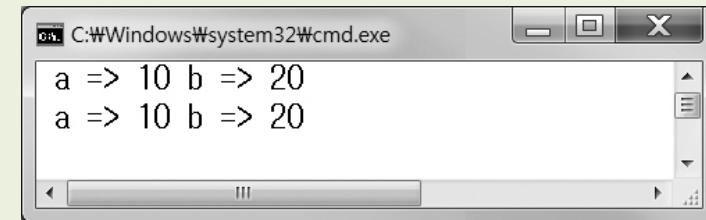
```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10, b=20;
06     cout<<" a => "<< a <<" b => "<< b <<"\n";
07     int t;
08     t=a;
09     a=b;
10     b=t;
11     cout<<" a => "<< a <<" b => "<< b <<"\n";
12 }
```



예제 5-7. 값에 의한 전달 방식으로 두 변수값을 교환하는 함수(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 }
```

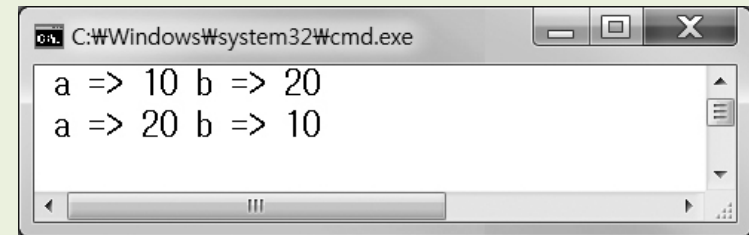
// Call by value case



예제 5-8. 주소에 의한 전달 방식으로 두 변수값을 교환하는 함수 작성하기(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);
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 }
```

// Call by address



02 함수에서 매개변수를 전달하는 방법

③ 참조에 의한 전달 방식

- 참조 변수란 별칭(일종의 다른 이름)을 의미한다. 참조 변수는 따로 기억공간이 할당되지 않는데, 이는 참조 변수 선언 시 부여한 변수명이 이미 선언되어 있는 변수의 별칭으로 사용되기 때문이다. 참조 변수는 메모리 상에 오로지 하나만 존재하는 변수를 여러 이름으로 접근해서 사용할 수 있게 한다.

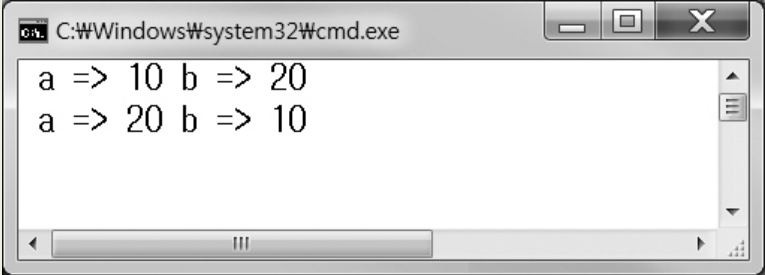
자료형 &참조 변수명 = 앞서 선언된 변수명

참조 변수 선언 기본 형식

- 참조 변수의 선언 방법은 별칭으로 사용할 변수명 앞에 & 기호를 덧붙인다. 이때 사용한 & 기호를 참조 연산자라고 한다.

예제 5-10. 참조에 의한 전달 방식으로 두 변수값을 교환하는 함수 작성하기(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 }
```



```
C:\Windows\system32\cmd.exe
a => 10 b => 20
a => 20 b => 10
```

Homework

- Chapter 5 Exercise: 14, 15, 16