

포인터와 함수의 매개변수 전달



● 주소

- 메모리에서 위치를 구분하기 위해 0번지부터 시작해서 붙이는 일련번호
- 정수 형태이며 단위는 바이트
- 컴퓨터는 데이터를 처리하기 위해 먼저 데이터를 메모리(램)로 옮김

● 데이터에 대한 메모리 주소와 변수와의 관계

a=10;

변수명	위치
a	1000번지

→ 1000번지

10

● 포인터 연산자

- 포인터는 컴퓨터의 메모리 번지(변수의 주소)
- 데이터가 어디에 저장되어 있는지 알림 : 포인터 연산자(&)를 제공
- 변수명 앞에 & 연산자를 붙이면 변수 a의 메모리 주소값 알기 가능

&일반변수명

포인터 연산자 & 기본 형식

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

- * 연산자는 주소 앞에 위치하며 해당 주소에 위치한 변수의 값

*포인터변수명

포인터 연산자 & 기본 형식

```
int *p;
cout<<p<<endl;
```

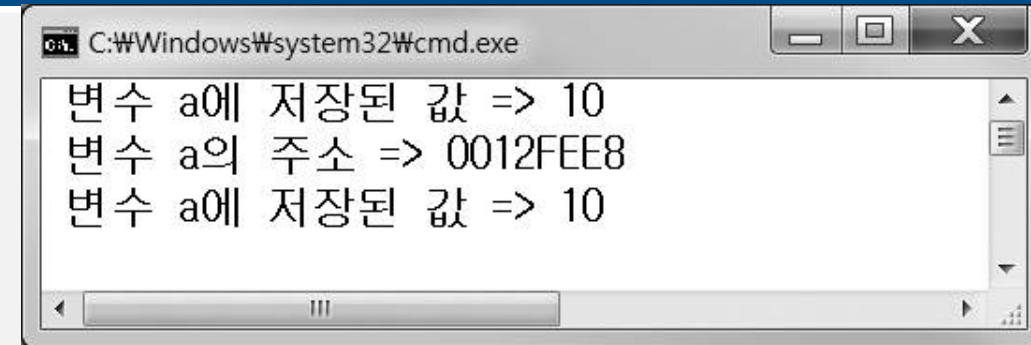
- 포인터 연산자의 종류와 의미

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

```
int *p;
p=&a;
cout<<p<<endl;
cout<<*p<<endl;
```

변수의 주소값 출력하기

```
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 }
```



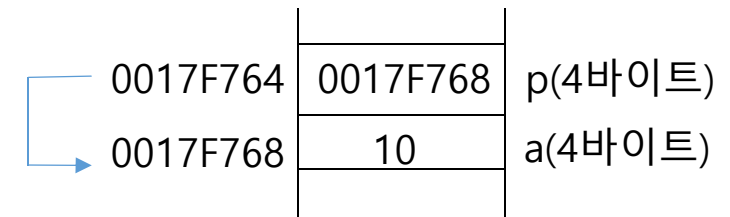
A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the C++ program, which prints three lines of text: "변수 a에 저장된 값 => 10", "변수 a의 주소 => 0012FEE8", and "변수 a에 저장된 값 => 10".

- C++에서는 주소만을 저장하는 포인터 변수가 별도 제공
- 일반 변수와 구분하기위해 * 연산자를 주소(&a) 앞에 붙여 표시
- 해당 주소를 찾아가 그 곳에 저장되어 있는 값을 알림
- * 연산자를 포인터 변수 p에 대해 사용하여 변수 a의 값을 출력

자료형 *포인터변수명;

포인터 연산자 & 기본 형식

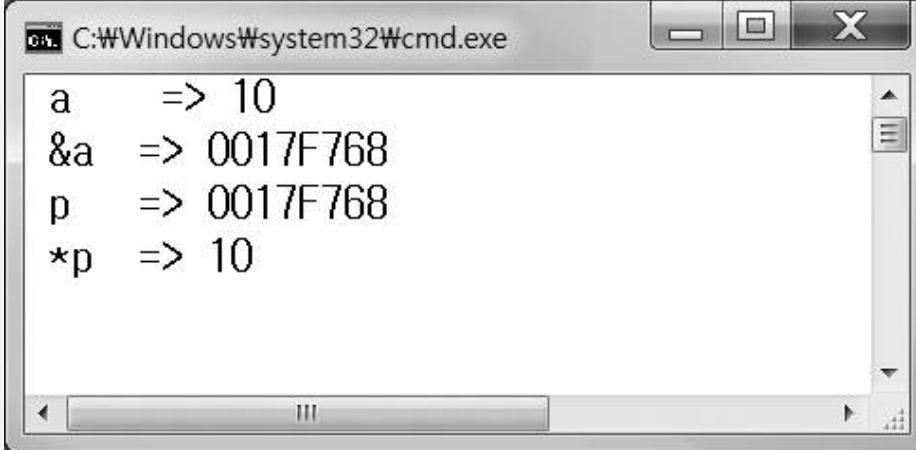
```
int a=10;  
int *p;  
p=&a;  
cout<<"변수 a에 저장된 값 => "<< *p <<"\n";
```



포인터 변수 p는 포인터 값(주소)을 의미하지만
포인터 변수에 *을 덧붙인 *p는 해당 주소에 저장된 값을 의미.
p에는 변수 a의 주소가 저장되어 있으므로 *p는 변수 a의 값을 출력

포인터 변수를 메모리에 할당

```
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 }
```



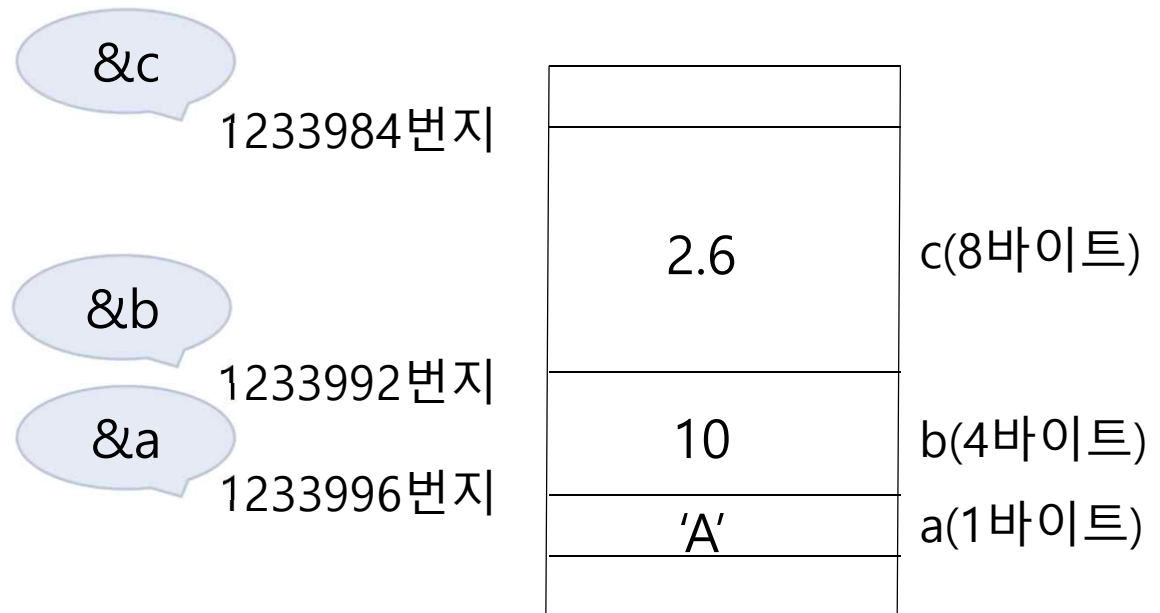
The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

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

● 포인터 변수의 자료형

- 변수 선언에서 자료형은 포인터가 가리키는 변수의 자료형에 의해 결정
- 저장할 변수의 자료형에 따라 포인터 변수의 자료형도 결정
- 다양한 자료형의 메모리 할당 가능

```
char a = 'A';  
int b = 10;  
double c = 2.6;
```

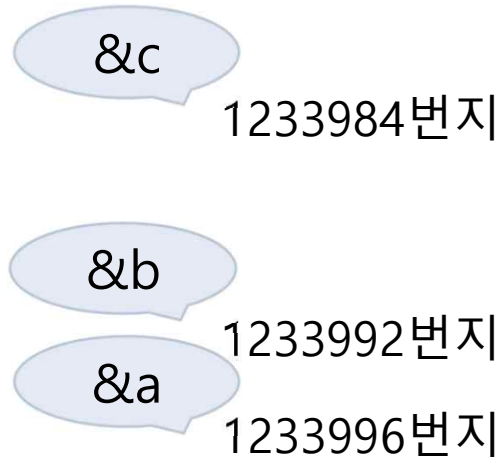


- 메모리 할당은 먼저 선언된 변수가 아래쪽(큰 주소)에 생성

• 포인터 변수로 저장했을 때의 메모리 상태

```
char a = 'A';
int b = 10;
double c = 2.6;
```

```
ptrA = &a;
ptrB = &b;
ptrC = &c;
```



1244984번지	ptcC(4바이트)
1244992번지	ptcB(4바이트)
1244996번지	ptcA(4바이트)
2.6	c(8바이트)
10	b(4바이트)
'A'	a(1바이트)

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


- 포인터 변수의 초기화

- 포인터 변수 선언과 동시에 주소값 대입

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

- 일반 변수에 포인터 변수 대입

㉠

```
int *p;  
int b;  
b=p; // 컴파일 에러
```

㉡

```
b=*p;
```

- 포인터 변수에 일반 상수값 저장

㉢

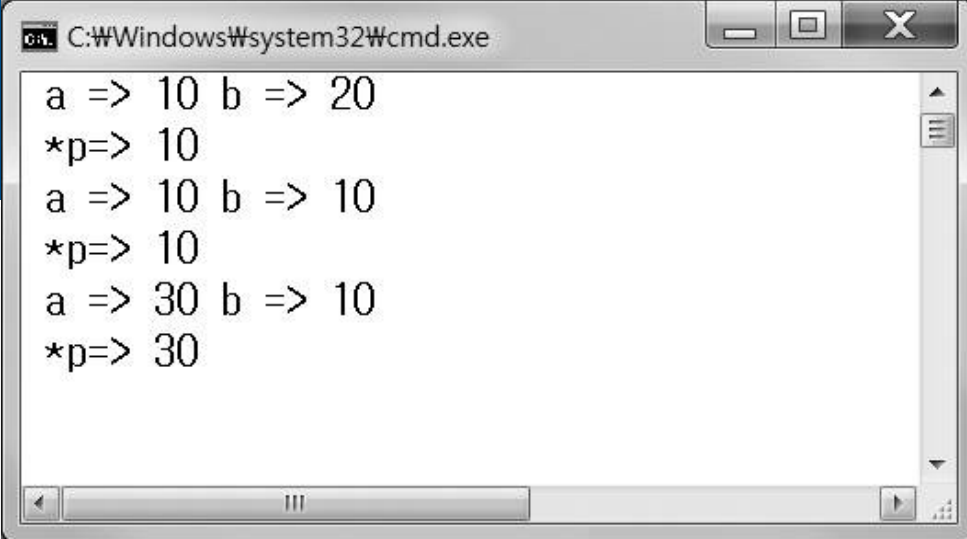
```
int *p;  
p=30; // 컴파일 에러
```

㉣

```
*p=30;
```

포인터 변수를 메모리에 할당

```
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
```

- **매개변수 전달방법**

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

- 함수 측에서 함수를 호출한 측으로 값을 전달하려면 return문으로 반환
- 형식 매개변수가 실 매개변수와 별개의 기억공간을 할당받아 값을 복사하므로 함수를 정의한 곳에서 형식 매개변수의 값을 변경해도 실 매개변수의 값은 변경되지 않음

함수 호출 측	
a=10, b=20; sum=add(a,b);	
a	10
b	20
sum	?

실 매개변수 a, b값을
형식 매개변수 x, y
값에 복사하고
결과 z값 반환

→ 복사

→ 복사

함수 측	
<pre>int add(int x, int y) { int z; z = x+y; return(z); }</pre>	
10	x
20	y
?	z

- 함수 측에서 함수를 호출한 측으로 값을 전달하려면 return문으로 반환
- 형식 매개변수가 실 매개변수와 별개의 기억공간을 할당받아 값을 복사하므로 함수를 정의한 곳에서 형식 매개변수의 값을 변경해도 실 매개변수의 값은 변경되지 않음

함수 호출 측	
a=10, b=20; sum=add(a,b);	
a	10
b	20
sum	?

실 매개변수 a, b값을
형식 매개변수 x, y
값에 복사하고
결과 z값 반환

→ 복사

→ 복사

함수 측	
<pre>int add(int x, int y) { int z; z = x+y; return(z); }</pre>	
10	x
20	y
30	z

- 함수 측에서 함수를 호출한 측으로 값을 전달하려면 return문으로 반환
- 형식 매개변수가 실 매개변수와 별개의 기억공간을 할당받아 값을 복사하므로 함수를 정의한 곳에서 형식 매개변수의 값을 변경해도 실 매개변수의 값은 변경되지 않음

함수 호출 측	
a=10, b=20; sum=add(a,b);	
a	10
b	20
sum	30

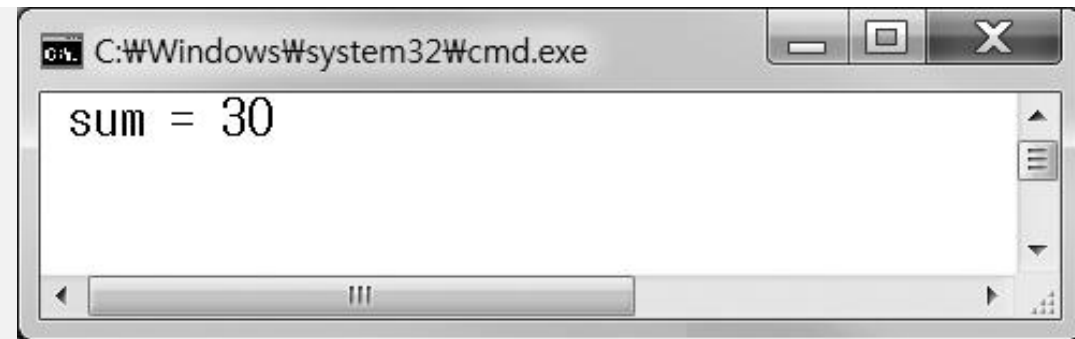
실 매개변수 a, b값을
형식 매개변수 x, y
값에 복사하고
결과 z값 반환

→ 복사
→ 복사
← 반환

함수 측	
int add(int x, int y) { int z; z = x+y; return(z); }	
10	x
20	y
30	z

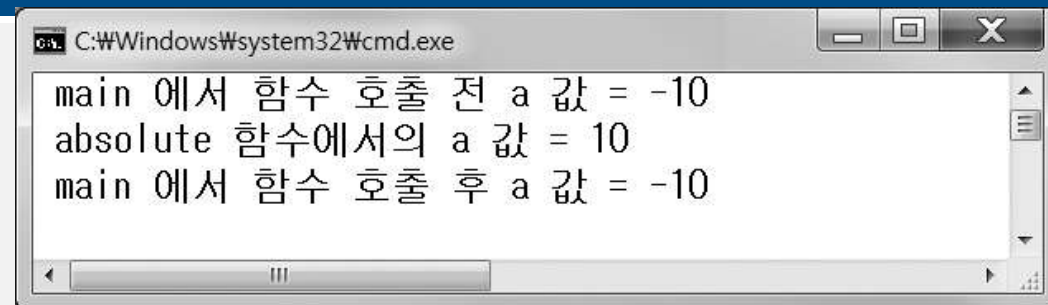
값에 의한 전달 방식의 함수

```
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);
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 }
```



값에 의한 전달 방식의 함수 (예 : 절대값 구하기)

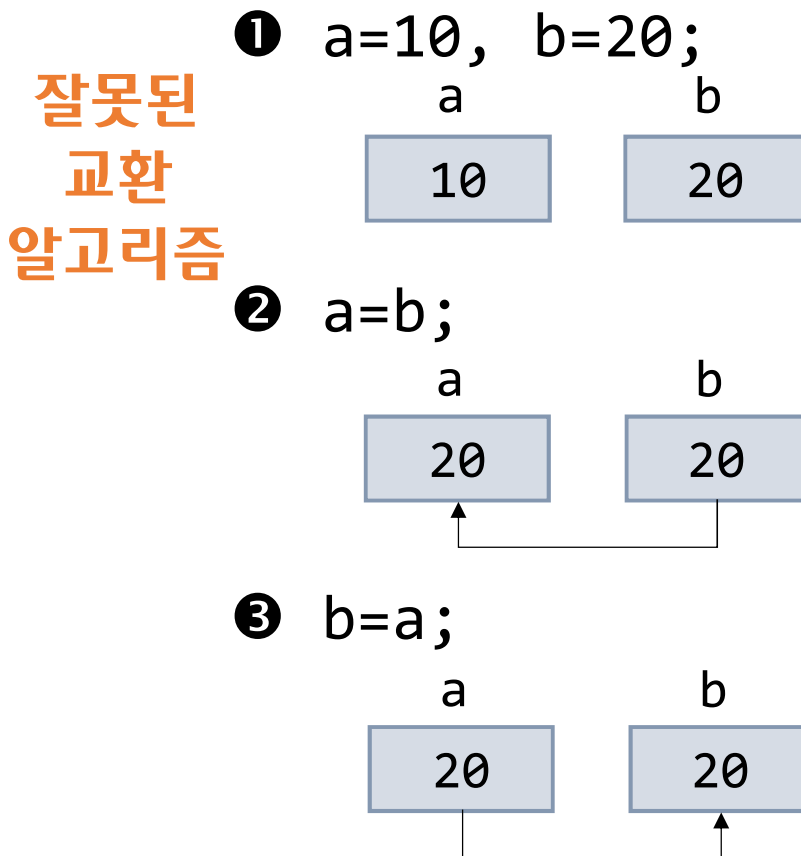
```
01 #include <iostream>
02 using namespace std;
03 void absolute(int a);
04 void main()
05 {
06     int a=-10;
07     cout<<" main 에서 함수 호출 전 a 값 = "<< a <<"\n";
08     absolute(a);
09     cout<<" main 에서 함수 호출 후 a 값 = "<< a <<"\n";
10 }
11
12 void absolute(int a)
13 {
14     if(a<0)
15         a=-a;
16     cout<<" absolute 함수에서의 a 값 = "<< a <<"\n";
17 }
```



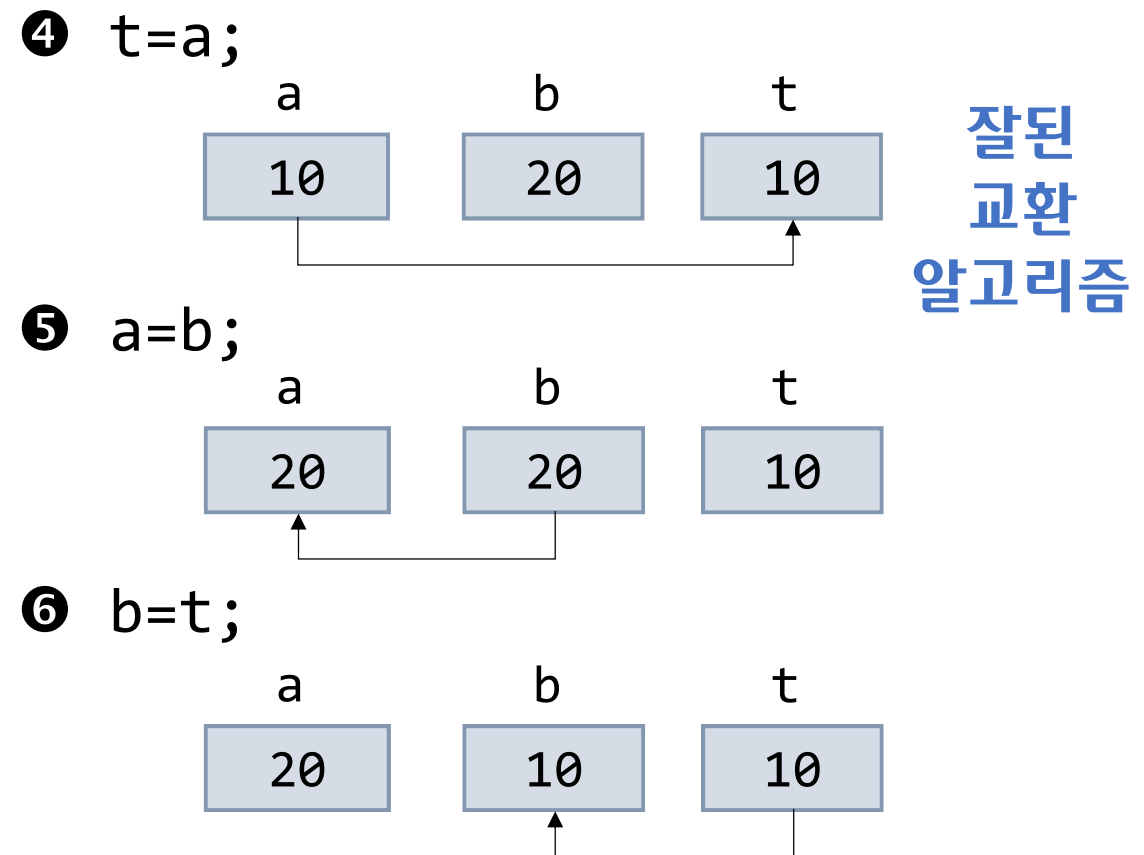
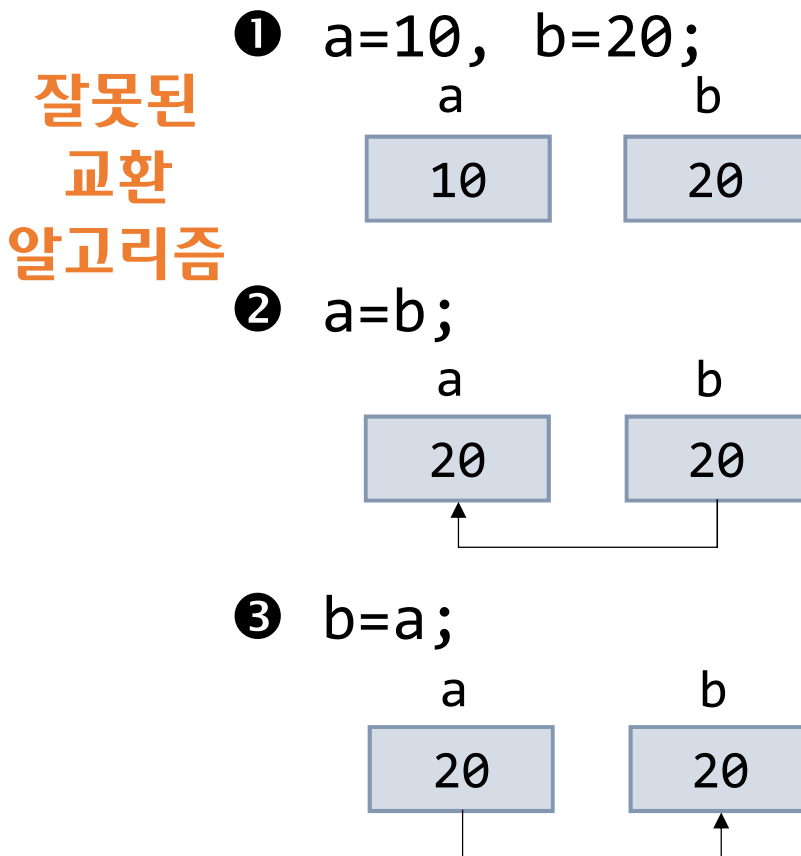
C:\Windows\system32\cmd.exe

```
main 에서 함수 호출 전 a 값 = -10
absolute 함수에서의 a 값 = 10
main 에서 함수 호출 후 a 값 = -10
```


- 함수의 일반적인 전달 방식은 값에 의한 전달 방식
- 한계 극복을 위해 주소에 의한 전달 방식 제공
- 예제 : 교환 알고리즘

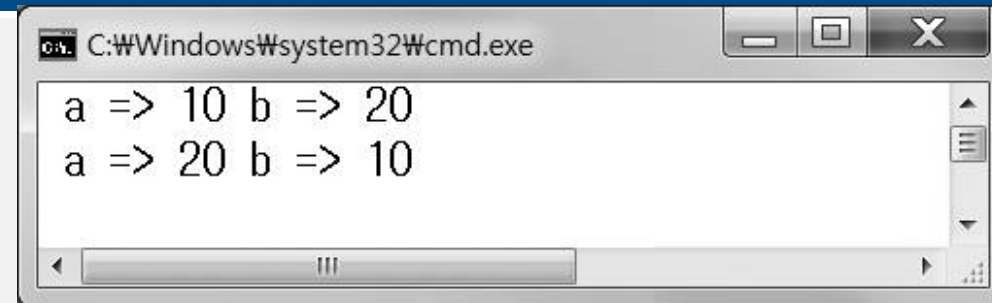


- 함수의 일반적인 전달 방식은 값에 의한 전달 방식
- 한계 극복을 위해 주소에 의한 전달 방식 제공
- 예제 : 교환 알고리즘



두 변수에 저장된 값 교환

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



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window displays the output of the C++ program, which is two lines of text: "a => 10 b => 20" on the first line and "a => 20 b => 10" on the second line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

값에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```

main함수측

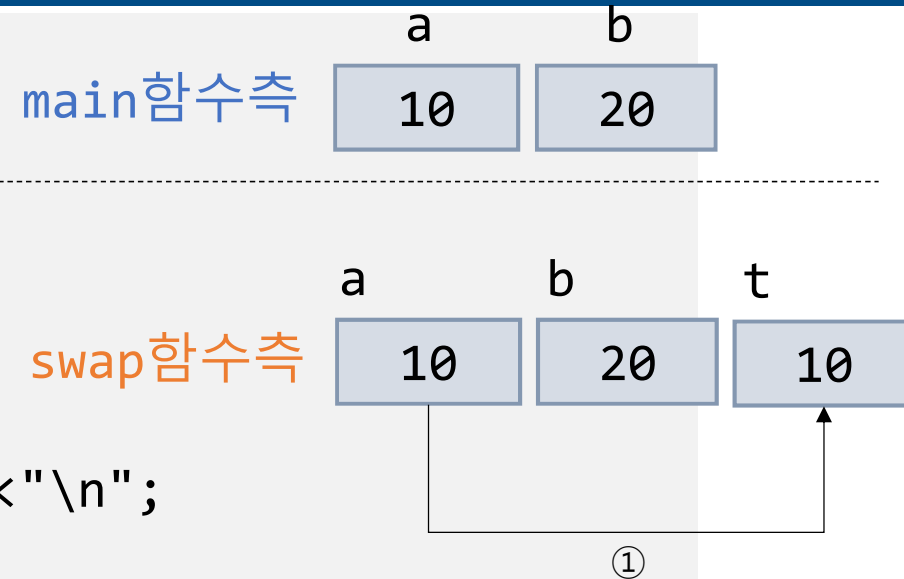
a	b
10	20

swap함수측

a	b	t
10	20	

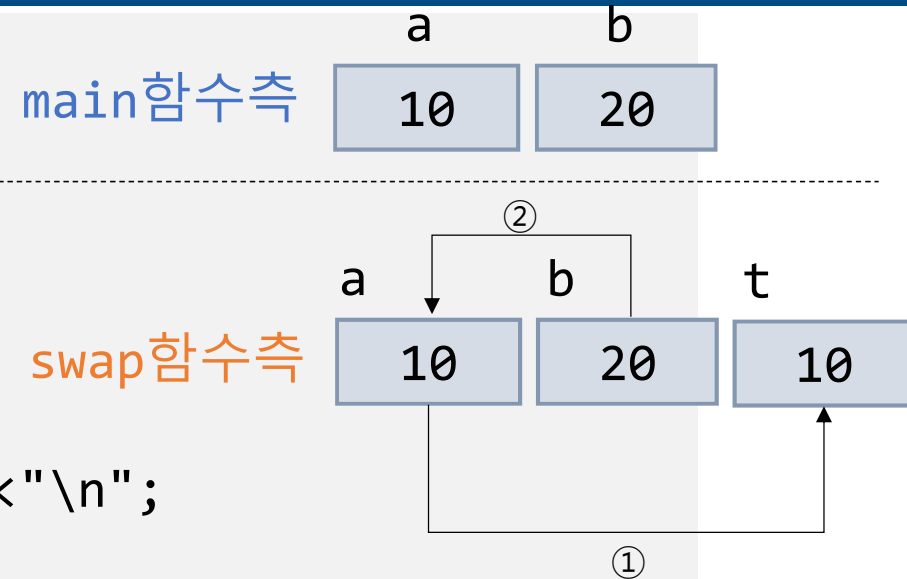
값에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



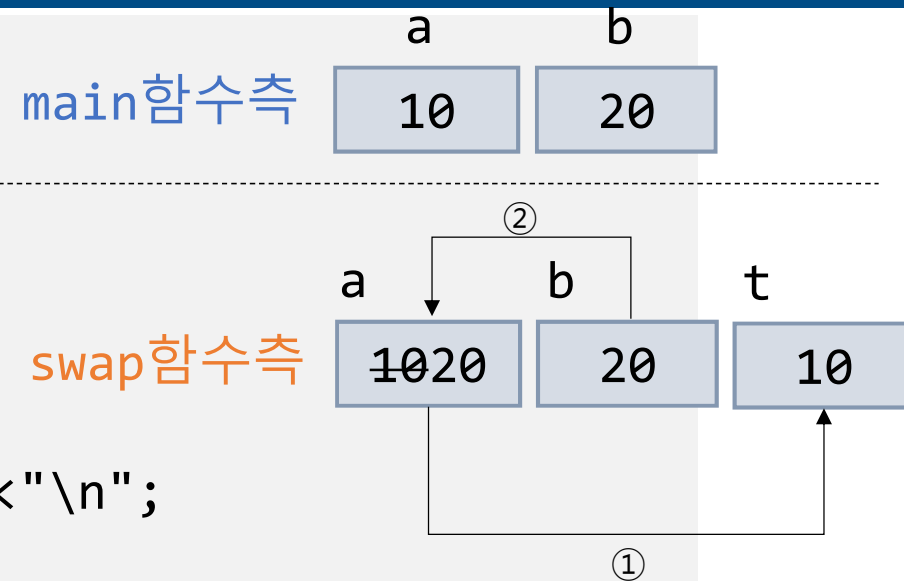
값에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



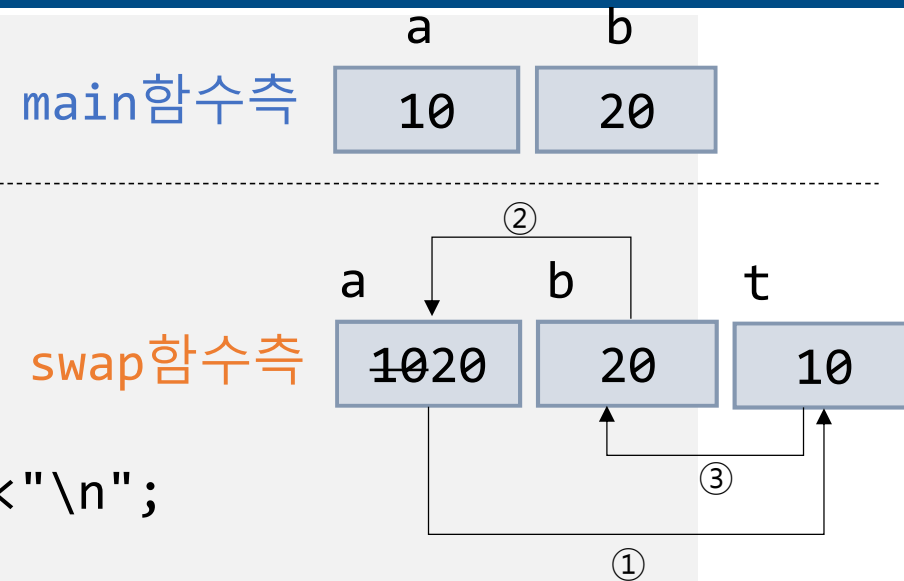
값에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



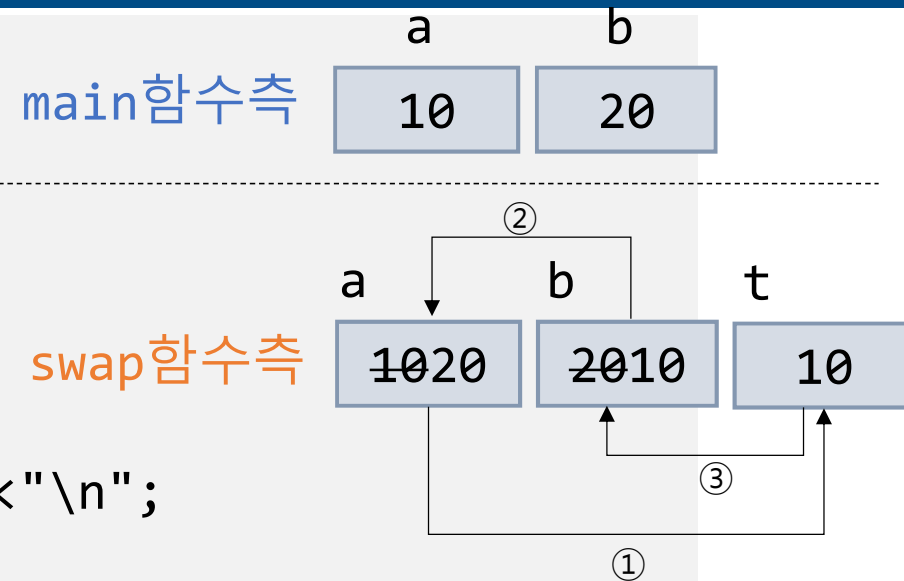
값에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



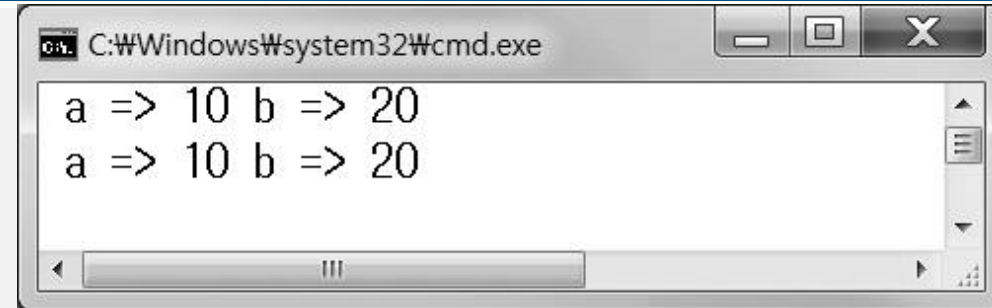
값에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



값에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window contains two lines of output: "a => 10 b => 20" on the first line and "a => 10 b => 20" on the second line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

주소에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```

<11행>

main함수측

a (0013ft7c) b (0013ft78)

10

20

swap함수측

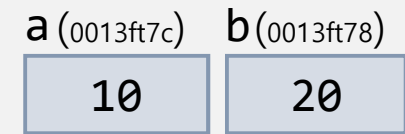
pa

pb

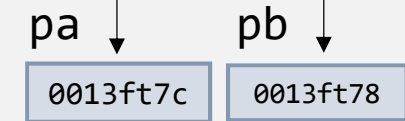
주소에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```

<11행>
main함수측

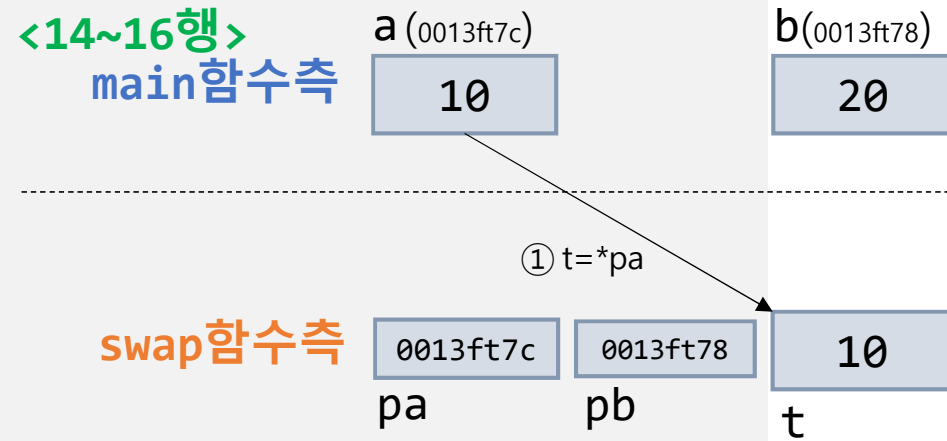


swap함수측



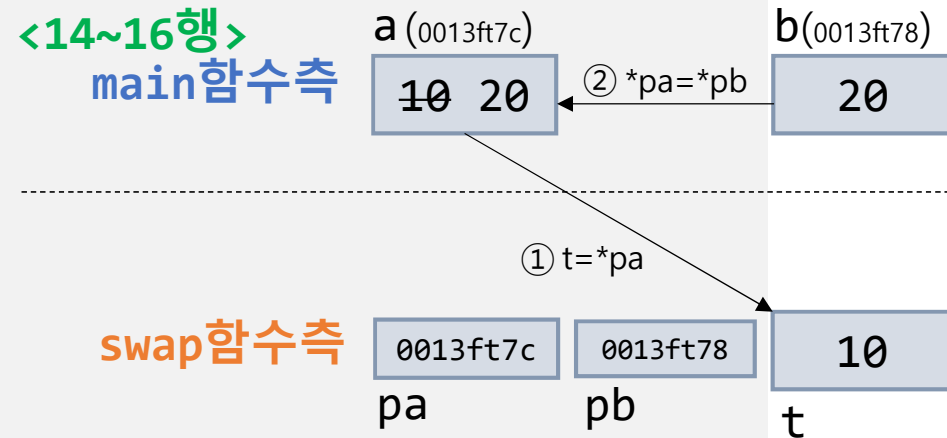
주소에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



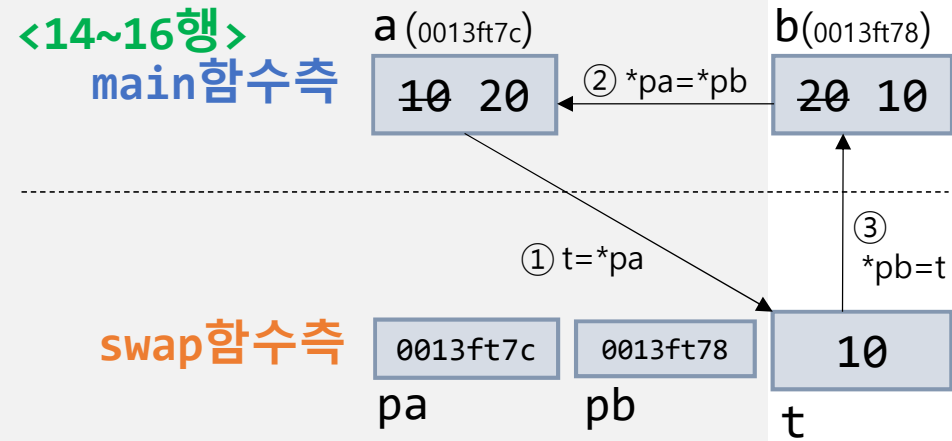
주소에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



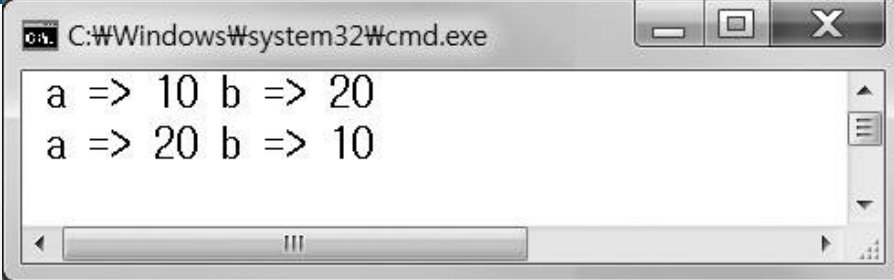
주소에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



주소에 의한 전달방식으로 두 변수의 값 교환

```
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 }
```



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


● 참조 변수

- 별칭(일종의 다른 이름)을 의미하며 따로 기억공간이 할당되지 않음
- 참조 변수 선언 시 부여한 변수명을 이미 선언되어 있는 변수의 별칭으로 사용
- 메모리상에 하나 존재하는 변수를 여러 이름으로 접근해서 사용하는 방법

● 참조 변수의 선언 방법

- 별칭으로 사용할 변수명 앞에 & 기호를 덧붙임
- 사용하는 & 기호는 참조 연산자

참조 변수 선언 기본 형식

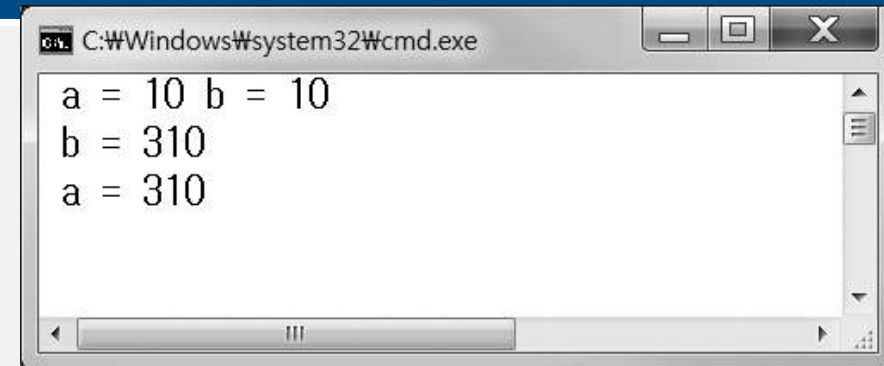
자료형 &별칭으로 사용할 변수명 = 앞서 선언된 변수명

● 참조 변수를 선언할 때 주의할 점

- 변수 선언 시 초기값을 설정해야 함

참조 변수 선언하기

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05     int a=10;
06     int &b = a;
07     cout<<" a = "<<a<<" b = "<<b<<endl;
08     b+=300;
09     cout<<" b = "<<b<<endl;
10     cout<<" a = "<<a<<endl;
11 }
```

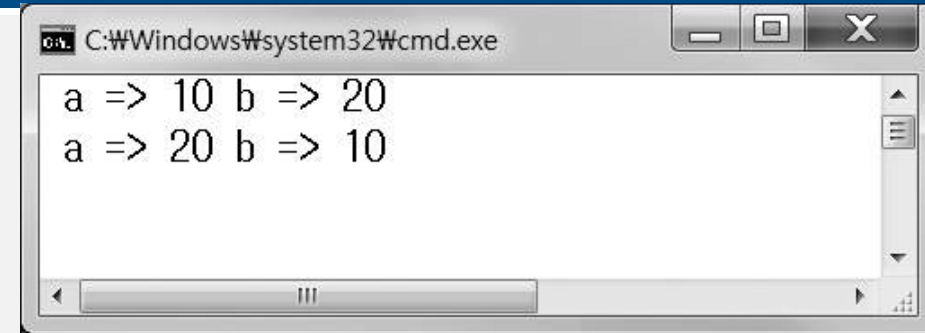


The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed as follows:

```
a = 10 b = 10
b = 310
a = 310
```

참조에 의한 전달 방식 – 두 변수값 교환 함수

```
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);
09     cout<<" a => " << a <<" b => " << b <<"\n";
10 }
11 void swap(int &x, int &y) /* x, y는 기억 공간을 따로 할당 받지 않고
12 {                               실 매개변수 a, b의 별칭으로 선언 */
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
```

형식매개변수 x,y는
실매개변수 a, b의 별칭이므로
swap함수 호출 후 변수 a,b의
값은 바뀐다.

**/* x, y는 기억 공간을 따로 할당 받지 않고
실 매개변수 a, b의 별칭으로 선언 */**