```
 1.  #include <iostream>
 2.  using namespace std;
 3.  void swap (int* m, int* w)
 4.  {   int b;
 5.      b=*m;
 6.      *m=*w;
 7.      *w=b;
 8.  }
 9.  int main()
10.  {   int a=1, b=2;
11.      cout << endl << "a=" << a << " b=" << b;
12.      swap(&a,&b);
13.      cout << endl << "a=" << a << " b=" << b;
14.      return 0;
15.  }
```

Figure 1. Program **p01**

Program **p01** produces:

a=1 b=2
a=2 b=1

Program **p01** notes:

1. Function *swap* is required to interchange the values of variables *a* and *b*.
2. Since all arguments are passed by value in C programs, passing copies of the values of variables *a* and *b* to parameters of *swap* will not accomplish the stated goal for function swap. Copies of the values assigned to the formal parameters of function *swap* will be discarded when function *swap* returns.
3. Thus, the addresses of variables *a* and *b* must be passed to function swap. The addresses of variables *a* and *b* in function swap will allow function *swap* to interchange their values. Please refer to figure 2.

	address	variable	type	value
swap	0003	w	int*	0001
	0002	m	int*	0000
main	0001	b	int	2
	0000	a	int	1

Figure 2. Activation records for program **p01**

4. The statements in function *swap* are illustrated in figures 3, 4, 5, and 6. The diagram in figure 3 represents the state of function *swap* just after line 4. The diagram in figure 4 shows the values of formal parameters *m* and *w* and local variable *b* after line 5. Figure 5 depicts parameters *m* and *w* and variable *b* after line 6 and figure 6 shows the values of parameters *m* and *w* and variable *b* just before function *swap* returns.

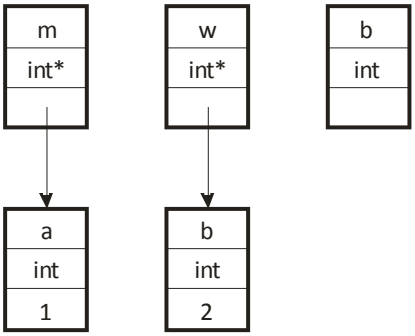

Figure 3. Function *swap* just after line 4

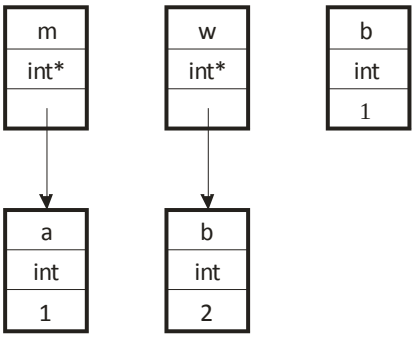

Figure 4. Function *swap* just after line 5

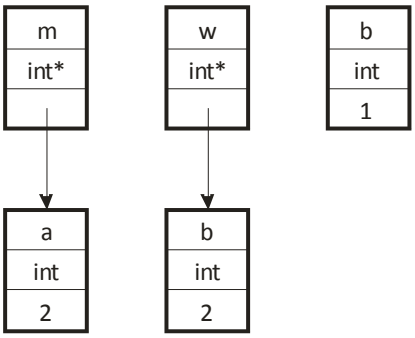

Figure 5. Function *swap* just after line 6

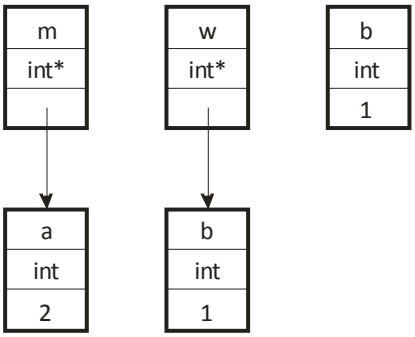

Figure 6. Function *swap* just before it returns

```
 1.  #include <iostream>
 2.  using namespace std;
 3.  void swap (int& m, int& w)
 4.  {   int b;
 5.      b=m;
 6.      m=w;
 7.      w=b;
 8.  }
 9.  int main()
10.  {   int a=1, b=2;
11.      cout << "\na=" << a << " b=" << b;
12.      swap(a,b);
13.      cout << "\na=" << a << " b=" << b;
14.      return 0;
15.  }
```

Figure 7. Program **p02**

Program **p02** produces:

a=1 b=2
a=2 b=1

Program **p02** notes:

1. Function *swap* is required to interchange the values of variables *a* and *b*.
2. Formal parameters *m* and *w* are references. References are known by the **&** operator following the type. The *type-name* and the **&** together are the complete type.
3. A reference is an alias. An alias is another name for an object. In figure 7, formal parameter *m* is an alias for variable *a*. Variable *a* is bound to formal parameter *m* when function *swap* is called. Variable *a* is the first argument of function *swap*.
4. The significance of a reference is that whenever an action is performed on the alias it is also performed on the original object. Thus, when the value of formal parameter *w* is accessed on line 6, the value of variable *a* declared on line 10 is actually retrieved. Formal parameter *w* is another name for variable *a*.
5. The reference syntax is an enhancement to C available in C++.
6. The actions in program **p02** are identical in every respect to the actions performed in program **p01**.