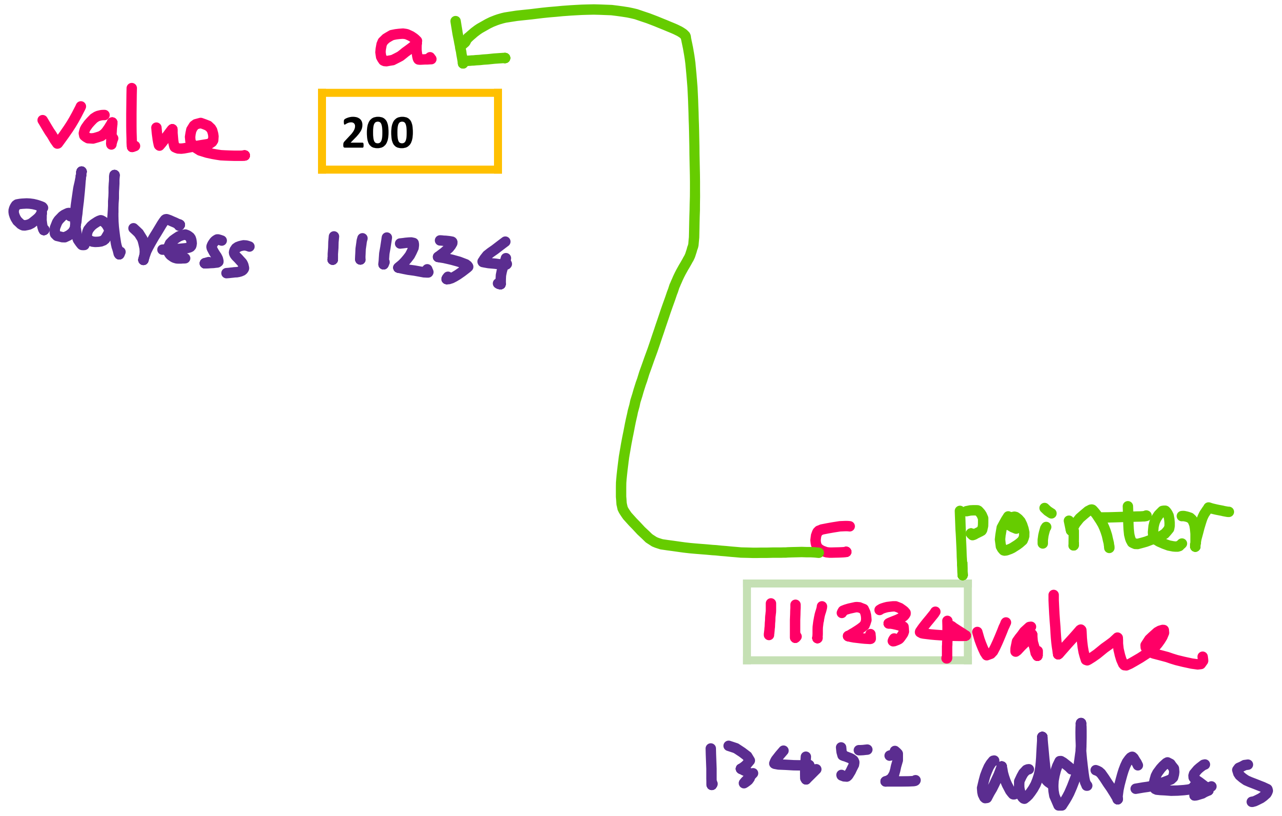
## Variables, Memory and Pointers


a
value
200
address
111234
c
pointer
111234
value
13452
address

## Variables, Memory and Pointers

- A variable is a named piece of memory
  - The name stands in for the *memory address*

```
int num;//allocate memory to it first
num = 10;
```

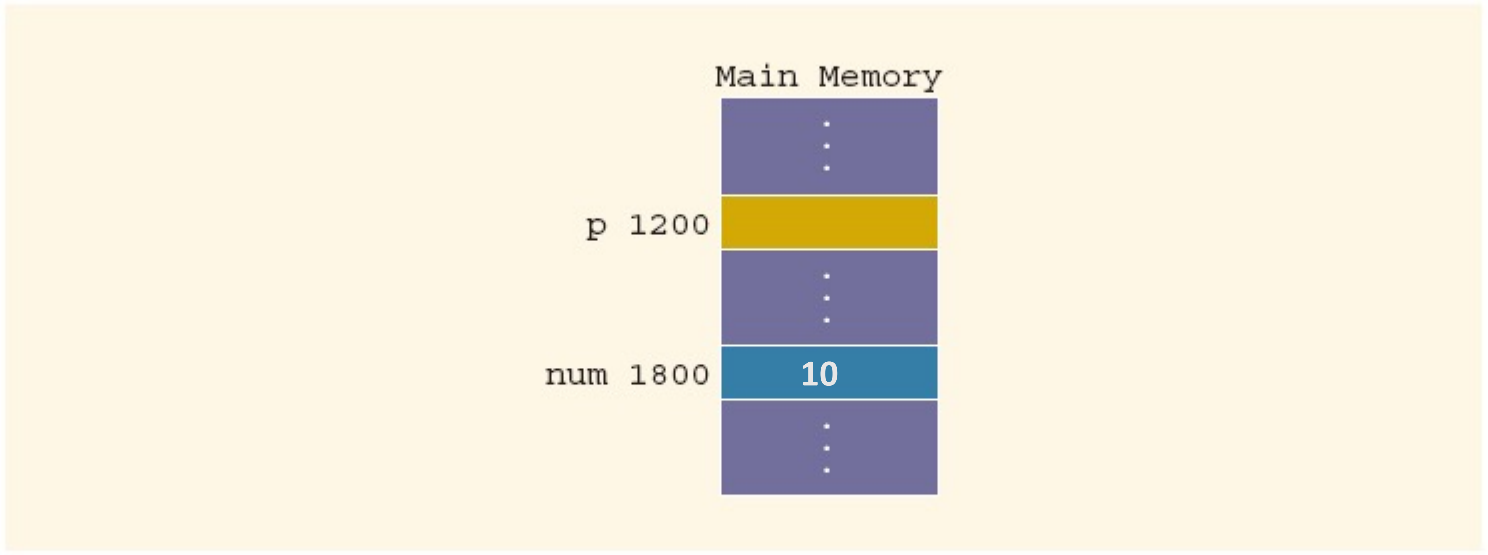


FIGURE 13-1 Main memory, p, and num

## Variables, Memory and Pointers

- When a value is assigned to a variable, it is stored at that address in memory

```
num = 78;
```

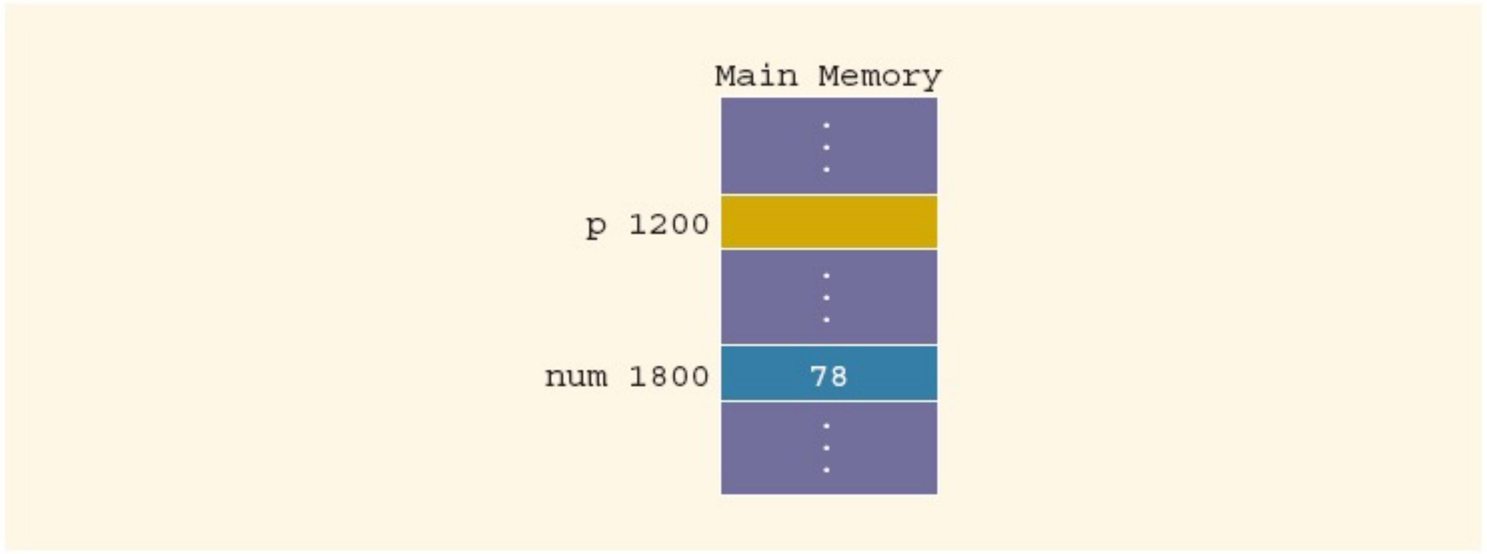


**FIGURE 13-2** num after the statement num = 78; executes

# Variables, Memory and Pointers

- A *pointer* is a variable that holds the address of another variable
  - It is declared in terms of the type of variable it points at:

```
int *p; // given a * in front of a variable, it means
  that this variable is a pointer.
```

- int num; num = 78;

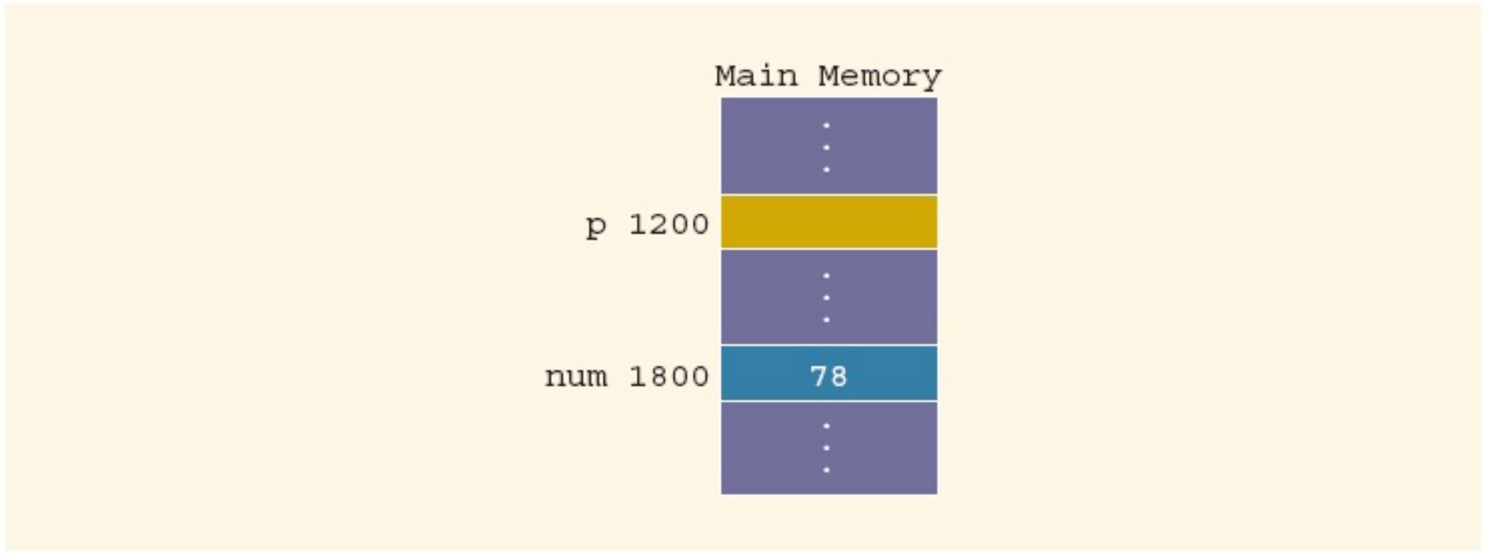


**FIGURE 13-2** num after the statement num = 78; executes

## Variables, Memory and Pointers

- The operator & returns the address of a variable
  - It can then be assigned to a pointer

```
p = &num;
// &num => the address of the variable num ⇔ 1800
// assign the address of num to the value of p.
```

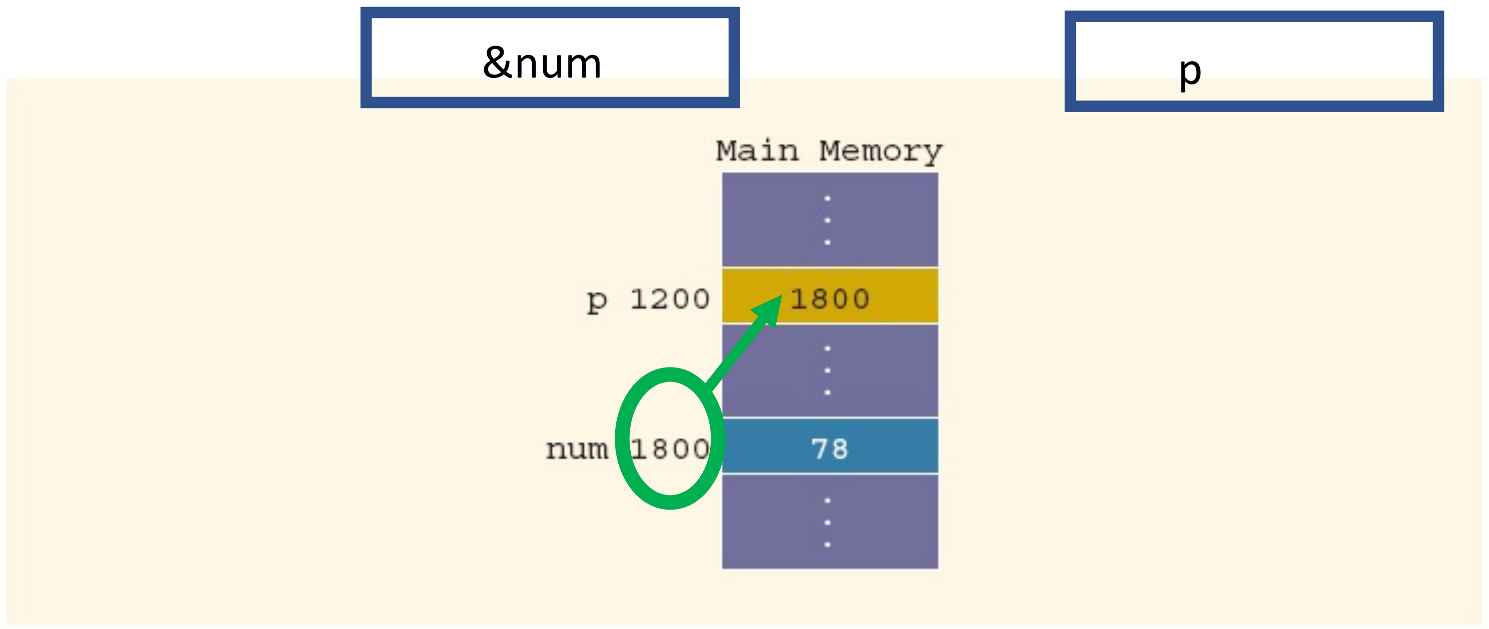


FIGURE 13-3 p after the statement p = &num; executes

## Variables, Memory and Pointers

- The operator `*` takes an address (a pointer) and returns the location in memory being pointed to
  - Can only be applied to a pointer

```
*p = 24;
int *q; // define a pointer;
*q = 30; // assign 30 to the variable that the pointer
  pointed to.
```

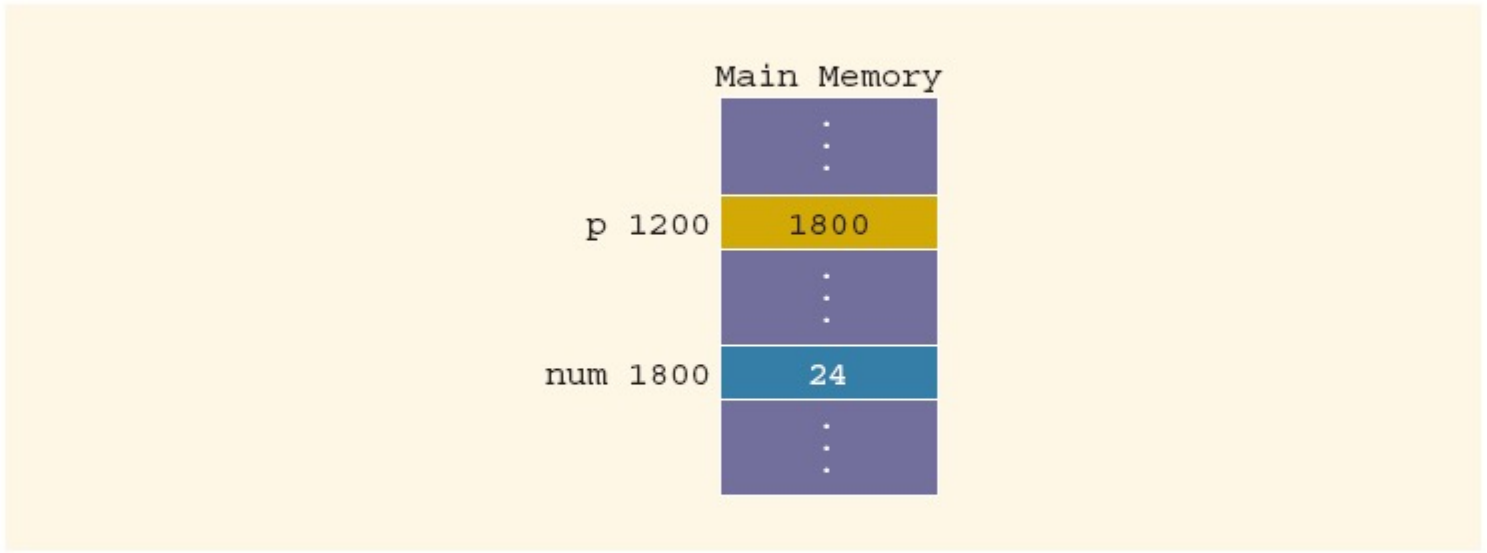


FIGURE 13-4 *p and num after the statement *p = 24; executes

## Declaring Pointer Variables

- Syntax:

```
dataType *identifier;
```

- Examples:

```
int *p; // this pointer will point to an integer variable
char *ch; // a pointer, ch, points to a character variable
```

- These statements are equivalent:

```
int  *p;
int*  p;
int * p;
```

## Declaring Pointer Variables (continued)

- In the statement:

```
int* p, q; // p is a pointer; q is variable
int num1, num2; ⇔ int num1; int num2;
```

  only `p` is the pointer variable, not `q`; here `q` is an `int` variable

- To avoid confusion, attach the character `*` to the variable name:

```
int   *p, q;
int   *p, *q;
int array1[100], array2[20];
```

# Address of Operator (`&`)

- The ampersand, `&`, is called the ***address of operator***
- The address of operator is a unary operator that returns the ***address of its operand***

- ***Binary operator***

  ***LHS = RHS, cout << "hello", cin >> a, +, -, *, /, %***

- ***Unary operator***

  ***Only need one side***

  ***&RHS;  &a***