THIRD EDITION

A First Book of C++

From Here to There

CHAPTER 4

Pointers

Addresses and Pointers

- High-level languages use memory addresses throughout executable programs
  - Keeps track of where data and instructions are physically located inside of computer
- **C++ Attribute:** Programmer provided access to addresses of program variables
  - This capability typically not provided in other high-level languages
- **Pointer (Pointer Variable):** a variable that stores the address of another variable

Addresses and Pointers (continued)

- Three major attributes of a variable:
  - **Data type:** declared in a declaration statement
  - **Value:** Stored in a variable by:
    - Initialization when variable is declared
    - Assignment
    - Input
  - **Address:** For most applications, variable name is sufficient to locate variable's contents
    - Translation of variable's name to a storage location done by computer each time variable is referenced

Addresses and Pointers (continued)

- Programmers are usually only concerned with a variable's value, not its address
- **Address operator &:** determines the address of a variable
  - & means "address of"
  - When placed in front of variable `num`, is translated as "the address of `num`"
- Program 9.2 uses the address operator to display the address of variable `num`

Addresses and Pointers (continued)

Program 9.2

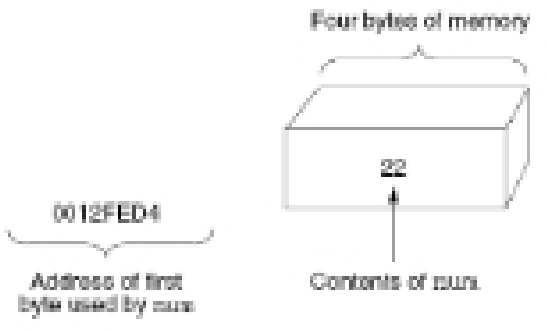
```
#include <iostream>
using namespace std;

int main()
{
  int num;

  num = 22;
  cout << "num = " << num << endl;
  cout << "The address of num = " << &num << endl;

  return 0;
}
```

The output of Program 9.2 is

```
num = 22
The address of num = 0012FED4
```


Addresses and Pointers (continued)

FIGURE 9.3 *A More Comprehensive Picture of the Variable num*



Storing Addresses

- Address can be stored in suitably declared variables
- **Example:** `numAddr = &num;`
  - Statement stores address of `num` in variable `numAddr`
  - `numAddr` is a pointer

Storing Addresses (continued)

FIGURE 9.4 *Storing num's Address in numAddr*

Variable name	Contents
numAddr	Address of num

Using Addresses

- **Indirection Operator** * : The * symbol, when followed by a pointer, means "the variable whose address is stored in"
  - If `y` is a pointer, then `*y` means "the variable whose address is stored in `y`"
  - Commonly shortened to "the variable pointed to by `y`"
- **Example** (Figure 9.6):
  - The content of `y` is the address `FFAA`
  - The variable pointed to by `y` – `qqqq`