PEN203

C++ Pointers

C++ How to Program
Deitel & Deitel

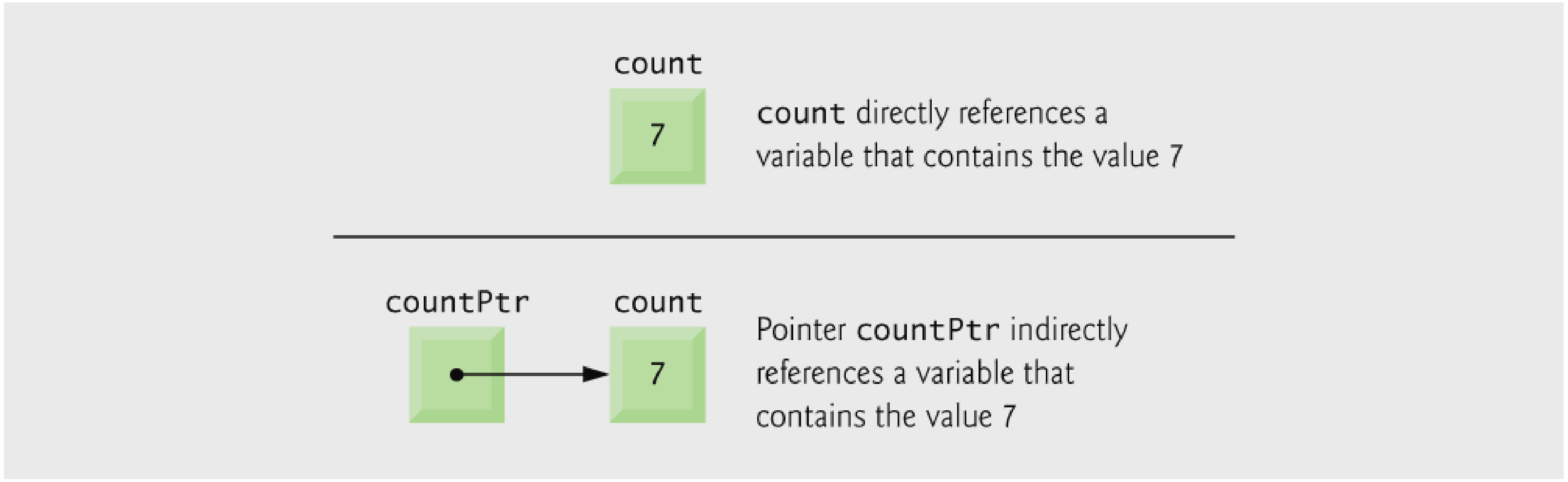
Outline

- Pointer Variable Definitions and Initialization
- Pointer Operators
- Passing Arguments to Functions by Reference
- Using const Qualifier with Pointers
- sizeof operator
- Pointer Expressions and Pointer Arithmetic
- Relationship between Pointers and Arrays
- Arrays of Pointers

Pointer Variable Definitions and Initialization

- Pointer variables store memory addresses as their values
- Pointer variables contain an address of a variable that has a specific value (indirect reference).

Pointer Variable Definitions and Initialization


count
7
count directly references a variable that contains the value 7
countPtr
count
7
Pointer countPtr indirectly references a variable that contains the value 7

Pointer Variable Definitions and Initialization

- Pointer variables store memory addresses as their values
- Pointer variables contain an address of a variable that has a specific value (indirect reference).
- Pointer definition:
  - int *myptr; statement defines a pointer of type int.
  - You may initialize pointers to 0, NULL or an address.

Pointer Operators

- & (address operator)
  - returns memory address of operand

```
int a=3;
int ptr;
ptr=&a;
```

  - With these declarations and assignments, ptr points to a.

Pointer Operators

- * (dereferencing operator)
  - Returns an alias of what its operand points to
  - * ptr returns a in our example
  - * can be used for assignment
  - *ptr= 10 modifies the value of a to 10
  - Dereferenced pointer must be a left value.
- * and & are inverses

Pointer Operators

```
// Fig. 5.4: fig05_04.cpp
// Using the & and * operators.
#include <iostream>

using std::cout;
using std::endl;

int main()
{
   int a;      // a is an integer
   int *aPtr;  // aPtr is a pointer to an integer

   a = 7;
   aPtr = &a;  // aPtr assigned address of a

   cout << "The address of a is " << &a
        << "\nThe value of aPtr is " << aPtr;

   cout << "\n\nThe value of a is " << a
        << "\nThe value of *aPtr is " << *aPtr;

   cout << "\n\nShowing that * and & are inverses of "
        << "each other.\n&*aPtr = " << &*aPtr
        << "\n*&aPtr = " << *&aPtr << endl;

```

Pointer Operators

```
    return 0;  // indicates successful termination

} // end main
```

```
The address of a is 0012FED4
The value of aPtr is 0012FED4

The value of a is 7
The value of *aPtr is 7

Showing that * and & are inverses of each other.
&*aPtr = 0012FED4
*&aPtr = 0012FED4
```

Passing Arguments to Functions by Reference

- Call functions by reference using pointer arguments
- To pass address of an argument, & operator will be used.
- Using * operator in function, you can modify the original value.
- Arrays are not passed with & operator: array name is already an address.

Passing Arguments to Functions by Reference

```
// Fig. 5.7: fig05_07.cpp
// Cube a variable using pass-by-reference
// with a pointer argument.
#include <iostream>

using std::cout;
using std::endl;

void cubeByReference( int * );   // prototype

int main()
{
   int number = 5;

   cout << "The original value of number is " << number;

   // pass address of number to cubeByReference
   cubeByReference( &number );

   cout << "\nThe new value of number is " << number << endl;

   return 0;  // indicates successful termination

} // end main

```

Passing Arguments to Functions by Reference

```
// calculate cube of *nPtr; modifies variable number in main
void cubeByReference( int *nPtr )
{
   *nPtr = *nPtr * *nPtr * *nPtr;  // cube *nPtr

} // end function cubeByReference
```

```
The original value of number is 5
The new value of number is 125
```