PEN203

C++ Pointers

C++ How to Program
Deitel & Deitel

Using const Qualifier with Pointers

- const qualifier is used if you do not need to modify a variable
- Any attemp to change a const variable causes a syntax error.
- const pointers must be initialized when declared. Can not be changed to point another location during program execution.

Using const Qualifier with Pointers

- int *const ptr = &a;
  - Constant pointer to an integer
- const int *ptr = &a;
  - Modifiable pointer to a constant integer
- const int *const ptr = &a;
  - Constant pointer to a constant integer.

sizeof Operator

- sizeof returns the size of operand in bytes.
- Sizeof can be used variable names and type names.
- Examples:
  - int x=5;
  - sizeof(int) and sizeof(x) return the same value that is the number of bytes allocated for integers.

Pointer Expressions and Pointer Arithmetic

- Arithmetic operations on pointers
  - Increment/decrement pointer
  - Add an integer to a pointer
  - A pointer can be subtracted from each other
- 5 element array:

  int ar[5];

  int *ptr=&ar[0];

  if the address of ar[0] is 2000, ptr+=2; sets ptr to 2008.

Pointer Expressions and Pointer Arithmetic

- 5 element array: (subtraction)

```
int ar[5];
int *ptr1=&ar[1];
int *ptr2=&ar[3];
ptr2-ptr1 returns 2
```

- Pointer comparison (<, ==, >)
  - Used to find which pointer points to greater numbered array element.
- Pointers of the same type can be assigned to each other.

Relationship between Pointers and Arrays

- Array names are constant pointers.

  int ar[5];

  int *ptr;

  ptr=ar; or ptr=&ar[0] assigns the address of first element on integer array ar to ptr.

Relationship between Pointers and Arrays

- Array element ar[2] can be accessed:
  - *(ptr+2) pointer/offset notation
  - ptr[2] pointer/subscript notation
  - Also can be accessed using pointer arithmetic on the array itself *(ar+3)
- You can not modify an array name with pointer arithmetic.

Relationship between Pointers and Arrays

```
// Fig. 5.20: fig05_20.cpp
// Using subscripting and pointer notations with arrays.

#include <iostream>

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

int main()
{
  int b[] = { 10, 20, 30, 40 };
  int *bPtr = b;   // set bPtr to point to array b

  // output array b using array subscript notation
  cout << "Array b printed with:\n"
       << "Array subscript notation\n";

  for ( int i = 0; i < 4; i++ )
    cout << "b[" << i << "] = " << b[ i ] << '\n';

  // output array b using the array name and
  // pointer/offset notation
  cout << "\nPointer/offset notation where "
       << "the pointer is the array name\n";

```

Relationship between Pointers and Arrays

```
   for ( int offset1 = 0; offset1 < 4; offset1++ )
      cout << "*(b + " << offset1 << ") = "
           << *( b + offset1 ) << '\n';

   // output array b using bPtr and array subscript notation
   cout << "\nPointer subscript notation\n";

   for ( int j = 0; j < 4; j++ )
      cout << "bPtr[" << j << "] = " << bPtr[ j ] << '\n';

   cout << "\nPointer/offset notation\n";

   // output array b using bPtr and pointer/offset notation
   for ( int offset2 = 0; offset2 < 4; offset2++ )
      cout << "*(bPtr + " << offset2 << ") = "
           << *( bPtr + offset2 ) << '\n';

   return 0;  // indicates successful termination

} // end main
```

Relationship between Pointers and Arrays

```
Array b printed with:

Array subscript notation
b[0] = 10
b[1] = 20
b[2] = 30
b[3] = 40

Pointer/offset notation where the pointer is the array name
*(b + 0) = 10
*(b + 1) = 20
*(b + 2) = 30
*(b + 3) = 40

Pointer subscript notation
bPtr[0] = 10
bPtr[1] = 20
bPtr[2] = 30
bPtr[3] = 40

Pointer/offset notation
*(bPtr + 0) = 10
*(bPtr + 1) = 20
*(bPtr + 2) = 30
*(bPtr + 3) = 40
```

Arrays of Pointers

- Arrays can contains pointers
- Example: an array of strings

```
char *suit[ 4 ] = { "Hearts", "Diamonds", "Clubs", "Spades" };
```

- An important issue here is the strings are not actually placed in the array. Only pointers to first character of strings are stored.