

Pointers

- To provide backwards compatibility with C and C++, C# allows the use of pointers in addition to references
- C and C++ use pointers to refer to objects
- A pointer provides the memory address of an object
 - Unlike references, pointers are not managed by C# and its automatic memory management system
 - A pointer value only changes when the program explicitly changes the value
- In C and C++, heap allocated objects must be manually allocated and de-allocated using new and delete operators

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brash, and C. C. W. Hall. All Rights Reserved

Using Pointers

- Pointers are a very powerful construct
- A pointer to an object of type int can be created as follows within C or C++:

```
int *ptr;      // The * denotes that this is a pointer to an integer
```

- Once a pointer has been created, it may be use to refer to an object or a specific memory address:

```
int a = 10;    // This allocates an ordinary variable a of type int
```

```
ptr = &a;      // The & denotes the address of a variable
               // In this case, ptr stores the address of variable a
```

```
ptr = 0xC000;  // In this case, ptr stores the address C000 in hexadecimal
```

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brash, and C. C. W. Hall. All Rights Reserved

The Power of Pointers

- If you wish to access a specific memory location within a computer, it is possible to do so using pointers
- For example, if I wanted to write an integer value of 65 to a parallel port mapped to location 0x378, I could do so using the following lines of C code...

```
int *a;
```

```
a = 0x378;      // This sets the address of pointer a to 0x378
```

```
*a = 65;        // This is called dereferencing a pointer
                 // The use of the * in this case denotes
                 // dereferencing. A value of 65 is written
                 // to the location pointed to by a
```

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brash, and C. C. W. Hall. All Rights Reserved

References vs. Pointers

References

- Provide access to specific objects in memory
- Can be modified by the automatic memory management system

Pointers

- Provide access to specific addresses in memory
- Cannot be modified by the automatic memory management system

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brash, and C. C. W. Hall. All Rights Reserved

Using Pointers in C#

- To use pointers in C#, you must do two things:
 - Use the unsafe modifier to identify the scope in which pointers will be used
 - Compile your program using the unsafe compiler option (/unsafe+)
- The unsafe modifier / compiler option simply indicates that C# does not manage the code in the typical way
 - For example, unsafe code is not subject to array bounds checking and some forms of type checking
- The unsafe modifier can be applied to classes, methods, blocks of code enclosed in braces, and specific variables

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brush, and C. C. W. Halls. All Rights Reserved

Unsafe Code Example

```
using System;

class MyTest
{
    unsafe static void Swap( int *a, int *b )
    {
        int tmp = *a;

        *a = *b;
        *b = tmp;
    }

    unsafe static void Main(string [] args)
    {
        int x = 5;
        int y = 12;

        Console.WriteLine( "x = {0}, y = {1}", x, y );

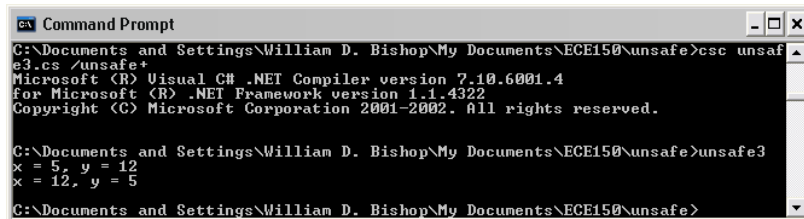
        Swap( &x, &y );

        Console.WriteLine( "x = {0}, y = {1}", x, y );
    }
}
```

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brush, and C. C. W. Halls. All Rights Reserved

Unsafe Code Example Output

- Here is the output of the previous example:



Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brush, and C. C. W. Halls. All Rights Reserved

- Note that since the addresses of the integers were passed to the Swap() method, the values could be swapped without the use of pass by reference

Tips on Using Pointers

- Avoid the use of pointers if possible
 - Pointers are rarely required in C# code
- The use of pointers can be justified in the following cases:
 - Legacy data structures
 - Legacy library functions
 - Performance critical code

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brush, and C. C. W. Halls. All Rights Reserved

Another Example of Pointer Usage

```
using System;

class UnsafeTest           // This is an example of an ugly use of pointers
{
    unsafe static void SquareIt( int *ptr )
    {
        *ptr *= *ptr; // This line multiplies the value pointed to
                       // by ptr by the value pointed to by ptr
    }

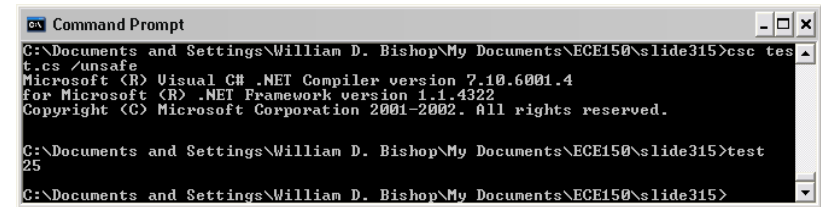
    unsafe static void Main()
    {
        int i = 5;

        SquareIt( &i );
        Console.WriteLine( i );
    }
}
```

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brash, and C. C. W. Halls. All Rights Reserved

Output of the Previous Example

- Here is the output of the previous example:



```
Command Prompt
C:\Documents and Settings\William D. Bishop\My Documents\ECE150\slide315>csc test.cs /unsafe
Microsoft (R) Visual C# .NET Compiler version 7.10.6001.4
for Microsoft (R) .NET Framework version 1.1.4322
Copyright (C) Microsoft Corporation 2001-2002. All rights reserved.

C:\Documents and Settings\William D. Bishop\My Documents\ECE150\slide315>test
25
C:\Documents and Settings\William D. Bishop\My Documents\ECE150\slide315>
```

- The program computes and displays the square of 5
- A pointer is used to manipulate the memory location associated with the variable

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brash, and C. C. W. Halls. All Rights Reserved

The Power of Pointers

- Pointers give developers access to memory directly
- Pointers are very powerful resources
 - Given a pointer to something on the program stack, it is possible to search for something nearby on the program stack
 - Hackers often use pointer arithmetic to exploit unencrypted data on the program stack
- Consider the following (somewhat scary) example...

Copyright © 2010 by W. D. Bishop, G. H. Freeman, R. E. Seivora, D. J. Brash, and C. C. W. Halls. All Rights Reserved