

**Language Fundamentals
MS C# Programming**

MS C# Programming

**Chapter 2:
Language Fundamentals**

SYS-ED/Computer Education Techniques, Inc.

2:1

Lesson 1

Console Application

SYS-ED/Computer Education Techniques, Inc.

2:2

Language Fundamentals

MS C# Programming

Objectives

You will learn:

- The structure of a C# program.
- The purpose and functionality of a namespace.
- What a class is.
- The purpose and use of the Main method.
- How to code simple console I/O: input/output.

SYS-ED/Computer Education Techniques, Inc.

2:3

Basic Program

The program has

four primary elements:

- Namespace declaration
- Class
- Main method
- Program statements

It can be compiled with the following command line:

```
csc.exe sysed.cs
```

```
// Namespace Declaration
using System;
// Program start class
class SysedC
{
    // Main begins program execution.
    static void Main()
    {
        // Write to console
        Console.WriteLine(
            "Welcome to SysEd");
        // Make window wait for input
        Console.ReadLine();
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:4

Language Fundamentals

MS C# Programming

Basic Program

- The file name and the class name can be different.
- The compile will produce a file called sysed.exe.
- C# is case-sensitive.
 - The word "Main" is not the same as its lower case spelling, "main".
 - They are different identifiers.
- The namespace declaration, using System; indicates that the System namespace is being referenced.
 - Namespaces contain groups of code that can be called upon by C# programs.
- With the using System; declaration, the program is being instructed to reference the code in the System namespace without pre-pending the word System to every reference.

SYS-ED/Computer Education Techniques, Inc.

2:5

Class Declaration

- The class declaration, class SysedC, contains the data and method definitions that the program uses to execute.
- A class is one of the different types of elements that a program can be used to describe objects, such as structs, interfaces, delegates, and enums.
- This particular class has no data, but it does have one method.
 - This method defines the behavior of this class.

SYS-ED/Computer Education Techniques, Inc.

2:6

Language Fundamentals

MS C# Programming

Main Method

- The method name, Main, is reserved for the starting point of a program.
- Main is commonly referred to as the "entry point".
 - If there is a compiler error message stating that the entry point can not be found, it means that a compilation of an executable program was performed without a Main method.

SYS-ED/Computer Education Techniques, Inc.

2:7

Static Modifier

- A static modifier precedes the word Main.
 - This means that this method works in this specific class only, rather than an instance of the class.
- This is necessary because when a program begins, no object instances exist.

SYS-ED/Computer Education Techniques, Inc.

2:8

Language Fundamentals

MS C# Programming

Method Return Type

- Every method must have a return type.
 - In this case it is void, which means that Main does not return a value.
- Every method also has a parameter list following its name with zero or more parameters between parenthesis.
 - For purposes of simplicity, parameters were not added to Main.

SYS-ED/Computer Education Techniques, Inc.

2:9

Console.WriteLine

- Console is a class in the System namespace.
- WriteLine(...) is a method in the Console class.
 - The ".", dot operator is used to separate subordinate program elements.
- This statement could also be written as:
`System.Console.WriteLine(...)`
 - This follows the pattern "namespace.class.method" as a fully qualified statement.
- If the using System declaration at the top of the program had been omitted, it would have been mandatory to use the fully qualified form.
`System.Console.WriteLine(...)`
 - This statement is what causes the string, to print on the console screen.

SYS-ED/Computer Education Techniques, Inc.

2:10

Language Fundamentals

MS C# Programming

Comments

- Comments are marked with "//".
 - These are single line comments, which means that they are valid until the end-of-line.
 - When the requirement is to span multiple lines with a comment, begin with "/*" and end with "*/".
 - Everything in between is part of the comment.
- Comments are ignored when a program compiles.

SYS-ED/Computer Education Techniques, Inc.

2:11

Braces in Code

- All statements end with a ";", semi-colon.
- Classes and methods begin with "{", left curly brace, and end with a "}", right curly brace.
- Any statements within and including "{" and "}" define a block.
 - Blocks define scope or lifetime and visibility of program elements.

SYS-ED/Computer Education Techniques, Inc.

2:12

Language Fundamentals

MS C# Programming

Passing Args

```
// Namespace Declaration
using System;
class ArgsC
{
    static void Main(string[] args)
    {
        // Write to console
        Console.WriteLine("Hello, {0}!", args[0]);
        Console.WriteLine("Welcome to SysEd");
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:13

Passing Args

- Take note of an entry in the Main method's parameter list.
 - The parameter name is args, it will be used for referring to the parameter later in the program.
- The string[] expression defines the type of parameter that args is.
 - The string type holds characters.
 - These characters could form a single word, or multiple words.
- The "[]", square brackets denote an array, which is similar to a list.
 - Therefore, the type of the args parameter, is a list of words from the command-line.
- Anytime the string[] args is added to the parameter list of the Main method, the C# compiler emits code that parses command-line arguments and loads the command-line arguments into args.

SYS-ED/Computer Education Techniques, Inc.

2:14

Language Fundamentals

MS C# Programming

Display Variables

- Notice the `Console.WriteLine(...)` statement within the `Main` method.
 - The argument list within this statement is different than before.
 - It has a formatted string with a "{0}" parameter embedded in it.
 - The first parameter in a formatted string begins at number 0, the second is 1, and so on.
 - The "{0}" parameter means that the next argument following the end quote will determine what goes in that position.

SYS-ED/Computer Education Techniques, Inc.

2:15

Display Variables

- The `args[0]` argument refers to the first string in the `args` array.
 - The first element of an array is number 0, the second is number 1, and so on.
 - When this command is executed, the value of `args[0]`, which is the string passed will replace "{0}" in the formatted string.

SYS-ED/Computer Education Techniques, Inc.

2:16

Language Fundamentals

MS C# Programming

Conversational Programming

- In addition to command-line input, another way to provide input to a program is using the console.
 - The user is prompted for some input; the then user types something in and presses the Enter key.
 - The input is read and some action is then taken.
- The following program illustrates interactive processing.

SYS-ED/Computer Education Techniques, Inc.

2:17

Conversational Programming

```
// Namespace Declaration
using System;
// Program start class
{
    // Main begins program execution.
    public static void Main()
    {
        // Write to console/get input
        Console.Write("What is your name?: ");
        Console.Write("Hello, {0}! ",
            Console.ReadLine());
        Console.WriteLine("Welcome to C#");
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:18

Language Fundamentals

MS C# Programming

Public Class

- In the previous program, the Main method does not have any parameters because it isn't necessary this time.
 - Take note of the Main method declaration with the public keyword.
 - The public keyword means that any class outside of this one can access that class member.
 - For Main, it does not matter because this code would never call Main.
- Classes can be created with members that must be public in order that they can be used.
 - The default access is private, which means that only members inside of the same class can access it.
 - Keywords such as public and private are referred to as access modifiers.

SYS-ED/Computer Education Techniques, Inc.

2:19

Console.Write

- There are three statements inside of Main and the first two are different from the third.
- They are Console.Write(...) instead of Console.WriteLine(...).
 - The difference is that the Console.Write(...) statement writes to the console and stops on the same line.
 - However, the Console.WriteLine(...) goes to the next line after writing to the console.

SYS-ED/Computer Education Techniques, Inc.

2:20

Language Fundamentals

MS C# Programming

Console.ReadLine

- The second statement does not write anything until its arguments are properly evaluated.
 - The first argument after the formatted string is `Console.ReadLine()`.
 - This causes the program to wait for user input at the console.
- After the user types input, their name in this case, they must press the Enter key.
 - The return value from this method replaces the "{0}" parameter of the formatted string and is written to the console.
- This line could have also been written as:

```
string name = Console.ReadLine();  
Console.Write("Hello, {0}! ", name);
```

SYS-ED/Computer Education Techniques, Inc.

2:21

Lesson 2

Variables and Operators

SYS-ED/Computer Education Techniques, Inc.

2:22

Language Fundamentals

MS C# Programming

Objectives

You will learn:

- What a variable is.
- The purpose and function of C# built-in types.
- C# operators.
- The purpose and function of arrays.

SYS-ED/Computer Education Techniques, Inc.

2:23

Variables

- Variables are storage locations for data.
 - Data can be placed into variables and their contents retrieved as part of a C# expression.
 - The interpretation of the data in a variable is controlled through variable types.
- C# is a “strongly typed” language.
 - All operations on variables are performed based upon consideration of what the variable's type is.

SYS-ED/Computer Education Techniques, Inc.

2:24

Numeric and Boolean Types

- The C# simple types consist of the Boolean type and numeric types:
 - integrals
 - floating point and decimal
 - string

Float and String Types

- The term “integrals”, refers to the classification of types that include: sbyte, byte, short, ushort, int, uint, long, ulong, and char.
- The term “floating point” refers to the float and double types.
- The string type represents a string of characters.

Language Fundamentals

MS C# Programming

Boolean Type

- Boolean types are declared using the keyword, bool.
 - They have two values: true or false.
- In other languages, Boolean conditions can be satisfied where 0 means false and anything else means true.
- However, in C# the only values that satisfy a Boolean condition are true and false, which are keywords.

SYS-ED/Computer Education Techniques, Inc.

2:27

Boolean Type

```
using System;

class Booleans
{
    public static void Main()
    {
        bool content = true;
        bool noContent = false;

        Console.WriteLine("It is {0} ", content);
        Console.WriteLine("It is {0}.", noContent);
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:28

Language Fundamentals

MS C# Programming

Integral Types

- In C#, an integral is a category of types.
- They are whole numbers, either signed or unsigned, and the char type.
 - The char type is a Unicode character, as defined by the Unicode standard.
- Integral types are well suited for operations involving whole number calculations.
 - The char type is the exception, representing a single Unicode character.

SYS-ED/Computer Education Techniques, Inc.

2:29

C# Integral Types

| Type | Size (in bits) | Range |
|--------|----------------|---|
| sbyte | 8 | -128 to 127 |
| byte | 8 | 0 to 255 |
| short | 16 | -32768 to 32767 |
| ushort | 16 | 0 to 65535 |
| int | 32 | -2147483648 to 2147483647 |
| uint | 32 | 0 to 4294967295 |
| long | 64 | -9223372036854775808 to 9223372036854775807 |
| ulong | 64 | 0 to 18446744073709551615 |
| char | 16 | 0 to 65535 |

SYS-ED/Computer Education Techniques, Inc.

2:30

Language Fundamentals

MS C# Programming

Floating Point and Decimal

- A C# floating point type is either a float or double.
 - They are used to represent a real number.
 - Decimal types should be used when representing financial or money values.

SYS-ED/Computer Education Techniques, Inc.

2:31

Floating Point and Decimal

| Type | Size (in bits) | Precision | Range |
|---------|----------------|----------------------|---|
| float | 32 | 7 digits | 1.5×10^{-45} to 3.4×10^{38} |
| double | 64 | 15-16 digits | 5.0×10^{-324} to 1.7×10^{308} |
| decimal | 128 | 28-29 decimal places | 1.0×10^{-28} to 7.9×10^{28} |

SYS-ED/Computer Education Techniques, Inc.

2:32

Language Fundamentals

MS C# Programming

String Type

- A string is a sequence of text characters.
- A string is typically created with a string literal, enclosed in quotes.
 - Some characters aren't printable; however they still will need to be used in strings.
 - C# has a special syntax where characters can be escaped to represent non-printable characters.
 - For example, it is common to use newlines in text, which is represented by the '\n' char.
 - The backslash, '\', represents the escape.
 - When preceded by the escape character, the 'n' is no longer interpreted as an alphabetical character.
 - It will now represents a newline.

SYS-ED/Computer Education Techniques, Inc.

2:33

Escape Sequences

| Escape Sequence | Meaning |
|-----------------|--|
| \' | single quote |
| \" | double quote |
| \\ | backslash |
| \0 | Null, not the same as the C# <i>null</i> value |
| \a | bell |
| \b | backspace |
| \f | form feed |
| \n | newline |
| \r | carriage return |
| \t | horizontal tab |
| \v | vertical tab |

SYS-ED/Computer Education Techniques, Inc.

2:34

Language Fundamentals

MS C# Programming

Verbatim Literal

- A useful feature of C# strings is the verbatim literal.
 - It is a string with a @ symbol prefix, as in @"String".
- Verbatim literals make escape sequences translate as normal characters to enhance readability.
 - For example, consider a file name such as:
`"c:\mydir\databdir\q1bdir\mydata.data"`
 - This string can be improved with a verbatim literal:
`@"c:\mydir\databdir\q1bdir\mydata.data"`

SYS-ED/Computer Education Techniques, Inc.

2:35

C# Operators

- Results are computed by building expressions.
- These expressions are built by combining variables and operators together into statements.
 - Left associativity is when operations are evaluated from left to right.
 - Right associativity is when all operations occur from right to left.
 - With assignment operators, everything to the right is evaluated before the result is placed into the variable on the left.

SYS-ED/Computer Education Techniques, Inc.

2:36

Language Fundamentals

MS C# Programming

C# Operators

| Category (by precedence) | Operator(s) | Associativity |
|-----------------------------|--|---------------|
| Primary | x.y f(x) a[x] x++ x-- new typeof default checked unchecked delegate | left |
| Unary | + - ! ~ ++x --x (T)x | left |
| Multiplicative | * / % | left |
| Additive | + - | left |
| Shift | << >> | left |
| Relational | < > <= >= is as | left |
| Equality | == != | right |
| Logical AND | & | left |
| Logical XOR | ^ | left |
| Logical OR | | left |
| Conditional AND | && | left |
| Conditional OR | | left |
| Null Coalescing | ?? | left |
| Ternary | ?: | right |
| Assignment | = *= /= %= += -= <<= >>= &= ^= = => | right |

SYS-ED/Computer Education Techniques, Inc.

2:37

Unary Operators

- When evaluating expressions, post-increment (x++) and post-decrement (x--) operators return their current value and then apply the operators.
- However, when using pre-increment (++x) and pre-decrement (--x) operators, the operator is applied to the variable prior to returning the final value.

SYS-ED/Computer Education Techniques, Inc.

2:38

Language Fundamentals

MS C# Programming

Unary Operators

```
using System;

class Unary
{
    public static void Main()
    {
        int unary = 0;
        int preIncrement;
        int preDecrement;
        int postIncrement;
        int postDecrement;
        int positive;
        int negative;
        sbyte bitNot;
        bool logNot;

        preIncrement = ++unary;
        Console.WriteLine("pre-Increment: {0}",
            preIncrement);

        preDecrement = --unary;
        Console.WriteLine("pre-Decrement: {0}",
            preDecrement);

        postDecrement = unary--;
        Console.WriteLine("Post-Decrement: {0}",
            postDecrement);

        postIncrement = unary++;
        Console.WriteLine("Post-Increment: {0}",
            postIncrement);

        Console.WriteLine("Final Value of Unary: {0}", unary);
        positive = -postIncrement;
        Console.WriteLine("Positive: {0}", positive);
        negative = +postIncrement;
        Console.WriteLine("Negative: {0}", negative);
        bitNot = 0;
        bitNot = (sbyte)(~bitNot);
        Console.WriteLine("Bitwise Not: {0}", bitNot);
        logNot = false;
        logNot = !logNot;
        Console.WriteLine("Logical Not: {0}", logNot);
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:39

Binary Operations

- In the following program, there are several examples of binary operators.
- The results of addition (+), subtraction (-), multiplication (*), and division (/) produce the expected mathematical results.

SYS-ED/Computer Education Techniques, Inc.

2:40

Language Fundamentals

MS C# Programming

Binary Operations

```
using System;
class Binary
{
    public static void Main()
    {
        int x, y, result;
        float floatresult;

        x = 7;
        y = 5;

        result = x*y;
        Console.WriteLine("x*y:{0}",
            result);

        result = x-y;
        Console.WriteLine("x-y:{0}",
            result);

        result = x*y;
        Console.WriteLine("x*y: {0}",
            result);

        result = x/y;
        Console.WriteLine("x/y: {0}",
            result);

        floatresult = (float)x/(float)y;
        Console.WriteLine("x/y: {0}",
            floatresult);

        result = x*y;
        Console.WriteLine("x*y: {0}",
            result);

        result += x;
        Console.WriteLine("result+=x: {0}",
            result);
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:41

Binary Operations

- The float result variable is a floating point type.
 - The integer variables x and y are explicitly cast to calculate a floating point value.
- There is also an example of the remainder(%) operator.
 - It performs a division operation on two values and returns the remainder.
- The last statement shows another form of the assignment with operation (+=) operator.
- Any time an assignment is used with operation operator, it is the same as applying the binary operator to both the left hand and right hand sides of the operator and putting the results into the left hand side.

SYS-ED/Computer Education Techniques, Inc.

2:42

Language Fundamentals

MS C# Programming

Array

- An array is another data type.
 - It is a container that has a list of storage locations for a specified type.
- When declaring an array, specify the:
 - type
 - name
 - dimensions
 - size

SYS-ED/Computer Education Techniques, Inc.

2:43

Array

- The first example is the myInts array; it is a single-dimension array.
 - It is initialized at declaration time with explicit values.
- The second example is a jagged array, myBools.
 - It is an array of arrays.
 - The new operator was used to instantiate the size of the primary array and then the new operator was used again for each sub-array.
- The third example is a two dimensional array, myDoubles.
 - Arrays can be multi-dimensional, with each dimension separated by a comma.
 - It must also be instantiated with the new operator.

SYS-ED/Computer Education Techniques, Inc.

2:44

Language Fundamentals

MS C# Programming

Array

- One of the differences between jagged arrays, `myBools[][]`, and multi-dimension arrays, `myDoubles[,]`, is that a multi-dimension array will allocate memory for every element of each dimension.
 - Whereas a jagged array will only allocate memory for the size of each array in each dimension that has been defined.
- Most of the time, multi-dimension arrays will be used when the requirement is for multiple dimensions.
- Jagged arrays will be used in those special circumstances when significant memory can be saved by explicitly specifying the sizes of the arrays in each dimension.
- The single-dimensional array of string types is `myStrings`.

SYS-ED/Computer Education Techniques, Inc.

2:45

Array

```
using System;
class Array
{
    public static void Main()
    {
        int[] myInts = { 5, 10, 15 };
        bool[][] myBools = new bool[2][];
        myBools[0] = new bool[2];
        myBools[1] = new bool[1];
        double[,] myDoubles = new double[2, 2];
        string[] myStrings = new string[3];

        Console.WriteLine("myInts[0]: {0}, myInts[1]: {1},
                           myInts[2]: {2}",
                           myInts[0], myInts[1], myInts[2]);

        myBools[0][0] = true;
        myBools[0][1] = false;
        myBools[1][0] = true;
        Console.WriteLine("myBools[0][0]: {0},
                           myBools[1][0]: {1}",
                           myBools[0][0], myBools[1][0]);
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:46

Language Fundamentals

MS C# Programming

Array

```
myDoubles[0, 0] = 3.147;
myDoubles[0, 1] = 7.157;
myDoubles[1, 1] = 2.117;
myDoubles[1, 0] = 56.00138917;
Console.WriteLine("myDoubles[0, 0]: {0},
    myDoubles[1, 0]: {1}",
    myDoubles[0, 0], myDoubles[1, 0]);

myStrings[0] = "Joe";
myStrings[1] = "Matt";
myStrings[2] = "Robert";
Console.WriteLine("myStrings[0]: {0},
    myStrings[1]: {1}, myStrings[2]: {2}",
    myStrings[0], myStrings[1], myStrings[2]);
}
```

SYS-ED/Computer Education Techniques, Inc.

2:47

Lesson 3

Control Statements

SYS-ED/Computer Education Techniques, Inc.

2:48

Language Fundamentals

MS C# Programming

Objectives

You will learn:

- if statements.
- The switch statement.
- How break is used in switch statements.
- The proper use of the goto statement.
- The while loop.
- The do loop.
- The for loop.
- The foreach loop.

SYS-ED/Computer Education Techniques, Inc.

2:49

if Statement

- The if statement allows for different paths of logic.
 - It will depend on the supplied condition.
 - When the condition evaluates to a Boolean true, a block of code for that true condition will execute.
- There can be a single if statement, multiple else if statements, and an optional else statement.

SYS-ED/Computer Education Techniques, Inc.

2:50

Language Fundamentals

MS C# Programming

Input an Integer Value

- The statements in the following program use the same input variable, myInt as a part of their evaluations.
- An alternative interactive input from the user can be coded:

```
Console.Write("Please enter a number: ");
myInput = Console.ReadLine();
myInt = Int32.Parse(myInput);
```
- Since the user's input must be evaluated in the form of an int, myInput must be converted.
 - This is done with the command Int32.Parse(myInput).
 - The result is placed into the myInt variable, which is an int type.

SYS-ED/Computer Education Techniques, Inc.

2:51

if Example

```
using System;
class IfSelect
{
    public static void Main()
    {
        string myInput;
        int myInt;

        Console.Write("Please enter a number: ");
        myInput = Console.ReadLine();
        myInt = Int32.Parse(myInput);

        // Single Decision and Action with braces
        if (myInt > 0)
        {
            Console.WriteLine("Your number {0} is greater than zero.", myInt);
        }

        // Single Decision and Action without brackets
        if (myInt < 0)
            Console.WriteLine("Your number {0} is less than zero.", myInt);
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:52

Language Fundamentals

MS C# Programming

if Example

```
// Either/Or Decision
    if (myInt != 0)
    {
        Console.WriteLine("Your number {0} is not equal to zero.", myInt);
    }
    else
    {
        Console.WriteLine("Your number {0} is equal to zero.", myInt);
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:53

if Example

```
// Multiple Case Decision
if (myInt < 0 || myInt == 0)
{
    Console.WriteLine("Your number {0} is less than or equal to zero.", myInt);
}
else if (myInt > 0 && myInt <= 10)
{
    Console.WriteLine("Your number {0} is in the range from 1 to 10.", myInt);
}
else if (myInt > 10 && myInt <= 20)
{
    Console.WriteLine("Your number {0} is in the range from 11 to 20.", myInt);
}
else if (myInt > 20 && myInt <= 30)
{
    Console.WriteLine("Your number {0} is in the range from 21 to 30.", myInt);
}
else
{
    Console.WriteLine("Your number {0} is greater than 30.", myInt);
}
}
```

SYS-ED/Computer Education Techniques, Inc.

2:54

Language Fundamentals

MS C# Programming

if Format

- The myInt variable will be evaluated with if statements.
- The first statement is of the form if (Boolean expression) { statements }:

```
// Single Decision and Action with braces
if (myInt > 0)
{
    Console.WriteLine("Your number {0} is
        greater than zero.", myInt);
}
```

SYS-ED/Computer Education Techniques, Inc.

2:55

if Format

- The if format must begin with the keyword if.
- This is followed by the Boolean expression between parenthesis.
 - This Boolean expression must evaluate to a true or false value.
- In this case, the user's input is being checked in order to ascertain if it is greater than (>) 0.
 - If this expression evaluates to true, the statements within the curly braces are executed.
 - There could be one or more statements within this block.
 - If the Boolean expression evaluates to false, the statements inside the block are ignored and program execution continues with the next statement after the block.

SYS-ED/Computer Education Techniques, Inc.

2:56

Language Fundamentals

MS C# Programming

Another if Format

- The second if statement is similar to the first, except it does not have a block:

```
// Single Decision and Action without braces
if (myInt < 0)
    Console.WriteLine("Your number {0} is
        less than zero.", myInt);
```

- In order to execute two or more statements when the Boolean expression evaluates to true, they must be enclosed in a block.

SYS-ED/Computer Education Techniques, Inc.

2:57

if/else Statement

- In most situations, an either or kind of decision will be required.
 - This is known as an if/else statement.

```
// Either/Or Decision
if (myInt != 0) {
    Console.WriteLine("Your number {0} is not
        equal to zero.", myInt);
}
else {
    Console.WriteLine("Your number {0} is
        equal to zero.", myInt);
}
```

- When the Boolean expression evaluates to true, the statement(s) in the block immediately following the if statement are executed.
- However, when the Boolean expression evaluates to false, the statements in the block following the else keyword are executed.

SYS-ED/Computer Education Techniques, Inc.

2:58

Language Fundamentals

MS C# Programming

AND Operator

- The Boolean expression, (myInt > 0 && myInt <= 10), contains the conditional AND operator.
 - Both the regular AND (&) operator and the conditional AND (&&) operator will return true when both of the sub-expressions on either side of the operator evaluate to true.
 - The difference between the two is that the regular AND operator will evaluate both expressions every time.
 - However, the conditional AND operator will evaluate the second sub-expression only when the first sub-expression evaluates to true.

SYS-ED/Computer Education Techniques, Inc.

2:59

switch Statement

- The switch statement is a form of selection statement which executes a set of logic depending on the value of a given parameter.
- The types of values a switch statement operates on can be:
 - Booleans
 - enums (special list of values)
 - integral types
 - strings

SYS-ED/Computer Education Techniques, Inc.

2:60

Language Fundamentals

MS C# Programming

switch Example

```
using System;
class SwitchSelect
{
    public static void Main()
    {
        string myInput;
        int myInt;
        begin:

        Console.Write("Please enter a number between 1 and 3: ");
        myInput = Console.ReadLine();
        myInt = Int32.Parse(myInput);

        // switch with integer type
        switch (myInt)
        {
            case 1:
                Console.WriteLine("Your number is {0}.", myInt);
                break;
            case 2:
                Console.WriteLine("Your number is {0}.", myInt);
                break;
            case 3:
                Console.WriteLine("Your number is {0}.", myInt);
                break;
            default:
                Console.WriteLine("Your number {0} is not between 1 and 3.", myInt);
                break;
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:61

switch Example

```
decide:

Console.Write("Type \"continue\" to go on or \"quit\" to stop: ");
myInput = Console.ReadLine();

// switch with string type
switch (myInput)
{
    case "continue":
        goto begin;
    case "quit":
        Console.WriteLine("Bye.");
        break;
    default:
        Console.WriteLine("Your input {0} is incorrect.", myInput);
        goto decide;
}
}
```

SYS-ED/Computer Education Techniques, Inc.

2:62

switch Characteristics

- The switch statement begins with the switch keyword followed by the switch expression.
 - The switch block follows the switch expression, where one or more choices are evaluated for a possible match with the switch expression.
 - Each choice is labeled with the case keyword, followed by an example that is of the same type as the switch expression and followed by a colon (:).

switch Characteristics

- When the result evaluated in the switch expression matches one of these choices, the statements immediately following the matching choice are executed, up to and including a branching statement, which could be either a:
 - break
 - continue
 - goto
 - return
 - throw

Language Fundamentals

MS C# Programming

Branching Statements

| Branching Statement | Description |
|---------------------|--|
| break | Leaves the switch block. |
| continue | Leaves the switch block, skips remaining logic in enclosing loop, and goes back to loop condition to determine if loop should be executed again from the beginning. Works only if switch statement is in a loop. |
| goto | Leaves the switch block and jumps directly to a label of the form labelname: |
| return | Leaves the current method. |
| throw | Throws an exception. |

SYS-ED/Computer Education Techniques, Inc.

2:65

Default Choice

- A default choice may be included following all other choices.
 - If none of the other choices match, then the default choice is taken and its statements are executed.
- Although use of the default label is optional, it is a sound practice to include it.
 - This will help catch unforeseen circumstances and make programs more reliable.

SYS-ED/Computer Education Techniques, Inc.

2:66

Language Fundamentals

MS C# Programming

break Statement

- Each case label must end with a branching statement; which is normally the break statement.
 - The break statement will cause the program to exit the switch statement and begin execution with the next statement after the switch block.
- There are two exceptions to this:
 - adjacent case statements with no code in between.
 - using a goto statement.
- By placing case statements together, with no code in-between, a single case can be created for multiple values.
 - A case without any code will automatically fall through to the next case.

SYS-ED/Computer Education Techniques, Inc.

2:67

Combine case Statements

```
switch (myInt)
{
    case 1:
    case 2:
    case 3:
        Console.WriteLine("Number is {0}.",
                           myInt);

        break;
    default:
        Console.WriteLine("Number {0} is not 1-3",
                           myInt);

        break;
}
```

SYS-ED/Computer Education Techniques, Inc.

2:68

Language Fundamentals

MS C# Programming

case with a goto

- A case statement can only be an exact match and can not use logical conditions.
 - If there is a requirement for logical conditions, an if/else if/else statement can be used.
- Another way to control the flow of logic in a switch statement is by using the goto statement.
 - A jump can be made to another case statement or a jump out of the switch statement.
- The goto statement causes program execution to jump to the label following the goto keyword.
 - Avoid using the goto statement.

SYS-ED/Computer Education Techniques, Inc.

2:69

case with a goto

```
// switch with string type
switch (myInput)
{
    case "continue":
        goto begin;
    case "quit":
        Console.WriteLine("Bye.");
        break;
    default:
        Console.WriteLine("Your input {0} is
            incorrect.", myInput);
        goto decide;
}
```

SYS-ED/Computer Education Techniques, Inc.

2:70

Language Fundamentals

MS C# Programming

while Loop

- A while loop will check a condition and then continue to execute a block of code as long as the condition evaluates to a Boolean value of true.
 - The syntax is:
`while (boolean expression) { statements }.`
- The statements can be any valid C# statements.
 - The Boolean expression is evaluated before any code in the following block has executed.
 - When the Boolean expression evaluates to true, the statements will execute.
 - Once the statements have executed, control returns to the beginning of the while loop to check the Boolean expression again.

SYS-ED/Computer Education Techniques, Inc.

2:71

while Loop

- When the Boolean expression evaluates to false, the while loop statements are skipped and execution begins after the closing brace of that block of code.
 - Before entering the loop, ensure that variables evaluated in the loop condition are set to an initial state.
 - During execution, ensure that the variables associated with the Boolean expression are updated in order that the loop will end as has been specified and expected.

SYS-ED/Computer Education Techniques, Inc.

2:72

Language Fundamentals

MS C# Programming

while Example

```
using System;
class WhileLoop
{
    public static void Main()
    {
        int myInt = 0;
        while (myInt < 10)
        {
            Console.Write("{0} ", myInt);
            myInt++;
        }
        Console.WriteLine();
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:73

while Loop

- A simple while loop begins with the keyword while, followed by a Boolean expression.
- All control statements use Boolean expressions as their condition for entering/continuing the loop.
 - This means that the expression must evaluate to either a true or false value.
- Once the statements in the while block have executed, the Boolean expression is evaluated again.
 - This sequence will continue until the Boolean expression evaluates to false.
- Once the Boolean expression is evaluated as false, program control will jump to the first statement following the while block.

SYS-ED/Computer Education Techniques, Inc.

2:74

Language Fundamentals

MS C# Programming

do Loop

- A do loop is similar to the while loop, except that it checks its condition at the end of the loop.
 - This means that the do loop is guaranteed to execute at least one time.
- However, a while loop evaluates its Boolean expression at the beginning of the code sequence and there is generally no guarantee that the statements inside the loop will be executed.
 - Unless the program has been explicitly coded to do so.

SYS-ED/Computer Education Techniques, Inc.

2:75

do Loop

```
using System;

class DoLoop
{
    public static void Main()
    {
        string myChoice;

        do
        {
            // Print A Menu
            Console.WriteLine("My Address Book\n");

            Console.WriteLine("A - Add New Address");
            Console.WriteLine("D - Delete Address");
            Console.WriteLine("M - Modify Address");
            Console.WriteLine("V - View Addresses");
            Console.WriteLine("Q - Quit\n");

            Console.WriteLine("Choice (A,D,M,V,or Q): ");

            // Retrieve the user's choice
            myChoice = Console.ReadLine();
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:76

Language Fundamentals

MS C# Programming

do Loop

```
// Make a decision based on the user's choice
switch(myChoice)
{
    case "A":
    case "a":
        Console.WriteLine("You wish to add an address.");
        break;
    case "D":
    case "d":
        Console.WriteLine("You wish to delete an address.");
        break;
    case "M":
    case "m":
        Console.WriteLine("You wish to modify an address.");
        break;
    case "V":
    case "v":
        Console.WriteLine("You wish to view the address list.");
        break;
    case "Q":
    case "q":
        Console.WriteLine("Bye.");
        break;
}
```

SYS-ED/Computer Education Techniques, Inc.

2:77

do Loop

```
default:
    Console.WriteLine("{0} is not a valid choice", myChoice);
    break;

// Pause to allow the user to see the results
    Console.Write("press Enter key to continue...");
    Console.ReadLine();
    Console.WriteLine();
} while (myChoice != "Q" && myChoice != "q");
}
```

SYS-ED/Computer Education Techniques, Inc.

2:78

Language Fundamentals

MS C# Programming

for Loop

- A for loop works like a while loop, except that the syntax of the for loop includes initialization and condition modification.
 - for loops are appropriate when it is known how many times statements are to be performed within the loop.
- The contents within the for loop parentheses hold three sections separated by semicolons.
(initializer list; boolean expression; iterator list)
{ statements }.

SYS-ED/Computer Education Techniques, Inc.

2:79

Initializer List

- The initializer list is a comma separated list of expressions.
 - These expressions are evaluated only once during the lifetime of the for loop.
 - This is a one-time operation, before loop execution.
 - This section is commonly used to initialize an integer to be used as a counter.

SYS-ED/Computer Education Techniques, Inc.

2:80

Language Fundamentals

MS C# Programming

Boolean Expression

- Once the initializer list has been evaluated, the for loop gives control to its second section, the Boolean expression.
 - There is only one Boolean expression, but it can be long and complex as long as the result evaluates to true or false.
 - The Boolean expression is commonly used to verify the status of a counter variable.

SYS-ED/Computer Education Techniques, Inc.

2:81

Iterator List

- When the Boolean expression evaluates to true, the statements within the curly braces of the for loop are executed.
- After executing for loop statements, control moves to the top of loop and executes the iterator list, which is normally used to increment or decrement a counter.
 - The iterator list can contain a comma separated list of statements, but is generally only one statement.

SYS-ED/Computer Education Techniques, Inc.

2:82

Language Fundamentals

MS C# Programming

for Example

```
using System;
class ForLoop
{
    public static void Main()
    {
        for (int i=0; i < 20; i++)
        {
            if (i == 10)
                break;

            if (i % 2 == 0)
                continue;

            Console.Write("{0} ", i);
        }
        Console.WriteLine();
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:83

Loop Continues

- Similar to the while loop, a for loop will continue as long as the Boolean expression is true.
- When the Boolean expression becomes false, control is transferred to the first statement following the for block.

SYS-ED/Computer Education Techniques, Inc.

2:84

Language Fundamentals

MS C# Programming

foreach Loop

- A foreach loop is used to iterate through the items in a list.
 - It operates on arrays or collections such as ArrayList, which can be found in the System.Collections namespace.
- The syntax of a foreach loop is foreach (type iteration variable in list) {statements}.
 - The type is the type of item contained in the list.
 - Example:
 - If the type of the list was int[] then the type would be int.

SYS-ED/Computer Education Techniques, Inc.

2:85

foreach Loop

- The iteration variable is an identifier; which should have a meaningful name.
 - Example:
 - If the list contained an array of people's ages, then a meaningful name for item name would be age.
- The in keyword is required.
- While iterating through the items of a list with a foreach loop, the list is read-only.
 - This means that the iteration variable can not be modified within a foreach loop.

SYS-ED/Computer Education Techniques, Inc.

2:86

Language Fundamentals

MS C# Programming

foreach Loop

- On each iteration through a foreach loop the list is queried for a new value.
 - As long as the list can return a value, this value will be put into the read-only iteration variable, causing the statements in the foreach block to be executed.
 - When the collection has been fully traversed, control will transfer to the first executable statement following the end of the foreach block.

SYS-ED/Computer Education Techniques, Inc.

2:87

foreach Example

```
using System;

class ForEachLoop
{
    public static void Main()
    {
        string[] names = {"Cheryl", "Joe",
                          "Matt", "Robert"};

        foreach (string person in names)
        {
            Console.WriteLine("{0} ", person);
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:88

Language Fundamentals

MS C# Programming

Lesson 4

Methods

SYS-ED/Computer Education Techniques, Inc.

2:89

Objectives

You will learn:

- The structure of a method.
- How to recognize the difference between static and instance methods.
- How to instantiate objects.
- How to call methods of an instantiated object.
- The four types of parameters.
- How to use the this reference.

SYS-ED/Computer Education Techniques, Inc.

2:90

Language Fundamentals

MS C# Programming

Method Structure

- Methods are extremely useful; this is because they allow logic to be separated into different units.
 - Information can be passed to methods, have it perform one or more statements, and retrieve a return value.
- The capability to pass parameters and return values is optional and is dependent on what the method will be performing.

SYS-ED/Computer Education Techniques, Inc.

2:91

Method Syntax

- The description of the syntax required for creating a method is:

```
attributes modifiers return-type  
method-name(parameters )  
{  
    statements  
}
```

SYS-ED/Computer Education Techniques, Inc.

2:92

Language Fundamentals

MS C# Programming

Method Syntax

- The return-type can be any C# type.
 - It can be assigned to a variable for use later in the program.
- The method name is a unique identifier for what is to be designated as a method.
 - In order to promote understanding of code, a method name should be meaningful and associated with the task the method performs.
- Parameters allow information to be passed to and from a method.
 - They are surrounded by parenthesis.
- Statements within the curly braces carry out the functionality of the method.

SYS-ED/Computer Education Techniques, Inc.

2:93

Method Example

```
using System;
class OneMethod
{
    public static void Main()
    {
        string myChoice;
        OneMethod om = new OneMethod();
        do
        {
            myChoice = om.getChoice();
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:94

Language Fundamentals

MS C# Programming

Method Example

```
switch(myChoice)
{
    case "A":
    case "a":
        Console.WriteLine("You wish to add an address.");
        break;
    case "D":
    case "d":
        Console.WriteLine("You wish to delete an address.");
        break;
    case "M":
    case "m":
        Console.WriteLine("You wish to modify an address.");
        break;
    case "V":
    case "v":
        Console.WriteLine("You wish to view the address list.");
        break;
    case "Q":
    case "q":
        Console.WriteLine("Bye.");
        break;
    default:
        Console.WriteLine("{0} is not a valid choice", myChoice);
        break;
}
```

SYS-ED/Computer Education Techniques, Inc.

2:95

Method Example

```
Console.WriteLine();
    Console.Write("press Enter key to continue...");
    Console.ReadLine();
    Console.WriteLine();
} while (myChoice != "Q" && myChoice != "q"); // Keep going until the user
wants to quit
}
```

SYS-ED/Computer Education Techniques, Inc.

2:96

Language Fundamentals

MS C# Programming

Method Example

```
string getChoice()
{
    string myChoice;

    // Print A Menu
    Console.WriteLine("My Address Book\n");

    Console.WriteLine("A - Add New Address");
    Console.WriteLine("D - Delete Address");
    Console.WriteLine("M - Modify Address");
    Console.WriteLine("V - View Addresses");
    Console.WriteLine("Q - Quit\n");

    Console.Write("Choice (A,D,M,V,or Q): ");

    // Retrieve the user's choice
    myChoice = Console.ReadLine();
    Console.WriteLine();

    return myChoice;
}
```

SYS-ED/Computer Education Techniques, Inc.

2:97

Method Example

- The menu functionality has been moved to a new method called `getChoice()`.
 - The return type is a string.
 - This string is used in the switch statement in `Main()`.
 - The method name "getChoice" describes what happens when it is invoked.
 - Since the parentheses are empty, no information will be transferred to the `getChoice()` method.

SYS-ED/Computer Education Techniques, Inc.

2:98

Language Fundamentals

MS C# Programming

Local Variables

- Within the method block, the variable myChoice is declared.
 - Although this is the same name and type as the myChoice variable in Main(), they are both unique variables.
 - They are local variables and are visible only in the block they are declared.
 - The myChoice in getChoice() knows nothing about the existence of the myChoice in Main(), and vice versa.

SYS-ED/Computer Education Techniques, Inc.

2:99

Return Type

- The return statement sends the data from the myChoice variable back to the caller, Main(), of getChoice().
 - The type that is returned by the return statement must be the same as the return-type in the function declaration.
 - In this case it is a string.

SYS-ED/Computer Education Techniques, Inc.

2:100

Language Fundamentals

MS C# Programming

Static and Instance Method

- In the Main() method, a new OneMethod object must be instantiated before getChoice() can be used.
 - This is because of the way getChoice() is declared.
 - Since a static modifier has not been specified, as for Main(), getChoice() becomes an instance method.
- The difference between instance methods and static methods is that multiple instances of a class can be created or instantiated and each instance has its own separate getChoice() method.
- When a method is static, there are no instances of that method, and that one definition of the static method can be invoked.
 - This is done with the declaration:
 - *OneMethod om = new OneMethod();*

SYS-ED/Computer Education Techniques, Inc.

2:101

Method Reference

- On the left hand side of the declaration is the object reference om which is of type OneMethod.
 - The distinction of om being a reference is important.
 - It is not an object itself, but it is a variable that can refer or point to an object of type OneMethod.
- On the right hand side of the declaration is an assignment of a new OneMethod object to the reference om.
- The keyword new is a C# operator that creates a new instance of an object on the heap.
 - A new OneMethod instance is being created on the heap and then being assigned to the om reference.

SYS-ED/Computer Education Techniques, Inc.

2:102

Language Fundamentals

MS C# Programming

Method Reference

- Methods, fields, and other class members can be accessed, identified, or manipulated through the "." (dot) operator.
 - In order to call `getChoice()`, this is done by using the dot operator through the object reference: `om.getChoice()`.
 - The program then executes the statements in the `getChoice()` block and returns.
 - In order to capture the value `getChoice()` returns, use the "=" (assignment) operator.

SYS-ED/Computer Education Techniques, Inc.

2:103

Methods with Parameters Example

```
using System;
class Address
{
    public string name;
    public string address;
}

class MethodParams
{
    public static void Main()
    {
        string myChoice;

        MethodParams mp = new MethodParams();

        do
        {
            // show menu and get input from user
            myChoice = mp.getChoice();

            // Make a decision based on the user's choice
            mp.makeDecision(myChoice);
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:104

Language Fundamentals

MS C# Programming

Methods with Parameters Example

```
// Pause to allow the user to see the results
    Console.Write("press Enter key to continue...");
    Console.ReadLine();
    Console.WriteLine();
} while (myChoice != "Q" && myChoice != "q"); // Keep going until the user
wants to quit
}
// show menu and get user's choice
string getChoice()
{
    string myChoice;
    // Print A Menu
    Console.WriteLine("My Address Book\n");
    Console.WriteLine("A - Add New Address");
    Console.WriteLine("D - Delete Address");
    Console.WriteLine("M - Modify Address");
    Console.WriteLine("V - View Addresses");
    Console.WriteLine("Q - Quit\n");
    Console.WriteLine("Choice (A,D,M,V,or Q): ");
    // Retrieve the user's choice
    myChoice = Console.ReadLine();
    return myChoice;
}
```

SYS-ED/Computer Education Techniques, Inc.

2:105

Methods with Parameters Example

```
// make decision
void makeDecision(string myChoice)
{
    Address addr = new Address();
    switch(myChoice)
    {
        case "A":
        case "a":
            addr.name = "Joe";
            addr.address = "C# Station";
            this.addAddress(ref addr);
            break;
        case "D":
        case "d":
            addr.name = "Robert";
            this.deleteAddress(addr.name);
            break;
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:106

Language Fundamentals

MS C# Programming

Methods with Parameters Example

```
case "M":
case "m":
    addr.name = "Matt";
    this.modifyAddress(out addr);
    Console.WriteLine("Name is now {0}.", addr.name);
    break;
case "V":
case "v":
    this.viewAddresses("Cheryl", "Joe", "Matt", "Robert");
    break;
case "Q":
case "q":
    Console.WriteLine("Bye.");
    break;
default:
    Console.WriteLine("{0} is not a valid choice", myChoice);
    break;
}
}
```

SYS-ED/Computer Education Techniques, Inc.

2:107

Methods with Parameters Example

```
// insert an address
void addAddress(ref Address addr)
{
    Console.WriteLine("Name: {0}, Address: {1} added.", addr.name, addr.address);
}

// remove an address
void deleteAddress(string name)
{
    Console.WriteLine("You wish to delete {0}'s address.", name);
}

// change an address
void modifyAddress(out Address addr)
{
    //Console.WriteLine("Name: {0}.", addr.name); // causes error!
    addr = new Address();
    addr.name = "Joe";
    addr.address = "C# Station";
}
```

SYS-ED/Computer Education Techniques, Inc.

2:108

Language Fundamentals

MS C# Programming

Methods with Parameters Example

```
// show addresses
void viewAddresses(params string[] names)
{
    foreach (string name in names)
    {
        Console.WriteLine("Name: {0}", name);
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:109

Passing Parameters

- In the previous example, the program has been modularized and additional implementation is added to show parameter passing.
- There are 4 kinds of parameters a C# method can handle:
 - out
 - ref
 - params
 - value
- In the program, an address class was created with two string fields.

SYS-ED/Computer Education Techniques, Inc.

2:110

Language Fundamentals

MS C# Programming

Passing Parameters

- In Main() the getChoice() is called to get the user's input and put that string in the myChoice variable.
 - Then myChoice is used as an argument to makeDecision().
- In the declaration of makeDecision(), its one parameter is declared as a string with the name myChoice.
 - This is a new myChoice, separate from the caller's argument and local only to this method.
- Since makeDecision()'s myChoice parameter does not have any other modifiers, it is considered a value parameter.
 - The actual value of the argument is copied on the stack.
- Variables given by value parameters are local and any changes to that local variable do not affect the value of the variable used in the caller's argument.

SYS-ED/Computer Education Techniques, Inc.

2:111

this Reference

- Instead of using the mp reference, the this keyword is used.
 - This is a reference to the current object.
 - It is known, that the current object has been instantiated because makeDecision() is not a static method.
 - Therefore, the this reference can be used to call methods within the same instance.

SYS-ED/Computer Education Techniques, Inc.

2:112

Language Fundamentals

MS C# Programming

ref Parameter

- The switch statement in `makeDecision()` calls a method for each case.
- The `addAddress()` method takes a ref parameter.
 - This means that a reference to the parameter is copied to the method.
 - This reference still refers to the same object on the heap as the original reference used in the caller's argument.
- This means any changes to the local reference's object also changes the caller reference's object.
 - The code can not change the reference, but it can make changes to the object being referenced.
- This is a de facto input/output parameter.

SYS-ED/Computer Education Techniques, Inc.

2:113

out Parameter

- Methods have return values; however, there will be times when there will be a requirement to return more than one value from a method.
 - An out parameter allows the return of additional values from a method.
 - `modifyAddress()` has an out parameter.
 - out parameters are only passed back to the calling function.
- As a result of definite assignment rules, the variable can not be used until it has a valid value assigned.
- The first line in `modifyAddress()` has been commented to illustrate this point.
 - Uncomment it and compile to see what happens.

SYS-ED/Computer Education Techniques, Inc.

2:114

Language Fundamentals

MS C# Programming

params Parameter

- The params parameter is a very useful addition to the C# language.
 - It provides the capability to define a method that can accept a variable number of arguments.
- The params parameter must be a single dimension or jagged array.
 - When calling `viewAddresses()`, four string arguments are passed.
 - The number of arguments is variable and will be converted to a `string[]` automatically.
- In `viewAddresses()` a foreach loop is used for printing each of these strings.
 - Instead of the list of string arguments, the input could have also been a string array.
- The params parameter is considered an input only parameter and any changes affect the local copy only.

SYS-ED/Computer Education Techniques, Inc.

2:115

Lesson 5

Namespaces

SYS-ED/Computer Education Techniques, Inc.

2:116

Language Fundamentals

MS C# Programming

Objectives

You will learn:

- What a namespace is.
- How to implement the using directive.
- Namespace members.

SYS-ED/Computer Education Techniques, Inc.

2:117

Namespaces - Purpose

- Namespaces are C# program elements designed to help organize programs.
 - They also provide assistance in avoiding name clashes between two sets of code.
- Implementing namespaces in code is a sound practice; it likely will serve to preclude problems when reusing code.
- Namespaces do not correspond to file or directory names.
 - If naming directories and files to correspond to namespaces helps in organizing code, then it can be undertaken.
 - However, it is not required.

SYS-ED/Computer Education Techniques, Inc.

2:118

Language Fundamentals

MS C# Programming

Namespace Example

- The following program illustrates how to create a namespace.
- The new namespace is declared by putting the word namespace in front of sysed.
 - Curly braces surround the members inside the sysed namespace.

SYS-ED/Computer Education Techniques, Inc.

2:119

Namespace Example

```
// Namespace Declaration
using System;
namespace sysed
{
    // Program start class
    class NamespaceCSS
    {
        // Main begins program execution.
        public static void Main()
        {
            // Write to console
            Console.WriteLine("New Namespace.");
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:120

Language Fundamentals

MS C# Programming

Nested Namespaces

- Namespaces allow for the creation of a system to organize code.
- A hierarchical system is a good way for organizing namespaces.
 - The general names are placed at the top of the hierarchy and the more specific names further down the hierarchy.
 - This hierarchical system can be represented by nested namespaces.
- The program on the next two pages demonstrates how to create a nested namespace.
 - The placement of code in different sub-namespaces can be used to keep code organized.
- There is another way of writing nested namespaces.
 - It specifies the nested namespace with the dot operator between sysed and subsysed.

SYS-ED/Computer Education Techniques, Inc.

2:121

Nested Namespaces

```
namespace sysed
{
    namespace subsysed
    {
        // Program start class
        class NamespaceCSS
        {
            // Main begins program execution.
            public static void Main()
            {
                // Write to console
                Console.WriteLine("New Namespace.");
            }
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:122

Nested Namespaces

```
using System;

namespace sysed.subsysed
{
    // Program start class
    class NamespaceCSS
    {
        // Main begins program execution.
        public static void Main()
        {
            // Write to console
            Console.WriteLine("New Namespace.");
        }
    }
}
```

SYS-ED/Computer Education Techniques, Inc.

2:123

Calling Namespace Members

- Fully qualified name can be used to call namespace members.
 - A fully qualified name contains every language element from the namespace name down to the method call.
- In order to call methods without typing their fully qualified name, the using directive can be implemented.

SYS-ED/Computer Education Techniques, Inc.

2:124