Name: _____ Class: _____ Date: _____

Chapter 02: Using Data

1. The equal sign (=) is the C# assignment operator.

a. True
b. False

ANSWER: True
POINTS: 1
REFERENCES: 50

2. You can concatenate two strings with the plus (+) sign.

a. True
b. False

ANSWER: True
POINTS: 1
REFERENCES: 53

3. Operator precedence rules ensure that addition and subtraction always take place prior to multiplication, division, or remainder in an expression.

a. True
b. False

ANSWER: False
POINTS: 1
REFERENCES: 64

4. `someValue++` is an example of the use of the prefix increment operator.

a. True
b. False

ANSWER: False
POINTS: 1
REFERENCES: 66

5. An escape sequence is used to represent special characters such as a carriage return or a tab, and always begins with a backslash.

a. True
b. False

ANSWER: True
POINTS: 1
REFERENCES: 74

6. A(n) ____ is a named location in computer memory that can hold different values at different points in time.

a. variable
b. constant
c. literal constant
d. alias

ANSWER: a
POINTS: 1
REFERENCES: 48

Chapter 02: Using Data

7. If a value is given to a variable at the time it is declared, this is called a(n) ____.

a. definition
b. allocation
c. assignment
d. initialization

ANSWER: d
POINTS: 1
REFERENCES: 50

8. A ____ is a string of characters that optionally contains fixed text and contains one or more format items or placeholders for variable values.

a. namespace
b. placeholder
c. format string
d. whitespace

ANSWER: c
POINTS: 1
REFERENCES: 53

9. A ____ consists of a pair of curly braces containing a number that indicates the desired variable's position in a list that follows the string.

a. format string
b. whitespace
c. namespace
d. placeholder

ANSWER: d
POINTS: 1
REFERENCES: 53

10. A(n) ____ number is one that contains decimal positions.

a. integer
b. whole
c. floating-point
d. **`int`**

ANSWER: c
POINTS: 1
REFERENCES: 59

11. In the C# statement `moneyString = someMoney.ToString("F3")`, the expression F3 is an example of a(n) ____.

a. literal constant
b. standard numeric format string
c. named constant
d. intrinsic type

ANSWER: b
POINTS: 1
REFERENCES: 60-61

12. A(n) ____ arithmetic operator has one argument to the left and another argument to the right of the operator.

a. binary
b. complex
c. composite
d. unary

ANSWER: a

POINTS: 1
REFERENCES: 63

13. += is the C# ____ operator; it adds the operand on the right to the operand on the left and assigns it to the operand on the left in one step.

a. prefix increment
b. postfix increment
c. add and assign
d. increment and assign

ANSWER: c
POINTS: 1
REFERENCES: 65

14. A prefix or postfix ____ operator reduces a variable's value by 1.

a. increment
b. sub and assign
c. add and assign
d. decrement

ANSWER: d
POINTS: 1
REFERENCES: 66

15. You declare a Boolean variable by using type ____.

a. **`char`**
b. **`bool`**
c. **`int`**
d. **`boolean`**

ANSWER: b
POINTS: 1
REFERENCES: 68

16. An expression containing a(n) ____ operator has a Boolean value.

a. decrement
b. comparison
c. assignment
d. unifying

ANSWER: b
POINTS: 1
REFERENCES: 68

17. In arithmetic operations with operands of dissimilar types, a(n) ____ is the automatic conversion of nonconforming operands to a unifying type.

a. implicit cast
b. explicit cast
c. manual cast
d. level cast

ANSWER: a
POINTS: 1
REFERENCES: 71

18. To purposefully override the unifying type in an arithmetic operation, use a(n) ____ by placing the desired result type in parentheses followed by the variable or constant to be cast.

a. implicit cast
b. automatic cast

c. intrinsic cast
d. explicit cast

ANSWER: d
POINTS: 1
REFERENCES: 72

19. You use the ____ data type to hold any single character.
a. **`char`**
b. **`string`**
c. **`bool`**
d. **`int`**

ANSWER: a
POINTS: 1
REFERENCES: 73

20. In C#, you use the ____ data type to hold a series of characters.
a. **`int`**
b. **`string`**
c. **`sbyte`**
d. **`char`**

ANSWER: b
POINTS: 1
REFERENCES: 79

21. The keyword ____ is used to declare a named constant.
a. **`final`**
b. **`static`**
c. **`const`**
d. **`fix`**

ANSWER: c
POINTS: 1
REFERENCES: 80

22. A program that accepts console input from the user is a(n) ____ program.
a. interactive
b. static
c. constant
d. better

ANSWER: a
POINTS: 1
REFERENCES: 83

23. You can use the ____ method to accept user input from the keyboard.
a. `Next()`
b. `NextLine()`
c. `ReadLine()`
d. `Input()`

ANSWER: c
POINTS: 1
REFERENCES: 84

24. An instruction to the user to enter data is called a ____.
a. line
b. prompt
c. cue
d. hint

Chapter 02: Using Data

ANSWER: b
POINTS: 1
REFERENCES: 84

25. The ____ method from the `Convert` class converts a specified value to an 8-bit unsigned integer.

a. `ToDecimal()` b. `ToSingle()`
c. `ToInt16()` d. `ToByte()`

ANSWER: d
POINTS: 1
REFERENCES: 86

26. As an alternative to the `Convert` class methods, you can use a(n) ____ method to convert a string to a number.

a. `Cast()` b. `Parse()`
c. `Assign()` d. `Write()`

ANSWER: b
POINTS: 1
REFERENCES: 86

27. A data item is _____ when it cannot be changed after a program is compiled.

ANSWER: constant
POINTS: 1
REFERENCES: 48

28. Giving a value to a variable, as in the statement `someVariable = 5`, is called a(n) _____.

ANSWER: assignment
POINTS: 1
REFERENCES: 50

29. The number of significant digits in a floating point number specifies the mathematical accuracy of the value. A(n) _____ data type can hold up to 16 significant digits of accuracy.

ANSWER: **`double`**
POINTS: 1
REFERENCES: 59

30. Operators are used to manipulate values in a program. For example, in the expression X + Y, + is an operator. The values that operators use, such as X and Y, are called _____.

ANSWER: operands
POINTS: 1
REFERENCES: 63

31. Operators that have only one argument, such as the prefix and postfix increment operators, are called _____ operators.

ANSWER: unary
POINTS: 1

REFERENCES: 66

32. What is a variable declaration, what is its purpose, and what does the declaration include?

ANSWER: A variable declaration is the statement that names a variable and reserves storage for it. The declaration includes:
* The data type that the variable will store
* The variable name (its identifier)
* An optional assignment operator and assigned value when you want a variable to contain an initial value
* An ending semicolon

POINTS: 1

REFERENCES: 50

33. What are the nine integral data types in C# and how do you decide which is the most appropriate to use in a particular situation?

ANSWER: The nine types are **`byte`**, **`sbyte`**, **`short`**, **`ushort`**, **`int`**, **`uint`**, **`long`**, **`ulong`**, and **`char`**. The first eight always represent whole numbers, and the ninth type, **`char`**, is used for characters like 'A' or 'a'.

The most basic of the integral types is **`int`**. An **`int`** uses four bytes of memory and can hold any whole number value ranging from 2,147,483,647 down to -2,147,483,648. Many programmers use **`int`** for most whole numbers. If you want to save memory and know you need only a small value, you can use one of the shorter integer types—**`byte`**, **`sbyte`** (which stands for signed byte), **`short`** (short **`int`**), or **`ushort`** (unsigned short **`int`**). For example, a payroll program might contain a variable named `numberOfDependents` that is declared as type byte, because `numberOfDependents` will never need to hold a negative value or a value exceeding 255; for that reason, you can allocate just one byte of storage to hold the value. If you use a type that is too large, you waste storage. If you use a type that is too small, your program won't compile.

POINTS: 1

REFERENCES: 57

34. What are the three floating-point data types in C#? Briefly describe the characteristics of each type.

ANSWER: A floating-point number is one that contains decimal positions. C# supports three floating-point data types: **`float`**, **`double`**, and **`decimal`**. A **`float`** data type can hold as many as seven significant digits of accuracy. A **`double`** data type can hold 15 or 16 significant digits of accuracy. Compared to **`float`**s and **`double`**s, the **`decimal`** type has a greater precision and a smaller range, which makes it suitable for financial and monetary calculations. For example, a **`decimal`** given the value 123456789.987654321 will appear as 123456789.987654321 (notice that it is accurate to the rightmost digit). A **`decimal`** cannot hold as large a value as a **`double`** can, but it can be accurate to more decimal places.

POINTS: 1

REFERENCES: 59-60

35. Explain how to use the C# shortcut arithmetic operators `-=`, `*=`, and `/=`.

ANSWER: Each of these operators is used to perform an operation and assign the result in one step. For example:

```
balanceDue -= payment
```

subtracts a payment from `balanceDue` and assigns the result to `balanceDue`.

```
rate *= 100
```

multiplies `rate` by 100 and assigns the result to `rate`. For example, this could be used to convert a fractional value stored in `rate`, such as 0.27, to a whole number, such as 27.

```
payment /= 12
```

divides `payment` by 12 and assigns the result to `payment`. This could be used to change a `payment` value from an annual amount to a monthly amount due.

POINTS: 1

REFERENCES: 65

36. What are the differences between the prefix increment operator and postfix increment operator?

ANSWER: When you only want to increase a variable's value by 1, there is no apparent difference between using the prefix and postfix increment operators. However, these operators function differently. When you use the prefix ++, the result is calculated and stored, and then the variable is used. In contrast, when you use the postfix ++, the variable is used, and then the result is calculated and stored.

POINTS: 1

REFERENCES: 66

37. Explain why implicit numeric conversions are necessary and how C# chooses operands to convert.

ANSWER: When you perform arithmetic with variables of the same type, the result of the arithmetic retains that type. For example, when you divide two **`int`**s, the result is an **`int`**. Often, however, you need to perform mathematical operations on different types. For example, you might need to multiply an **`int`** by a **`double`**. In this case, C# chooses a unifying type for the result and implicitly converts nonconforming operands to the unifying type, which is the type with the higher type precedence. In the case of multiplying an **`int`** by a **`double`**, the result will be a **`double`**.

POINTS: 1

REFERENCES: 71

38. How can you compare strings using the `Compare()` method?

ANSWER: The `Compare()` method requires two string arguments and returns an integer result. When it returns 0, the two strings are equivalent; when it returns a positive number, the first string is greater than the second; and when it returns a negative number, the first string is less than the second. A string is considered equal to, greater than, or less than another string lexically, which in the case of letter values means alphabetically. That is, when you compare two strings, you compare each character in turn from left to right. If each Unicode value is the same, then the strings are equivalent. If any corresponding character values are different, the string that has the greater Unicode value earlier in the string is considered greater.

POINTS: 1

REFERENCES: 77-78

39. How is a named constant different from a variable? How would you create a named constant (and show an example)?

ANSWER: By definition, a variable's value can vary, or change. Sometimes you want to create a named constant (often simply called a constant), an identifier whose contents cannot change. You create a named constant similar to the way you create a named variable, but by using the keyword **`const`**. Although there is no requirement to do so, programmers usually name constants using all uppercase letters, inserting underscores for readability. This convention makes constant names stand out so that the reader is less likely to confuse them with changeable variable names. For example, the following

declares a constant named `TAX_RATE` that is assigned a value of 0.06:

```
const double TAX_RATE = 0.06;
```

POINTS: 1

REFERENCES: 80-81

40. What is an enumeration and what are the advantages of creating an enumeration type? Show an example of an enumeration definition.

ANSWER: An enumeration is a set of constants represented by identifiers. An example of an enumeration definition is the following:

```
enum DayOfWeek
{
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY,
    SATURDAY
}
```

Creating an enumeration type provides several advantages. For example, the `DayOfWeek` type above improves a program as follows:

1. Only the seven allowed values can be assigned to a `DayOfWeek` object.
2. Using an enumeration type makes a program type-safe (for example, you couldn't use an integer method such as addition on an object of type `DayOfWeek`).
3. The **`enum`** constants provide clearer self-documentation (it is more apparent what you mean when you assign a variable the value `WEDNESDAY` rather than the number 4, for example).

POINTS: 1

REFERENCES: 81

41. What is the purpose of the `ReadLine()` method? How would you use it with a variable of type **`string`**? How would you use it with a variable of type **`double`**?

ANSWER: You use the `ReadLine()` method to accept user input from the keyboard. This method accepts all of the characters entered by a user until the user presses Enter. The characters can be assigned directly to a string. For example, the following statement accepts a user's input and stores it in the variable `myString`:

```
myString = ReadLine();
```

If you want to use the data as a string—for example, if the input is a word or a name—then you simply use the variable to which you assigned the value. If you want to use the data as a number, then you must use a conversion method to convert the input string to the proper type, as in:

```
itemPrice = Convert.ToDouble(myString);
```

POINTS: 1

REFERENCES: 83-84

Match each item with a statement below:

a. data type

b. integral data type

c. precision specifier

d. culture

e. operator precedence
f. Boolean variable
g. comparison operator
h. Unicode
i. named constant

REFERENCES: 48
56
61
62
64
68
69
74
80

42. An identifier whose contents cannot change
ANSWER: i
POINTS: 1

43. A set of rules that determines how to format values such as money and dates that can be represented differently based on locale and tradition
ANSWER: d
POINTS: 1

44. Controls the number of significant digits or zeros to the right of a decimal point
ANSWER: c
POINTS: 1

45. A 16-bit coding scheme for characters
ANSWER: h
POINTS: 1

46. Can hold only one of two values: true or false
ANSWER: f
POINTS: 1

47. Stores a whole number
ANSWER: b
POINTS: 1

48. Rules that determine the order in which parts of a mathematical expression are evaluated
ANSWER: e
POINTS: 1

49. Describes the format and size (amount of memory occupied) of a data item and defines a set of relevant operations
ANSWER: a

Chapter 02: Using Data

POINTS: 1

50. <= is an example of this
ANSWER: g
POINTS: 1