1. abstract: Used to declare abstract classes or abstract members that must be implemented by derived classes.
2. as: Used for type casting or to perform null-safe type conversions.
3. base: Refers to the base class or calls a method in the base class.
4. bool: Represents a Boolean value, which can be either true or false.
5. break: Exits from a loop or switch statement.
6. byte: Represents an 8-bit unsigned integer value.
7. case: Used in switch statements to specify different cases.
8. catch: Catches exceptions that occur in try blocks for error handling.
9. char: Represents a single Unicode character.
10. checked: Checks for overflow in integral-type arithmetic operations.
11. class: Declares a class, which is a blueprint for creating objects.
12. const: Declares a constant value that cannot be changed.
13. continue: Skips the rest of the loop iteration and proceeds to the next iteration.
14. decimal: Represents a decimal floating-point number.
15. default: Specifies the default case in a switch statement.
16. delegate: Represents a reference to a method with a specific signature.
17. do: Executes a block of code repeatedly until a specified condition is no longer true.
18. double: Represents a double-precision floating-point number.
19. else: Specifies an alternative block of code to execute if the condition of an if statement is false.
20. enum: Declares an enumeration, which is a set of named values.
21. event: Declares an event, which enables objects to communicate with each other.
22. explicit: Defines a user-defined type conversion that must be invoked explicitly.
23. extern: Indicates that a method is implemented externally in another programming language.
24. false: Represents the Boolean value false.
25. finally: Defines a block of code that is executed after a try block, regardless of whether an exception is thrown.
26. fixed: Declares a fixed-size buffer to store data in unsafe code.
27. float: Represents a single-precision floating-point number.
28. for: Executes a block of code repeatedly for a specified number of times.
29. foreach: Iterates over elements in a collection or array.
30. goto: Transfers control to a labeled statement.
31. if: Executes a block of code based on a specified condition.
32. implicit: Defines a user-defined type conversion that can be invoked implicitly.
33. in: Specifies the iteration variable in a foreach loop.
34. int: Represents a 32-bit signed integer value.
35. interface: Declares an interface, which defines a contract for implementing classes.
36. internal: Limits the accessibility of a type or member to within the same assembly.
37. is: Checks if an object is compatible with a given type.
38. lock: Synchronizes access to a shared resource by acquiring the exclusive lock.
39. long: Represents a 64-bit signed integer value.
40. namespace: Declares a namespace, which organizes code and prevents naming conflicts.
41. new: Creates an instance of a class or hides an inherited member.
42. null: Represents a null reference.
43. object: Represents a base type from which all other types are derived.
44. operator: Declares an operator for a user-defined type.
45. out: Used in method parameters to pass a value by reference.
46. override: Provides a new implementation of a virtual or abstract member in a derived class.
47. params: Allows a variable number of arguments to be passed to a method.
48. private: Restricts access to a type or member to within the same class.
49. protected: Specifies that a member can only be accessed within its containing class or derived classes.
50. public: Specifies that a type or member is accessible from any code in the same assembly or another assembly.
51. readonly: Declares a field as read-only, meaning its value can only be assigned during initialization or within the constructor.
52. ref: Used in method parameters to pass arguments by reference, allowing the method to modify the value of the original variable.
53. return: Terminates the execution of a method and returns a value (if specified) to the caller.
54. sbyte: Represents a signed 8-bit integer value.
55. sealed: Specifies that a class or method cannot be derived from or overridden, respectively.
56. short: Represents a signed 16-bit integer value.
57. sizeof: Returns the size in bytes of a value type.
58. stackalloc: Allocates a block of memory on the stack.
59. static: Declares members that belong to the type itself rather than instances of the type.
60. string: Represents a sequence of characters.
61. struct: Declares a value type.
62. switch: Evaluates an expression and executes the code block associated with a matching case.
63. this: Refers to the current instance of a class or struct.
64. throw: Generates an exception.
65. true: Represents the Boolean value true.
66. try: Encloses a block of code that may throw exceptions, allowing for exception handling.
67. typeof: Gets the System.Type object for a specified type.
68. uint: Represents an unsigned 32-bit integer value.
69. ulong: Represents an unsigned 64-bit integer value.
70. unchecked: Disables overflow checking in integral-type arithmetic operations.
71. unsafe: Declares an unsafe context, allowing the use of pointers and unsafe code.
72. ushort: Represents an unsigned 16-bit integer value.
73. using: Provides a convenient way to use namespaces and dispose of resources.
74. virtual: Specifies that a method, property, or event can be overridden in derived classes.
75. void: Specifies that a method does not return a value.
76. volatile: Specifies that a field can be modified by multiple threads.
77. while: Executes a block of code repeatedly as long as a specified condition is true