Glossary

<https://csci-1301.github.io/about#authors>

November 2, 2023 (01:03:21 PM)

Table of Contents

[Keywords/Reserved Words](https://csci-1301.github.io/book.html#reserved-words-and-identifiers):

words defined by the C# language and used for one thing

[Datatypes](https://csci-1301.github.io/book.html#datatypes-and-variables):

categories in C# used to define types of values, such as strings

[Variable](https://csci-1301.github.io/book.html#datatypes-and-variables):

values that be changed

[Constant](https://csci-1301.github.io/book.html#constants):

values that can not be changed

[Identifier](https://csci-1301.github.io/book.html#reserved-words-and-identifiers):

words defined by the programmer to refer to an object or variable.

Operations:

[Operators](https://csci-1301.github.io/book.html#operators):

symbols used to perform operations

[Modulo](https://csci-1301.github.io/book.html#arithmetic-operators):

the % used to divide two numbers and return the remainder

[Escape Sequence](https://csci-1301.github.io/book.html#escape-sequences):

used to represent a non-printable character

[Reference Types (Objects and Strings)](https://csci-1301.github.io/book.html#value-and-reference-types-1=):

a variable of a class object holds a reference to the address of the object on the managed heap.

[Value Types (all other reserved words)](https://csci-1301.github.io/book.html#value-and-reference-types-1=):

a variable of a class object stores the exact data value held by the variable

[Numeric Types](https://csci-1301.github.io/book.html#sizes-of-numeric-datatypes-1):

[Booleans](https://csci-1301.github.io/book.html#boolean-variables-and-values):

a binary datatype that can only be true or false

[Decision Structures (if/else/switch)](https://csci-1301.github.io/book.html#decisions-and-decision-structures):

[Control Structures (loops)](https://csci-1301.github.io/book.html#loops-increment-operators-and-input-validation):

[Instantiation (instance of a class)](https://csci-1301.github.io/book.html#using-our-class-1):

the act of creating a object, an instance of a C# class

[Initialization](https://csci-1301.github.io/book.html#initialization-declaration-assignment-1):

the act of both declaring a variable’s datatype and identifier and assigning it value

[Declaration](https://csci-1301.github.io/book.html#declaration-1):

the act of creating a variable’s datatype and identifier

[Assignment](https://csci-1301.github.io/book.html#assignment-1):

the act of giving a value to an identifier

[Implicit Conversion](https://csci-1301.github.io/book.html#implicit-and-explicit-conversions-between-datatypes):

the act of automatically storing the value of one identifier into another identifier that differs from its own

[Explicit Conversion (casting)](https://csci-1301.github.io/book.html#implicit-and-explicit-conversions-between-datatypes):

the act of storing the value of one identifier into another identifier that differs from its own using additional syntax

[Rules](https://csci-1301.github.io/book.html#rules-of-c-syntax):

are required syntactical ways to write a program for it to function [Conventions](https://csci-1301.github.io/book.html#conventions-of-c-programs):

are not required for the program to function, but are heavily encouraged for the readability and comprehension of other programmers

[Format Specifiers (C, N, P, E)](https://csci-1301.github.io/book.html#format-specifiers-1):

added to variable calls in strings to format the numeric variable in various ways (see link for specifics)

[Constructor](https://csci-1301.github.io/book.html#constructors):

a method used to instantiate an object and assign it’s attributes

[Parameter](https://csci-1301.github.io/book.html#writing-our-first-class-1):

any variable declared within a method

[Argument](https://csci-1301.github.io/book.html#first-program-1):

any value that must passed to a method in order for it to be called

[Attribute](https://csci-1301.github.io/book.html#the-object-oriented-paradigm-1):

the variables declared within a class to act as the characteristics of any of its instantiated object

[Method](https://csci-1301.github.io/book.html#the-object-oriented-paradigm-1):

a code block that contains a series of statements

[Class Member (attributes and methods)](https://csci-1301.github.io/book.html#class-and-object-basics-1):

anything defined within a class that can be access within and outside of the class

[Scope](https://csci-1301.github.io/book.html#definition-of-scope-1):

Time and place in program where the variable exists

Iterator:

an object that traverses an array or list

[Sentinel Value](https://csci-1301.github.io/book.html#vocabulary-1):

a special value in the context of an algorithm which uses its presence as a condition of termination, typically in a loop or recursive algorithm.

Guard Condition:

boolean expressions (predicates) found at the top of a method or function that determine whether the function should continue to execute.

[Accumulator](https://csci-1301.github.io/book.html#vocabulary-1):

[Counter](https://csci-1301.github.io/book.html#vocabulary-1):

a variable used to count the number of times a certain condition is met

[Complex Condition](https://csci-1301.github.io/book.html#while-loop-with-complex-conditions-1):

a condition consisting of multiple conditions

[Method Signature](https://csci-1301.github.io/book.html#method-signatures-1):

the way a computer reads a method by its name and the datatype of its parameters

[Method Overloading](https://csci-1301.github.io/book.html#calling-overloaded-methods-1):

the act of creating multiple methods with the same signature

[Return Type](https://csci-1301.github.io/book.html#writing-our-first-class-1):

the datatype of any value returned from a called method

[UML Diagram](https://csci-1301.github.io/book.html#introduction-to-uml):

a written diagram used to display a class and all of its members

[Input Validation](https://csci-1301.github.io/book.html#loops-and-input-validation):

whenever a program checks if the user gave a usable input and responds accordingly to avoid errors implementation