List of Topics

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

November 2, 2023 (01:03:19 PM)

Table of Contents

## General Concepts

*Students should understand the meaning and importance of the following notions. This statement should be read as “understand the first sentence or paragraph on a wikipedia article”, taking* [*high-level programming language*](https://en.wikipedia.org/wiki/High-level_programming_language) *as an example.*

* Programming languages types and paradigms
	+ Machine language instructions
	+ Assembly instructions
	+ High-Level Programming Languages
	+ Object-oriented paradigm and data hiding
* The difference between roles (user, tester, programmer)
* How complex piece of software *reuse* previous pieces.
* The importance of security 🛡
	+ Types of attack (malware, phishing, social engineering, zero-day)
	+ Types of loss (loss of integrity / availability / confidentiality)

## Writing and Compiling Programs

* Understand what the “flow of development” is:
	+ Having a goal
	+ Writing down specifications
	+ Creating the source code
	+ Running the compiler
	+ Reading the compiler’s output, warning and error messages
	+ Looking for documentation and help on-line and off-line
	+ Testing
	+ Making sure the program is secure 🛡
	+ Editing
	+ Reusing
* Using an IDE to
	+ Create a project,
	+ Perform some of the steps of the “flow of development”,
	+ Correctly save and re-open projects,
	+ Understand basic features of break points and debugging. ❓

*The IDE used can be* [*MonoDevelop*](https://www.monodevelop.com/) *or* [*Visual Studio*](https://visualstudio.microsoft.com/)*, the student can pick other IDEs if they wish but they will not be supported.*

## Computer Usage

* How to download and install an IDE in a secure way 🛡
* How to share and zip a project
* How to use shortcuts ❓
* How to look for on-line documentation

## The Structure of a Program

### First Program - Hello World

*The students should understand all the components of a simple “Hello World” program:*

* Comments (in line and block)
* using statements and namespace / API concepts
* blank lines and spacing
* indentation
* intro to classes and methods’ structures (body / header)
* status of Main method
* intro to Console’s Write and WriteLine
* string literal

### Rules and Conventions

* The difference between a “rule” (e.g. case-sensitivity) and a “convention” (commenting your code).
* Reserved words
* Identifiers and naming conventions
* That the distinction can vary with the programming language
* Importance and role of { and }

# Datatypes and Operators

## Variable

* Datatype (numerical, boolean, string, character) – including a mention of reference datatypes
* Declaration, assignment, initialization
* Naming variables correctly
* The absence of default value after declaration (un-assigned variables)

## Numerical Values

* Integers (int, long) – range and size, signature (uint)
* Floating Point (float, double, and decimal) – range, size and precision,
* Type casting (e.g. from int to double, and legal operations between different datatypes) and casting operator (e.g. (int)).
* Overflow and underflow 🛡

## Booleans

* Possible values (true, false)
* Usage
* That boolean variables are called “switches”

## Operators

* Binary arithmetic operators: \*, /, %, +, -
* Unary arithmetic operators: ++, --
* The difference between postfix and infix notation for unary operators ❓
* Comparison operators: !=, ==, >, >=, <, <=
* Boolean logical operators: &&, ||, !
* Precedence and “validity” of some expressions (typically, ! 2 < 3 is not a valid expression)
* Combined assignment operators: +=, \*=, -=, /=, %=

## Strings

* ReadLine method
* Concatenation (+)
* Interpolation
* Additional methods: ToLower, ToUpper, Contains, StartsWith, EndsWith ❓

### Displaying Strings on the Screen

* [Format specifiers](https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings) for numbers: – Currency (C),
	+ Fixed-point (F) or Number (N)
	+ Percent (P) ❓
	+ Exponential (E) ❓
* The String.Format method ❓

## Characters

* Possible values and the existence of binary, oct, dec and hex representation (cf. for instance [wikipedia](https://en.wikipedia.org/wiki/ASCII#Printable_characters))
* Escape character and sequences: \n, \t, \\
* Conversion between glyph and decimal value.
* Various methods: ToLower, ToUpper ❓

# Lists ❓

* Creating a list of numbers or strings
* Adding items using the Add method
* Accessing items using []
* Removing and Inserting (Remove, RemoveAt, Insert)
* Count property

# Basic Control Structures

## Selection Statements

For each of the following structure:

* if
* if-else
* if-else if
* nested ifs
* switch

The student should understand

* Their importance,
* Their usage,
* Their syntax,
* Their flow,
* When to use one or the other,
* The common pitfalls (e.g., writing a condition in a switch).

## Repetition Statements

For each of the following structure:

* foreach
* while
* for
* do{…}while(…) ❓

The student should understand:

* Their importance,
* Their usage,
* Their syntax,
* Their flow,
* When to use one or the other,
* The common pitfalls (e.g. = instead of ==, <= n vs < n)

As well as being capable of identifying the difference between

* Counter-controlled,
* Sentinel-controlled,
* User-controlled

and defining the term “accumulator”

# Object-oriented programming

## Class Conception

* Need and interest of specification 🛡
* UML Class diagram: interest, usage, and simple case (single class with attributes, methods and constructor).
* Access modifier (private, public)
* Principle of least privilege (private variables and methods where possible) 🛡

## Class Implementation

* Attributes (and their default value, as well as how to change them)
* Get and Set methods
* Properties ❓
* Method signature
* Overloading
* [Variable shadowing](https://en.wikipedia.org/wiki/Variable_shadowing) ❓
* Constructors: default constructor and “custom” constructor

## Class Usage

* The new keyword
* Object creation using default and custom constructors
* Object manipulation: calling a method, setting an attribute, calling the ToString method implicitly.

## Additional Considerations

* ToString method
* static class and methods
* [Math Class](https://docs.microsoft.com/en-us/dotnet/api/system.math?view=net-5.0) (Abs, Sqrt, Pow) ❓

# Random Class

* Creating a generator with new Random()
* Generating non-negative integers,
* Generating integers between ranges,
* Generating double,
* Generating a random word ❓
* Potential problems with deterministic generators 🛡

# Testing and Debugging

* How to test intelligently
* How to test every instruction
* How to test boundary conditions

# Interacting with Users

* Input validation 🛡
* TryParse in the int and decimal classes.
* Reading a single character from the user ❓

# Data structures

## Constant

* The const keyword
* Example usages (Avogadro constant, miles-to-kilometer ratio, speed of light) and use case.
* Math.PI ❓
* Static constant ❓

## Enumerated Datatype ❓

* Define enumerated datatypes using enum
* Enum values (i.e. numerical values assigned to enumerated values by default) ❓
* Use enumerated datatypes (variable declaration, assignment, displaying).

## Arrays

Only one-dimensional arrays should be discussed.

* Vocabulary: index (starting at 0), bounds.
* Length property
* Resize method ❓
* Different syntaxes for initializing and declaring arrays ❓
* Buffer overflow 🛡

# Exceptions 🔜

* try…catch blocks
* Types of exceptions
* finally
* Defining your own exception

# File I/O 🔜

* StreamWriter and StreamReader classes
* Manipulating binary and text files
* File class ❓