List of Topics

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

April 5, 2024 (12:47:55 PM)

Contents

0.1 General Concepts . . . 2
0.2 Writing and Compiling Programs . . . 2
0.3 Computer Usage . . . 3
0.4 The Structure of a Program . . . 3
0.4.1 First Program - Hello World . . . 3
0.4.2 Rules and Conventions . . . 3

1 Datatypes and Operators **3**
1.1 Variable . . . 3
1.2 Numerical Values . . . 3
1.3 Booleans . . . 4
1.4 Operators . . . 4
1.5 Strings . . . 4
1.5.1 Displaying Strings on the Screen . . . 4
1.6 Characters . . . 4

2 Lists ? **5**

3 Basic Control Structures **5**
3.1 Selection Statements . . . 5
3.2 Repetition Statements . . . 5

4 Object-oriented programming **6**
4.1 Class Conception . . . 6
4.2 Class Implementation . . . 6
4.3 Class Usage . . . 6
4.4 Additional Considerations . . . 6

5 Random Class **6**

6 Testing and Debugging **7**

7 Interacting with Users **7**

8 Data structures **7**
8.1 Constant . . . 7
8.2 Enumerated Datatype ? . . . 7
8.3 Arrays . . . 7

9 Exceptions SOON **7**

10 File I/O ➡ SOON **8**

For an explanation of the shortcodes labelling, please refer to our user guide[1].

0.1 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[2] 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)

0.2 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[3] or Visual Studio[4], the student can pick other IDEs if they wish but they will not be supported.

[1] user_guide#labelling-with-shortcodes-1
[2] https://www.wikiwand.com/en/High-level_programming_language
[3] https://www.monodevelop.com/
[4] https://visualstudio.microsoft.com/

0.3 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

0.4 The Structure of a Program

0.4.1 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

0.4.2 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 `}`

1 Datatypes and Operators

1.1 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)

1.2 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 ⛉

1.3 Booleans

- Possible values (`true`, `false`)
- Usage
- That boolean variables are called "switches"

1.4 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: `+=`, `*=`, `-=`, `/=`, `%=`

1.5 Strings

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

1.5.1 Displaying Strings on the Screen

- Format specifiers[5] for numbers: – Currency (`C`),
  - Fixed-point (`F`) or Number (`N`)
  - Percent (`P`) **?**
  - Exponential (`E`) **?**
- The `String.Format` method **?**

1.6 Characters

- Possible values and the existence of binary, oct, dec and hex representation (cf. for instance wikipedia[6])
- Escape character and sequences: `\n`, `\t`, `\\`
- Conversion between glyph and decimal value.
- Various methods: `ToLower`, `ToUpper` **?**

[5] https://learn.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings
[6] https://www.wikiwand.com/en/ASCII#Printable_characters

2 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

3 Basic Control Structures

3.1 Selection Statements

For each of the following structure:

- **`if`**
- **`if-else`**
- **`if-else if`**
- nested **`if`**s
- **`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`**).

3.2 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"

4 Object-oriented programming

4.1 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) ♡

4.2 Class Implementation

- Attributes (and their default value, as well as how to change them)
- Get and Set methods
- Properties **?**
- Method signature
- Overloading
- Variable shadowing[7] **?**
- Constructors: default constructor and "custom" constructor

4.3 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.

4.4 Additional Considerations

- `ToString` method
- static class and methods
- `Math` Class[8] (`Abs`, `Sqrt`, `Pow`) **?**

5 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 ♡

[7] https://www.wikiwand.com/en/Variable_shadowing
[8] https://learn.microsoft.com/en-us/dotnet/api/system.math

6 Testing and Debugging

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

7 Interacting with Users

- Input validation ⛉
- `TryParse` in the `int` and `decimal` classes.
- Reading a single character from the user **?**

8 Data structures

8.1 Constant

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

8.2 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).

8.3 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 ⛉

9 Exceptions ➡ SOON

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

10 File I/O ➡ SOON

- `StreamWriter` and `StreamReader` classes
- Manipulating binary and text files
- `File` class **?**