C Programming Language - Associate Programmer

- CLA - C Programming Language Certified Associate
  - **0 Installing and using your programming environment**
    - 0.1 Introduction
    - 0.2 Installing compiler
    - 0.3 Installing NetBeans
  - **1 Introduction to computer programming**
    - 1.1 Different languages for different purposes
    - 1.2 Your first program
    - 1.3 Integer values, integer variables and comments
  - **2 Data types, their operations and basics of flow control**
    - 2.1 Floating-point numbers
    - 2.2 Computer arithmetic and arithmetic operators
    - 2.3 Characters as another kind of data
    - 2.4 Controlling the flow – absolute basics
    - 2.5 Formatted input/output
  - **3. Flow control (continued), more data types and computer logic**
    - 3.1 if-else statement
    - 3.2 More data types
    - 3.3 Conversions
    - 3.4 Loops
    - 3.5 Computer logic
  - **4 Aggregating data into arrays**
    - 4.1 switch - the different face of 'if'
    - 4.2 Vectors: why do you need them?
    - 4.3 Sorting data: in real life and in the computer memory
    - 4.4 Initiators - the simple way to set an array
    - 4.5 Not only ints
    - 4.6 Pointers: another kind of data in the "C" language
    - 4.7 Pointers vs. arrays: different forms of the same phenomenon
    - 4.8 The string: a very special vector
    - 4.9 Assigning values to strings
    - 4.10 Processing strings
  - **5 Arrays vs. structures: different aggregates for different purposes**

- 5.1 The real meaning of array indexing
- 5.2 Using pointers: perils and disadvantages
- 5.3 Arrays of arrays: multidimensional arrays
- 5.4 Memory allocation and deallocation: malloc() and free()
- 5.5 Arrays of pointers as multidimensional arrays
- 5.6 Declaring arrays: traps and puzzles
- 5.7 The structures: why?
- 5.8 Declaring and initializing structures
- 5.9 Pointers to structures and arrays of structures
- 5.10 Basics of recursive data collections

- **6 Functions**
  - 6.1 Functions: why do we need them?
  - 6.2 Our first function
  - 6.3 Variables, parameters and results
  - 6.4 Scalars as function parameters
  - 6.5 Structures and strings as function parameters
  - 6.6 Arrays as function parameters
  - 6.7 Parameterizing the main function
  - 6.8 The basics of disjoint compilation
- **7 Connecting to the real world: files and streams**
  - 7.1 File systems: definitions and conventions
  - 7.2 Introduction to files and streams
  - 7.3 Opening streams
  - 7.4 Pre-opened streams
  - 7.5 Closing the stream and error handling
  - 7.6 Reading from the stream
  - 7.7 Writing to the stream
  - 7.8 Dealing with the stream's position
- **8 Preprocessor and declarations**
  - 8.1 Preprocessor: absolute basics
  - 8.2 Preprocessor: the #include directive
  - 8.3 Preprocessor: the #define directive
  - 8.4 Preprocessor: the parameterized #define directive
  - 8.5 Preprocessor: the third variant of the #define directive and #undef directive
  - 8.6 Preprocessor: predefined identifiers
  - 8.7 Preprocessor: operators

- 8.8 Preprocessor: conditional compilation
- 8.9 The scope of the declaration
- 8.10 Storage classes
- 8.11 Pointers to functions
- 8.12 Complex declarations