ICS0004 Fundamentals of programming

Overview

The primary goals of this course are:
1. Introduce the software development terms, tools and problems.
2. Teach the basics of design of algorithms and their implementation using the C programming language.
3. Train the students to give them some practical experience in programming.

On successful completion of the course the students will be able to construct small scale algorithms, code them in C and implement in Microsoft Visual Studio environment.

Number of ECTS credits awarded for this course is 6.0.

Presumptions

The course is for absolute beginners and therefore has no compulsory prerequisites. Some limited experience in Windows 10/11 (file handling and text editing) is expected.

Students who have learned C language before

The students who have learned C language may test their knowledge. The optional mock examination will be held at the beginning of semester (presumably on the second week). The excellent mark in mock examination means that the student has carried out the actual examination and finished the course. For those who get lower mark the mock examination simply points out their shortcomings in knowledge.

Time and place

The course is offered in the autumn term. The lessons (once per week, 4 * 45 minutes) include lectures as well as exercises and practical work.

The lessons are held in classroom ICT-121 / 122 on each Friday from 12:00 until 15:15.

Language

The courseware and the lectures are in English.

Software support

In our computer class we use Windows 10 and Microsoft Visual Studio. It may happen that the students need to finish a more complicated exercise at home. Therefore it is advised to download and install the Visual Studio 2022 Community Edition (https://visualstudio.microsoft.com/downloads/) into his/her own computer. Remark that students who’s computer runs macOS or Linux may meet rather serious difficulties.

Course themes

1. First steps:
    a. Basic terms of programming: source code, object code, executive, library, header files, compiler, linker, debugger, wizard, integrated development environment, bit, byte, etc.
    b. Simple C program, the analysis and explanation of its source code statements. Functions and calls to them. Preprocessor directives.
    c. Positional numeral systems. Converting between binary and hexadecimal systems. Integers and floating-point values in computer memory.

   d. Basic knowledge about programming in C: variables and their declaration, arrays, ASCII characters and strings, converting from text to numeral types and vice versa.
   e. Branching. *if* and *if else* statements. Flowcharts.

2. Fundamentals of C language:
   a. C and C++. A brief history.
   b. Components of the source code (statements, expressions, blocks, functions).
   c. Standard data types and declarations.
   d. Constants.
   e. Arrays.
   f. Assignment. Arithmetical, logical and relational expressions.
   g. Loops. *While* and *break* statements.
   h. Functions: parameters, return values, prototypes, calls.
   i. Preprocessor directives.
   j. Standard functions for input and output.

3. Going deeper:
   a. Visibility and lifetime. Local and global variables.
   b. Incrementing and decrementing operators.
   c. Loops. *for*, *do while* and *continue* statements.
   d. Labels. *goto* and *switch* statements.
   e. Expressions with operands of different types. Type conversions. Casting operator.
   f. Compound assignments.
   g. Conditional expressions.
   h. Operator precedense table.

4. Pointers:
   a. Dynamical memory allocation. *sizeof* operator.
   b. Declaration of pointers.
   c. Dereferencing and pointer arithmetics.
   d. Address-of operator. Functions with several output values.
   e. Pointers to pointers and simple data structures.
   f. Syntactic shorthands.
   g. Standard functions for memory management.
   h. Constant pointers.
   i. String constants. Initialization of strings and string arrays.
   j. Standard functions for string handling.

5. Structs:
   a. Declaration of structs and pointers to structs. Arrays of structs. Initialization.
   b. Accessing the struct attributes.
   c. Structs consisting of separate memory fields.
   d. Standard functions for time handling.
   e. Standard functions for file handling.

6. Programming technologies: a short overview:
   a. Dynamic link libraries in C
   b. Python as interpreted language
   c. Java and the Virtual Machine
   d. .NET family and the Common Language Runtime
   e. Version control tools.

f. Software life cycle.

Courseware

The range of books on programming is very wide and it is impossible to say which of them are the best to acquire the material of the current course. Nowadays almost all the books dealing with programming in C cover also programming in C++. You can use them, but skip chapters speaking about classes, objects, inheritance, etc. See https://www.taltech.ee/raamatukogu and the Amazon bookshop.

There is a lot of online courses to learn C. See https://www.quora.com/Where-can-I-find-an-online-course-to-learn-C-language. However, do not reckon on them: they may help you but they do not replace the lectures and especially practicing in class.

The participants will be supplied with course material including copies of PowerPoint slides and example code snippets. The complete printed notes, however, are not provided and the students are expected to participate the classwork.

Consultations

In case of problems write viktor.leppikson@liewenthal.ee

Assessment

There are neither prerequisites for examination nor evaluated tests during the lessons. The programming exercises performed in the classroom are not assessed. At the examination each student gets a specification presenting a problem and has to design the algorithm and implement it in C programming language.

Duration of the examination is 4 hours. The student has access to Internet. Usage of textbooks, notes written during lessons, etc. is allowed.

Marks:

- "5" – all the tests checking the correctness of program passed, the code corresponds to the rules of good programming practice and the student gave correct answers to orally presented additional questions.
- "4" – the tests passed but the style of code and / or the answers to additional questions were not satisfactory.
- "3" – the executable file is built but the some of the tests did not pass.
- "2" – the executable is not working (no one test passed).
- "1" – the code is finished but its compiling and / or linking is not possible.
- "0" – code is not finished.