Module Specification

Foundation of Programming [TSI]

Version: 2023-24, v1.0, 07 Aug 2023

Contents

Part 1: Information

Module title: Foundation of Programming [TSI]

Module code: UFCE3U-36-0

Level: Level 3

For implementation from: 2023-24

UWE credit rating: 36

ECTS credit rating: 18

College: College of Arts, Technology and Environment

School: CATE School of Computing and Creative Technologies

Partner institutions: Transport and Telecommunication Institute

Field: Computer Science and Creative Technologies

Module type: Module

Pre-requisites: None

Excluded combinations: None

Co-requisites: None

Continuing professional development: No

Professional, statutory or regulatory body requirements: None

Part 2: Description

Overview: Not applicable

Features: Not applicable

Educational aims: The aim of this module is to enable students to develop algorithmic thinking, learn basic concepts and programming terms, and to become acquainted with high level programming constructs and software development processes.

Outline syllabus: The module covers the basic concepts of programming such as: data types, variables, constants, expressions, branching, loops, arrays, functions and others, using the high-level programming language C++ as an example. Also, some of the topics are devoted to the types and types of software, the classification of programming languages, and a brief overview of the programming paradigms. For comparison, an example of the interpreted Python programming language is considered separately. Modules consist of 2 parts:

Part 1:

•Methodology and paradigms of programming. Main definitions.

•Basic principles of algorithmicising. Forms of algorithm representation. Basic programming constructs

•Introduction to programming languages.

•Composition of a high-level programming language: identifiers, keywords, constants. Composition of C++

•Basic data types. Variables and expressions.

•Operations. Math functions.

•Branching. Conditional and unconditional jumps, multiple choice operators.

•Loop constructs and operators.

•Memory addressing and pointers.

•Data structures - arrays. Single and multidimensional arrays.

•Basic algorithms for working with arrays.

•Char type, working with strings.

•User defined functions & types.

•Working with files.

•Standard Template Library (STL).

•Type of errors in the programming code. Exception handling.

•Pre-processor directives.

•Introduction to the GUI.

Part 2:

•Defining types of Programming Languages and limitations of programming language(s),

•The expressions in programming languages

•The syntax and grammars. Parsing and derivations

•The grammar for expressions. Versions of grammars
•The actions and statements in the programs
•Semantics of Programming Languages
•Objects and data types. Scalar and array types
•Records and sets. Variable attributes. Pointers
•Data type conversions
•Parameter-passing methods
•Scopes and activations. Abstract data types
•Object-oriented programming

Part 3: Teaching and learning methods

Teaching and learning methods: Learning and teaching will be provided to students in two forms: lectures, practical classes and labs. During lectures, theoretical aspects of the course will be provided to students by the teaching staff. Lectures will be supported by presentation published and available to the students on e.tsi.lv under the module section. Also, additional materials, like code examples, text books, publications on the internet, videos etc will be presented in e.tsi.lv. During labs, each student receives an individual task to perform.

During practical classes detailed discussion about specific algorithms is conducted & and algorithms work demonstration is done by the teaching staff.
C++ is studied, as an example of high-level programming language. In addition to learning activities during taught sessions, students are expected to spend time outside of class on independent learning activities. These might include completing assignment tasks, independent reading, practising new skills on personal projects and watching informative videos, completing self-assessment test etc.

Module Learning outcomes: On successful completion of this module students will achieve the following learning outcomes.

MO1 Apply principles and concepts that define the construction and application of a programming language.

MO2 Apply principles of algorithmic thinking,

MO3 Use high level programming language (as example C++, Python) to solve programming problems.

MO4 Use development environments and debuggers for program development and testing.

MO5 Implement algorithms and data processing methods using high level programming language.

Hours to be allocated: 360

Contact hours:

Independent study/self-guided study = 288 hours

Face-to-face learning = 192 hours

Total = 480

Reading list: The reading list for this module can be accessed at readinglists.uwe.ac.uk via the following link https://rl.talis.com/3/uwe/lists/940C82D6-FC89-1CF3-C1F4-9DCC83D9E181.html?lang=en&login=1

Part 4: Assessment

Assessment strategy: AT both first sit and resit, to assess the learning outcomes of this course, several types of activities are provided, which include:
Part 1) Performing practical / laboratory work.
Part 2) A midterm evaluation and examination.
Part 3) Course project. Practical / laboratory work is carried out by students independently.
Part 4) End Examination

The main task is the acquisition of practical skills and the application of theoretical knowledge gained during the classes. Based on the results of the implementation, a report is prepared, which is evaluated by the teacher using the rubrics of assessment and grading scale. In addition to the assessment, the student receives feedback on the work done. Rating and review are published in e.tsi.lv and are available to

students.

Automated tests are used as a formative type of knowledge assessment and are designed for continuous self-assessment of the knowledge acquired by the student. This will allow students to pay attention to material that they have not mastered enough. The course ends with an exam, which is aimed at assessing the theoretical knowledge and practical skills acquired by the student in the process of studying the course.

Students will be required to resit any failed assessment tasks during the resit.

Assessment tasks:

Examination (First Sit)
Description: Mid-term Examination (2 Hours)
Weighting: 30 %
Final assessment: No
Group work: No
Learning outcomes tested: MO1, MO2

Examination (First Sit)
Description: Final Examination (2 Hours)
Weighting: 30 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO1, MO2

Practical Skills Assessment (First Sit)
Description: Set of practical assessments
Weighting: 20 %
Final assessment: No
Group work: No
Learning outcomes tested: MO5

Project (First Sit)
Description: Report of course project
Weighting: 20 %
Final assessment: No
Group work: No
Learning outcomes tested: MO3, MO4

Examination (Resit)
Description: Mid-term examination (2 hours)
Weighting: 30 %
Final assessment: No
Group work: No
Learning outcomes tested: MO1, MO2

Examination (Resit)
Description: Final Exam (2 hours)
Weighting: 30 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO1, MO2

Practical Skills Assessment (Resit)
Description: A set of practical assessments (students are only required to resit the failed work)
Weighting: 20 %
Final assessment: No
Group work: No
Learning outcomes tested: MO5

Project (Resit)
Description: Report on course project
Weighting: 20 %
Final assessment: No
Group work: No

Learning outcomes tested: MO3, MO4

Part 5: Contributes towards

This module contributes towards the following programmes of study:

Computer Science and Software Development {Double Degree} {Foundation} [TSI] BSc (Hons) 2023-24

Computer Science and Software Development {Double Degree} {Foundation} [TSI] BSc (Hons) 2023-24