Module name: Methodology and programming techniques 1


Faculty of: Computer Science, Electronics and Telecommunications

Field of study: Electronics and Telecommunications  Specialty: —

Study level: First-cycle studies  Form and type of study: Full-time studies

Lecture language: English  Profile of education: Academic (A)  Semester: 1

Course homepage: —

Responsible teacher: prof. dr hab. inż. Zieliński Tomasz (tzielin@agh.edu.pl)

Academic teachers: prof. dr hab. inż. Zieliński Tomasz (tzielin@agh.edu.pl)
dr inż. Bułat Jarosław (kwant@agh.edu.pl)
dr inż. Dańda Jacek (danda@agh.edu.pl)
dr inż. Orzechowski Tomasz Marcin (tomeko@agh.edu.pl)
Wszolek Jacek (jwszolek@kt.agh.edu.pl)

Description of learning outcomes for module

<table>
<thead>
<tr>
<th>MLO code</th>
<th>Student after module completion has the knowledge/ knows how to/is able to</th>
<th>Connections with FLO</th>
<th>Method of learning outcomes verification (form of completion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_K001</td>
<td>Understands necessity of self-learning (progress)</td>
<td>ES1A_K01</td>
<td>Project</td>
</tr>
</tbody>
</table>

Skills

| M_U001    | Can solve simple algorithmic tasks                                                                                                   | ES1A_U24             | Project                                                      |
| M_U002    | Can code algorithms in procedural programming language                                                                                | ES1A_U24             | Project                                                      |
| M_U003    | Can use simple and structural data types                                                                                             | ES1A_U24             | Project                                                      |
| M_U004    | Can apply problem decomposition                                                                                                      | ES1A_U24             | Project                                                      |

Knowledge

| M_W001    | Student knows and understands fundamental definitions and concepts of computer science, knows architecture and operation of a simple computer, knows basic tasks of the operating system | ES1A_W07             | Examination                                                  |
| M_W002    | Knows and understands procedure-based programming language                                                                           | ES1A_W07             | Examination                                                  |
FLO matrix in relation to forms of classes

<table>
<thead>
<tr>
<th>MLO code</th>
<th>Student after module completion has the knowledge/ knows how to/is able to</th>
<th>Form of classes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lectures</td>
<td>Auditorium classes</td>
</tr>
<tr>
<td>Social competence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_K001</td>
<td>Understands necessity of self-learning (progress)</td>
<td>+</td>
</tr>
<tr>
<td>Skills</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_U001</td>
<td>Can solve simple algorithmic tasks</td>
<td>-</td>
</tr>
<tr>
<td>M_U002</td>
<td>Can code algorithms in procedural programming language</td>
<td>-</td>
</tr>
<tr>
<td>M_U003</td>
<td>Can use simple and structural data types</td>
<td>-</td>
</tr>
<tr>
<td>M_U004</td>
<td>Can apply problem decomposition</td>
<td>-</td>
</tr>
<tr>
<td>Knowledge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_W001</td>
<td>Student knows and understands fundamental definitions and concepts of computer science, knows architecture and operation of a simple computer, knows basic tasks of the operating system</td>
<td>+</td>
</tr>
<tr>
<td>M_W002</td>
<td>Knows and understands procedure-based programming language</td>
<td>+</td>
</tr>
<tr>
<td>M_W003</td>
<td>Knows and understands fundamental rules of algorithm development</td>
<td>+</td>
</tr>
<tr>
<td>M_W004</td>
<td>Knows and understands classical algorithms</td>
<td>+</td>
</tr>
</tbody>
</table>

Module content
**Lectures**

**LECTURES (28h)**

1. Practical usage of UNIX operating system: OS tasks, OS features (multi-tasking, multi-user), UNIX main commands, shells, scripts.
4-6. Fundamentals of C language programming: simple data types, input/output operations, control instructions. Tables and complex data structures. Procedure, function, sending parameters to procedures and functions, local variables, system stack and its role during calling procedures/functions. Top-down and bottom-up approach in programming.
7. Information, algorithmic task, algorithm, its block diagram. Examples of simple mathematical problems, algorithms for their solving and corresponding programs.
10. Static and dynamic data structures. One and two-directional list, queue, stack, tree. Comparison of different algorithms.
13. Dynamic structures in C++ language on the example of STL „list” and „vector” containers.

**Laboratory classes**

**LABORATORY (28h)**

During laboratory students write scripts of LINUX commands for one of available shells as well as programs in C/C++ and Matlab (next semester). Laboratory exercises aim at clarification and extension of knowledge given during the lectures.

Short introduction into UNIX:
- Introduction to C language programming:
  - Pre-processor. Data types – declaration, initialization, projection. Control instructions.
  - Standard input/output functions. Functions operating on string of characters.
  - Operators and expressions. Tables and pointers. Data structures and unions.
  - Functions. Operations on files.
Details:
1. Introduction to Linux. Main shell commands, regular expressions, system variables, shell variables.
3. Operating upon scripts and pipes in Linux.
4. Introduction to C language. Declaration, definition, program structure.
5. Different loops, input/output operations.
7. Control instructions, simple and complex logical conditions, binary operators.
8. One and multi-dimensional tables. Operating upon tables.
10. Complex data types – structures and unions.
11. Operations on files. Different access methods to data written into files.
13. Practical example: single variables, strings of characters, complex data structures.
14. Functions. Definition, declaration, sending data by value and pointer.

Method of calculating the final grade
1. Positive final evaluation from both, laboratory exercises and examination is required.
2. Mean value is calculated from marks obtained by a student during all final laboratory evaluations and all examination dates.
3. Final mark is calculated using the following formulae:
   if mean>4.75 then OK:=5.0 else
   if mean>4.25 then OK:=4.5 else
   if mean>3.75 then OK:=4.0 else
   if mean>3.25 then OK:=3.5 else OK:=3
4. If positive results from laboratory and examination are obtained during the first attempt (date) and the final mark is lower than 5.0, then the final mark is increased by 0.5.

Prerequisites and additional requirements
Elementary/basic knowledge of mathematics (positional systems, combinatorics, logarithms, etc.)

Recommended literature and teaching resources

Scientific publications of module course instructors related to the topic of the module


**Additional information**

None

**Student workload (ECTS credits balance)**

<table>
<thead>
<tr>
<th>Student activity form</th>
<th>Student workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participation in lectures</td>
<td>28 h</td>
</tr>
<tr>
<td>Realization of independently performed tasks</td>
<td>28 h</td>
</tr>
<tr>
<td>Participation in laboratory classes</td>
<td>28 h</td>
</tr>
<tr>
<td>Preparation of a report, presentation, written work, etc.</td>
<td>50 h</td>
</tr>
<tr>
<td>Summary student workload</td>
<td>134 h</td>
</tr>
<tr>
<td>Module ECTS credits</td>
<td>5 ECTS</td>
</tr>
</tbody>
</table>