Module name: Object-Oriented Programming Language

Academic year: 2016/2017 Code: IES-1-514-s ECTS credits: 4

Faculty of: Computer Science, Electronics and Telecommunications

Field of study: Electronics and Telecommunications Specialty: -

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

Lecture language: English Profile of education: Academic (A) Semester: 5

Course homepage: 

Responsible teacher: dr hab. inż. Cyganek Bogusław (cyganek@agh.edu.pl)

Academic teachers: dr hab. inż. Cyganek Boguslaw (cyganek@agh.edu.pl)
Rydoś Artur (rydosz@agh.edu.pl)
Smolarek Łukasz (smolarek@agh.edu.pl)

Description of learning outcomes for module

<table>
<thead>
<tr>
<th>MLO code</th>
<th>Social competence</th>
<th>Skills</th>
<th>Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_K001</td>
<td>Student is aware of the responsibility for its work, is ready to obey the rules of team work and be liable for common tasks realization.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_K002</td>
<td>Student is aware of the need for continuous skills development.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_K003</td>
<td>Student is able to use of known technologies in companies and commercial projects.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_U001</td>
<td>Student is able to design and implement specified algorithm using basic methods of the object programming technique; Student is able to use STL library resources, template functions and classes in software development process.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_U002</td>
<td>Student is able to construct software in accordance with specified programming standards, creating simple graphic user interfaces and documentation of the software project.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Connections with FLO

Method of learning outcomes verification (form of completion)

Social competence

ES1A_K04 Involvement in teamwork, Test results

ES1A_K01 Test results

ES1A_K05 Test results

Skills

ES1A_U24 Test results

Knowledge

ES1A_U22 Test results
**Module card - Object-Oriented Programming Language**

<table>
<thead>
<tr>
<th>M_W001</th>
<th>Student has knowledge about basics of the scripting object programming languages, available libraries and their application.</th>
<th>ES1A_W21</th>
<th>Test results</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_W002</td>
<td>Student has knowledge about the data representation methods, use of the basic instructions, operators, expressions and functions in C/C++ programming language.</td>
<td>ES1A_W07</td>
<td>Test results</td>
</tr>
<tr>
<td>M_W003</td>
<td>Student has knowledge about classes and objects, inheritance, polymorphism and operator overloading methods, template functions and classes, elements of the STL library, rules of the software development and testing.</td>
<td>ES1A_W17</td>
<td>Test results</td>
</tr>
</tbody>
</table>

**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>
</tbody>
</table>

**Social competence**

| M_K001 | Student is aware of the responsibility for its work, is ready to obey the rules of team work and be liable for common tasks realization. | + | - | + | - | - | - | - | - | - | - |
| M_K002 | Student is aware of the need for continuous skills development. | + | - | + | - | - | - | - | - | - | - |
| M_K003 | Student is able to use of known technologies in companies and commercial projects. | + | - | + | - | - | - | - | - | - | - |

**Skills**

| M_U001 | Student is able to design and implement specified algorithm using basic methods of the object programming technique; Student is able to use STL library resources, template functions and classes in software development process. | + | - | + | - | - | - | - | - | - | - |
| M_U002 | Student is able to construct software in accordance with specified programming standards, creating simple graphic user interfaces and documentation of the software project. | + | - | + | - | - | - | - | - | - | - |
Module card - Object-Oriented Programming Language

| Knowledge | M_W001 | Student has knowledge about basics of the scripting object programming languages, available libraries and their application. | + | - | + | - | - | - | - | - | - | - | - | - |
| Knowledge | M_W002 | Student has knowledge about the data representation methods, use of the basic instructions, operators, expressions and functions in C/C++ programming language. | + | - | + | - | - | - | - | - | - | - | - | - |
| Knowledge | M_W003 | Student has knowledge about classes and objects, inheritance, polymorphism and operator overloading methods, template functions and classes, elements of the STL library, rules of the software development and testing. | + | - | + | - | - | - | - | - | - | - | - | - |

### Module content

**Lectures**

Classes within the module consist of lectures (30 hours) and laboratories (30 hours).

**Lectures**

1. **Basics – Introduction to object programming in C++ – 2 hours**
   - Introduction to programming; Review of object programming languages; Stages of software development; Elements of Unified Modeling Language (UML): use case and activity diagrams; Basics of data representation: constants and variables, basic types of data and rules of use (int, char, long, double); Basics of the C++: instructions if and for, basic arithmetic and logic operators, elementary functions, useful objects (cout, cin, vector<>, string) with examples; Example of using basic language constructions for implementation of simple program.

2. **Basic instructions, operators, expressions and functions – 6 hours**
   - Basic instructions of the C/C++: conditional (if, switch-case), loops (while, for, do-while), throw-try-catch; Review of all groups of operators, hierarchy and associativity law: arithmetic, logic, bit, conditional, others; Pointers: role, syntax, semantics; Arrays; Structures; References; Creating expressions; Functions: role, structure, passing arguments to functions, returns values from functions; Project organization in C/C++: preprocessing, compilation, consolidation, debugging; Examples with real applications;

3. **Object programming – 8 hours**
   - Class and object conception: essential paradigms of object programming (encapsulation, polymorphism, code reusability, code safety), ‘knows’, ‘has-a’ and ‘is-a’ relations; Class anatomy: class components, ‘this’ pointer, component protection, group of construction methods, data access methods, input/output operations, static members, member pointers; Enumerated type; Class design rules with examples; Operators overloading: basic constructions, examples; Inheritance and polymorphism: basic concepts, virtual functions, class hierarchy design, virtual base classes; Types conversions; Special classes: functors, friends; Exceptions handling; Namespaces;
analysis of existing examples.

4. C++ Templates – 4 hours
Concepts of meta-programming, introduction to templates; Template functions: parameters declaration; Template classes: template parameters declaration, template-template parameters, typical templates, unusual templates, virtual functions, template members, default parameters, template class hierarchy. Instantiation of the template classes; Template arguments deduction process; Special template classes: traits and policy, functions for type specification, smart pointers, functors and callback functions; Template constructions design methodology; Examples with real applications.

5. Standard library STL – 6 hours
Introduction to STL: examples of using basic constructions (vector, string), library resources (containers, iterators, algorithms); Basic containers – properties and usage: vector, list, set and multiset, maps and multimaps; Iterators: iterators categories (input, output, forward, bidirectional, random access), iterators adapters; Functional objects of the STL: construction and using of predictors; STL algorithms: non-modifying (elements counting, minimum/maximum, containers searching, range comparing), modifying (copying, transformation, conversion, assigning values to elements), removing containers components, mutating (reversing, rotating, permuting), sorting, numeric algorithms; Special containers: queue, stack; Input/Output operations: basic I/O classes, stream operators overloading, manipulators, formatting, access to files, data streams.

6. Elements of the software design. Advanced constructions – 2 hours
Advanced UML; Software design: stages of development, documentation; Programming standards; Library construction; Software testing: types of errors, tests constructing, programming by contract, methods of code profiling and debugging.

7. Others object languages – 2 hours
Introduction to Python: scripting languages specification, basic constructions, modes of language usage, differences related to C++, object programming, introduction to Python libraries; Examples with real applications.

Laboratory classes

Laboratories
1. Elaboration and implementation of quadratic trinomial roots computing and representation problem – 4 hours – Work in Linux environment, documentation of the project, software implementation, compilation, consolidation and debugging methods, methods of errors removing.
2. Elaboration and implementation of currency calculator – 4 hours – Work in Visual C++ environment, exploration of exchange rate application, internet connection implementation, graphic user interface project (GUI), using of MFC and/or FLTK.
3. Elaboration and implementation of class register – 4 hours – Implementation of simple data base with adding/removing/sorting functions; input/output and files operations.
4. Elaboration and implementation of complex numbers class – 4 hours – Implementations of the complex numbers class, operators overloading, streaming operations with files, type conversion operators. Design with programming-by-contract methodology.
5. Elaboration and implementation of matrix representation class, handle-body and proxy patterns – 4 hours – Implementation of specified project patterns in matrix class application. Examples of using in numerical algebra applications.
6. Elaboration and implementation of computer graphic application using OpenGL – 4 hours – Introduction to OpenGL, design of simple application using resources of the
library, work with class hierarchy and virtual functions. Examples of using in graphic applications.

7. Elaboration and implementation of a simple computer game – 4 hours – Design of application in accordance with programming patterns rules, UML problem modelling, documentation of the project.

8. Student’s own project – 2 hours – Design and implementation of the own application in Python; Using of the Python libraries, project management and project group management.

Project classes

- Method of calculating the final grade

1. To obtain a course credit a positive grade of the laboratory and lecture test should be obtained.
2. It is possible to obtain a grade by presentation about selected issue during a lecture.
3. The arithmetic average of the all tests is calculating (retake tests with weight 2).
4. Final grade is obtained using the following algorithm:
   - if av>4.75 then OK:=5.0 else
   - if av>4.25 then OK:=4.5 else
   - if av>3.75 then OK:=4.0 else
   - if av>3.25 then OK:=3.5 else
   - if av>=3.0 then OK:=3.0

Prerequisites and additional requirements

- Basics of programming methods and techniques.

Recommended literature and teaching resources

2. Stroustrup B. The C++ Programming Language, 2000,
3. Vandervoorde D., Josuttis N.M. C++ Templates. Addison Wesley, 2003,
5. Lippman S. Essential C++ . Addison Wesley, 2005,

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

Additional scientific publications not specified

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>24 h</td>
</tr>
<tr>
<td>Participation in laboratory classes</td>
<td>28 h</td>
</tr>
<tr>
<td>Preparation for classes</td>
<td>20 h</td>
</tr>
<tr>
<td>Summary student workload</td>
<td>100 h</td>
</tr>
<tr>
<td>Module ECTS credits</td>
<td>4 ECTS</td>
</tr>
</tbody>
</table>