## STUDY MODULE DESCRIPTION FORM

| Name of the module/subject **Languages and paradigms of programming** | | Code **1010334521010334960** |
|---|---|---|

| Field of study **Information Engineering** | Profile of study (general academic, practical) **(brak)** | Year /Semester **1 / 2** |
|---|---|---|

| Elective path/specialty **-** | Subject offered in: **Polish** | Course (compulsory, elective) **obligatory** |
|---|---|---|

| Cycle of study: **First-cycle studies** | Form of study (full-time,part-time) **part-time** |
|---|---|

| No. of hours | | | | No. of credits **6** |
|---|---|---|---|---|
| Lecture: **20** | Classes: **-** | Laboratory: **20** | Project/seminars: **-** | |

| Status of the course in the study program (Basic, major, other) **(brak)** | (university-wide, from another field) **(brak)** |
|---|---|

| Education areas and fields of science and art **technical sciences** | ECTS distribution (number and %**)** **6   100%** |
|---|---|

**Responsible for subject / lecturer:**

PH.D.Eng. Beata Jankowska
email: beata.jankowska@put.poznan.pl
tel. +48 61 665 37 24
Wydział Elektryczny
ul. Piotrowo 3A 60-965 Poznań

**Prerequisites  in terms of knowledge, skills and social competencies:**

| 1 | **Knowledge** | Student has an elementary mathematical knowledge, including algebra, analysis, logics, theory of probability, elements of discrete maths and applied maths. |
|---|---|---|
| 2 | **Skills** | Student can: use programming environments and platforms for coding, running and testing simple programs in imperative laguages; prepare and show a short presentation of the results of an executed engineering task. |
| 3 | **Social competencies** | Student realises the responsibility for his/her work done individually or in a team; also, he/she is ready to accept the rules of group work. |

**Assumptions and objectives of the course:**

the understanding of different programming styles (and languages); a mastery of choosing an appropriate style and language to solve a specific problem; a particular competence to design and implement various algorithms in object-oriented style and language; the clever using of constructs that are typical of object-oriented language C++.

### Study outcomes and reference to the educational results for a field of study

**Knowledge:**

1. Student has an organized and theoretically grounded knowledge in the fields of: basic algorithms and their analysing, techniques of designing algorithms, abstract data structures and their implementation, hard computational problems. - [K_W04]

2. Student has an organized and theoretically grounded knowledge in the fields of: basic programming constructs, algorithms implementation, paradigms and styles of programming, methods of verifying program correctness, formal languages and compilers, programming platforms. - [K_W05]

**Skills:**

1. Student can design algorithms (with the use of basic algorithmic techniques) and estimate their complexity.  - [K_U09]

2. Student can use programming environments and platforms for coding, running and testing simple programs in imperative, object-oriented and declarative languages.  - [K_U10]

3. Student can prepare the documentation of an executed engineering task, including the discussion of the obtained results.  - [K_U03]

**Social competencies:**

1. Student realises the importance of: executing projects precisely, preserving notational standards and linguistic correctness, and completing works on time. - [K_K07]

2. Student realises the importance and understands non-technical aspects and effects of computer engineer - [K_K02]

| **Assessment  methods of study outcomes** |
| --- |
| Lecture: written exam. |
| Labs: rating student's results of input tests, internal tests, programming activity, and his/her solution of an optional project task (implementation in C++, written documentation). |
| More than 50% points are necessary for passing the exam and labs. |

| **Course description** |
| --- |
| Lectures: |
| Different styles of programming and their classification. Basic paradigms of object-oriented programming (encapsulation, inheritance, polymorphism) and their implementation in C++ language. Implementation of input-output instructions in C++. Handling errors and exceptions in object-oriented languages. Overloading functions and operators. Dynamic storage management in object-oriented languages and systems. Rules of multi-thread programming. |
| Labs: |
| Designing and implementing algorithms in C++ language. |

| **Basic bibliography:** |
| --- |
|  |

| **Additional bibliography:** |
| --- |
|  |

### Result of average student's workload

| Activity | Time (working hours) |
| --- | --- |
| 1. Lectures | 20 |
| 2. Labs | 20 |
| 3. Final exam and consultations | 10 |
| 4. Preparing for labs | 30 |
| 5. Preparing for internal tests | 25 |
| 6. Preparing for the final exam | 45 |

### Student's workload

| Source of workload | hours | ECTS |
| --- | --- | --- |
| Total workload | 150 | 6 |
| Contact hours | 75 | 3 |
| Practical activities | 75 | 3 |