



This learning unit is not open to incoming exchange students!

| | |
|---------------------|--|
| Teacher(s) | Van Roy Peter ; |
| Language : | French |
| Place of the course | Charleroi |
| Prerequisites | <p>This course assumes that the student already masters basic programming skills targeted by courses LINFO1101 or LEPL1401 and concepts on algorithmics and simple data structures covered by course LEPL1402.</p> <p><i>The prerequisite(s) for this Teaching Unit (Unité d'enseignement – UE) for the programmes/courses that offer this Teaching Unit are specified at the end of this sheet.</i></p> <p><i>The prerequisite(s) for this Teaching Unit (Unité d'enseignement – UE) for the programmes/courses that offer this Teaching Unit are specified at the end of this sheet.</i></p> |
| Main themes | <ul style="list-style-type: none"> • Programming paradigms: functional programming, object-oriented programming and declarative dataflow programming; • Formal semantics and reasoning techniques on programs; • Core language and abstract machine; • Data Abstractions and Object-Oriented Modeling; • Recursive algorithms and programming with invariant using linear and tree data structures; • Analysis of the temporal complexity of an algorithm and the spatial complexity of a data structure; • Non-determinism, scheduling and equity; • Implementation of medium complexity programs with a focus on test and program validation methods. |
| Learning outcomes | <p>At the end of this learning unit, the student is able to :</p> <p>Regarding the learning outcomes of the program of Bachelor in Engineering, this course contributes to the development and the acquisition of the following learning outcomes:</p> <ul style="list-style-type: none"> • LO 1.1, 1.2 • LO 2.3, 2.4, 2.5, 2.6, 2.7 • LO 4.2, 4.3, 4.4 <p>Given the learning outcomes of the "Bachelor in Computer Science" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes:</p> <ul style="list-style-type: none"> • S1.I2., S1.I3, S1.I5 • S2.2., S2.3, S2.4 • S5.3, S5.4., S5.5. <p>Students completing successfully this course will be able to</p> <ul style="list-style-type: none"> • specify the problems, divide them into their basic steps, and design algorithms and abstractions to solve them; • choose the right programming paradigm and write a program in this paradigm to solve a problem; • use formal semantics to explain the accuracy of the program; • write small concurrent programs in the deterministic dataflow paradigm. • think using abstractions (reason correctly on a system that includes several layers of abstractions, and define new abstractions to simplify the resolution of a problem) |
| Evaluation methods | <ul style="list-style-type: none"> • Dispensatory test 25% (around week 7) • Project 25% • Final exam 50% (or 75% if redoing test part) <p>The project is mandatory and is done during the quadrimester. It can only be done only once and it counts for the whole academic year. The optional dispensatory test and the final exam may be done in auditorium or online, depending on university requirements. The teacher reserves the right to give an oral examination to certain students.</p> |
| Teaching methods | <ul style="list-style-type: none"> • Weekly lectures (in auditorium or online, according to university requirements) • Practical lab sessions in the computer room every week, to solve simplified problems using concepts explained during the lectures. • One major design and programming project to apply these concepts to a more complex application |

| | |
|------------------------------------|--|
| <p>Content</p> | <p>The goal of this course is to broaden and deepen the programming knowledge acquired in preceding courses. The course treats the following subjects:</p> <ul style="list-style-type: none"> • The course gives a uniform framework for all programming concepts, organised as programming paradigms. • The course gives a formal semantics and reasoning techniques for all paradigms. • The course gives an introduction to lambda calculus as foundation of functional programming and higher-order programming. • Higher-order programming is used as organizing principle for the construction of procedural abstractions. • Concurrent programming is presented in two forms, namely deterministic dataflow and message-passing concurrency. • Data abstraction is presented in its general form and with its two principal derived forms, namely object-oriented programming and abstract data types. • Symbolic programming and algorithm design principles are used throughout the course. • Five important programming paradigms are presented in the course: functional programming, object-oriented programming, deterministic dataflow programming, actor dataflow, and active object (multi-agent) programming. <p>Examples of practical applications are given for all concepts and all paradigms.</p> |
| <p>Faculty or entity in charge</p> | <p>SINC</p> |

| Programmes containing this learning unit (UE) | | | | |
|--|-------------------------|---------|---------------------------|---|
| Program title | Acronym | Credits | Prerequisite | Learning outcomes |
| Bachelor in Computer Science | SINC1BA | 5 | LSINC1101 |  |