# linfo2132

*2022*

# Languages and translators

| 6.00 credits | 30.0 h + 30.0 h | Q2 |
|---|---|---|

| | |
|---|---|
| Teacher(s) | Sadre Ramin ; |
| Language : | English<br>> French-friendly |
| Place of the course | Louvain-la-Neuve |
| Main themes | • Methods to analyze context-free languages, upstream and downstream methods<br>• Generators of lexical analyzers and parsers<br>• Statistical semantics and attributed grammars<br>• Methods to translate a source code in a target code, and generation of target code<br>• Machine virtuelle et byte-code (JVM)<br>• Garbage Collection et gestion mémoire<br>• Domain Specific Languages (DSL) |
| Learning outcomes | **At the end of this learning unit, the student is able to :**<br>Given the learning outcomes of the "Master in Computer Science and Engineering" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes:<br>• INFO1.1-3<br>• INFO2.2-4<br>• INFO5.2, INFO5.4, INFO5.5<br>• INFO6.1, INFO6.4<br>Given the learning outcomes of the "Master [120] in Computer Science" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes:<br>• SINF1.M2<br>• SINF2.2-4<br>• SINF5.2, SINF5.4, SINF5.5<br>• SINF6.1, SINF6.4<br>Given the learning outcomes of the "Master [60] in Computer Science" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes:<br>1 • 1SINF1.M2<br>• 1SINF2.2-4<br>• 1SINF5.2, 1SINF5.4, 1SINF5.5<br>• 1SINF6.1, 1SINF6.4<br>Students completing successfully this course will be able to<br>• explain in a practical way the structure of compilers dealing with algorithmic languages<br>• design and implement a compiler for a practical language which solves a interesting problem<br>• show the interest of compiling techniques in problem resolving<br>Students will have developed skills and operational methodology. In particular, they have developed their ability to<br>• treat rigorously a problem, justifying and validating each step of a project to be able to rely on it to implement the following one<br>• explain in practical terms how a source code (Java) is finally translated into byte-code.<br>• explain the enforcement mechanisms byte code by JVM<br>• explain memory management during the execution of a program<br>• explain how garbage collection mechanisms |

| | |
|---|---|
| Evaluation methods | June session:<br>The evaluation consists of two components: The project (done in groups) accounts for 60% of the course's final grade. A written exam accounts for 40%.<br>August session:<br>If the student did not successfully pass the course in the first session (i.e., they did not obtained at least 10/20 for the final grade), the student is allowed to redo those components (project or exam or both) of the evaluation for which they did not obtain at least 50% of the respective points. They will keep the points for the component that they passed (if any). The same weights as in the June session are applied for the calculation of the final grade.<br>Both sessions: The professor may request a student to go through an additional oral exam as a complement of the exam and/or of the project activities, in cases including, but not limited to, technical issues, or suspicion of irregularities. |
| Teaching methods | The course consists in a series of pre-recorded video lectures, as well as weekly or bi-weekly consolidation or lecture sessions. There will also be a couple of lab sessions to best prepare the students for the project.<br>During the semester, students will have to complete the project, which consists of extending a basic programming language interpreter/compiler with new paradigms. |
| Content | The course presents the theory and practice of programming language implementation, as well as compiler architecture. We will review the standard components of a compiler, from front-end (parsing, lexical analysis) to back-end (machine code generation, or interpreters), also touching on static semantics and type systems. Ultimately, students should be able to understand the ins and outs of the various programming language implementation techniques in use today.<br>During the course, the students will implement their own programming language. |
| Inline resources | Teams and/or Moodle |
| Bibliography | Ouvrage(s) recommandé(s) :<br>• Crafting Interpreters, Bob Nystrom (https://craftinginterpreters.com/)<br>• How To Create Pragmatic Lightweight Languages, Federico Tomassetti<br>• Introduction to Compiler Construction in a Java World, Bill Campbell, Swamilyer, Bahar Akbal-Deliba<br>• Modern Compiler Implementation in Jaav, Andrew W. Appel |
| Other infos | Background :<br>• LINGI1122 : Program design<br>• LSINF1121 : High-level programming language, algorithmics and data structures<br>• LINGI1101 : Logic and discrete structures |
| Faculty or entity in charge | INFO |

## Programmes containing this learning unit (UE)

| Program title | Acronym | Credits | Prerequisite | Learning outcomes |
|---|---|---|---|---|
| Master [120] in Computer Science and Engineering | INFO2M | 6 |  | 🔍 |
| Master [120] in Computer Science | SINF2M | 6 |  | 🔍 |
| Master [60] in Computer Science | SINF2M1 | 6 |  | 🔍 |