

Due to the COVID-19 crisis, the information below is subject to change, in particular that concerning the teaching mode (presential, distance or in a comodal or hybrid format).



6 credits

30.0 h + 30.0 h

Q2

Teacher(s)	Laurent Nicolas ;
Language :	English
Place of the course	Louvain-la-Neuve
Main themes	<ul style="list-style-type: none"> <li>• Methods to analyze context-free languages, upstream and downstream methods</li> <li>• Generators of lexical analyzers and parsers</li> <li>• Statistical semantics and attributed grammars</li> <li>• Methods to translate a source code in a target code, and generation of target code</li> <li>• Machine virtuelle et byte-code (JVM)</li> <li>• Garbage Collection et gestion mémoire</li> <li>• Domain Specific Languages (DSL)</li> </ul>
Aims	<p>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:</p> <ul style="list-style-type: none"> <li>• INFO1.1-3</li> <li>• INFO2.2-4</li> <li>• INFO5.2, INFO5.4, INFO5.5</li> <li>• INFO6.1, INFO6.4</li> </ul> <p>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:</p> <ul style="list-style-type: none"> <li>• SINF1.M2</li> <li>• SINF2.2-4</li> <li>• SINF5.2, SINF5.4, SINF5.5</li> <li>• SINF6.1, SINF6.4</li> </ul> <p>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:</p> <p>1</p> <ul style="list-style-type: none"> <li>• 1SINF1.M2</li> <li>• 1SINF2.2-4</li> <li>• 1SINF5.2, 1SINF5.4, 1SINF5.5</li> <li>• 1SINF6.1, 1SINF6.4</li> </ul> <p>Students completing successfully this course will be able to</p> <ul style="list-style-type: none"> <li>• explain in a practical way the structure of compilers dealing with algorithmic languages</li> <li>• design and implement a compiler for a practical language which solves a interesting problem</li> <li>• show the interest of compiling techniques in problem resolving</li> </ul> <p>Students will have developed skills and operational methodology. In particular, they have developed their ability to</p> <ul style="list-style-type: none"> <li>• treat rigorously a problem, justifying and validating each step of a project to be able to rely on it to implement the following one</li> <li>• explain in practical terms how a source code (Java) is finally translated into byte-code.</li> <li>• explain the enforcement mechanisms byte code by JVM</li> <li>• explain memory management during the execution of a program</li> <li>• explain how garbage collection mechanisms</li> </ul> <p>----</p> <p><i>The contribution of this Teaching Unit to the development and command of the skills and learning outcomes of the programme(s) can be accessed at the end of this sheet, in the section entitled "Programmes/courses offering this Teaching Unit".</i></p>

Evaluation methods	<p><b>Due to the COVID-19 crisis, the information in this section is particularly likely to change.</b></p> <p>The major part of the course's evaluation will consist of the aforementioned language implementation project. A written exam is also under consideration, depending on circumstances.</p> <p>The modalities for the second session will depend on the number of students involved. It may consist of a written exam, oral exam, or a second project.</p> <p>In case of doubt about the personal work of a student during the project, or in case of a technical issues, the instructor reserves the right to submit students to an additional oral exam, which complements or replaces the grade obtained during the project(s) and exam(s).</p>
Teaching methods	<p><b>Due to the COVID-19 crisis, the information in this section is particularly likely to change.</b></p> <p>The course consists of a series of lectures, as well as practical lab session to introduce technologies, and/or office hours with teaching assistants. During the teaching period, the students will have to implement their own programming language.</p> <p>Face-to-face classes as well as remote teaching or a mix of the two methods are possible, depending on the circumstances.</p>
Content	<p>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.</p> <p>During the course, the students will implement their own programming language.</p>
Inline resources	<p><a href="http://moodleucl.uclouvain.be/course/view.php?id=5423">http://moodleucl.uclouvain.be/course/view.php?id=5423</a></p>
Bibliography	<p>Ouvrage(s) recommandé(s) :</p> <ul style="list-style-type: none"> <li>• Crafting Interpreters, Bob Nystrom (<a href="https://craftinginterpreters.com/">https://craftinginterpreters.com/</a>)</li> <li>• How To Create Pragmatic Lightweight Languages, Federico Tomassetti</li> <li>• Introduction to Compiler Construction in a Java World, Bill Campbell, Swamilyer, Bahar Akbal-Deliba</li> <li>• Modern Compiler Implementation in Jaav, Andrew W. Appel</li> </ul>
Other infos	<p>Background :</p> <ul style="list-style-type: none"> <li>• LINGI1122 : Program design</li> <li>• LSINF1121 : High-level programming language, algorithmics and data structures</li> <li>• LINGI1101 : Logic and discrete structures</li> </ul>
Faculty or entity in charge	INFO

<b>Programmes containing this learning unit (UE)</b>				
Program title	Acronym	Credits	Prerequisite	Aims
Master [120] in Computer Science and Engineering	INFO2M	6		
Master [120] in Computer Science	SINF2M	6		
Master [60] in Computer Science	SINF2M1	6		