2026 年度シラバス

科目分類/Subject Categories			
学部等/Faculty	/工芸科学部 : /School of Science and Technology	今年度開講/Availability	/有 : /Available
学域等/Field	/設計工学域 : /Academic Field of Engineering Design	年次/Year	/3年次 : /3rd Year
課程等/Program	/情報工学課程・課程専門科目 : /Specialized Subjects for Undergraduate Program of Information Science	学期/Semester	/後学期 : /Second term
分類/Category	/ : /	曜日時限/Day & Period	/金 5 : /Fri.5

科目情報/Course Information				
時間割番号 /Timetable Number	12221201			
科目番号 /Course Number	12260093			
単位数/Credits	2			
授業形態 /Course Type	講義・演習 : Lecture/Practicum			
クラス/Class				
授業科目名 /Course Title	言語処理プログラミング : Programming Language Processing			
担当教員名 / Instructor(s)	/水野　修 : MIZUNO Osamu			
その他/Other	インターンシップ実施科目 Internship	国際科学技術コース提供科目 IGP	PBL 実施科目 Project Based Learning	DX 活用科目 ICT Usage in Learning
				○
	実務経験のある教員による科目 Practical Teacher			
科目ナンバリング /Numbering Code				

授業の目的・概要 Objectives and Outline of the Course	
日	小さなプログラミング言語のコンパイラを作成する。これを通して、コンパイラと言語処理手法の実際を学ぶ。また、数千行の大きさのプログラムを作成することにより、比較的大きな規模のプログラミングの経験を得る。
英	Each student implements a compiler for a tiny language and learns the practice of text processing, especially compiler. Moreover, he/she gets the experiences of large scale programming.

学習の到達目標 Learning Objectives	
日	字句解析法を理解し、実際に字句解析系を作成できる。 LL(1)構文解析法を理解し、実際に構文解析系を作成できる。 名前表の役割を理解し、実際に型検査系を作成できる。 コンパイラの構造を理解し、簡単なコンパイラが作成できる。 比較的大きなソフトウェアの作成経験を得て、大きなソフトウェアの設計、作成、テスト、デバッグの問題点を体感する。
英	Able to implement a scanner of a simple compiler. Able to implement a syntax analyzer of a simple compiler based on LL (1) syntax analysis method. Able to implement a type checker of a simple compiler. Able to implement a simple compiler including code generator. To experience large software development and know the problems of software design, implementation, test, debug, and so on.

学習目標の達成度の評価基準 / Fulfillment of Course Goals（JABEE 関連科目のみ）	
日	
英	

授業計画項目 Course Plan			
No.		項目 Topics	内容 Content
1	日	字句解析(1)	［課題 1］テキスト処理技法の基本を学習する。それを利用して字句解析系を作成し、字句の出現頻度表を生成するプログラムを作成する。そのため，まず字句の分析と字句解析系の設計を行う。
	英	Lexical analysis(1)	analyzing tokens and designing the scanner to implement Token-Counter
2	日	字句解析(2)	［課題 1］テキスト処理技法の基本を学習する。それを利用して字句解析系を作成し、字句の出現頻度表を生成するプログラムを作成する。 そのため、次に字句解析系のプログラミングを行う。
	英	Lexical analysis(2)	coding the scanner to implement Token-Counter
3	日	字句解析(3)	［課題 1］テキスト処理技法の基本を学習する。それを利用して字句解析系を作成し、字句の出現頻度表を生成するプログラムを作成する。 そのため、最後に字句解析系のテストを行う。
	英	Lexical analysis(3)	testing the scanner to implement Token-Counter
4	日	構文解析(1)	［課題 2］LL(1)構文解析手法を学習する。それを利用して構文解析系を作成し、構文が正しいかどうかを調べ、プリティプリントするプログラムを作成する。そのため、まず構文の分析を行う.
	英	Syntax analysis(1)	analyzing the syntax to implement Pretty-Printer using LL (1) syntax analysis method
5	日	構文解析(2)	［課題 2］LL(1)構文解析手法を学習する。それを利用して構文解析系を作成し、構文が正しいかどうかを調べ、プリティプリントするプログラムを作成する。そのため、次に構文解析系の設計を行う。
	英	Syntax analysis(2)	Syntax analysis(2)
6	日	構文解析(3)	［課題 2］LL(1)構文解析手法を学習する。それを利用して構文解析系を作成し、構文が正しいかどうかを調べ、プリティプリントするプログラムを作成する。そのため、3 番目に構文解析系のプログラミングを行う。
	英	Syntax analysis(3)	coding the parser to implement Pretty-Printer using LL (1) syntax analysis method
7	日	構文解析(4)	［課題 2］LL(1)構文解析手法を学習する。それを利用して構文解析系を作成し、構文が正しいかどうかを調べ、プリティプリントするプログラムを作成する。そのため，最後に構文解析系のテストを行う。
	英	Syntax analysis(4)	testing the parser to implement Pretty-Printer using LL (1) syntax analysis method
8	日	型検査(1)	［課題 3］ポインタを利用したデータ構造作成の基本を学習する。プログラミング言語の制約を満たしているかどうかを調べ、クロスリファレンス表を出力するプログラムを作成する。そのため、まず型システムの分析を行う。
	英	Type inspection(1)	analyzing the type system to implement Cross-Referencer using symbol table
9	日	型検査(2)	［課題 3］ポインタを利用したデータ構造作成の基本を学習する。プログラミング言語の制約を満たしているかどうかを調べ、クロスリファレンス表を出力するプログラムを作成する。そのため、次に型検査系の設計を行う。
	英	Type inspection(2)	designing the type checker to implement Cross-Referencer using symbol table
10	日	型検査(3)	［課題 3］ポインタを利用したデータ構造作成の基本を学習する。プログラミング言語の制約を満たしているかどうかを調べ、クロスリファレンス表を出力するプログラムを作成する。そのため、3 番目に型検査系のプログラミングを行う。
	英	Type inspection(3)	coding the type checker to implement Cross-Referencer using symbol table
11	日	型検査(4)	［課題 3］ポインタを利用したデータ構造作成の基本を学習する。プログラミング言語の制約を満たしているかどうかを調べ、クロスリファレンス表を出力するプログラムを作成する。そのため、最後に型検査系のテストを行う。
	英	Type inspection(4)	testing the type checker to implement Cross-Referencer using symbol table
12	日	コード生成(1)	［課題 4］以上の課題の成果にコード生成部を加えてコンパイラを完成させる。そのため、まず目的コードの分析を行う。

	英	Code generation(1)	analyzing the object language to implement a small compiler
13	日	コード生成(2)	［課題 4］以上の課題の成果にコード生成部を加えてコンパイラを完成させる。そのため、次にコード生成部の設計を行う。
	英	Code generation(2)	designing the code generator to implement a small compiler
14	日	コード生成(3)	［課題 4］以上の課題の成果にコード生成部を加えてコンパイラを完成させる。そのため、3 番目にコード生成部のプログラミングを行う。
	英	Code generation(3)	coding the code generator to implement a small compiler
15	日	コード生成(4)	［課題 4］以上の課題の成果にコード生成部を加えてコンパイラを完成させる。そのため、最後にコード生成部のテストを行う。
	英	Code generation(49	testing the code generator to implement a small compiler

履修条件 Prerequisite(s)	
日	「プログラミング I,II」及び「ソフトウェア演習 I,II」を履修し、C 言語の機能を深く理解していること。この演習は、科目「コンパイラ」と対応しているので、「コンパイラ」を履修していることが望ましい。
英	Courses students need to have taken courses "Programming I," "Programming II," and "data structures and algorithms" before.　Students had better to take "Compiler."

授業時間外学習（予習・復習等） Required study time, Preparation and review	
日	時間内の演習だけではできない。平均して毎週 5 時間程度の時間外演習をすることが必要である。その他、レポート作成のための時間も必要である。
英	About 5 hours or more are needed for the programming each week. About 4 hours are needed for writing documents for each of 4 implementations.

教科書／参考書 Textbooks/Reference Books	
日	教科書はなし。プリントやサンプルプログラムを配布。参考書「コンパイラ，第 2 版」(辻野嘉宏著，オーム社)。
英	No text book.　Printed materials will be distributed. Reference book : Y.Tsujino: "Compiler, Second Edition" Ohmsha (2019).

成績評価の方法及び基準 Grading Policy	
日	各課題レポート各 12.5%、課題 1 から課題 3 のプログラムの評価各 10%、最終課題のコンパイラの評価 20%により評価する。 レポートは配付資料で要求された事項を満たしているかで評価する。プログラムはその完成度で評価する。合計点が 60 点以上を合格とする。 出席を義務づけた演習日を欠席した場合には減点する。 レポートやプログラムを一つでも提出しない学生は成績評価対象外となることがある。
英	All programs which each student will develop are tested and evaluated The evaluation of this course will be conducted based on the evaluation of programs (50%) and the evaluation of the required documents (50%). The students whose result of the evaluation is 60 points or higher will pass.

留意事項等 Point to consider	
日	わからないことがあれば，自分から調べたり質問したりして能動的に演習する必要がある。 他人が作成したレポートを自分が作成したとして提出しないこと. 他人が作成したソースコードをコピーして利用することは禁止する.
英	If you have questions, investigate by yourself or ask professor for your question. Do not submit a report that the other one write. Do not copy and paste other one's source code into your code.