Types de Données Abstraits & Base de la Programmation Orienté Objet ***Abstract Data Type & Base of Object-Oriented Programming***			
Code cours *Course code*: **TAB**		**Crédits ECTS** *ECTS Credits:* **2,5**	
Département *Department*	: IA	**Cours** *Lectures*	: 20H
Coordonnateurs *Lecturers*	: M. RICHARD	**T.D.** *Tutorials*	: 20H
		T.P. *Laboratory sessions*	:
Période *Year of study*	: 3ième année *3rd year*	**Projet** *Project*	:
		Non encadré *Homework*	:
Semestre *Semester*	: 5ième semester – A *5th semester - A*	**Horaire global** *Total hours*	: 40H
Evaluation *Assessment method(s)*	: 1 examen écrit, *1 written exam*		
Langue d'instruction *Language of instruction*	: Français *French*		
Type de cours *Type of course*	: Obligatoire *Compulsory*		
Niveau *Level of course*	: Avancé *graduate*		

Compétences attendues :

À l'issue de ce module, l'étudiant possédera :

- une maitrise avancée du langage de programmation C (modularité, gestion mémoire, …)
- la capacité d'implémenter et d'utiliser des structures de données dynamiques linéaires et hiérarchiques (Pile, File, Deque, Arbre, Graphe)
- une bonne connaissance des concepts de base de la programmation orientée objet (Classes, Associations, Encapsulation, Abstraction, …)

Pré-requis : Une pratique d'un langage de programmation procédural.

Contenu :

Ce module est composé de trois grandes partie permettant l'acquisition des compétences ci-dessus :

1. Langage C :
   Ce chapitre permettra l'acquisition de compétences avancées dans l'utilisation du langage C, en particulier concernant la modularité, la gestion mémoire et les bonnes pratiques pour la mise en place de d'algorithmes robustes
2. Types Abstraits de Données linéaires et hiérarchiques :
   Nous étudierons ici les grandes familles de types abstraits de données dans un premier temps linéaire (Pile, file, listes, queue, deque, …) puis hiérarchique (arbre, arbre binaire, arbre binaire de recherche, introduction aux graphes). L'étude des algorithmes permettra une initiation à la complexité en espace et en temps de ces derniers.
3. Des types de données abstraits au paradigme objet :
   De manière très naturelle nous découvrirons, à partir des connaissances acquises dans la partie précédente, le paradigme de la programmation objet et les éléments de base de ce type de programmation (classes, relation d'association, d'agrégation et composition, encapsulation, abstraction et héritage) permettant d'améliorer les propriétés d'un code selon différents critères définis en génie logiciel.

Toutes ces notions seront largement mises en œuvre au cours des différentes séances de travaux dirigés.

Bibliographie :

Expected competencies:

At the end of this module, the student will have :

- an advanced knowledge of the C programming language (modularity, memory management, ...)
- the ability to implement and use dynamic linear and hierarchical data structures (Stack, File, Deque, Tree, Graph)
- a good knowledge of the basic concepts of object-oriented programming (Classes, Associations, Encapsulation, Abstraction, ...)

Prerequisites: A practice of a procedural programming language.

Content:

This module is composed of three main parts allowing the acquisition of the above skills:

1. C language :

This chapter will allow the acquisition of advanced skills in the use of the C language, in particular concerning modularity, memory management and good practices for the implementation of robust algorithms

2. Linear and hierarchical Abstract Data Types:

We will study here the main families of abstract data types in a first step linear (stack, queue, lists, tail, deque, ...) then hierarchical (tree, binary tree, binary search tree, introduction to graphs). The study of algorithms will allow an introduction to their space and time complexity.

3. From abstract data types to the object paradigm:

In a very natural way we will discover, from the knowledge acquired in the previous part, the object programming paradigm and the basic elements of this type of programming (classes, association, aggregation and composition relationships, encapsulation, abstraction and inheritance) allowing to improve the properties of a code according to different criteria defined in software engineering.

All these notions will be widely implemented during the different tutorial sessions.

Recommended reading: