# Tiger

http://tiger.lrde.epita.fr

## Enseignement de techniques de programmation par la construction de compilateur

### Le projet et son rôle dans la scolarité

Étape importante dans le cursus de tronc commun de l'EPITA, le projet Tiger vise à enseigner aux étudiants des notions d'informatique ayant trait aux langages de programmation, à travers l'écriture d'un compilateur pour un langage « jouet », Tiger.

Tiger a été conçu par Andrew W. Appel, et est utilisé comme support dans ses livres « Modern Compiler Implementation in C, Java, ML » [1].
Il s'agit d'un langage simple mais complet, dérivé de Pascal et adoptant une syntaxe claire à la ML. La variante du langage utilisé a l'EPITA diffère de celle d'Appel et dispose d'extensions originales.

Écrire un compilateur n'est pas l'objectif premier du projet : il s'agit surtout d'un excellent support pour l'enseignement de concepts et de savoir-faire fondamentaux en informatique [2] : découverte de paradigmes de langages de programmation, apprentissage du C++ et de designs patterns, manipulation d'outils de développement, gestion d'un gros projet sur le long terme au sein d'une équipe de quatre étudiants, etc. Le projet fonctionne par « tranches » d'une à trois semaines, pour lesquelles du code « à trous » est fourni.

### Bibliographie


[1] Andrew W. Appel. Modern Compiler Implementation in C, Java, ML.
Cambridge University Press. 1998.

[2] Akim Demaille. Making Compiler Construction Projects Relevant to Core Curriculums.
10th Annual Conference on Innovation and Technology in Computer Science Education
http://www.iticse05.unl.pt/
Universidade Nova de Lisboa, Portugal
June 27-29, 2005.


### Étapes du projet Tiger

**TC-0 : construction d'un analyseur lexical et syntaxique**

- écrire/débugger un analyseur lexical avec Flex
- **écrire/débugger un analyseur syntaxique LALR(1) avec Bison**
- utiliser quelques fonctionnalités du C++

**TC-1 : complétion du scanner/parser et empaquetage du projet**

- équiper le projet d'un systèmes de tâches pourvues de dépendances
- apprendre de nouveaux outils (Autoconf, Automake, contrôle de versions) et intégration du projet dans un cadre existant
- **mettre en œuvre un premier design pattern (poids-plume)**

**TC-2 : construction de l'arbre de syntaxe abstraite**

- compléter le parser en GLR
- **comprendre des traits de l'objet et du C++ (héritage, polymorphisme d'inclusion, construction, destruction)**
- **découvrir de nouveaux design patterns (composite, visiteur)**
- utiliser des conteneurs standards (STL)
- écrire une documentation développeur

**TC-3 : calcul des liaisons (bindings)**

- appliquer le design pattern commande pour ajouter une tâche au compilateur
- écrire un visiteur complet
- **écrire une classe de conteneur paramétrée**
- **découvrir la métaprogrammation via les « traits »**

**TC-4 : vérification des types**

- comprendre les fonctions et méthodes paramétrées et la spécialisation de fonctions/méthodes paramétrées
- appliquer le design pattern patron de méthode
- comprendre et mettre en œuvre le contrôle de types dans Tiger

**TC-5 : traduction vers la représentation intermédiaire**

- comprendre la représentation intermédiaire de Tiger
- comprendre les blocs d'activation
- **gérer la mémoire à l'aide de « smart pointers » et de comptage de références**
- **utiliser les « variants » de Boost**
- découvrir de nouveaux traits du C++

**TC-6 : canonisation de la représentation intermédiaire**

- **appréhender la réécriture de termes**
- **découvrir une programmation à saveur fonctionnelle en C++ via la STL**

**TC-7 : Sélection d'instructions**

- découvrir l'assembleur et les architectures processeur de type RISC et CISC
- écrire/débugger un générateur de code assembleur avec MonoBURG
- implémenter la gestion de la mémoire dans les langages autorisant les fonctions récursives, comme Tiger

Options

**TC-8 : Analyse de vivacité**

- assimiler des notions nécessaires à l'allocation de registres : gestion de graphes, graphe de flux, vivacité des variables, graphe d'interférences/de conflits
- **manipuler les graphes de Boost**
- **pratiquer le parcours de graphes**

**TC-9 : Allocation de registres**

- **travailler sur un problème NP-complet**
- **découvrir l'allocation de registres par coloration de graphes**

### Développement du projet

Outils

- HAVM : un interpréteur du langage de représentation intermédiaire de Tiger
- Nolimips : simulateur d'architecture MIPS, utilisé pour exécuter de l'assembleur MIPS
- MonoBURG : un générateur de générateur de code issu du projet Mono, enrichi pour le projet Tiger
- GNU Bison : ce générateur de parser libre a été amélioré à l'occasion du projet Tiger

Quelques chiffres

- projet démarré en l'an 2000
- 40000 lignes de code
- 1500 étudiants ont fait le projet Tiger
- plus de 200 pages de documentation (sujet du projet et spécifications)