

INFO0947 : Compléments de Programmation Récursivité & Types Abstraits de Données

B. Donnet, S. Liénardy
Université de Liège

1 Type Abstrait

Le 29 octobre 1959 (il y a donc 60 ans) paraissait la première aventure d’un petit gaulois, Astérix, et son ami, Obélix. Cette première aventure est parue le 29 octobre 1959 (il y a donc 60 ans), dans le journal français *Pilote*. La première bande dessinée du Gaulois, “Astérix Le Gaulois”¹, sera elle publiée deux ans plus tard. L’objectif de ce travail est de rendre hommage à cette bande dessinée en implémentant le “Tour de Gaule d’Astérix”² sous la forme d’un type abstrait de données.

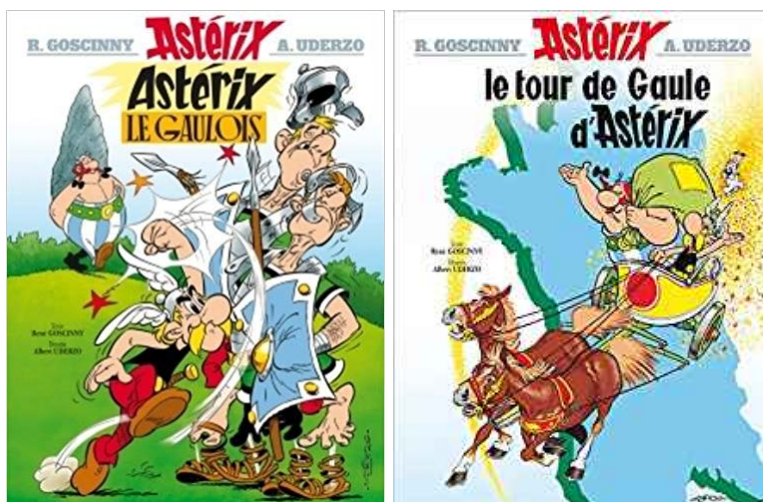


FIGURE 1 – Couverture des albums d’Astérix (©Dargaud).

“Le Tour de Gaule” raconte l’histoire d’Astérix et Obélix qui font le tour de plusieurs villes de Gaule, suite à un défi lancé à un envoyé de César. Avec Obélix, ils franchiront la palissade qui entoure leur village, feront le tour de la Gaule et ramèneront comme preuves des spécialités gastronomiques des villes gauloises. Les villes visitées, dans l’histoire, à partir de leur village sont Rotomagus (Rouen), Lutèce (Paris), Camaracum (Cambrai), Durocortorum (Reims), Divodurum (Metz), Lugdunum (Lyon), Nicae (Nice), Massilia (Marseille), Tolosa (Toulouse), Aginum (Agen), Burdigala (Bordeaux), Gésocribate (Le Conquet), pour enfin revenir à leur village.

Dans ce travail, il vous est demandé de spécifier deux types abstraits, une *ville* (Sec. 1.1) visitée par nos amis gaulois et un *Tour de Gaule* basé sur une séquence de ville (Sec. 1.2).

1. R. Goscinny, A. Uderzo. *Astérix le Gaulois*. Astérix, Tome 1. Ed. Dargaud, 1961.

2. R. Goscinny, A. Uderzo. *Le Tour de Gaule d’Astérix*. Astérix, Tome 5. Ed. Dargaud, 1965.

1.1 Ville

Une *ville*, sur la carte de Gaule, est définie par ses coordonnées géographiques X et Y , sous la forme de deux nombres réels. Le type abstrait **Ville** doit permettre de :

- créer une ville à partir de ses deux coordonnées X et Y et de son nom ;
- obtenir la coordonnée X d’une escale ;
- obtenir la coordonnée Y d’une escale ;
- obtenir le nom de la ville ;
- obtenir la spécialité gastronomique de la ville
- calculer la distance géographique entre deux ville ;
- enregistrer la spécialité gastronomique de la ville³ ;
- obtenir la spécialité gastronomique de la ville.

1.2 Tour de Gaule

Un *Tour de Gaule* est défini comme une suite ordonnée d’escales. Par définition, le point de départ du tour (i.e., le village d’Astérix et d’Obélix) est le même que le point d’arrivée. Le tour de Gaule forme donc ce qu’on appelle ce qu’on appelle un *circuit*.

Le type abstrait **Gaule** doit permettre de :

- créer un tour de Gaule sur base de deux villes. Par définition, un tour nouvellement créé ne peut pas constituer un circuit ;
- déterminer si un tour de Gaule constitue un circuit ;
- déterminer le nombre de villes visitées durant le tour ;
- déterminer le nombre totale de ventes d’un voyage ;
- déterminer la spécialité gastronomique d’une ville du tour ;
- ajouter une ville à un tour ;
- supprimer une ville d’un tour ;

2 Représentation Concrète

À l’instar de ce qui a été fait au cours théorique, vous devrez proposer deux représentations concrètes implémentant vos types abstraits.

La première devra être une représentation basée sur un **tableau**.

La deuxième représentation devra être basée sur une **liste chaînée**.⁴ Pour cette représentation, quand cela s’avère possible et pertinent, il vous est demandé d’implémenter les fonctionnalités de manière récursive.

Attention, n’oubliez pas de fournir une représentation concrète pour tous les types abstraits que vous devez définir.

3 Tests Unitaires

Nous vous demandons, aussi, dans ce travail de tester la validité d’une partie de vos deux implémentations à l’aide de tests unitaires. Pour ce faire, vous utiliserez la librairie **seatest** présentée dans le cours INFO0030. En particulier, nous voulons que vous testiez la fonctionnalité qui retourne le nombre de villes visitées lors d’un tour de Gaule et celle qui permet d’ajouter une ville à un tour.

3. peu importe laquelle, ce n’est pas vraiment la question ici

4. Il n’est pas nécessaire, ici, de passer par une représentation en TAD de la liste chaînée.

4 Aspects Pratiques

Le travail à rendre se compose d'une archive `tar.gz`⁵. Votre archive portera la nom suivant : `tad-groupeXX.tar.gz`, où `XX` fait référence à l'identifiant de votre groupe.⁶

Une fois décompressée, votre archive devra donner naissance à deux répertoires : `rapport/` (cfr. Sec. 4.1) et `code/` (Sec. 4.2). Tout non-respect des consignes se verra sanctionné par 2 points en moins dans la note finale de votre projet.⁷

4.1 Rapport

Votre rapport devra être rédigé via l'outil LaTeX en utilisant l'en-tête du template LaTeX disponible sur la page web du cours. Dans le template, vous trouverez les commandes suivantes :

```
\title{INF00947: Titre du Projet}
\author{Groupe XX: Prénom\_Nom, Prénom\_Nom}
```

où `Titre du Projet` doit être remplacé par le titre de ce projet et où `XX` désigne votre numéro de binôme. Vous ferez suivre votre numéro de groupe par les prénoms et noms des différents membres du groupe. Une fois compilé, votre document LaTeX devra produire un fichier PDF dont le nom sera `tad-XX.pdf`, où `XX` représente toujours votre numéro de groupe. Ce rapport ne pourra pas compter plus de **12 pages**.⁸

Le dossier `rapport/` sera composé des éléments suivants :

- votre rapport au format `.tex` ainsi que toutes les sources utiles à la compilation ;
- votre rapport déjà compilé au format `.pdf`.

Votre rapport devra contenir tous les éléments nécessaires à la compréhension de votre code source.

Soit :

- la spécification complète des types abstraits (i.e., signature et sémantique) ainsi que la justification de la complétude et consistance des axiomes ;
- la description de la structure implémentant vos TADs ;
- la description des structures de données basées sur un tableau et sur une liste que vous mettrez en place ;
- la discussion des avantages et inconvénients de chaque structure de données ;
- votre découpe en sous-problèmes, la description de chaque sous-problème et comment ils s'emboîtent ;
- la spécification (formelle) de chaque procédure/fonction concrète (pour les deux représentations) ;
- quand cela s'avère nécessaire, un schéma représentant la situation générale et l'invariant formel que vous tirez de cet invariant. Si nécessaire, vous pouvez introduire de nouvelles notations (à condition de bien les expliquer). Il n'est pas demandé d'écrire les assertions intermédiaires ;
- pour les implémentations récursives, il est demandé de justifier votre construction récursive en répondant aux trois questions (paramètre récursif, cas de base, formulation récursive – cfr. Chapitre 4, Slides 42 → 64) ;
- l'évaluation et la justification de la complexité de chacune des opérations en fonction du type d'implémentation ;
- une explication de la façon dont vous avez écrit votre librairie de tests unitaires.

Soyez clairs et précis dans vos explications. N'hésitez pas à ajouter un schéma si vous le jugez nécessaire. Dans ce cas, expliquez-le : un schéma seul ne suffit pas !

Soignez votre orthographe : au delà de trois fautes, chaque faute vous fera perdre 0,25 points.

5. Nous vous renvoyons à la formation sur la ligne de commande, donnée au Q1, pour la compression des fichiers dans une archive `tar.gz`.

6. Pour rappel, il est interdit de changer de groupe entre les différents projets. Votre identifiant doit nécessairement être un nombre composé de deux chiffres (compris entre 01 et 50).

7. Comme pour les projets précédents, les archives ne se décompressant pas convenablement engendrent une note finale nulle pour ce projet.

8. Sans compter la page de garde, ni son verso.

4.2 Code

Votre code source devra être rédigé en langage C. Votre code source devra être accompagné d'un Makefile⁹ rédigé par vos soins.

Votre code source sera adéquatement découpé en headers et modules. Les différents sous-problèmes identifiés dans votre rapport devront apparaître clairement dans votre code. Les headers décrivant les interfaces de vos types abstraits devront s'appeler `ville.h` et `gaule.h`. Vous devrez en outre fournir les modules implémentant ces headers. Le premier, appelé `ville.c` implémente le point dans l'espace à deux dimensions. Ensuite, vous devrez fournir deux autres modules, soit `gaule_tableau.c` correspond à l'implémentation basée sur un tableau. Le deuxième, appelé `gaule_liste.c`, correspond à l'implémentation basée sur une liste.

En outre, vous fournirez une librairie de tests unitaires basée sur `seatest`¹⁰ comme indiqué à la Sec. 3.

L'exécution de la commande

```
1 $>make test_array
```

doit produire un fichier binaire exécutable appelé `gaule_tableau_test`. Ce fichier exécutable sera situé dans le même répertoire que celui où se trouve le code source. Son exécution doit permettre de tester la validité de votre code pour l'implémentation des deux fonctionnalités demandées pour le TAD via un tableau. La fonction `main` de cet exécutable devra être implémentée dans un fichier appelé `gaule_tableau_test.c`

L'exécution de la commande

```
1 $>make test_list
```

doit produire un fichier binaire exécutable appelé `gaule_liste_test`. Ce fichier exécutable sera situé dans le même répertoire que celui où se trouve le code source. Son exécution doit permettre de tester la validité de votre code pour l'implémentation des deux fonctionnalités demandées pour le TAD via une liste. La fonction `main` de cet exécutable devra être implémentée dans un fichier appelé `gaule_liste_test.c`.

Ces différents fichiers source doivent pouvoir être compilés et utilisés sans la présence du fichier contenant la fonction `main`.

Il est **interdit**¹¹ d'utiliser des fonctions ou procédures de librairies tierces (y compris la librairie standard du C), à l'exception de `assert.h`, `stdlib.h`, `math.h` et `seatest.h`.

5 Agenda

5.1 Milestones

Pendant la durée du projet, nous vous proposons deux *milestones*.¹² L'objectif de ces milestones est de rencontrer chaque groupe individuellement afin de discuter de l'avancement du projet et corriger le tir le cas échéant. Ces rencontres sont organisées resp. la 3^e et la 2^e semaine qui précèdent la remise du travail final. À l'issue de ces rencontres, aucune note ne sera affectée. L'objectif n'est donc pas d'évaluer formellement votre travail mais bien de vous aider à réaliser le projet et à en tirer des enseignements.

Les deux milestones sont les suivants :

- **Semaine du 20/04/2020.** Le milestone portera sur les axiomes. Afin de permettre à l'équipe pédagogique de préparer au mieux cette rencontre, il vous est demandé de nous (i.e., Benoit Donnet et Simon Liénardy) envoyer par email un petit document (2 pages maximum, format PDF, rédigé en LaTeX) décrivant votre compréhension du problème et vos axiomes. En particulier, il peut être intéressant de fournir un schéma illustrant les concepts derrière votre TAD. La deadline pour

9. cfr. le cours INFO0030, « Partie 2 – Chapitre 1 : Compilation ».

10. cfr. le cours INFO0030, « Partie 2 – Chapitre 2 : Tests ». La page web du cours INFO0030 contient les sources de la librairie `seatest`. Vous n'oublierez pas de joindre ces sources à votre archive.

11. Cette restriction ne s'applique pas aux fichiers `*-test.c` où sont implémentés les exécutables de tests.

12. « Étapes » en français.

l'envoi de ce document est le samedi 18/04/2020, 12h00 (n'oubliez pas d'indiquer la chaîne de caractères [INFO0947] dans le sujet de votre email).

- **Semaine du 27/04/2020.** Le milestone portera sur les structures de données et une partie de votre code (et donc les invariants). Afin de permettre à l'équipe pédagogique de préparer au mieux cette rencontre, il vous est demandé de nous (i.e., Benoit Donnet et Simon Liénardy) envoyer par email un petit document (format PDF, rédigé en LaTeX) décrivant vos structures de données et vos invariants (situation générale, dessin, et invariant formel). La deadline pour l'envoi de ce document est le samedi 25/04/2020, 12h00 (n'oubliez pas d'indiquer la chaîne de caractères [INFO0947] dans le sujet de votre email).

La participation à ces milestones est obligatoire pour tous les membres du binôme. Aucun groupe ni étudiant ne peut s'y soustraire. Pour définir le moment auquel votre groupe viendra rencontrer l'équipe pédagogique, deux sujets de discussion sont créés dans le forum du cours, sur la plateforme e-campus. Il vous suffit de lire et de suivre les instructions détaillées dans le premier message de chaque sujet.

Veuillez faire débiter les sujets des mails envoyés à l'équipe pédagogique par la chaîne “[INFO0947]”.

5.2 Deadline

Les archives `tar.gz` finales doivent être soumises sur la plateforme `http://submit.montefiore.ulg.ac.be` avant le **11/05/2020 à 18h00** (l'heure du serveur faisant foi). Toute soumission postérieure à cette date ne sera pas prise en compte. Comme l'heure de l'horloge du serveur de soumission et celle de votre ordinateur peuvent être légèrement différentes, il est **fortement déconseillé** de soumettre votre projet à 17h59 et 59 secondes : ne prenez pas de risque inutile.