Claire BAUCHU – Damien CHANCEREL – APP4 INFO

# RAPPORT PROJET DE COMPILATION – 2020

## Retours sur le cours

Ce que l’on a aimé :

* La clarté du cours : les explications étaient toujours exhaustives, la quantité de schémas était très appréciables surtout pour les passages un peu ardus.
* La méthode d’enseignement : utiliser une méthode incrémentale comme celle employée était bénéfique pour notre compréhension et était très adaptée au contenu. Le fait d’ajouter des couches au fur et à mesure (enrober d’abord les Atomes, puis les Expressions, puis les Instructions, puis les Fonctions) nous permettait de corriger au fur et à mesure : il fallait que la couche inférieure soit fonctionnelle et débuggée pour s’assurer du fonctionnement de la suivante. De même, il était très gratifiant sur le plan intellectuel de voir son programme s’améliorer petit à petit et de constater les fruits du travail sur la couche précédente.
* Le thème du cours : c’était inédit pour nous. L’intitulé a pu nous faire peur (nous pensions notamment que nous allions devoir faire des TP où il aurait fallu compiler à longueur de temps et avec différentes options), mais nous sommes impressionnés d’être capables de coder un compilateur, que nous considérions auparavant comme quelque chose d’intouchable pour nous.

Ce que l’on a moins aimé :

* Le rythme : parfois, le rythme était un peu rapide. Nous savons que le cours était dense, cela laissait peu de temps de TP en classe. Heureusement, notre groupe arrivait à terminer les parties TP en classe (sauf une fois, où il a fallu travailler le soir à la maison). L’organisation aurait pu être très difficile pour se retrouver en dehors des cours. De plus, sans enseignant pour poser des questions, nous étions un peu perdus : ce n’est pas une matière qui nous permet de chercher sur Internet pour trouver des pistes pour débugger. Mais cela nous force en un sens à avoir une réflexion poussée seuls.

## Difficultés rencontrées

* Les pointeurs : les pointeurs nous ont paru difficiles, à cause de bugs que nous avions. En plus de ces bugs (qui étaient des fautes dans le code dans la génération de code), notre connaissance (complexe) des pointeurs nous empêchait de correctement débugger. Nous nous trompions en effet entre adresse et valeur assez souvent, ce qui ralentissait notre progression.
* Le debug de la pile : cela reste une difficulté normale rencontrée surement par tout le monde, mais le debug de la pile était ardu. En effet, nous sommes habitués aux compilateurs plutôt bavards sur l’origine d’un problème. Ici, c’était le pire des scénarios : le code « fonctionnait », mais pas comme nous le voulions. Ajouter des détails sur le déroulement du code en ajoutant des -d à l’exécution de la machine MSM nous permettait de débugger en détails. Cependant, au début nous n’avions pas beaucoup de code. A partir de l’ajout du runtime (qui compilait donc plusieurs fonctions en plus à chaque fois), le debug devenait difficile (et surtout très long, ce qui mettait à rude épreuve notre patience).

## Ce qui marche ou pas

|  |  |
| --- | --- |
| **Thème** | **Marche, Marche à peu près, Marche pas, Pas fait** |
| **Analyseur Lexial** | Marche |
| **Atomes (Maths)** | Marche |
| **Expressions (sauf affectations)** | Marche |
| **Variables (Déclaration et affectation)** | Marche |
| **Conditionnelles** | Marche |
| **Boucles (dont break et continue)** | Marche |
| **Fonctions (définitions)** | Marche |
| **Fonctions (appels)** | Marche |
| **Pointeurs** | Marche |
| **Tableaux** | Marche |
| **Runtime -> qu'est-ce qui est disponible et comment l'utiliser** | Puissance : a^2 ; Printf : printf(a) ;  Scanf : a=scanf();  Malloc :  int t ;  t=malloc(3) ; //pour réserver 3 cases mémoire  Free :  free(t) ; -> ne fait qu’un return 0 |