Projet de compilation

Nicolas Bedon et Arnaud Lefebvre

2021–2022

LaTeX est un programme permettant d'écrire des documents scientifiques. Ce sujet a été écrit en utilisant LaTeX. Vous trouverez dans les sources accompagnants ce sujet des exemples de fichiers écrits en LaTeX. L'extension classique des fichiers LaTeX est `.tex`. Un fichier `.tex` contient des commandes décrivants un document. Par exemple, si le fichier contient la commande `$\sqrt n$` le document final contiendra $\sqrt{n}$. La commande `pdflatex toto.tex` permet de produire `toto.pdf`.

Il est possible, dans un document LaTeX, d'utiliser des paquetages qui offrent des possibilités supplémentaires. Par exemple, le paquetage nommé AlGo permet d'écrire des algorithmes. Ainsi, le code LaTeX suivant :

```
\begin{algo}{puissance}{a,b}
  \SET{p}{1}
  \DOFORI{k}{1}{b}
    \SET{p}{p*a}
  \OD
  \RETURN{p}
\end{algo}
```

produit le résultat:

PUISSANCE(a,b)
1 $p \leftarrow 1$
2 **for** $k \leftarrow 1$ **to** b **do**
3 $\quad p \leftarrow p * a$
4 **return** p

On souhaite introduire deux nouvelles commandes LaTeX que nous appellerons `\ASIPRO` et `\SIPRO`.

- `\ASIPRO` prends en argument un fichier `.tex` contenant la description d'une fonction f dans la syntaxe du paquetage AlGo. Elle traduit la description de cet algorithme en ASIPRO, en produisant un fichier `.asm` correspondant au `.tex`.
- `\SIPRO` prends en argument un nom de fonction dont on supposera qu'elle aura été précédemment décrite avec le paquetage AlGo, ses arguments, exécute la fonction en utilisant SIPRO, et récupère le résultat.

Vous trouverez un exemple de ce qu'on veut faire dans le répertoire `Exemple`.

Pour simplifier, le but du projet n'est pas d'implanter entièrement ces deux commandes, mais juste d'écrire deux programmes (dont l'utilisation est ici illustrée par l'exemple de la puissance décrite plus haut):

- `algo2asm puissance.tex` génère un fichier `puissance.asm` contenant le code source ASIPRO correspondant au contenu de `puissance.tex`, dont on supposera qu'il ne contient que la description d'une fonction décrite en utilisant AlGo. Pour simplifier, on supposera que le nom du fichier est le même que la fonction décrite dans le fichier (dans l'exemple précédent, on suppose donc que le fichier `puissance.tex` contient la description AlGo d'une fonction nommée `puissance`);
- `run '\SIPRO{puissance}{2,3}'` utilise SIPRO pour calculer la valeur de la fonction dont le nom est donné en premier argument de la commande `\SIPRO`, en utilisant les arguments donnés en second

argument de la commande `\SIPRO`. Ainsi, `run '\SIPRO{puissance}{2,3}'` produit 8 sur sa sortie standard. Pour ce faire:

- on suppose que `algo2asm puissance.tex` a préalablement été exécuté avec succès, et donc qu'un fichier `puissance.asm` a déjà été produit;
- on génère un second fichier `puissance_main.asm` obtenu en ajoutant à `puissance.asm`, qui ne contient que la description d'une fonction, le code ASIPRO nécessaire pour constituer un programme qui applique la fonction à ses arguments et affiche le résultat sur la sortie standard;
- on compile `puissance_main.asm` en utilisant ASIPRO, et on exécute le fichier obtenu avec SIPRO.

Le paquetage ALgo et sa documentation sont donnés dans le répertoire `Exemple`.

Travail minimum à rendre : votre projet devra au minimum gérer les cas des algorithmes simples (n'appelant aucun autre algorithme) et les algorithmes récursifs ne manipulant que des entiers. Les conditions dans les boucles sont exprimées avec des expressions entières; la valeur 0 signifie que la condition est fausse, toute autre valeur qu'elle est vraie. Les opérateurs dans les expressions sont au minimum les opérations arithmétiques et logiques usuels (somme, produit, différence et quotient, et, ou, négation). Les structures de contrôle minimales sont:

- `\IF{condition}...\FI`
- `\IF{condition}...\ELSE...\FI`
- `\DOWHILE{condition}...\OD`

mais vous pouvez aussi implanter en supplément d'autres structures de contrôles décrites dans la documentation du paquetage qui vous est donnée.

1 Ce qu'il vous est demandé

Vous devez écrire en C11, et en utilisant `flex` et `bison` pour les parties concernant les analyses lexicales et syntaxiques, les commandes `algo2asm` et `run` présentées dans le sujet.

Bien entendu, votre projet devra être écrit le plus proprement possible: algorithmique adaptée, code clair et commenté.

Vous pouvez étendre le projet si vous le souhaitez, en rajoutant des fonctionnalités par exemple. Cependant, ne le faites que si la base qui vous est demandée est implantée et fonctionne correctement: il est préférable d'avoir un projet qui fait correctement le minimum plutôt que d'avoir un projet étendu dont le minimum demandé ne fonctionne pas.

Votre projet devra être rendu avec un jeu d'exemples illustrant le mieux possible ses fonctionnalités, ainsi qu'un rapport contenant un rapport de développement et un manuel d'utilisation.

Il devra être développé individuellement ou par binôme, et rendu au plus tard le 3 avril 2022 au soir, dans une archive au format `tar` gzippé de nom `FrancoisDupontJacquesDurant.tar.gz` pour un binôme (si Francois Dupont et Jacques Durant sont vos noms), envoyée en pièce jointe à un courriel de sujet « Projet de compilation L3 Info » à `Nicolas.Bedon@univ-rouen.fr` et `Arnaud.Lefebvre@univ-rouen.fr`). Vous vous mettrez en copie du courriel pour vérifier que vous n'oubliez pas la pièce jointe. L'extraction du fichier d'archive devra produire un répertoire de nom `FrancoisDupontJacquesDurant` contenant le code source de votre projet, un makefile, des jeux d'exemples et un rapport de projet.

Votre projet sera peut être l'objet d'une soutenance sur machine. Il devra en particulier compiler et s'exécuter sans erreur et sans avertissement dans les salles de travaux pratiques.