Rapport de project d'architecture des ordinateurs

C. DEFRÉTIÈRE

15 janvier 2019

Résumé

Dans le cadre de la licence d'informatique proposée par l'Université Jean-Monnet, l'architecture des ordinateurs est une unité d'enseignement proposé aux étudiants. Celle-ci permet de mieux comprendre les technologies utilisées actuellement, et de se forger une culture informatique plus solide.
Nous allons, à travers ce document, découvrir l'implémentation faite par moi-même, d'un vérificateur de parenthèses dans une chaine de caractères, en langage assembleur.

Table des matières

1 Introduction

1.1 Description du programme

Ce programme prend une chaine de caractère quelconque en argument, il vous suffira de taper, si votre exécutable se nomme 'main' par exemple :

```
./main 'ma chaine (de) c[a]racteres {(test)}'
```

afin de vérifier si la chaine est bien parenthésé.

Cette implémentation gère les types de parenthèses suivants :

- (et)
- { et }
- [et]

1.2 Une chaine bien parenthésé

Une chaine est dite 'bien parenthésé', si pour un type de parenthèse, il y a autant de parenthèse ouvrante que fermante.
De plus, pour éviter les chaines comme ')test(', on vérifiera également que pour tout préfixe de la chaine, toujours pour un type de parenthèse donné, il y a plus de parenthèse ouvrante que fermante.

1.3 Compilation et exécution

Avant de tenter toute commande, veuillez décompresser le fichier tar. Tout se passera désormais dans le dossier obtenu.
Pour compiler ce projet, aucun soucis, utilisez simplement l'utilitaire *make* à la racine du dossier.
Les options disponible pour *make* sont :

- *clean*
- *build*
- *run*

2 Algorithme

2.1 Pseudo-code

Algorithme Vérifier_parenthèses($\mathcal{S}$)

Données : une chaine de caractères $\mathcal{S}$

Résultat(s) : un affichage dans le terminal disant si oui ou non la chaine est bien parenthésé

```
début
    chaine ← S;
    vider(pile);
    tant que non_vide(chaine) faire
        c ← retirer_premier_caractère(S);
        si est_ouvrante(c) alors
            empiler(c);
        si est_fermante(c) alors
            si est_vide(pile) alors
                afficher("mal parenthésé");
                arrêt_programme() ;
            p ← dépiler(c);
            écart ← écart_ascii(p, c);
            si écart > 2 alors
                afficher("mal parenthésé");
                arrêt_programme();
    si est_vide(pile) alors afficher("bien parenthésé");
    si n'est_pas_vide(pile) alors afficher("mal parenthésé");
```