Programmation en language C - Projet

Licence informatique 2ème année

Université de La Rochelle

2021-2022_1

Table des matières

Liste des exercices

1 Consignes

1.1 Groupes

Le projet se fait par groupe de maximum 6 étudiants et est à rendre par un dépôt *moodle* le vendredi 10 décembre 2021 à 23h00. Un retard raisonnable est toléré mais il sera pénalisé.

1.2 Langue utilisée

La langue utilisée dans le code et la documentation devra être exclusivement l'anglais. Si vous avez des difficultés dans la langue de Shakespeare, vous pourrez utiliser les traducteurs automatiques :

— https://www.deepl.com/translator
— https://translate.google.fr/

1.3 Nommage

1.3.1 Fichiers

- les noms de fichiers du langage C seront tous en minuscules et en anglais. S'ils sont composés de plusieurs mots, ils devront être séparés par un tiret (-) ;
- les fichiers contenant le code devront avoir l'extension `.c` ;
- les fichiers d'en-têtes (exportables) devront avoir l'extension `.h` ;
- les fichiers destinés à être inclus dans votre code mais non exportables devront avoir l'extension `.inc`

1.3.2 Types

Les noms de types devront faire commencer chaque mot qui les compose par une majuscule. Il n'y a pas de sous-tirets. Les structures et les énumérations devront commencer par un sous-tiret (_) pour ne pas les confondre avec les noms de types.

1.3.3 Macros

Les macros (avec ou sans arguments) s'écrivent tout en majuscule en séparant les mots par des sous-tirets (_).

1.3.4 Variables et fonctions

Les variables et les fonctions s'écrivent toutes en minuscules en séparant les mots par des sous-tirets.

1.4 Style

1.4.1 Marque d'inclusion unique

Chaque fichier d'en-tête devra posséder une marque permettant d'éviter les conséquences d'un fichier inclus plusieurs fois. Voir https://google.github.io/styleguide/cppguide.html#The__define_Guard

1.4.2 Ordre des inclusions

L'inclusion des fichiers d'en-tête devra respecter la logique suivante :

1. Inclusion du fichier directement lié au fichier `.c` qui l'inclut suivi d'une ligne vide ;
2. inclusion des fichiers d'en-tête du C standard suivis d'une ligne vide ;
3. inclusion des fichiers d'en-tête provenant d'autres librairies suivis d'une ligne vide ;
4. inclusion des fichiers d'en-tête du projet suivi d'une ligne vide ;
5. inclusion des fichiers d'inclusion (extension `.inc`)

1.4.3 Indentation

Le style d'indentation devra être celui préconisé par Google https://google.github.io/styleguide/cppguide.html#Formatting. L'utilitaire `clang-format` (https://clang.llvm.org/docs/ClangFormat.html) supporte le style `Google`.

Vous pourrez utiliser l'utilitaire `cclint` pour vérifier votre code.

1.5 Structuration du code

Les champs des structures seront protégés à la manière de la librairie `fraction` vue en travaux pratiques. Un soin sera tout particulièrement apporté à la structuration du code notamment en ce qui concerne les structures de données utilisées.

1.6 Documentation

La documentation sera générée avec l'outil `sphinx` et les fonctions seront documentées avec la norme de `doxygen`.

1.7 Tests

Des tests unitaires devront être implémentés, ils testeront chaque fonction et s'efforceront de vérifier que la mémoire est bien libérée au moyen de l'utilitaire `valgrind`.

Vous pourrez vous inspirer du projet https://github.com/chdemko/c-test.

D'une manière générale, toutes les options possibles décrites dans ce projet devront être implémentées.

2 Sujet

Le but du projet est de produire :

— une librairie permettant d'effectuer de la déduction par un système de chainage avant [1] ;
— un logiciel capable d'appliquer un fichier de règles à un fichier une base de faits initiaux.

Le projet devra fournir une documentation produite avec

```
$ make docs
```

Il pourra être installé avec

```
$ make install
```

Le logiciel devra s'appeler `rules` (et devra utiliser une librairie partagée de même nom) et devra prendre en arguments le fichier de règles à utiliser, le fichier de faits initiaux et le fichier dans lequel écrire les résultats :

1. https://fr.wikipedia.org/wiki/Chaînage_avant

```
$ ./rules --rules example.rules.txt \
          --facts example.facts.txt \
          --output example.output.txt
$
```

Chaque fait sera identifié par un entier unique. Chaque règle dans le fichier de règles sera représentée par deux lignes contenant des numéros de faits séparés par des espaces. La première ligne représentera les prémisses de la règle et la deuxième ligne les conclusions de la règle. Les règles seront séparées entre elles par une ligne vide.

Si le fichier de règles `example.rules.txt` contient les lignes suivantes

```
1 2
3

4 5
6

3
7

6
8
```

Si le fichier de faits `example.facts.txt` contient les lignes suivantes

```
1 2
```

Alors le fichier de sortie `example.output.txt` contiendra les lignes suivantes

```
1 2 3 7
```

Historique des modifications

2021-2022_1 *Mercredi 8 octobre 2021*

Dr Christophe Demko <christophe.demko@univ-lr.fr>

— Version initiale