projet de mini-interpréteur FORTH

Modifications structurelles

Table des matières

1 Changement de stratégie

1.1 Conception initiale

Le projet FORTH a été conçu dans une optique autant de conception logicielle que de codage C :

- un cahier des charges général
- une interface suggérée pour quelques modules de bas niveau
- beaucoup d'autonomie pour les solutions d'implémentation choisies
- le tout complété par des explications en direct durant les TD

Il implique beaucoup d'interactions durant les TD pour expliciter le cahier des charges, proposer des pistes, voir pour chaque projet ce qui bloque etc.

1.2 Changement de fusil d'épaule

L'impossibilité d'assurer les TD en présentiel fait que ni le calendrier, ni même la méthode, ne sont tenable. Il a donc été décidé de réduire la voilure, de laisser les étudiants se concentrer sur l'analyse et le codage des modules **pile** et **dictionnaire,** et d'une partie du **noyau**, en fournissant le reste du projet.

1.2.1 Fourni

Pour ce faire, un fichier ***zip*** est fourni, qui contient les fichiers ***.h*** du projet, ainsi qu'une partie des fichiers ***.c*** . L'ensemble provient d'une implémentation opérationnelle , qui a donc été dépouillée de ce vous aurez à faire.

1.2.2 A faire

Le but pour les étudiants est désormais de :

- compléter le ou les fichiers ***.h*** et ***.c*** là ou c'est indiqué (/* ***TODO */***)
- implémenter les fichiers ***.c*** manquants
- intégrer avec l'ensemble du code fourni pour tester le projet

1.2.3 A rendre

Bien entendu, il faudra toujours m'envoyer les livrables :

- rapport de projet
- ensemble des codes sources

tels qu'indiqué dans la documentation d'origine

2 Détails techniques

2.1 Liste des fichiers

Nom	Contenu	Etat
forth.h	Déclarations générales, codes d'erreur	Complet
interface.h	Interface de la gestion de l'analyse syntaxique et des E/S	Complet
interprete.h	Interface de l'interpréteur principal	Complet
pile.h	Interface de la gestion de la pile de données	Complet
dictionnaire.h	Interface de la gestion du dictionnaire des mots	A compléter
noyau.h	Déclarations des fonctions implémentant les mots système	A compléter
main.c	Programme principal	Complet
interface.c	Gestion de l'analyse syntaxique et des E/S	Complet
interprete.c	Interpréteur principal	A compléter
pile.c	Gestion de la pile de données	A faire
dictionnaire.c	Gestion du dictionnaire des mots	A faire
noyau.c	Fonctions implémentant les mots système	À compléter

La compilation de l'ensemble se fait en passant l'ensemble des fichiers ***.c*** ci-dessus en paramètre au compilateur.

2.2 Ce qu'il faut compléter

2.2.1 Déclarations à compléter

Dans **dictionnaire.h** : le contenu de la structure représentant une entrée du dictionnaire (*typedef struct { … } Entree ;*)

Dans **noyau.h** : la déclaration de toutes les fonctions implémentant les mots de la liste de vocabulaire spécifiée (il y en a déjà quelques unes pour servir de modèle).

2.2.2 Définitions à ajouter

Dans **pile.c** : la structure de la pile de données et les variables globales internes attenantes

Dans **dictionnaire.c** : la structure du dictionnaire d'entrées et les variables globales attenantes

2.2.3 Fonctions à compléter

Dans **interprete.c** : compléter le code de la fonction *Retcode Interprete(char * element)*, selon votre implémentation des entrées du dictionnaire

Dans **noyau.c** : compléter le code de la fonction *Retcode Code_RefValue(RefEntree ref)*, selon votre implémentation des entrées du dictionnaire

2.2.4 Fonctions à implémenter

Dans **pile.c** : toutes les fonctions déclarées dans **pile.h**, plus des fonctions internes si besoin est

Dans **dictionnaire.c** : toutes les fonctions déclarées dans **dictionnaire.h**, plus des fonctions internes si besoin est

Dans **noyau.c** : le code de toutes les fonctions déclarées dans noyau.h (il y en a quelques une d'implémentées, pour servir de modèle).

2.3 Conséquences par rapport au travail déjà effectué

La conséquence la plus immédiate est que certains choix techniques ont été opérés, qui ne sont peut-être pas compatibles avec ceux que certains d'entre vous ont effectué. Vous pouvez dans ce cas bien sûr tenter de modifier le code fourni dans votre sens, mais attention de ne rien casser !

Le choix des noms de fonctions de **pile** et **dictionnaire**, notamment, est pratiquement gelé, puisqu'elles sont utilisées dans des modules fournies. Ce point n'est pas si embêtant que ça : comme vous avez suivi les consignes sémantiques initiales, il ne devrait y avoir que le nom de vos fonctions à changer, ce qui se fait facilement en une fois avec Notepad++ ou un éditeur de code intégré.

L'autre point bloqué est la liste des codes d'erreurs retournés par les diverses fonctions. Vous pouvez modifier bien sur cette liste, mais attention là encore à ne rien casser !