Université de Yaoundé I
Faculté des Sciences

University of Yaounde I
Faculty of Science

Département d'Informatique

TP final ICT106, Structures de Données, 20 Juin 2024

Dr NZEKON NZEKO'O Armel Jacques , Dr MESSI NGUÉLÉ Thomas

Consignes : Le TP sera réalisé en langage C. Vous déveez avoir un dossier nommé **premierNom_matricule** qui aura trois sous-dossiers nommé respectivement pile_premierNom_matricule, verifExpression_premierNom_matricule et tableHachage_premierNom_matricule. Chacun de ces sous dossiers doit contenir quatre fichiers : Makefile, main_premierNom_matricule.c, functions_premierNom_matricule.c, functions_premierNom_matricule.h.

1 Pile

1. Implémenter en C une pile à l'aide d'un tableau. On devra avoir les contraintes suivantes :
   (a) le type de la pile devra être : *PileTab_t*
   (b) les primitives devront s'appeler :
      i. *pushTab(char c, PileTab_t* P)* ;
      ii. *char pullTab(PileTab_t* P)*
      iii. *char headTab(PileTab_t* P)*
      iv. *initTab(PileTab_t* P)*
      v. *isEmptyTab(PileTab_t P)*
      vi. *isPlentyTab(PileTab_t P)*
2. Implémenter en C une pile à l'aide d'un tableau. On devra avoir les contraintes suivantes :
   (a) le type de la pile devra être : *PileList_t*
   (b) les primitives devront s'appeler :
      i. *pushList(char c, PileList_t* P)* ;
      ii. *char pullList(PileList_t* P)*
      iii. *char headList(PileTab_t* P)*
      iv. *initList(PileList_t* P)*
      v. *isEmptyList(PileList_t P)*
3. Calculer le nombre d'instructions réalisés par chaque opération de la pile suivant chaque implémentation.

2 Vérificateur des expressions bien formées

On voudrait écrire un programme permettant de dire si les expressions constituées de parenthèses, de crochets et d'accolades sont bien formées : l'algorithme parcourt la chaine et

détermine s'il y a équilibre (et dans un bon ordre) entre les parenthèses ouvrantes "(" et fermantes ")", crochets ouvrants "[" et fermants "]", accolades ouvrantes "{" et fermantes "}", les barres ouvrantes "|" et fermantes "|".

Exemple :

— L'expression $(1+2)|+3$ est mal formée car il y a une parenthèse fermante de plus.

— L'expression $\{[(1+2)+3]+13\}$ est bien formée.

1. **Principe de résolution.** Dire (sur papier et en au plus 5 lines) comment un tel programme peut être écrit en se servant de la structure de données pile.
2. **Structure de données.**
   (a) Proposer deux structures de données Pile *PileList_t* et *PileTab_t* permettant de résoudre le problème ainsi posé.
   (b) Donner les primitives de base permettant de manipuler cette pile (Vous servir des mêmes noms des primitives qu'à l'exercice 1 Pile).
3. **Programme de vérification.**
   (a) Écrire (sur papier) l'algorithme permettant de vérifier les expressions bien formées.
   (b) Écrire la procédure C correspondant à cet algorithme. On vous impose de vous servir des structures alternatives *switch - case*.
   (c) Écrire le programme principale qui lit un ensemble d'expressions (chaine de caractères) dans un fichier et vérifie si ces expressions sont bien formées.

3 Stockage des nouveaux étudiants

Enoncé : On aimerait stoquer les nouveaux étudiants pré-inscrit à l'université de yaoundé 1 pour l'année 2024-2025. L'un des dirigeants de l'Université décide de mettre le matricule sur le format $24LCCCCC$ avec C un chiffre de [1-9] et L une lettre [a-z]. On ne fait pas de disctinction entre une lettre miniscule et une lettre majuscule.

1. Répondre sur feuille aux questions suivantes :
   (a) Combien d'étutiants potentiels peut-on enrégistrer avec ce format ?
   (b) Donner les avantages et les inconvénients lorsqu'on décide d'utiliser une structure table ou une strucutre de données liste chainée pour stoquer les étudiants.
   (c) Dites comment utiliser une table de hachage pour stoquer les étudiants.
2. Code C
   (a) Donner une strucutre de données *student_t* permettant de représenter un étudiant (caractérisé par son age, son matricule, son nom).
   (b) Donner une strucutre de données *hashTable_t* implémentant une table de hachage.
   (c) Donner une fonction permettant d'ajouter un étudiant dans la table de hachage *insertInHashTable(student_t Stud, hashTable_t* hashT)*.
   (d) Donner une fonction permettant de rechercher un étudiant dans la table de hachage *findInHashTable(char* matricule, hashTable_t hashT)*.
   (e) Donner une fonction permettant de supprimer un étudiant dans la table de hachage *removeFromHashTable(char* matricule, hashTable_t* hashT)*.
   (f) Ajouter un main qui permet de gérer les étudiants en appelant toutes les fonctions définies.

..... Bon Courage ! .....