Structures de données en C

TD/TP 2 – Les Piles et les Files

Problème 1 : Vérification des expressions parenthésées avec des types multiples

Écrire un programme en C qui vérifie si une expression contenant plusieurs types de parenthèses, crochets et accolades ((), [], {}) est correctement équilibrée. Une expression est dite équilibrée si chaque parenthèse ouvrante a une parenthèse fermante correspondante du même type, et les parenthèses s'imbriquent correctement.

Par exemple :

- L'expression {[(a+b)*(c-d)]} est équilibrée.
- L'expression {a*[b-c+(d]-e)/g} ne l'est pas (les crochets et parenthèses ne sont pas correctement imbriqués).

1. Commencer par la déclaration d'une structure pour la pile comme suivant :

```
typedef struct Pile {
        char items[Max];
        int top;
} Pile;
```

   Le tableau items[Max] sera utilisé pour stocker les parenthèses d'ouverture ('(' ou '[' ou '{')

2. Développer les fonctions principales de traitement des piles :
   a. initPile(Pile* pile)
   b. isEmpty(Pile* pile)
   c. push(Pile* pile, char c)
   d. pop(Pile* pile)

3. Développer une fonction isBalanced(char* expression) qui prend comme argument une expression parenthésées. En parcourant l'expression caractère par caractère, la fonction fait ce qui suit :
   - S'il s'agit d'un caractère d'ouverture ('(' ou '[' ou '{'), elle l'ajoute à la pile
   - S'il s'agit d'un caractère de fermeture (')' ou ']' ou '}'), elle vérifie la correspondance à l'élément au sommet de la pile en utilisant la fonction
   - Vers la fin, si la pile est vide, ce qui veut dire que l'expression est équilibrée, la fonction retourne 1, sinon el retourne 0.

Problème 2 : Développement d'un système de navigation

Développez un système de navigation qui permettre à l'utilisateur de :

- Naviguer vers de nouvelles pages
- Revenir en arrière (avec la fonction " **backPage** ")
- Avancer à nouveau (avec la fonction " **forwardPage** ") après avoir utilisé la fonction "**backPage**")

Consignes : Utilisez deux piles pour gérer l'historique de navigation :

1. **Pile des pages précédentes (Back Stack)** : Contient les pages visitées avant la page actuelle.
2. **Pile des pages suivantes (Forward Stack)** : Contient les pages que l'on peut revisiter après êtr e revenu en arrière.

Fonctions principales :

1. **visitPage** : Ajoute une nouvelle page à la backStack et vide la forwardStack lorsqu'une nouvel le page est visitée.
2. **backPage et forwardPage** : Utilisent les opérations de pop et push pour déplacer des pages e ntre les piles backStack et forwardStack en fonction des actions de navigation de l'utilisateur.