**Data Structure**

COMPTE RENDU DES EXERCICES DES PILES

**Réalisé par :**

ELYOUSFI Mohamed (BDCC1)

**Supervisé par :**

Mr A. NAJI

**Année universitaire 2020/2021**

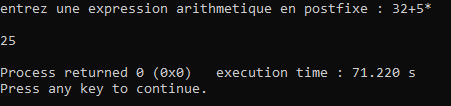
**Exercice 1 :**

* Objectif: utilisation des piles pour les expressions arithmétiques en postfixé

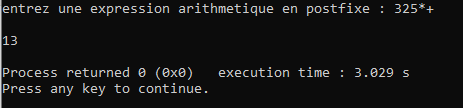
1. Code source : dans le fichier **Exercice1.c** attaché avec ce document.

Exécution :

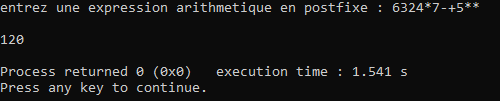
Pour **32+5\***



Pour **325\*+**



Pour **6324\*7-+5\*\***



1. Schématiser les étapes, à l’aide d’une pile, pour évaluer 3+(2\*5). Sachant que cette expression s’écrit en postfixé de la manière suivante : 325\*+ :

Tableau T de caractères qui contient toute l’expression arithmétique

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 2 | 5 | \* | + |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
| 13 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| 10 |
| 3 |

|  |
| --- |
|  |
|  |
|  |
|  |
| 5 |
| 2 |
| 3 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| 2 |
| 3 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
| 3 |

1. Schématiser les étapes, à l’aide d’une pile, pour évaluer l’expression en postfixé 6324\*7-+5\*\* :

Tableau T de caractères qui contient toute l’expression arithmétique

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 3 | 2 | 4 | \* | 7 | - | + | 5 | \* | \* |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
| 6 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| 3 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
|  |
| 2 |
| 3 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
|  |
| 8 |
| 3 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
| 7 |
| 8 |
| 3 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
|  |
| 1 |
| 3 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
| 60 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| 20 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
|  |
| 5 |
| 4 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
| 4 |
| 6 |

|  |
| --- |
|  |
|  |
|  |
| 4 |
| 2 |
| 3 |
| 6 |

1. A partir de ce qui précède, implémenter une fonction verifier\_si\_postfixe(chaine) qui permet de vérifier si une expression lue au clavier est en postfixé.

Code source : dans le fichier **Exercice1.c** attaché avec ce document.

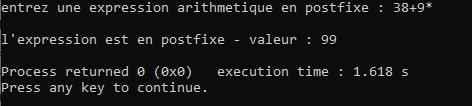
Exécution :

Principe utilisé : en lisant l'expression de gauche à droite on incrémente un compteur à chaque fois qu'on rencontre un nombre et on décrémente le compteur si on rencontre un opérateur.

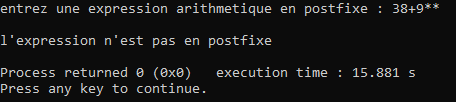
il s'agit d'une expression postfixe si et seulement si le compteur est toujours positif et sa valeur finale est égale à 1 (car il faut que le nombre des opérandes soit plus grande par 1 que le nombre des opérateurs).

Aussi, on vérifie que au moins les deux premiers caractères sont des opérandes, et le dernier caractère est un opérateur

**Cas d’expression en postfixé :** exemple 38+9\*



**Cas d’expression en postfixé :** exemple 38+9\*\*



**Exercice 2 :**

* Objectif : vérifier si un fichier HTML est bien structuré

Code source : dans le fichier **Exercice2.c** attaché avec ce document.

Exécution :

Principe utilisé : si le caractère égal à ‘ **<** ’ et son suivant différent de ‘ **/** ’, on empile **<,** sinon on la dépile.

**Cas d’expression HTML valide :**

<html>

<head>

<title>my page</title>

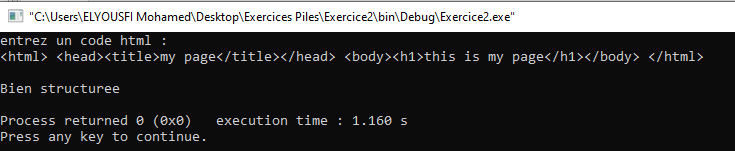
</head>

<body>

<h1>this is my page</h1>

</body>

</html>



**Cas d’expression HTML non valide :**

<html>

<head>

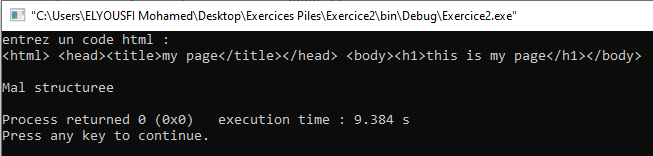
<title>my page</title>

</head>

<body>

<h1>this is my page</h1>

</body>



**Schématisation : cas 1**

|  |
| --- |
|  |
|  |
|  |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
| **<** |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
|  |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
| **<** |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
|  |
| **<** |

|  |
| --- |
|  |
|  |
|  |
|  |

**IsEmpty** dans ce cas return 1, donc l’expression est bien structurée

**Schématisation : cas 2**

|  |
| --- |
|  |
|  |
|  |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
| **<** |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
|  |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
| **<** |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
| **<** |
| **<** |

|  |
| --- |
|  |
|  |
|  |
| **<** |

**IsEmpty** dans ce cas return 0, donc l’expression n’est pas bien structurée

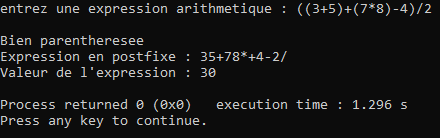
**Exercice 3 :**

* Objectif : transformer une expression infixée en une expression postfixée après qu’on a vérifié que c’est bien parenthésée.

Code source : dans le fichier **Exercice3.c** attaché avec ce document.

Exécution :

**Cas d’expression bien parenthésée :** ((3+5) +(7\*8) -4) /2



**Cas d’expression mal parenthésée** : ((3+5)+(7\*8)-4/2

