ASSEMBLEUR : Projet

DELAFORGE Jean, PELOUX Louis

17 décembre 2024

Table des matières

1 Programme Jules Cesar

1.1 Pseudo-code

Algorithme Jules César 2
Entrée : 2 arguments, le premier un entier et le second une chaîne de caractères
Sortie : Affichage de la chaîne modifiée

Algorithm 1 Décalage des caractères d'une chaîne

début :
$i \leftarrow 0$
$cl \leftarrow$ depiler(P)
$cl \leftarrow$ atoi(cl)
$chane \leftarrow$ depiler(P)
while $chane \neq$ vide **do**
    $c \leftarrow$ chaîne[i]
    while $c > 122$ **do**
        $c \leftarrow c - 26$
    end while
    chaîne[i] $\leftarrow c + cl$
    $i \leftarrow i + 1$
end while
Afficher la chaîne modifiée
fin

1.2 Fonctionnement du code

L'algorithme prend en parametre 2 arguments, le premier correspond a la clé, celle-ci doit être passée en argument d'une fonction atoi pour pouvoir avoir sa valeur en entier. Le second argument est une chaîne de caractères.

Premièrement, on verifie qu'il y est bien le bon nombre d'arguments, c'est-à-dire 2 arguments. Si ce n'est pas le cas, on affiche le message d'erreur "Il faut 2 arguments !!" dans stdout et on finit le programme.
On initialise un registre un 0 qui aura pour but d'être l'indice de la chaîne de caractères.

Ensuite, on recupère la chaîne de caractères contenue dans la pile et on la stocke dans un registre. On parcours le registre jusqu'à arriver au bout de la

chaîne. On recupère le caractère à l'indice, on lui ajoute a son code ascii la clé et on réécrit la valeur dans la chaîne.
On recommence cette boucle tant que la chaîne argv2 soit vide. Enfin, on affiche la chaîne de caractères modifiée.

2 Programme Vigenère

2.1 Pseudo-code

Algorithme Vigenere 2
Entrée : Deux chaînes de caractères, une qui fait office de clé, et l'autre à décoder
Sortie : Chaîne de caractères modifiée

Algorithm 2 Algorithme Vigenère

début :
$i \leftarrow 0$
$j \leftarrow 0$
$k \leftarrow 0$
$cl \leftarrow$ depiler(P)
$chane \leftarrow$ depiler(P)
$chane3 \leftarrow' \backslash 0'$ ▷ Initialisation de la chaîne codée
while *chane* n'est pas vide **do**
    $c \leftarrow$ chaîne[i]
    if $clef[j]$ est vide **then**
        $j \leftarrow 0$ ▷ Boucle sur la clé
    end if
    if $A \leq c \leq Z$ **then**
        $c \leftarrow c + 32$ ▷ Convertir en minuscule
    end if
    if $a \leq c \leq z$ **then**
        $h \leftarrow cl[j]$
        $h \leftarrow h - 97$ ▷ Décalage selon la clé
        $c \leftarrow c + h$
        while $c > 122$ **do**
            $c \leftarrow c - 26$ ▷ Reste dans la plage des minuscules
        end while
        $chane3[k] \leftarrow c$
        $j \leftarrow j + 1$
    else
        Appeler `sous_vigenere`
        $chane3[i] \leftarrow' .'$
    end if
    $i \leftarrow i + 1$
    $k \leftarrow k + 1$
end while
Afficher *chane*3
fin

2.2 Fonctionnement du code

L'algorithme de Vigenère commence par dépiler la clé et le message à chiffrer. Il initialise trois registres : i, j, et k, qui seront utilisés pour parcourir respectivement le message, la clé, et la chaîne codée.

À chaque itération, l'algorithme récupère un caractère du message (chaine[i]) et, si le caractère est valide (lettre de l'alphabet), le décalage est effectué en fonction de la clé. Si le caractère est une lettre majuscule, il est d'abord converti en minuscule.

Le décalage est effectué en ajoutant la valeur de la clé à la valeur ASCII du caractère, et si cette somme dépasse la lettre 'z', une boucle permet de revenir dans la plage des lettres minuscules.

Si un caractère n'est pas valide (pas une lettre), l'algorithme passe à la fonction `sous_vigenere` pour vérifier et passer au caractère suivant. Enfin, la chaîne codée est affichée à la fin du processus.

2.3 Pseudo-code Sous Vigenère

Algorithm 3 Vérification de la validité du caractère

```
1: début :
2: fin_chaine ← 1
3: while la chaîne n'est pas terminée do
4:     i ← i + 1                                  ▷ (i = indice chaîne)
5:     c ← chaine[i]
6:     if c = 0 then
7:         fin_chaine ← 0
8:     end if
9:     if A ≤ c ≤ Z then
10:        return
11:    else if a ≤ c ≤ z then
12:        return
13:    end if
14: end while
15: fin
```

2.4 Fonctionnement du code sous_vigenere

La fonction `sous_vigenere` est utilisée pour vérifier si un caractère dans la chaîne à chiffrer (`chaine`) est une lettre valide (majuscule ou minuscule).

Elle commence par initialiser une variable `fin_chaine` à 1, ce qui indique que la chaîne est valide au début. Ensuite, elle boucle sur chaque caractère de la chaîne à partir de l'indice i.

- Si le caractère est nul (fin de chaîne), la fonction définit `fin_chaine` à 0 et termine la vérification.
- Si le caractère est une lettre majuscule (de A à Z), la fonction `return` pour indiquer qu'il s'agit d'un caractère valide.
- Si le caractère est une lettre minuscule (de a à z), la fonction `return`.
- Si le caractère n'est ni une lettre majuscule ni une lettre minuscule, la fonction continue la boucle pour vérifier le prochain caractère.

Si la boucle se termine sans trouver de caractère valide, la fonction quitte sans retourner de résultat explicite. Sa logique sert à gérer la progression à travers les caractères de la chaîne jusqu'à ce qu'un caractère valide soit trouvé.