Faculté d'ingénierie et de technologie – Génie informatique

Algorithmique et programmation

Damien Berthet & Vincent Labatut

TP 51 | détection de palindromes

Présentation

Le but de ce TP est d'approfondir des notions déjà étudiées en cours et lors des TP précédents, en particulier : chaînes de caractères, pointeurs, allocation dynamique, fonctions récursives, piles de données.

1 Notion de palindrome

Un *palindrome* est un mot symétrique, c'est-à-dire que l'on peut le lire dans les deux sens (en partant du début ou de la fin). Exemples de mots constituants des palindromes : rotor, kayak, radar, selles, été… Ce TP porte sur l'implémentation de différents algorithmes permettant de déterminer si une chaîne de caractères donnée est un palindrome.

2 Par renversement

Exercice 1

Écrivez une fonction `inverse_chaine(char *chaine)` qui reçoit en paramètre un pointeur sur une chaîne de caractères `chaine` et construit une nouvelle chaîne correspondant à l'inverse de `chaine`. L'espace mémoire occupé par cette nouvelle variable doit être *minimal*. La fonction doit renvoyer un pointeur vers la nouvelle chaîne, ou la valeur `NULL` en cas d'erreur.

exemple : appliquée à une chaîne `"abcd"`, la fonction doit renvoyer un pointeur sur une chaîne `"dcba"`.

Exercice 2

Écrivez une fonction récursive `compare_chaines(char *chaine1, char *chaine2)` qui compare caractère par caractère les deux chaînes passées en paramètres. La fonction renvoie 1 si les deux chaînes sont identiques et 0 sinon.

exemples :

- L'appel `compare_chaines("abcd","abce")` renvoie 0 ;
- L'appel `compare_chaines("abcd","abcd")` renvoie 1.

Exercice 3

En utilisant les fonctions `inverse_chaine` et `compare_chaines`, écrivez une fonction `int est_palindrome1(char *chaine)` qui détermine si une chaîne de caractères est un palindrome : elle doit renvoyer 1 si la chaîne est un palindrome, et 0 sinon.

exemples :

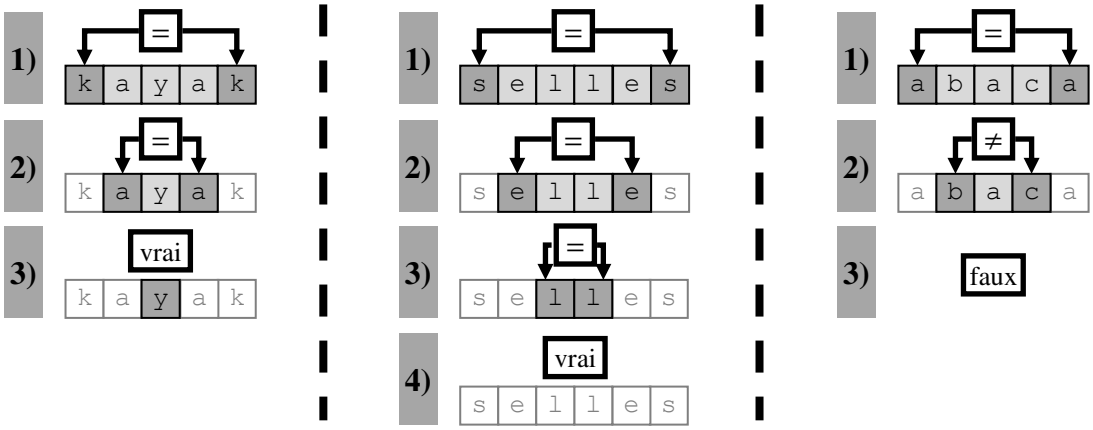
- L'appel `est_palindrome1("abcd","eizyuepzep")` renvoie 0 ;
- L'appel `est_palindrome1("abcd","dcba")` renvoie 1.

3 Par double parcours

Un autre algorithme pour déterminer si une chaîne de caractères est un palindrome consiste à comparer les caractères en partant des deux extrémités (le début et la fin) et en se déplaçant vers le centre :

- On continue à se déplacer vers le centre tant que les caractères sont égaux.
- On s'arrête quand on arrive au centre (c'est un palindrome) ou quand les deux caractères comparés ne sont pas égaux (ce n'est pas un palindrome).

exemples : on considère successivement les chaînes `"kayak"`, `"selles"` et `"abaca"` :



Exercice 4

Écrivez une fonction itérative `int est_palindrome2_it(char *chaine)` qui prend une chaîne de caractères en paramètre et implémente l'algorithme du double parcours pour déterminer s'il s'agit d'un palindrome. La fonction renvoie 1 si le mot est un palindrome et 0 sinon.

Exercice 5

Exprimez l'algorithme précédent sous forme récursive, en précisant les éventuels cas d'arrêt, cas d'erreur et cas général (ou cas généraux).

Écrivez ensuite la fonction récursive `int est_palindrome2_rec_sec(char *debut, char *fin)` qui implémente cet algorithme. Les paramètres `debut` et `fin` pointent respectivement sur le premier et le dernier caractères de la chaîne à traiter.

Exercice 6

Enfin, écrivez la fonction `int est_palindrome2_rec(char *chaine)`, qui initialise les paramètres `debut` et `fin`, puis effectue le premier appel de la fonction `est_palindrome2_rec_sec`.

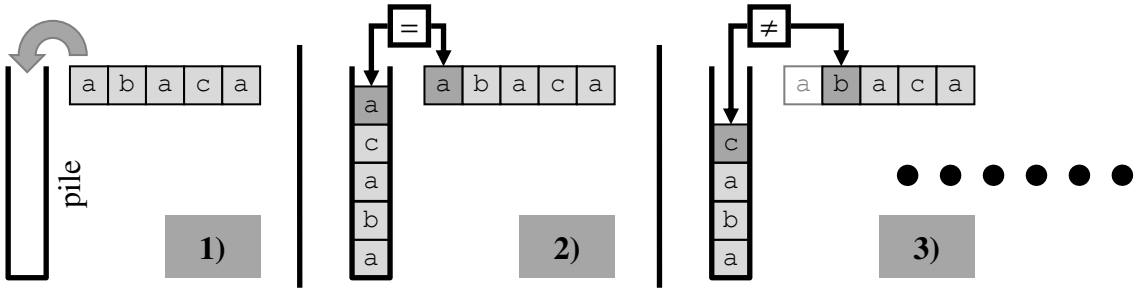
4 Par empilement

Le dernier algorithme utilise une pile de caractères pour effectuer l'analyse. Pour ces exercices, vous disposez de la bibliothèque `pile_liste` définie en cours et utilisée en TP. Elle a été adaptée pour pouvoir traiter des piles de valeurs de type `char`.

La première étape de l'algorithme consiste à empiler le contenu de la chaîne dans la pile, caractère par caractère. La deuxième étape consiste à comparer la chaîne avec le contenu de la pile, caractère par caractère. On considère le premier caractère de la chaîne et le sommet de la pile :

- S'ils sont différents, alors la chaîne n'est pas un palindrome.
- Sinon, on dépile et on recommence avec le caractère suivant dans la chaîne.

- Si la pile est vide, c'est qu'il s'agit d'un palindrome.



Exercice 7

Écrivez une fonction `int initialise_pile(char *chaine, pile *p)` qui reçoit un pointeur sur une pile vide et y recopie les caractères composant la chaîne reçue en paramètre. La fonction renvoie la valeur −1 si une erreur se produit, et 0 sinon.

Exercice 8

En utilisant la fonction `initialise_pile` et l'algorithme présenté ci-dessus, écrivez une fonction `int est_palindrome3(char *chaine)` qui détermine si une chaîne de caractères est un palindrome. La fonction renvoie les mêmes valeurs que les autres fonctions `est_palindromex`.

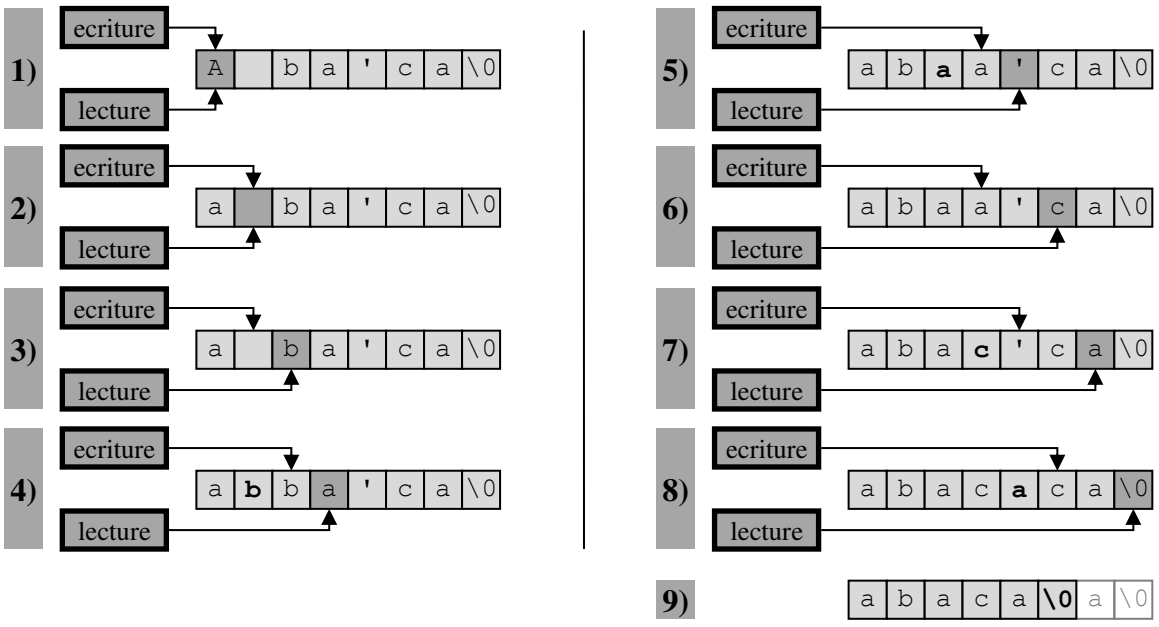
5 Généralisation

On peut généraliser la notion de palindrome à une phrase. Dans ce cas-là, on ignore généralement la ponctuation, les accents et les espaces, par exemple : "Élu par cette crapule !", "Engage le jeu, que je le gagne.", "La mère Gide digère mal."…

Exercice 9

On veut utiliser les fonctions précédentes pour traiter ces palindromes. Pour cela, il suffit de nettoyer les chaines de caractères, de manière à ne garder que les lettres. De plus, les lettres majuscules doivent être transformées en minuscules. On veut effectuer ce traitement en modifiant directement la chaine concernée (i.e. sans passer par un autre tableau).

exemple : sur la phrase `"A ba'ca"` (phrase qui n'a aucune signification)



Écrivez une fonction récursive `void nettoie_chaine(char *lecture, char *ecriture)` qui effectue le nettoyage décrit ci-dessus. Le pointeur `lecture` indique quel est le prochain caractère à traiter, le pointeur `ecriture` indique à quel endroit le prochain caractère devra être recopié.

Exercice 10

Écrivez une fonction `int est_palindrome_phrase(char *phrase)` qui teste si la phrase passée en paramètre est un palindrome, en utilisant les fonctions des exercices précédents.

exemple : pour la phrase "Engage le jeu, que je le gagne !", la fonction doit renvoyer la valeur 1.

Remarque : par souci de simplification, on supposera que la phrase ne contient pas de caractères accentués.