Niveau 3

TP
S4. Développement logiciel
4.3 - Structure et gestion des données (Types dérivés : tableaux, chaînes de caractères)

Les chaînes de Dick

TABLE DES MATIÈRES

1 - CHAÎNES DE CARACTÈRES

Historiquement, en C, une chaîne de caractères est en fait un tableau à une dimension, dont chaque case contient un unique caractère.

La fin d'une telle chaîne de caractères est définie par un caractère 'spécial '\0'. Il est alors nécessaire que ce tableau possède une case supplémentaire par rapport à la longueur maximale de la chaîne de caractères désirée.

La syntaxe en C pour déclarer une chaîne de caractères est :

```
char sMaChaine[] = "Bonjour" ;
```

Dans ce cas, le tableau sMaChaine est automatiquement dimensionné à 7 caractères ("Bonjour") suivis d'un 8ème pour indiquer la fin de chaîne ('\0'). Chaque lettre occupe une case du tableau :

- sMaChaine[0] contient la première lettre de la chaîne ('B') ;
- sMaChaine[1] contient la deuxième ('O') ;
- etc.
- sMaChaine[7] contient automatiquement le caractère fin de chaîne ('\0').

Il n'est pas possible de modifier la totalité de la chaîne par une affectation unique : c'est un tableau, on doit alors le scruter case par case ! Donc sMaChaine[] = "Salut" ne fonctionnera pas ! Pas plus que les comparaisons !

Par contre, grâce aux flux d'entrée/sortie cin et cout, l'affichage et la saisie au clavier s'effectuent comme pour une variable classique.

Pour la saisie au clavier, tous les caractères tapés seront placés dans le tableau lorsque l'on appuie sur la touche entrée.

```
cin >> sMaChaine ;
```

Attention :

- Seuls les caractères saisis avant une espace ou une tabulation seront pris en compte ;
- le caractère de fin de chaîne '\0' est ajouté à la fin de la chaîne automatiquement ;
- si le tableau est de taille plus petite que la chaîne saisie, on risque de faire planter la mémoire du programme.

Pour l'affichage, le flux de sortie cout effectue une boucle Tant Que qui permet de parcourir le tableau chaîne et d'afficher chaque caractère les uns après les autres sans se poser plus de question. La boucle s'arrête dès que le caractère de fin de chaîne est trouvé (mais, lui, n'est pas affiché !). Si pour une raison ou une autre, la chaîne n'est pas terminée correctement par le caractère '\0', le cout continuera d'afficher le contenu de la mémoire situé après le tableau de la chaîne, au risque de faire exploser la mémoire.

```
cout << sMaChaine << endl ;
```

En C, une chaîne de caractères devra obligatoirement être travaillée comme un tableau classique.

En C++, il existe une classe string spécifique pour faciliter le travail sur les chaînes de caractères. Elle est très puissante, mais elle reste inadaptée dans le cadre de notre type de programmation : la plupart des données que nous aurons à traiter sont des trames de communication (cf. TP précédent !) que nous recevront sous forme de tableaux de caractères, et donc de chaînes de caractères au sens du C classique.

Monsieur Técenfote doit résoudre de petits problèmes par rapport à des mots ou de courtes phrases. En effet, Dick, dont la passion est les mots croisés et autres jeux de lettres, souhaite...

1) ... connaître la taille d'une chaîne de caractères. Proposez-lui un programme pour lui simplifier la tâche.

2) ... savoir combien il y a de voyelles dans une chaîne de caractères. Notez que les accentuées ne seront pas prises en compte dans cet exercice.

3) ... recopier une chaîne de caractères dans une autre. La première chaîne (`sSource`) sera à déclarer comme indiqué ci-dessus, tandis que l'autre (`sDestination`) le sera comme un tableau de caractères de taille largement supérieure, 100 caractères maxi par exemple. Pour bien voir si votre programme fonctionne, il aura fallu, en tout premier, initialiser la chaîne `sDestination` avec des valeurs aléatoires comprises entre 1 et 255 !

4) ... créer des mots de passes automatiquement à partir d'une petite phrase. Le principe de création est simple : les voyelles, espaces et ponctuations sont tout bonnement supprimées, les autres caractères seront copiés dans le mot de passe. Par exemple la phrase de base "`Salut a tous !`" deviendra le mot de passe "`Sltts`". Note : les majuscules et minuscules ne sont pas changées ! Et les accentuées ne sont pas non plus utilisées.

5) ... vérifier si une chaîne de caractères est un palindrome. C'est un mot ou une phrase qui peut se lire indifféremment de droite à gauche ou de gauche à droite. Par exemple : `laval`, `kayak`, `Ésope reste ici et se repose`... Note : les accents, les majuscules, les espaces, la ponctuation... ne sont pas utilisés dans cet exercice !

6) ... tester si une chaîne de caractère est intégralement dans une autre chaîne plus grande. Par exemple "`jour`" est bien dans "`Bonjour`" ! Dick aimerait aussi connaître à quel emplacement (indice du tableau), le début de la petite chaîne a été trouvé. Si elle n'y est pas en entier voire pas du tout, l'emplacement sera indiqué à `-1`. Dans l'exemple précédent, l'emplacement trouvé sera le `3` !

2 - TRI D'UN TABLEAU À 1 DIMENSION

Il existe pléthores méthodes de tri d'un tableau. Nous allons en étudier une en particulier.

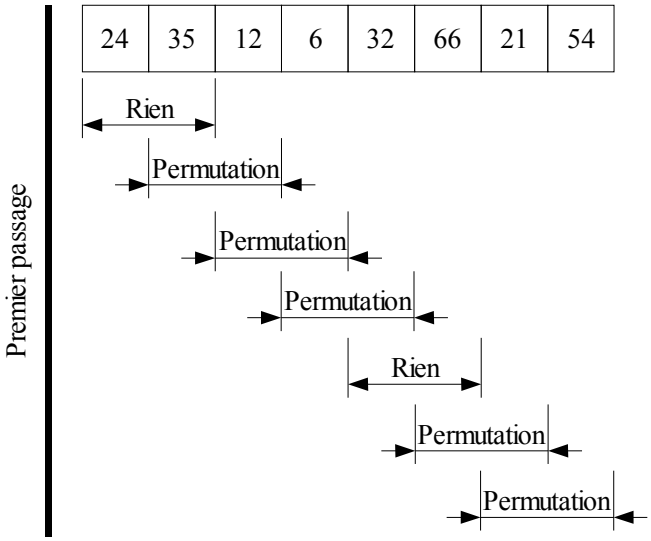
Le tri est ici réalisé en propageant par permutations successives le plus grand élément du tableau vers la fin de celui-ci et donc le plus petit se déplace vers le début. De la même façon que les bulles d'air montent à la surface de l'eau, le plus petit élément, « plus léger » que les autres, gagne de proche en proche la « surface » (extrémité du tableau d'indice 0).

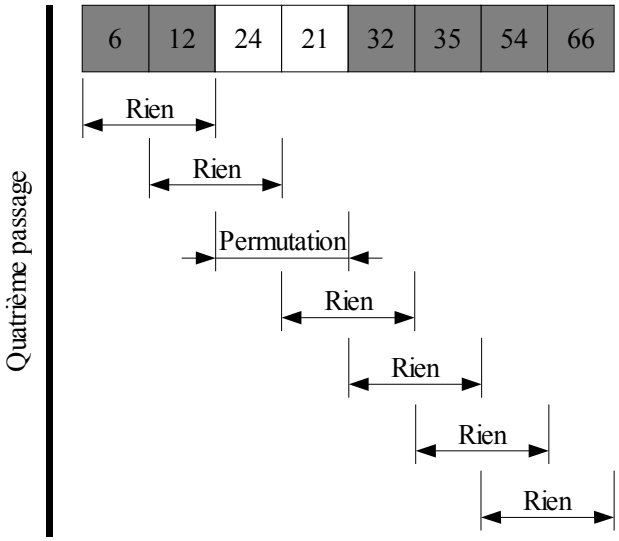
Tout le tableau est parcouru au moins une fois. L'élément courant est comparé avec son voisin suivant immédiat. Si ces deux valeurs sont classées entre-elles, il ne se passe rien et on passe à l'élément suivant. Si par contre, elles ne sont pas bien ordonnées, on les permute et on passe à l'élément suivant du tableau. Un élément plus lourd que les autres s'enfonce vers la fin du tableau jusqu'à ce qu'il rencontre un élément encore plus grand.

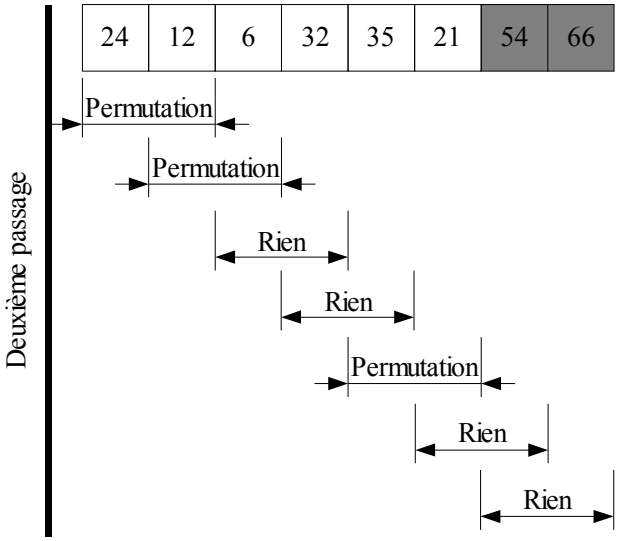
Une fois que le tableau a été parcouru en entier de cette façon, on pourrait croire qu'il est trié. Eh bien pas forcément : s'il y a eu une permutation, il se peut que l'élément qui est monté (si le plus grand s'enfonce, le plus petit, lui, va monter) ne soit pas bien placé. Il faut donc tout reprendre depuis le début... Et ceci jusqu'à ce qu'aucune permutation ne soit réalisée dans le parcours du tableau.

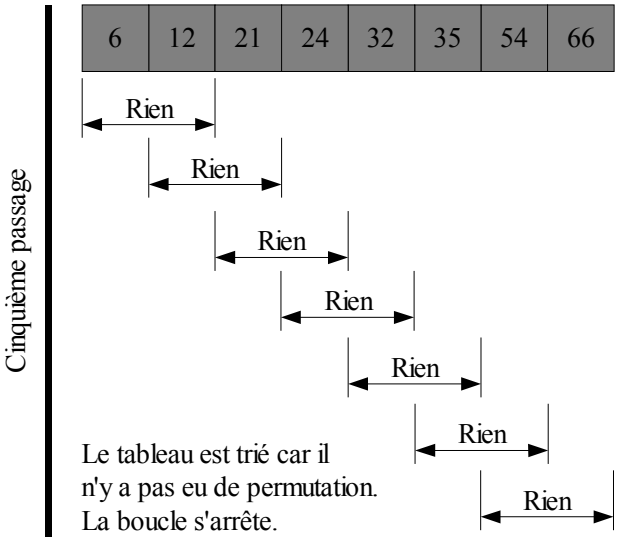
7) Codez le tri de cette façon d'un tableau de 20 entiers, initialisé de manière aléatoire. Vous afficherez le contenu complet du tableau avant puis après le tri.

Exemple de ce type de tri d'un tableau :









Le tableau est trié car il n'y a pas eu de permutation. La boucle s'arrête.