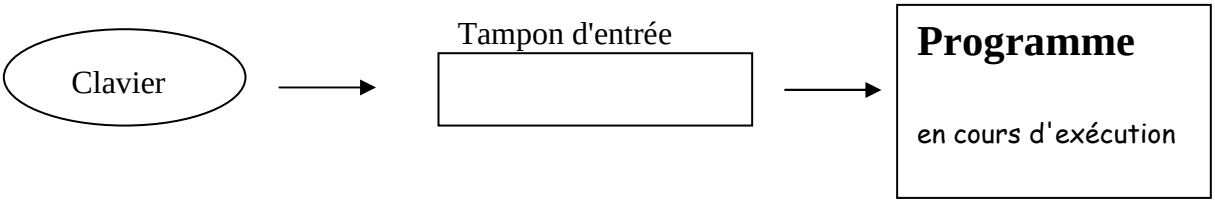
TP Algorithmique : Chaînes de caractères C

Il faudra pour tous les exercices effectuer le codage en C++ et rendre un fichier source d'extension .cpp

Toutes les saisies se font par l'intermédiaire d'un **tampon** (ou buffer) **d'entrée** :



Le contenu du tampon est "analysé" lorsque l'utilisateur "tape" sur la touche Entrée (le caractère Line Feed (LF, '\n' en langage C) est alors écrit dans le tampon).

Dans tous les exercices, les **saisies des chaînes de caractères devront être sécurisées** sans pour autant surdimensionner les tableaux de caractères. Si l'utilisateur saisit un mot trop long, il ne doit pas y avoir de débordement de tableau.
Cela nécessite l'utilisation du **manipulateur paramétrique setw()** dans le cas de saisies de chaînes de caractères avec **l'opérateur** >> (=> il faut inclure le fichier <iomanip>).

Exemple :

```
#include <iostream>
#include <iomanip>          // utilisation du manipulateur paramétrique setw()
using namespace std;

int main()
{
        char Mot1 [5]= {'B','o','n', '\0'} ;
        char Mot2 [5];

        cout << "Mot1?\n";
        cin >> setw(5)>> Mot1;      // la longueur max de la chaîne saisie est égale à 4 : 5 - 1
                                    // => la saisie conduira à écrire au maximum 5 caractères dans le tableau Mot1
                                    // (en comptant le caractère de fin de chaîne : le caractère '\0')

        cout << "Mot2?\n";
        cin >> setw(5)>>Mot2;

        cout << "Le mot 1 : " << Mot1 << endl;
        cout << "Le mot 2 : " << Mot2 << endl;

        return 0;
}
```

Cet exemple permet bien de sécuriser la saisie de la chaîne *Mot1*.
En revanche, si l'utilisateur saisit une chaîne trop longue, il n'a pas la main pour saisir la seconde chaîne de caractères.

Explication :
S'il **saisit "123456\n"** à la suite de l'invitation "Mot1?", ces 7 caractères sont transmis dans le tampon d'entrée et analysés au moment de l'appui sur la touche Entrée. Les 4 premiers caractères sont extraits du tampon et affectés dans le tableau *Mot1*. Le système place également le **caractère '\0' dans le tableau Mot1** après le caractère '4'.
A ce niveau là, les autres caractères (**"56\n"**) sont toujours présents dans le tampon d'entrée.
Conséquence : lors de la seconde saisie, l'utilisateur n'a pas la main et les 2 caractères "56" sont à leur tour extraits du tampon pour être affectés dans le tableau *Mot2*. Le système place également le **caractère '\0' dans le tableau Mot2** après le caractère '6'.

Solution :
Pour corriger ce problème, il faut **vider le tampon d'entrée entre les 2 saisies**. Pour cela, vous pouvez utiliser l'instruction :

```
cin.ignore(255, '\n');
```

Remarque :
Pour **saisir une chaîne de caractères contenant des espaces,** on ne peut pas utiliser l'opérateur >>. Il faut utiliser la fonction getline() :
→ syntaxe :

```
char ch[5];
cin.getline(ch, 5) ; // saisie d'une chaîne de caractères ch pouvant contenir des
                     // espaces et de longueur maximale 4 (sans compter le caractère
                     // terminateur '\0').
```

Travail à réaliser :

Exercice n°1 :
Écrire un programme qui calcule et affiche la longueur d'une chaîne de caractères saisie au clavier, **sans utiliser la fonction strlen().**

Exercice n°1bis :
Écrire un programme qui calcule et affiche la longueur d'une chaîne de caractères saisie au clavier, en utilisant la fonction strlen().

Exercice n°2 :
Écrire un programme qui lit au clavier un mot (d'au plus 30 caractères) et qui l'affiche à l'envers.

Exercice n°3 :
Ecrire un programme qui lit au clavier 2 mots (d'au plus 10 caractères), concatène le second mot à la suite du premier **sans utiliser la fonction strcat()** et affiche le résultat.

Exercice n°3bis :
Ecrire un programme qui lit au clavier 2 mots (d'au plus 10 caractères), les concatènent en utilisant la fonction strcat() et affiche le résultat.

Exercice n°4 :
Écrire un programme déterminant le nombre de lettres « e » (minuscules) présentes dans un texte de moins d'une ligne (supposée ne pas dépasser 80 caractères) fournie au clavier.

Exercice n°5 :
Ecrire un programme qui demande à l'utilisateur de saisir une phrase constituée de lettres minuscules (longueur max 80 caractères), puis décale chaque caractère de 2 lettres (**a** devient **c**, **b** devient **d**, **c** devient **e**,…, **y** devient **a** et **z** devient **b**.) et affiche le résultat.

Exemple : si l'utilisateur saisit « **ma petite phrase** » le résultat à afficher est « **oc rgvkvg rjtcug** »

Exercice n°6 : Conjugaison d'un verbe du 1er groupe
Écrire un programme qui lit un verbe du premier groupe et qui en affiche la conjugaison au présent de l'indicatif, sous la forme :

je chante
tu chantes
il chante
nous chantons
vous chantez
ils chantent

Le programme devra vérifier que le mot fourni se termine bien par « er ». On supposera qu'il ne peut comporter plus de 26 lettres et qu'il s'agit d'un verbe régulier. Autrement dit, l'utilisateur ne fournira pas un verbe tel que « manger » (le programme afficherait alors : « nous mangons »).

Exercice n°6 bis : Conjugaison d'un verbe du 1er groupe
Même problème à résoudre mais on vous demande de stocker dans une chaîne de caractères nommée par exemple *ligne* la chaîne « je chante » puis d'afficher la chaîne *ligne*. Ensuite, il faut stocker dans *ligne* « tu chantes » et afficher *ligne* etc…
Cela vous permettra de manipuler les fonctions strlen(), strcpy(), strcat(),…

Il est possible de stocker dans une seule variable tableau plusieurs chaînes de caractères. Pour cela, il faut utiliser un **tableau à 2 dimensions contenant des éléments de type caractère**.
Le fichier source « tableauChar2dim.cpp » ci-joint illustre comment on peut manipuler un tableau contenant plusieurs chaînes de caractères.

Exercice n°7 : Système de contrôle d'accès au bâtiment

Suite de l'exercice 6 du TP7.

On souhaite modifier l'identifiant du gardien.
Nouvel identifiant du gardien : 0123456789.

Pour pouvoir différencier l'identifiant 0123456789 (valide) de l'identifiant 123456789 (non valide), il est nécessaire de **stocker chaque identifiant dans une chaîne de caractères**.
Idem pour les codes d'accès.

De plus, on souhaite ajouter une nouvelle fonctionnalité : un même employé ne doit être autorisé à entrer dans le bâtiment qu'une seule fois : s'il passe son badge et saisit son code d'accès une seconde fois, l'accès au bâtiment doit lui être refusé. Il est donc nécessaire de **gérer la présence (ou absence)** des différents employés.

Dans cette nouvelle version du programme, les identifiants, les codes d'accès et la présence des employés devront être mémorisés dans 3 tableaux.