# Chapitre 6 : Chaînes de caractères

## *Exercice 6.1 Analyse d'un caractère*

Ecrire un programme C++ qui, après avoir demandé à l'utilisateur de saisir un caractère, détermine / affiche à l'écran si ce caractère :

1°) est une lettre de l'alphabet
2°) est une lettre minuscule de l'alphabet
3°) est un chiffre
4°) est un symbole de ponctuation

Le résultat doit être affiché comme dans l'exemple d'exécution suivant :

```
Entrez un caractere : a

Code ASCII de 'a'                   : 97
'a' est une lettre de l'alphabet    : vrai
'a' est une lettre minuscule        : vrai
'a' est un chiffre                  : faux
'a' est un caractere de ponctuation : faux
```

## *Solution exercice 6.1*

```
#include <cctype>
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

string vraiFaux(bool b) {
   return b ? "vrai" : "faux";
}

int main() {
   unsigned char c;
   cout << "Entrez un caractere : ";
   cin >> c;
   cout << endl
        << "Code ASCII de '" << c << "'                   : "
        << (int) c << endl
        << "'" << c << "' est une lettre de l'alphabet    : "
        << vraiFaux(isalpha(c)) << endl
        << "'" << c << "' est une lettre minuscule        : "
        << vraiFaux(islower(c)) << endl
        << "'" << c << "' est un chiffre                  : "
        << vraiFaux(isdigit(c)) << endl
        << "'" << c << "' est un caractere de ponctuation : "
        << vraiFaux(ispunct(c)) << endl;

   return EXIT_SUCCESS;
}
```

## Exercice 6.2 Constructeurs

Indiquer ce que va afficher[1] chacune des séquences d'instructions suivantes :

[1] *Il se peut qu'une séquence provoque une erreur à la compilation ou à l'exécution, voire ait un comportement indéfini. Le cas échéant, répondez "Erreur à la compilation", "Erreur à l'exécution" ou "Comportement indéfini"*

1. ```
   string s1 = "ABC";
   string s2(s1, 1);
   cout << '|' << s2 << '|' << endl;
   ```

2. ```
   string s1 = "ABC";
   string s2(s1, 2, 3);
   cout << '|' << s2 << '|' << endl;
   ```

3. ```
   string s1 = "ABC";
   string s2(s1, 3, 2);
   cout << '|' << s2 << '|' << endl;
   ```

4. ```
   string s1 = "ABC";
   string s2(s1, 4, 1);
   cout << '|' << s2 << '|' << endl;
   ```

5. ```
   string s("ABC", 3);
   cout << '|' << s << '|' << endl;
   ```

6. ```
   string s("ABC", 4);
   cout << '|' << s << '|' << endl;
   ```

7. ```
   string s("ABC", 5);
   cout << '|' << s << '|' << endl;
   ```

8. ```
   string s(3, 'A');
   cout << '|' << s << '|' << endl;
   ```

## Solution exercice 6.2

1. `|BC|`
2. `|C|`
3. `||`
4. `Erreur à l'exécution`
5. `|ABC|`
6. `|ABC\0|`
7. `Comportement indéfini`
8. `|AAA|`

## *Exercice 6.3 assign, '+', '+=' et append*

Indiquer ce que va afficher[1] chacune des séquences d'instructions suivantes :

*[1] Il se peut qu'une séquence provoque une erreur à la compilation ou à l'exécution, voire ait un comportement indéfini. Le cas échéant, répondez "Erreur à la compilation", "Erreur à l'exécution" ou "Comportement indéfini"*

1. ```
   string s1 = "ABC";
   string s2;
   s2.assign(s1, 2, 3);
   cout << '|' << s2 << '|' << endl;
   ```

2. ```
   string s1 = "ABC";
   string s2;
   s2.assign(s1, 3, 2);
   cout << '|' << s2 << '|' << endl;
   ```

3. ```
   string s;
   s.assign("ABC", 2);
   cout << '|' << s << '|' << endl;
   ```

4. ```
   string s;
   s.assign(2, 65);
   cout << '|' << s << '|' << endl;
   ```

5. ```
   string s;
   s.assign(4, '\101');
   cout << '|' << s << '|' << endl;
   ```

6. ```
   string s;
   s.assign(4, '\x42');
   cout << '|' << s << '|' << endl;
   ```

7. ```
   string s = string("A") + string("BC");
   cout << '|' << s << '|' << endl;
   ```

8. ```
   string s = string("A") + "BC";
   cout << '|' << s << '|' << endl;
   ```

9. ```
   string s = 'A' + string("BC");
   cout << '|' << s << '|' << endl;
   ```

10. ```
    string s = "A" + "BC";
    cout << '|' << s << '|' << endl;
    ```

11. ```
    string s = "AB";
    s += 'C';
    cout << '|' << s << '|' << endl;
    ```
    ```

12.
```
string s1 = "";
string s2 = "123";
s1.append(s2);
cout << '|' << s1 << '|' << endl;
```

13.
```
string s1 = "";
string s2 = "123";
s1.append(s2, 1, 3);
cout << '|' << s1 << '|' << endl;
```

14.
```
string s;
s.append("123", 2);
cout << '|' << s << '|' << endl;
```

15.
```
string s;
s.append(2, '2');
cout << '|' << s << '|' << endl;
```

## *Solution exercice 6.3*

1. `|C|`
2. `||`
3. `|AB|`
4. `|AA|`
5. `|AAAA|`
6. `|BBBB|`
7. `|ABC|`
8. `|ABC|`
9. `|ABC|`
10. `Erreur à la compilation`
11. `|ABC|`
12. `|123|`
13. `|23|`
14. `|12|`
15. `|22|`

## *Exercice 6.4 Affichage d'un petit dessin*

Sans utiliser le concept de boucles, écrire, le plus judicieusement possible, un programme C++ permettant d'afficher à l'écran le dessin suivant :

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

## *Solution exercice 6.4*

```
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

int main() {

   const char SYMBOLE_1 = ' ';
   const char SYMBOLE_2 = '*';

   const string LIGNE_1 = string(3, SYMBOLE_1) + string(1, SYMBOLE_2);
   const string LIGNE_2 = string(2, SYMBOLE_1) + string(3, SYMBOLE_2);
   const string LIGNE_3 = string(1, SYMBOLE_1) + string(5, SYMBOLE_2);
   const string LIGNE_4 =                        string(7, SYMBOLE_2);

   const string DESSIN = LIGNE_1 + '\n' +
                         LIGNE_2 + '\n' +
                         LIGNE_3 + '\n' +
                         LIGNE_4 + '\n' +
                         LIGNE_3 + '\n' +
                         LIGNE_2 + '\n' +
                         LIGNE_1;

   cout << DESSIN << endl;

   return EXIT_SUCCESS;
}
```

A relever dans le code ci-dessus :

1) L'usage de constantes symboliques en général
2) L'usage de constantes symboliques pour les 2 symboles (caractères) utilisés dans le dessin.
   Avantage : Si le programme doit afficher des '1' au lieu des '*', par ex, il suffit de changer la valeur de SYMBOLE_2
3) L'usage de constantes symboliques pour les lignes
   Avantage : Permet d'éviter une certaine redondance du code (LIGNE_1 par ex est défini 1x mais utilisé 2x)

## Exercice 6.5 Vérifier qu'une chaîne de caractères est vide

On souhaite tester si une chaîne de caractères *str* de type string est vide ou non.
Proposer 4 manières différentes d'écrire le test.

## Solution exercice 6.5

```
if (str.empty())        // La meilleure formulation
if (str == "")
if (str.length() == 0)
if (str.size() == 0)
```

## Exercice 6.6 [ ], at, length, size, resize et substr

Indiquer ce que va afficher[1] chacune des séquences d'instructions suivantes :

[1] *Il se peut qu'une séquence provoque une erreur à la compilation ou à l'exécution, voire ait un comportement indéfini. Le cas échéant, répondez "Erreur à la compilation", "Erreur à l'exécution" ou "Comportement indéfini"*

1.
```
string s = "ABC";
cout << '|' << s[1] << '|' << endl;
```

2.
```
string s1 = "ABC";
string s2 = s1[1];
cout << '|' << s2 << '|' << endl;
```

3.
```
string s1 = "ABC";
string s2;
s2 = s1[1];
cout << '|' << s2 << '|' << endl;
```

4.
```
string s = "ABC";
cout << '|' << s[3] << '|' << endl;
```

5.
```
string s = "ABC";
cout << '|' << s[4] << '|' << endl;
```

6.
```
string s = "ABC";
cout << '|' << s.at(3) << '|' << endl;
```

7.
```
string s;
cout << s.length() << endl;
```

8.
```
string s = "ABC";
cout << s.length() << endl;
```

9.
```
string s = "ABC";
cout << s.size() << endl;
```

10.
```
string s = "ABC";
s.resize(5);
cout << '|' << s << '|' << endl;
```

11.
```
string s = "ABC";
s.resize(2, 'x');
cout << '|' << s << '|' << endl;
```

12.
```
string s = "ABC";
s.resize(4, 'x');
cout << '|' << s << '|' << endl;
```

13.
```
string s = "ABCDE";
cout << '|' << s.substr(1, 2) << '|' << endl;
```

14.
```
string s = "ABCDE";
cout << '|' << s.substr(0, 10) << '|' << endl;
```

15.
```
string s = "ABCDE";
cout << '|' << s.substr(3) << '|' << endl;
```

16.
```
string s = "ABCDE";
cout << '|' << s.substr() << '|' << endl;
```

## Solution exercice 6.6

1. `|B|`
2. `Erreur à la compilation (pas de constructeur string(char))`
3. `|B|`
4. `|\0|`[1]
5. `Comportement indéfini`
6. `Erreur à l'exécution`
7. `0`
8. `3`
9. `3`
10. `|ABC\0\0|`
11. `|AB|`
12. `|ABCx|`
13. `|BC|`
14. `|ABCDE|`
15. `|DE|`
16. `|ABCDE|`

[1] If *pos* is equal to the string length, the function returns a reference to the null character that follows the last character in the string.
A noter que avec.at(3) => erreur à l'exécution (cf question 6)

## Exercice 6.7 Milieu d'une chaîne de caractères

Ecrire une fonction C++ qui prend en paramètre une chaîne de caractères *str* (de type string) et qui retourne une chaîne de caractères (de type string) contenant :

- le caractère médian de *str*, si *str* comporte un nombre impair de caractères
- les 2 caractères médians de *str*, si *str* comporte un nombre pair de caractères

**Exemples** milieu("ABC") renvoie "B"; milieu("ABCD") renvoie "BC"

## Solution exercice 6.7

```
#include <cstdlib>
#include <iomanip>
#include <iostream>
#include <string>
using namespace std;

string milieu(const string& str);

int main() {

   const string SOURCE_TESTS = "ABCDEF";
   const size_t TAILLE = SOURCE_TESTS.length();
   const int W = (int) ("s = \"" + SOURCE_TESTS + "\"").length();

   string s;

   for (size_t len = 0; len <= TAILLE; ++len) {
      s = SOURCE_TESTS.substr(0, len);
      cout << setw(W) << left << "s = \"" + s + "\""
           << "  milieu = \"" << milieu(s) << "\"" << endl;
   }

   return EXIT_SUCCESS;
}

string milieu(const string& str) {
   const size_t TAILLE = str.length();
   string milieu;
   if (TAILLE > 0) { // Attention, test important pour éviter...
      if (TAILLE % 2 == 1) {
         milieu = str[TAILLE / 2]; // ... un problème ici
      } else {
         milieu = str.substr(TAILLE / 2 - 1, 2);
      }
   }
   return milieu;
}

// Autre variante possible
// string milieu(const string& str) {
//    const size_t TAILLE = str.length();
//    return TAILLE == 0 ? str :
//                         str.substr((TAILLE - 1) / 2, (TAILLE - 1) % 2 + 1);
// }

// s = ""        milieu = ""
// s = "A"       milieu = "A"
// s = "AB"      milieu = "AB"
// s = "ABC"     milieu = "B"
// s = "ABCD"    milieu = "BC"
// s = "ABCDE"   milieu = "C"
// s = "ABCDEF"  milieu = "CD"
```

## *Exercice 6.8 Inversion récursive d'une chaîne de caractères*

Ecrire une fonction récursive :

```
void inverser(string& str)
```

permettant d'inverser le contenu de la chaîne de caractères *str* passée en paramètre.

Ecrire également un petit programme de test permettant de vérifier le bon fonctionnement de la fonction *inverser*.

## *Solution exercice 6.8*

### Variante 1

```
#include <cstdlib>
#include <iomanip>
#include <iostream>
#include <string>
using namespace std;

void inverser(string& str);

int main() {

   const string SOURCE_TESTS = "ABCDEF";
   const size_t TAILLE = SOURCE_TESTS.length();
   const int W = int(("s = \"" + SOURCE_TESTS + "\"").length());

   string s, inverse;

   for (size_t len = 0; len <= TAILLE; ++len) {
      s = SOURCE_TESTS.substr(0, len);
      cout << setw(W) << left << "s = \"" + s + "\"";
      inverser(s);
      cout << "  inverse = \"" << s << "\"" << endl;
   }

   return EXIT_SUCCESS;
}

void inverser(string& str) {
   if (!str.empty()) {
      char c = str.at(0);
      inverser(str.erase(0, 1));
      str = str + c;
   }
}

// void inverser(string& str) {
//    if (!str.empty()) {
//       const size_t TAILLE = str.size();
//       char c = str.at(TAILLE - 1);
//       inverser(str.erase(TAILLE - 1, 1));
//       str = c + str;
//    }
// }

// s = ""        inverse = ""
// s = "A"       inverse = "A"
// s = "AB"      inverse = "BA"
// s = "ABC"     inverse = "CBA"
// s = "ABCD"    inverse = "DCBA"
// s = "ABCDE"   inverse = "EDCBA"
// s = "ABCDEF"  inverse = "FEDCBA"
```

## Variante 2

```
#include <cstdlib>
#include <iomanip>
#include <iostream>
#include <string>
using namespace std;

void inverser(string& str);
void inverserR(string& str, size_t pos1, size_t pos2);
void echanger(char& c1, char& c2);

int main() {

   const string SOURCE_TESTS = "ABCDEF";
   const size_t TAILLE = SOURCE_TESTS.length();
   const int W = int(("s = \"" + SOURCE_TESTS + "\"").length());

   string s, inverse;

   for (size_t len = 0; len <= TAILLE; ++len) {
      s = SOURCE_TESTS.substr(0, len);
      cout << setw(W) << left << "s = \"" + s + "\"";
      inverser(s);
      cout << "  inverse = \"" << s << "\"" << endl;
   }

   return EXIT_SUCCESS;
}

void inverser(string& str) {
   if (str.size() > 1) {
      inverserR(str, 0, str.size() - 1);
   }
}

void inverserR(string& str, size_t pos1, size_t pos2) {
   if (pos1 < pos2) {
      echanger(str.at(pos1), str.at(pos2));
      inverserR(str, pos1 + 1, pos2 - 1);
   }
}

void echanger(char& c1, char& c2) {
   char c3 = c1;
   c1 = c2;
   c2 = c3;
}

// s = ""        inverse = ""
// s = "A"       inverse = "A"
// s = "AB"      inverse = "BA"
// s = "ABC"     inverse = "CBA"
// s = "ABCD"    inverse = "DCBA"
// s = "ABCDE"   inverse = "EDCBA"
// s = "ABCDEF"  inverse = "FEDCBA"
```

## *Exercice 6.9 Numération romaine*

**Wikipédia** *La **numération romaine** est un système de numération additive utilisé par les Romains de l'Antiquité. Les chiffres romains sont représentés à l'aide de symboles combinés entre eux, notamment par les signes I, V, X, L, C, D et M, représentant respectivement les nombres 1, 5, 10, 50, 100, 500 et 1000.*
*Un nombre écrit en chiffres romains se lit de gauche à droite. En première approximation, sa valeur se détermine en faisant la somme des valeurs individuelles de chaque symbole, sauf quand l'un des symboles précède un symbole de valeur supérieure; dans ce cas, on soustrait la valeur du premier symbole au deuxième.*
*Exemple :* XIV = X + IV = 10 + (5 - 1) = 14

En s'inspirant de ce qui précède, on demande ici d'écrire une fonction (non récursive) permettant de convertir un nombre romain (supposé correct) en entier décimal.
Ecrire également un petit programme de test (*main*) permettant de vérifier le bon fonctionnement de votre fonction de conversion.

## Solution exercice 6.9

```
#include <cstdlib>
#include <iostream>
using namespace std;

int valeur(char chiffreRomain);
int conversion(const string& nombreRomain);

int main() {

   const string TESTS[] = {"I", "II", "III", "IV", "V",
                           "VI", "VII", "VIII", "IX", "X",
                           "MMMMDCCCLXXXVIII"};

   for (string s : TESTS) {
      cout << "\"" << s << "\" = " << conversion(s) << endl;
   }

   return EXIT_SUCCESS;
}

int valeur(char chiffreRomain) {
   switch (chiffreRomain) {
      case 'I' : return 1;
      case 'V' : return 5;
      case 'X' : return 10;
      case 'L' : return 50;
      case 'C' : return 100;
      case 'D' : return 500;
      case 'M' : return 1000;
      default : return 0;
   }
}

int conversion(const string& nombreRomain) {
   string s(nombreRomain);
   int somme = 0;
   while (!s.empty()) {
      if (s.length() == 1 || valeur(s.at(0)) >= valeur(s.at(1))) {
         somme += valeur(s.at(0));
         s.erase(0, 1);
      } else {
         somme += valeur(s.at(1)) - valeur(s.at(0));
         s.erase(0, 2);
      }
   }
   return somme;
}
```

```
// Autre variante un peu plus longue mais plus efficace
// car s.at(1) évalué que quand nécessaire
// int conversion(const string& nombreRomain) {
//     string s(nombreRomain);
//     int somme = 0;
//     while (!s.empty()) {
//        int chiffre_0 = valeur(s.at(0));
//        if (s.length() == 1) {
//           somme += chiffre_0;
//           s.erase(0, 1);
//        } else {
//           int chiffre_1 = valeur(s.at(1));
//           if (chiffre_0 >= chiffre_1) {
//              somme += chiffre_0;
//              s.erase(0, 1);
//           } else {
//              somme += chiffre_1 - chiffre_0;
//              s.erase(0, 2);
//           }
//        }
//     }
//     return somme;
// }

// Autre variante possible
// int conversion(const string& nombreRomain) {
//     if (nombreRomain.empty()) {
//        return 0;
//     } else if (nombreRomain.length() == 1) {
//        return valeur(nombreRomain.at(0));
//     } else {
//        const size_t TAILLE = nombreRomain.length();
//        int somme = 0;
//        for (size_t i = 0; i < TAILLE - 1; ++i) {
//           if (valeur(nombreRomain.at(i)) >= valeur(nombreRomain.at(i + 1))) {
//              somme += valeur(nombreRomain.at(i));
//              if (i == TAILLE - 2) {
//                 somme += valeur(nombreRomain.at(i + 1));
//              }
//           } else {
//              somme += valeur(nombreRomain.at(i + 1)) -
//                       valeur(nombreRomain.at(i));
//              ++i;
//           }
//        }
//        return somme;
//     }
// }

// "I" = 1
// "II" = 2
// "III" = 3
// "IV" = 4
// "V" = 5
// "VI" = 6
// "VII" = 7
// "VIII" = 8
// "IX" = 9
// "X" = 10
// "MMMMDCCCLXXXVIII" = 4888
```

## *Exercice 6.10 Analyse d'un nombre entier (2)*

Sans utiliser la librairie *cmath*, ni la librairie *sstream*, ni le concept de boucle mais en utilisant exclusivement les services offerts par la classe *string*, écrire un programme C++ qui, après avoir demandé à l'utilisateur de saisir un nombre entier (de type *int*) ≥ 0, affiche à l'écran combien de chiffres contient ce nombre et ce que valent le premier chiffre et le dernier chiffre du nombre.

**NB** On supposera ici la saisie utilisateur correcte.

### Exemple d'exécution

```
Entrez un nombre entier >= 0 : 123

Nombre saisi       : 123
Nombre de chiffres : 3
Premier chiffre    : 1
Dernier chiffre    : 3
```

## *Solution exercice 6.10*

```
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

int main() {

   int n;
   string s;
   size_t nbChiffres;
   char premierChiffre,
        dernierChiffre;

   cout << "Entrez un nombre entier >= 0 : ";
   cin >> n;

   s = to_string(n);

   nbChiffres = s.length(); // ou s.size()
   premierChiffre = s[0];
   dernierChiffre = s[nbChiffres - 1];

   cout << endl
        << "Nombre saisi       : " << n << endl
        << "Nombre de chiffres : " << nbChiffres << endl
        << "Premier chiffre    : " << premierChiffre << endl
        << "Dernier chiffre    : " << dernierChiffre << endl;

   return EXIT_SUCCESS;
}
```

NB Bien que la fonction to_string soit définie par la norme C++11, certains compilateurs ne l'implémentent pas encore. Dans ce cas, on peut résoudre le problème comme suit :

```
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

int main() {

   string s;
   size_t nbChiffres;
   char premierChiffre,
        dernierChiffre;

   cout << "Entrez un nombre entier >= 0 : ";
   cin >> s;

   nbChiffres = s.length(); // ou s.size()
   premierChiffre = s[0];
   dernierChiffre = s[nbChiffres - 1];

   cout << endl
        << "Nombre saisi       : " << s << endl
        << "Nombre de chiffres : " << nbChiffres << endl
        << "Premier chiffre    : " << premierChiffre << endl
        << "Dernier chiffre    : " << dernierChiffre << endl;

   return EXIT_SUCCESS;
}
```

## *Exercice 6.11 Prénom, nom et acronyme (1)*

Ecrire un programme C++ qui après avoir demandé à l'utilisateur de saisir son prénom et son nom (sur la même ligne), affiche à l'écran son prénom, son nom ainsi que son acronyme.

**NB** L'acronyme s'obtient en concaténant la première lettre du prénom, la première lettre du nom et la dernière lettre du nom, le tout mis entièrement en majuscules.

**IMPORTANT**
On supposera :
- que l'utilisateur n'entre qu'un seul prénom et qu'un seul nom
- que le nom ne contient aucun espace blanc

Il n'est pas demandé ici de tenir compte des cas où l'utilisateur n'entrerait que son prénom, que son nom, voire ni son prénom, ni son nom.

### Exemple d'exécution

```
Entrez votre prenom et votre nom : Alexandre Dumas

Votre prenom est    : Alexandre
Votre nom est       : Dumas
Votre acronyme est : ADS
```

## *Solution exercice 6.11*

```
#include <cctype>
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

int main() {

   string prenom,
          nom,
          acronyme;

   cout << "Entrez votre prenom et votre nom : ";
   cin >> prenom >> nom;

   acronyme = string("") // ou ""s mais pas ""
              + (char) toupper(prenom[0])
              + (char) toupper(nom[0])
              + (char) toupper(nom[nom.length() - 1]);

   cout << endl
        << "Votre prenom est    : " << prenom << endl
        << "Votre nom est       : " << nom << endl
        << "Votre acronyme est : " << acronyme << endl;

   return EXIT_SUCCESS;
}
```

## *Exercice 6.12 insert, replace et erase*

Indiquer ce que va afficher[1] chacune des séquences d'instructions suivantes :

*[1] Il se peut qu'une séquence provoque une erreur à la compilation ou à l'exécution, voire ait un comportement indéfini. Le cas échéant, répondez "Erreur à la compilation", "Erreur à l'exécution" ou "Comportement indéfini"*

1.
```
string s1 = "ABC";
string s2 = "123";
s1.insert(1, s2);
cout << '|' << s1 << '|' << endl;
```

2.
```
string s1 = "ABC";
string s2 = "123";
s1.insert(4, s2);
cout << '|' << s1 << '|' << endl;
```

3.
```
string s1 = "ABC";
string s2 = "123";
s1.insert(2, s2, 1, 2);
cout << '|' << s1 << '|' << endl;
```

4.
```
string s1 = "ABC";
string s2 = "123";
s1.insert(2, s2, 0, string::npos);
cout << '|' << s1 << '|' << endl;
```

5.
```
string s = "ABC";
s.insert(3, "123", 2);
cout << '|' << s << '|' << endl;
```

6.
```
string s = "ABC";
s.insert(1, 2, '3');
cout << '|' << s << '|' << endl;
```

7.
```
string s1 = "ABCDEF";
string s2 = "123";
s1.replace(1, 2, s2);
cout << '|' << s1 << '|' << endl;
```

8.
```
string s1 = "ABCDEF";
string s2 = "123";
s1.replace(2, 4, s2);
cout << '|' << s1 << '|' << endl;
```

9.
```
string s1 = "ABCDEF";
string s2 = "123";
s1.replace(1, 2, s2, 1, 2);
cout << '|' << s1 << '|' << endl;
```

10.
```
string s = "ABCDEF";
s.replace(2, 3, "123", 2);
cout << '|' << s << '|' << endl;
```

11.
```
string s = "ABCDEF";
s.erase();
cout << '|' << s << '|' << endl;
```

12.
```
string s = "ABCDEF";
s.erase(2);
cout << '|' << s << '|' << endl;
```

13.
```
string s = "ABCDEF";
s.erase(2, 2);
cout << '|' << s << '|' << endl;
```

## *Solution exercice 6.12*

1. `|A123BC|`
2. `Erreur à l'exécution (NB s1.insert(3, s2) serait OK)`
3. `|AB23C|`
4. `|AB123C|`
5. `|ABC12|`
6. `|A33BC|`
7. `|A123DEF|`
8. `|AB123|`
9. `|A23DEF|`
10. `|AB12F|`
11. `||`
12. `|AB|`
13. `|ABEF|`

## *Exercice 6.13 Compter le nombre d'occurrences*

Ecrire une fonction qui renvoie combien de fois un caractère donné se trouve dans une chaîne de caractères (de type *string*) donnée.

Proposer deux implémentations de la fonction :

- L'une utilisant la fonction *find* de la classe *string*
- L'autre utilisant la librairie *algorithm*

## *Solution exercice 6.13*

### Variante 1

```
size_t find (char c, size_t pos = 0) const;
```

```
size_t nbOcc(const string& str, char c) {
   size_t nbOcc = 0,
          found = 0;
   while ( (found = str.find(c, found)) != string::npos ) {
      nbOcc++;
      found++;
   }
   return nbOcc;
}
```

### Variante 2

```
template <class InputIterator, class T>
  typename iterator_traits<InputIterator>::difference_type
    count (InputIterator first, InputIterator last, const T& val);
```

```
size_t nbOcc(const string& str, char c) {
   return count(str.begin(), str.end(), c);
}
```

## *Exercice 6.14 Analyse d'une adresse*

Ecrire un programme C++ qui, après avoir demandé à l'utilisateur de saisir (sur la même ligne) un nom et un numéro de rue, affiche séparément à l'écran, d'une part le nom de la rue, d'autre part le numéro de la rue.

### Exemple d'exécution

```
Entrez le nom et le numero de la rue : Impasse des Acacias 24a

Le nom de la rue est : Impasse des Acacias
Le no de la rue est  : 24a
```

## *Solution exercice 6.14*

```
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

int main() {

   const string CHIFFRES = "0123456789";

   string rue;
   size_t pos_no_rue;

   cout << "Entrez le nom et le numero de la rue : ";
   getline(cin, rue);

   pos_no_rue = rue.find_first_of(CHIFFRES);

   cout << endl
        << "Le nom de la rue est : " << rue.substr(0, pos_no_rue - 1) << endl
        << "Le no de la rue est  : " << rue.substr(pos_no_rue) << endl;

   return EXIT_SUCCESS;
}
```

A noter que la solution ci-dessous (contrairement à celle-ci-dessus) posera un problème si l'utilisateur entre malencontreusement des espaces blancs après le no de la rue.

```
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

int main() {

   string rue;
   size_t pos_dernier_espace;

   cout << "Entrez le nom et le numero de la rue : ";
   getline(cin, rue);

   pos_dernier_espace = rue.find_last_of(" ");

   cout << endl
        << "Le nom de la rue est : " << rue.substr(0, pos_dernier_espace) << endl
        << "Le no de la rue est  : " << rue.substr(pos_dernier_espace + 1) << endl;

   return EXIT_SUCCESS;
}
```

## *Exercice 6.15 Prénom, nom et acronyme (2)*

Même énoncé que celui de l'exercice 6.11, à la différence importante près que, ici, il est demandé de tenir compte des cas où le nom serait constitué de plusieurs mots.

### Exemples d'exécution

```
Entrez votre prenom et votre nom : Ludwig van Beethoven

Votre prenom est   : Ludwig
Votre nom est      : van Beethoven
Votre acronyme est : LVN


Entrez votre prenom et votre nom : Valery Giscard d'Estaing

Votre prenom est   : Valery
Votre nom est      : Giscard d'Estaing
Votre acronyme est : VGG
```

## *Solution exercice 6.15*

```
#include <cctype>
#include <cstdlib>
#include <iostream>
#include <string>
using namespace std;

int main() {

   string s,
          prenom,
          nom,
          acronyme;

   size_t posPremiereLettrePrenom,
          posPremierBlancApresPrenom,
          posPremiereLettreNom,
          posDerniereLettreNom;

   cout << "Entrez votre prenom et votre nom : ";
   getline(cin, s);

   posPremiereLettrePrenom    = s.find_first_not_of(" \t");
   posPremierBlancApresPrenom = s.find_first_of(" \t", posPremiereLettrePrenom);
   posPremiereLettreNom       = s.find_first_not_of(" \t",
                                                    posPremierBlancApresPrenom);
   posDerniereLettreNom       = s.find_last_not_of(" \t");

   prenom = s.substr(posPremiereLettrePrenom,
                     posPremierBlancApresPrenom - posPremiereLettrePrenom);

   nom = s.substr(posPremiereLettreNom,
                  posDerniereLettreNom - posPremiereLettreNom + 1);

   acronyme = string("") // ou ""s mais pas ""
              + (char) toupper(prenom[0])
              + (char) toupper(nom[0])
              + (char) toupper(nom[nom.length() - 1]);

   cout << endl
        << "Votre prenom est   : " << prenom << endl
        << "Votre nom est      : " << nom << endl
        << "Votre acronyme est : " << acronyme << endl;

   return EXIT_SUCCESS;
}

// Entrez votre prenom et votre nom : Ludwig van Beethoven
//
// Votre prenom est   : Ludwig
// Votre nom est      : van Beethoven
// Votre acronyme est : LVN
//
// Entrez votre prenom et votre nom : Valery Giscard d'Estaing
//
// Votre prenom est   : Valery
// Votre nom est      : Giscard d'Estaing
// Votre acronyme est : VGG
//
```

## *Exercice 6.16 Saison correspondant à une date donnée*

Ecrire le plus proprement et judicieusement possible un programme C++ qui, après avoir demandé à l'utilisateur de saisir une date au format *jj.mm* (par ex 31.12), détermine / affiche la saison à laquelle correspond cette date.

### Exemple d'exécution

```
Entrez une date sous la forme jj.mm (par ex 31.12) : 21.3
Le 21.03 se situe au printemps.
```

### IMPORTANT

- On suppose la saisie utilisateur correcte
- On suppose les saisons définies comme suit :
  - 21.12 – 20.03 hiver
  - 21.03 – 20.06 printemps
  - 21.06 – 20.09 été
  - 21.09 – 20.12 automne

## Solution exercice 6.16

```
#include <cstdlib>
#include <iomanip>
#include <iostream>
#include <string>
using namespace std;

enum class Saison {HIVER, PRINTEMPS, ETE, AUTOMNE};

int main() {

   string str_jour, str_mois;
   int jour, mois;
   Saison saison;

   // Lire la date saisie par l'utilisateur
   cout << "Entrez une date sous la forme jj.mm (par ex 31.12) : ";
   getline(cin, str_jour, '.');
   getline(cin, str_mois);

   // Déterminer la saison correspondant à cette date
   jour = stoi(str_jour); // (*)
   mois = stoi(str_mois);

   saison = (Saison)((mois - 1) / 3);

   if (mois % 3 == 0 && jour >= 21) {
      if (saison == Saison::AUTOMNE) {
         saison = Saison::HIVER;
      } else {
         saison = (Saison)(static_cast<int>(saison) + 1);
      }
   }

   // Afficher le résultat
   cout << "Le " << setfill('0') << setw(2) << jour << "." << setw(2) << mois
        << " se situe ";
   switch (saison) {
      case Saison::HIVER:
         cout << "en hiver";
         break;
      case Saison::PRINTEMPS:
         cout << "au printemps";
         break;
      case Saison::ETE:
         cout << "en ete";
         break;
      case Saison::AUTOMNE:
         cout << "en automne";
         break;
   }
   cout << "." << endl;

   return EXIT_SUCCESS;
}
```

(*) … ou, si *stoi* n'est pas disponible

```
stringstream ss_jour(str_jour), ss_mois(str_mois);
ss_jour >> jour;
ss_mois >> mois;
```

## *Exercice 6.17 Analyse d'un nombre entier (3)*

Sans utiliser la librairie *cmath*, ni la fonction *to_string()* de la librairie *string*, ni le concept de boucle, écrire un programme C++ qui, après avoir demandé à l'utilisateur de saisir un nombre entier (de type *int*) ≥ 0, affiche à l'écran combien de chiffres contient ce nombre et ce que valent le premier chiffre et le dernier chiffre du nombre.

**NB** On supposera ici la saisie utilisateur correcte.

**IMPORTANT**
Le nombre saisi par l'utilisateur doit obligatoirement être stocké dans une variable de type *int*

### Exemple d'exécution

```
Entrez un nombre entier >= 0 : 123

Nombre saisi       : 123
Nombre de chiffres : 3
Premier chiffre    : 1
Dernier chiffre    : 3
```

## Solution exercice 6.17

```
#include <cstdlib>
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

int main() {

   int n;
   stringstream ss;
   string s;
   size_t nbChiffres;
   char premierChiffre,
        dernierChiffre;

   cout << "Entrez un nombre entier >= 0 : ";
   cin >> n;

   // Conversion de n en string
   ss << n;
   s = ss.str();

   nbChiffres = s.length(); // ou s.size()
   premierChiffre = s[0];
   dernierChiffre = s[nbChiffres - 1];

   cout << endl
        << "Nombre saisi       : " << n << endl
        << "Nombre de chiffres : " << nbChiffres << endl
        << "Premier chiffre    : " << premierChiffre << endl
        << "Dernier chiffre    : " << dernierChiffre << endl;

   return EXIT_SUCCESS;
}
```

Autre façon d'écrire les choses :

```
#include <cstdlib>
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

int main() {

   cout << "Entrez un nombre entier >= 0 : ";
   int n;
   cin >> n;

   // Conversion de n en string
   stringstream ss;
   ss << n;
   const string S = ss.str();

   const size_t NB_CHIFFRES = S.length(); // ou S.size()
   const char PREMIER_CHIFFRE = S[0],
              DERNIER_CHIFFRE = S[NB_CHIFFRES - 1];

   cout << endl
        << "Nombre saisi       : " << n << endl
        << "Nombre de chiffres : " << NB_CHIFFRES << endl
        << "Premier chiffre    : " << PREMIER_CHIFFRE << endl
        << "Dernier chiffre    : " << DERNIER_CHIFFRE << endl;

   return EXIT_SUCCESS;
}
```

## *Exercice 6.18 Conversions en base 8 et en base 16*

Ecrire un programme C++ qui, après avoir demandé à l'utilisateur de saisir un entier *n* (de type *int*) ≥ 0, affiche à l'écran (exactement comme dans l'exemple d'exécution ci-dessous) ce que vaut *n* en base 8 et en base 16.

**NB** On supposera ici la saisie utilisateur correcte.

**Indication** Consulter la documentation de la librairie *ios*

### Exemple d'exécution

```
Entrez un nombre entier >= 0 : 255

(255)10 = (377)8 = (FF)16
```

## *Solution exercice 6.18*

```
#include <cstdlib>
#include <iostream>
#include <sstream>
using namespace std;

int main() {

   int n;
   stringstream ss;

   cout << "Entrez un nombre entier >= 0 : ";
   cin >> n;

   // Affichage de n en base 8 et 16
   ss << oct << n;
   cout << "\n(" << n << ")10 = (" + ss.str() + ")8";
   ss.str("");
   ss << uppercase << hex << n;
   cout << " = (" + ss.str() + ")16\n";

   return EXIT_SUCCESS;
}
```

## *Exercice 6.19 Saisie contrôlée d'un entier entre 2 bornes (1)*

Ecrire un programme C++ qui :

1) demande à l'utilisateur de saisir un entier (de type *int*) entre deux bornes données (*min* et *max* également de type *int*)

2) affiche à l'écran la valeur saisie par l'utilisateur

**IMPORTANT**

- **La saisie utilisateur doit être entièrement contrôlée** (l'utilisateur doit être invité à refaire sa saisie aussi longtemps que cette dernière est incorrecte).
- Les bornes min et max sont à définir en tant que constantes dans le programme.
- Indications
    - A chaque flux est associé un "statut d'erreur" composé de 4 bits. Lorsque le flux est "sain", ces 4 bits sont tous à 0. Lorsque, par exemple, une tentative de lecture sur le flux *cin* échoue, l'un de ces 4 bits est mis à 1. Pour pouvoir continuer à utiliser le flux (pour refaire une nouvelle tentative de lecture), il s'agit au préalable de le remettre en état en remettant les 4 bits du statut d'erreur à 0. Ceci se réalise à l'aide de l'instruction : `cin.clear();`
    - Mais cela ne suffit pas. Il faut encore prendre soin de vider le buffer associé au flux afin de supprimer, notamment, le caractère responsable de l'erreur de lecture (qui, puisqu'il n'a pas pu être extrait du flux, s'y trouve toujours).
      Pour ce faire, on peut procéder de diverses manières.
      Il est demandé ici d'utiliser la méthode *ignore*.

**Exemple d'exécution**

```
Donnez un entier dans l'intervalle [1, 10] : a
Saisie incorrecte. Veuillez SVP recommencer.

Donnez un entier dans l'intervalle [1, 10] : 0
Saisie incorrecte. Veuillez SVP recommencer.

Donnez un entier dans l'intervalle [1, 10] : 11
Saisie incorrecte. Veuillez SVP recommencer.

Donnez un entier dans l'intervalle [1, 10] : 5
L'entier que vous avez saisi est 5.
```

## *Solution exercice 6.19*

```
#include <cstdlib>
#include <iostream>
#include <limits>
#include <string>
using namespace std;

int main() {

   const int MIN = 1,
             MAX = 10;

   // NB Si to_string() pas disponible, utiliser alors un stringstream
   const string MSG_INVITE = "Donnez un entier dans l'intervalle ["
                             + to_string(MIN) + ", " + to_string(MAX) + "] : ",
                MSG_ERREUR = "Saisie incorrecte. Veuillez SVP recommencer.";

   const streamsize MAX_STREAMSIZE = numeric_limits<streamsize>::max();

   int n;
   bool saisieOk;

   do {
      cout << MSG_INVITE;
      if (!(saisieOk = cin >> n && n >= MIN && n <= MAX)) {
         cin.clear(); // reset des bits d'erreur
         cout << MSG_ERREUR << endl << endl;
      }
      cin.ignore(MAX_STREAMSIZE, '\n'); // vider le buffer dans tous les cas !
      // char c;
      // while ( (c = (char)cin.get()) != '\n' );
   } while (!saisieOk);

   cout << "L'entier que vous avez saisi est " << n << "." << endl;

   return EXIT_SUCCESS;
}
```

## *Exercice 6.20 Saisie contrôlée d'un entier entre 2 bornes (2)*

Même énoncé que l'exercice 6.19, à la différence près que, ici, le programme est censé non plus lire un *int* mais un *unsigned int.*

## *Solution exercice 6.20*

```
#include <cstdlib>
#include <iostream>
#include <limits>
#include <string>
using namespace std;

int main() {

   const unsigned MIN = numeric_limits<unsigned>::lowest(),
                  MAX = numeric_limits<unsigned>::max();

   // NB Si to_string() pas disponible, utiliser alors un stringstream
   const string MSG_INVITE = "Donnez un entier dans l'intervalle ["
                             + to_string(MIN) + ", " + to_string(MAX) + "] : ",
                MSG_ERREUR = "Saisie incorrecte. Veuillez SVP recommencer.";

   const streamsize MAX_STREAMSIZE = numeric_limits<streamsize>::max();

   unsigned n;
   long long m;
   bool saisieOk;

   do {
      cout << MSG_INVITE;
      if (!(saisieOk = cin >> m && m >= MIN && m <= MAX)) {
         cin.clear(); // reset des bits d'erreur
         cout << MSG_ERREUR << endl << endl;
      }
      cin.ignore(MAX_STREAMSIZE, '\n'); // vider le buffer dans tous les cas !
   } while (!saisieOk);

   n = (unsigned) m;
   cout << "L'entier que vous avez saisi est " << n << "." << endl;

   return EXIT_SUCCESS;
}
```

## *Exercice 6.21 Saisie contrôlée d'un entier entre 2 bornes (3)*

Même énoncé que l'exercice 6.19, sauf qu'ici la saisie contrôlée est à implémenter en tant que fonction et que l'on souhaite mettre en œuvre les possibilités de la compilation séparée.

**NB** On supposera ici que la borne minimale est inférieure ou égale à la borne supérieure

## *Solution exercice 6.21*

```
#ifndef SAISIE_CONTROLEE
#define SAISIE_CONTROLEE

#include <string>

/**
 * @brief            Saisie contrôlée d'un entier dans l'intervalle [a, b]
 * @param a          borne minimale de l'intervalle de lecture
 * @param b          borne maximale de l'intervalle de lecture
 * @param msgInvite  message d'invite
 * @param msgErreur  message affiché en cas d'erreur de saisie
 * @return           l'entier saisi par l'utilisateur
 * @details          L'utilisateur est invité à refaire sa saisie aussi longtemps
 *                   qu'il n'entre pas une valeur correcte. <br>
 *                   On suppose ici a <= b.
 */
int lireEntier(int a, int b,
               const std::string& msgInvite = "",
               const std::string& msgErreur = "");

#endif
```

```
#include <iostream>
#include <limits>
#include "saisie_controlee.h"

using namespace std;

int lireEntier(int a, int b,
               const string& msgInvite,
               const string& msgErreur) {
   int n;
   bool saisieOk;
   do {
      cout << msgInvite;
      if (!(saisieOk = cin >> n && n >= a && n <= b)) {
         cin.clear(); // reset des bits d'erreur
         if (!msgErreur.empty()) {
            cout << msgErreur << endl << endl;
         }
      }
      cin.ignore(numeric_limits<streamsize>::max(), '\n'); // vider le buffer
   } while (!saisieOk);
   return n;
}
```

```
#include <cstdlib>
#include <iostream>
#include <string>
#include "saisie_controlee.h"

using namespace std;

int main() {

   const int A = 1,
             B = 10;

   // NB Si to_string() pas disponible, utiliser alors un stringstream
   const string MSG_INVITE = "Donnez un entier dans l'intervalle ["
                             + to_string(A) + ", " + to_string(B) + "] : ",
                MSG_ERREUR = "Saisie incorrecte. Veuillez SVP recommencer.";

   int n = lireEntier(A, B, MSG_INVITE, MSG_ERREUR);

   cout << "\nL'entier que vous avez saisi est " << n << "." << endl;

   return EXIT_SUCCESS;
}
```