Cours d'introduction à la programmation (en C++)
Types avancés I (en C++)
`string`

Jamila Sam
Jean-Cédric Chappelier
Vincent Lepetit

Faculté I&C

Le type `string`

Les chaînes de caractères C++ sont définies par le type `string`.

Pour utiliser des chaînes de caractères, il faut importer leur définition :

```
#include <string>
```

La déclaration d'une chaîne de caractères se fait alors avec :

```
string identificateur;
```

Exemple :

```
#include <string>
...
// déclaration (chaîne vide)
string un_nom;

// déclaration avec initialisation
string message("Bonjour à tous !");
...
```

Le type `char`

Les caractères (constituants d'une chaîne) peuvent aussi se représenter en tant que tels :
☞ le type `char`

Leur valeurs s'écrivent avec des guillemets simples : `'a'`

Exemple :

```
char c('x');
char u;
//...
u = 's';
```

Affectation de chaînes

Une variable de type `string` déclarée mais non initialisée est automatiquement initialisée à la chaîne vide (`""`).

Et comme pour n'importe quel autre type, une variable de type `string` (qui n'a pas été déclarée comme constante) peut être modifiée par une affectation.

Exemples :

```
string chaine ;           // -> chaine vaut ""
string chaine2("test") ;  // -> chaine2 vaut "test"
chaine = "test3" ;        // -> chaine vaut "test3"
chaine = chaine2 ;        // -> chaine vaut "test"
chaine = 'a' ;            // -> chaine vaut "a"
```

Remarque : dans le cas de l'affectation d'un caractère, la valeur affectée à la chaîne est la chaîne réduite au caractère affecté.

Concaténation

`chaine1 + chaine2` correspond à une **nouvelle chaîne** dont la valeur est la **concaténation** des valeurs de `chaine1` et de `chaine2`.

Exemple : constitution du nom complet à partir du nom de famille et du prénom :

```
string nom;
string prenom;
string famille;
...
nom = famille + " " + prenom;
```

Concaténation

Les combinaisons suivantes sont possibles pour la concaténation de deux chaînes :

`string + string`

`string + "..."` `"..." + string`

`string + char` `char + string`

où `string` correspond à une variable de type `string`, `"..."` correspond à une valeur littérale et `char` à une variable ou une valeur littérale de type `char`.

Exemple revisité (avec `char`) :

```
string nom;
string prenom;
string famille;
...
nom = famille + ' ' + prenom;
```

Concaténation

Remarque : les concaténations de la forme `string+char` constituent un moyen très pratique pour ajouter un caractère à la fin d'une chaîne.

Exemple : Ajout d'un 's' final au pluriel :

```
string reponse("solution");
...
if (n > 1) {
  reponse = reponse + 's';
}
...
```

La forme `char+string` permet d'ajouter un caractère au début.

Comparaison de chaînes

Comme pour les autres types, on utilise `==` pour tester l'égalité et `!=` pour la différence.

Exemples :

```
// il faut être sûr qu'ils ont compris :
do {
  reponse = poser_question();
} while (reponse != "oui");
```

Indexation

Si `chaine` est une `string`, alors `chaine[i]` est le `(i+1)`ème caractère de `chaine` (de type `char`).

Attention ! Comme pour les tableaux, les éléments d'une chaîne sont indicés de 0 à (*taille* − 1), où *taille* est le nombre de caractères de la chaîne.

Exemple 12 :

```
string demo("ABCD");
char premier;
char dernier;

premier = demo[0];
// premier reçoit 'A'

dernier = demo[3];
// dernier reçoit 'D'
```

```
string essai("essai");
string test;

for(int i(1); i <= 3; ++i) {
    test = test + essai[6-2*i];
    test = essai[i] + test;
}

cout << test << endl;
```

affiche : `assise`

Fonctions spécifiques aux chaînes

Certaines fonctions *propres aux* `string` sont définies.
Elle s'utilisent avec la syntaxe suivante :

nom_de_chaine.*nom_de_fonction*(arg1,arg2,...);

Les fonctions suivantes sont définies (où `chaine` est une variable de type `string`) :

`chaine.size()` : renvoie la taille (c'est-à-dire le nombre de caractères) de `chaine`.

`chaine.insert(position, chaine2)` : insère, à partir de la position (indice) `position` dans la chaîne `chaine`, la `string chaine2`

Exemple :

```
string exemple("abcd");  // exemple vaut "abcd"
exemple.insert(1, "xx"); // exemple vaut "axxbcd"
```

construit la chaîne `"axxbcd"`.

Fonctions spécifiques aux chaînes

`chaine.replace(position, n, chaine2)` : remplace les `n` caractères d'indice `position`, `position+1`, ..., `position+n-1` de `chaine` par la `string chaine2`.

Exemple :

```
string exemple("abcd");
exemple.replace(1, 2, "1234");
```

construit, dans `exemple`, la chaîne `"a1234d"`.

Remarque : la fonction `replace()` peut également servir à supprimer des caractères dans une chaîne.

Exemple :

```
string exemple("abcd");
exemple.replace(1, 2, "");
```

`exemple` vaut `"ad"`.

Fonctions spécifiques aux chaînes

`chaine.find(souschaine)` : renvoie l'indice dans `chaine` du 1er caractère de l'occurrence *la plus à gauche* de la `string souschaine`.

Exemple :
`string exemple("baabbaab");`
`exemple.find("ab")` renvoie 2.

`chaine.rfind(souschaine)` : renvoie l'indice dans `chaine` du 1er caractère de l'occurrence *la plus à droite* de la `string souschaine`.

Exemple :
`string exemple("baabbaab");`
`exemple.rfind("ab")` renvoie 6.

Fonctions spécifiques aux chaînes

Dans les cas où les fonctions `find()` et `rfind()` ne peuvent s'appliquer, elles renvoient la valeur prédéfinie `string::npos`

Exemple :

```
if (exemple.find("xy") != string::npos) ...
```

Fonctions spécifiques aux chaînes

`chaine.substr(depart, longueur)` : renvoie la sous-chaîne de `chaine`, de longueur `longueur` et commençant à la position `depart`.

Exemple :

```
string exemple("Salut à tous !");
```

`exemple.substr(8, 4)` renvoie la `string` `"tous"`.