Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

String - traitements (partie 2)

Concepts (extraits des sous-titres générés automatiquement) :

Chaîne de caractères essai. Fonctions propres. Chaîne de caractères. Nom de la fonction. Chaîne vide. Chaîne exemple. Chaîne test. Fonction size. Caractère d'indice. Type string. Fonction find. Variable exemple. Éventuels arguments. Fonction rfind. Nom de la variable de type.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en C++).)

vers la vidéo

EPFL

String : traitements
(Partie 2

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Indexation

EPFL

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 2 :

```
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;
```

Je commence par déclarer ici une variable qui s'appelle essai et qui contient la chaîne de caractères essai, et je déclare ici une autre chaîne de caractères qui est initialisée à la chaîne vide. Je continue avec cette boucle for où la variable 'i' va prendre les valeurs 1, 2, et 3, pour l'instant 'i' vaut 1. J'entre dans la boucle for : essai [6 - 2 * i] qu'est ce que ça vaut ? Alors ça sera un caractère déjà, comme 'i' vaut 1, l'indice vaut 4 et le caractère d'indice 4 de essai c'est 0, 1, 2, 3, 4 c'est le caractère 'i'. Tout ceci, est le caractère 'i' que je vais ajouter à ce que contient la chaîne test . La chaîne test est vide donc j'ai tout simplement à obtenir la chaîne 'i', que je vais mettre dans la variable test. Ensuite, ici dans cette affectation, essai [i] , comme 'i' vaut 1 et bien, c'est le caractère 's' ici. Test, pour l'instant, vaut la chaîne de caractères "i". Donc tout cet ensemble vaut la chaîne "si" que je vais affecter à test. 'i' vaut maintenant 2, je continue dans la boucle for. J'exécute cette affectation ici, cette expression vaut 2. Ceci est donc le caractère de la chaîne essai d'indice 2, qui est ce 's' ci. Ceci vaut 's', test vaut 'si'. Donc le résultat de cette affectation, c'est je vais mettre "sis" dans la variable test. Je continue ici, essai[i] c'est l'élément d'indice 2 de ma variable essai c'est à dire, ce 's' ci. test vaut "sis", et donc quand je fais cette affectation ici j'obtiens "ssis" dans la variable test. Je continue, cette fois-ci, 'i' vaut 3 dans cette affectation et dans cette expression qui vaut donc maintenant 0.

notes

résumé

0m 1s

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`.

Tout ceci vaut donc le caractère d'indice 0 de ma chaîne essai, c'est à dire, ce 'e' ci que je vais ajouter à test. J'obtiens donc, 6 pour l'instant, je passe à cette affectation. essai[3] c'est, 0, 1, 2, 3, ce caractère-ci. C'est à dire un 'a'. Je vais lui ajouter "ssise" et j'obtiens la chaîne "assise" que je vais mettre dans ma variable test. Et cet affichage me donne donc l'affichage 'd'assise'. Plusieurs fonctions propres au type string sont définis en c++ comme pour les vectors, elles s'utilisent avec la syntaxe suivante, tout d'abord le nom de la variable de type chaîne de caractères, suivi d'un point, suivi du nom de la fonction q'on veut utiliser, suivi des éventuels arguments dont a besoin la fonction pour pouvoir fonctionner. On dispose entre autres de la fonction size qui va s'appeler ainsi, si je suppose que chaine est une variable de type chaîne de caractères et cette fonction va me renvoyer la taille, c'est à dire, le nombre de caractères de ma variable chaine.

notes

résumé

2m 37s

Fonctions spécifiques aux chaînes

EPFL

`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 la chaîne `"a1234d"` (dans `exemple`).

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"`.

Je dispose également de la fonction insert qui a elle, deux arguments. Tout d'abord la position d'un caractère qui définit là où on va insérer une deuxième chaîne. Et qui est définit par cet argument ici. Donc l'appel à ces fonctions va insérer la chaine2 dans la variable chaine à partir de cette position-ci. Par exemple, je déclare ici une variable chaîne de caractères qui contient la chaîne: "abcd", et j'appelle ici la fonction insert avec la position 1, qui est celle-ci. Et la chaîne "xx". Et bien, cette fonction insert va insérer la chaîne "xx" entre le caractère d'indice 0 et le caractère d'indice 1, c'est à dire, ici. Et on va mettre le résultat dans la chaîne exemple C'est à dire que après l'appel à la fonction insert ma variable exemple contient la chaîne "axxbcd". On dispose également de la fonction replace, qui a elle besoin de 3 arguments. Tout d'abord, position, un peu comme la fonction insert que nous venons de voir. Un nombre de caractères, et le troisième argument est une chaîne de caractères. Cette fonction replace fonctionne un peu comme la fonction insert, sauf que, elle va remplacer 'n' caractères de la variable chaîne. Alors plus précisément, je vais prendre un exemple, après cette déclaration et initialisation la variable exemple contient la chaîne "abcd". J'appelle ma fonction replace avec pour position la valeur 1, qui correspond à ce 'b' ci. La fonction replace va remplacer deux caractères à partir de ce caractère 'b' C'est à dire, ce 'b' et ce 'c', et va les remplacer par cette chaîne-ci qui est "1234" et je vais donc obtenir dans ma variable exemple la chaîne "a1234d". Au passage je peux utiliser cette fonction replace, pour supprimer les caractères dans une chaîne. Pour cela il nous suffit d'utiliser la chaîne vide pour le troisième argument.

notes

résumé

4m 1s

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.

Dans ce nouvel exemple, ma variable exemple contient la chaîne "abcd". Je vais remplacer comme avant les caractères 'b' et 'c', mais je vais les remplacer avec la chaîne vide. Ça veut dire, qu'au final, je veux obtenir la chaîne 'ad' dans ma variable exemple. La fonction find attend un seul argument de type chaîne de caractères et renvoie l'indice dans chaine du premier caractère de l'occurrence la plus à gauche de la chaîne passé en argument. Alors qu'est ce que ça veut dire ? Je vais prendre un exemple, Où ma variable exemple est initialisée à cette chaîne de caractère-ci, et j'appelle la fonction find en lui passant "ab" en paramètre. Et bien "ab", en partant de la gauche apparaît pour la première fois dans exemple ici, l'indice du premier caractère c'est l'indice 2, puisqu'on part de 0. Et ma fonction find va donc me renvoyer la valeur 2. Il existe également la fonction inverse qui part de la droite,

notes

résumé

6m 49s

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) ...
```

baabbaab xy

`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"`.

c'est la fonction rfind. Et donc, quand j'appelle cette fonction rfind sur cette même chaîne exemple j'ai dit que je partais de la droite et la première fois que "ab" apparaît dans cette chaîne-ci c'est ici et l'indice de ce caractère-ci c'est '0, 1, 2, 3, 4, 5, 6', et donc ma fonction rfind dans ce cas ci, va me renvoyer la valeur 6. Notez qu'il n'y a pas de garantie que la chaîne passée en argument aux fonctions find et rfind se trouve dans la chaîne sur laquelle ces fonctions s'appliquent. Dans ce cas, les fonctions find et rfind vont envoyer une valeur prédéfinie qui se note string::npos pour signaler qu'elle n'ont pas trouvé la chaîne passée en argument. Par exemple si je fais cette instruction-ci avec exemple qui vaut comme dans les exemples précédents "baabbaab" et que j'essaye de chercher la chaîne "xy". Comme cette chaîne "xy" n'apparaît pas dans la chaîne exemple et bien, la fonction find va renvoyer la valeur prédéfinie string::npos .

notes

résumé

8m 13s

Enfin, une dernière fonction qui s'appel substr qui a deux paramètres : depart et longueur qui s'applique dans cet exemple sur la chaîne de caractères chaine, et bien, cet appel à la fonction substr renvoie la sous-chaîne de chaine qui a, comme longueur, cette longueur passée en argument et qui va commencer à la position depart. Par exemple, si je considère cette chaîne de caractères exemple, qui contient cette valeur l'appel à cette fonction exemple . substr (8, 4) va me renvoyer une sous-chaîne qui va commencer aux caractères d'indice 8. Alors, 0,1,2,3,4,5,6,7,8 c'est à dire sauter ici et va prendre 4 caractères à partir du 't' et donc l'appel à cette fonction va renvoyer la chaîne "tous". Voila, c'est tous ce que nous allons voir sur les fonctionnalités du type string. Sachez qu'il y en a d'autres mais qu'elle fonctionnent toutes sur le même principe. sur le même principe.

notes

résumé

9m 37s