

Travaux Dirigés Programmation C++ : Feuille 4

Informatique 2ème année.

—Julien Allali - allali@enseirb.fr —

Nous allons développer un modèle permettant d'implanter différents types de parcours sur une chaîne de caractères.

Repartir de la version de la classe `Chaine` disponible à l'adresse :

<http://www.enseirb.fr/~allali/Enseignement/cplusplus/chaine4.tar>

►Exercice 1. La Classe de Base :

La classe de base de nom `Curseur` va disposer des fonctions suivantes :

- `operator++` fait aller le curseur à la position suivante
- `operator--` fait aller le curseur à la position précédente
- `operator*` permet d'obtenir le caractère de la position courante
- `fini` renvoie vrai si la position actuelle du curseur est la fin de la chaîne de caractère.
- `debut` retour au début de parcours
- `fin` aller à la fin du parcours

À part pour `operator*`, les autres méthodes dépendent toutes du type du parcours et doivent donc être implantées dans la sous-classe.

En C++, lorsqu'une méthode n'a pas d'implémentation et doit être redéfinie dans une sous-classe on dit qu'elle est virtuelle pure :

```
class Mere{
public:
    virtual void m() const =0;
};
```

La méthode `m` n'est pas implantée dans `Mere` mais doit obligatoirement être redéfinie dans une sous classe pour que celle-ci soit instanciable.

Outre ces méthodes, la classe `Curseur` contiendra un attribut `_chaine` de type `Chaine`, copie de la `Chaine` sous-jacente, ainsi qu'un entier `_position` indiquant la position actuelle dans la `Chaine`. Vous écrirez un constructeur prenant en argument une instance de `Chaine` et la position de départ.

►Exercice 2. Parcours classique :

Écrire la classe `CurseurClassique` comme sous-classe de `Curseur`. Vous écrirez entre autre un constructeur prenant en argument une `Chaine` ainsi que l'ensemble des méthodes virtuelles pures.

Écrire une fonction `afficheCurseur` qui affiche un `Curseur` pris en argument par référence.

Tester bien votre implémentation, avec `valgind` entre autre.

►Exercice 3. Parcours inverse :

Écrire la classe `CurseurInverse` qui permet de parcourir la chaîne à l'envers. Tester avec la fonction `affiche`

►Exercice 4. Parcours avec pas :

Écrire la classe `CurseurPas` qui prend en argument la `Chaine` à parcourir, la position de début, la position de fin et le pas d'incrément. La valeur de fin pourra être négative, dans ce cas cela indiquera une position relativement à la fin. La valeur 0 pour fin indiquera la fin de la chaîne..

```
ParcoursPas p("un_texte_long",0,-2,10);
affiche(p);
// affiche:"u et o"

affiche(ParcoursPas("un_texte_long",0,0,3));
// affiche:"uttlg"
```

► **Exercice 5. Exceptions :**

Pour lever une exception en C++ on utilise le mot clé `throw` :

```
int main(){

    try {
        throw 1;
    }
    catch(int i){
        // gestion de l'exception de type int
        throw; // relance de l'exception à l'appelant.
    }
    catch(...){
        // attrape tout type d'exception
    }
}
```

Bien que l'on puisse lever une exception de tout type, il existe déjà des exceptions dans l'espace de nom `std`. La classe de base de ces exceptions est la classe `exception` déclarée dans `#include<exception>`. Deux classes dérivées servent de base à toutes les autres exceptions standards : `logic_error` et `runtime_error` (`#include<stdexcept>`).

Enfin, il existe deux sous-classes à `logic_error` : `out_of_range` et `invalid_argument`.

Ajouter une gestion des erreurs dans la classe `Chaine` là où vous le jugez nécessaire.

Vous testerez le fonctionnement des exceptions dans le programme principal en affichant des messages d'erreurs.