Travaux Dirigés Programmation C++ : Feuille 1

Informatique 2ème année.

—*Julien Allali* - `allali@enseirb.fr` —

Le but de ces 5 TD de C++ est de découvrir la programmation C++ en implémentant entre autre une classe pour la représentation des chaînes de caractères.

On utilisera la convention de nommage Java expection faite des attributs qui seront préfixés par `_`.

1 Classes.

▸**Exercice 1.** *Chaîne de caractère*

On se propose d'écrire une classe pour représenter des chaînes de caractères.

Dans un fichier `Chaine.hpp`*, déclarer une classe* `Chaine` *avec deux attributs privés :* `_taille` *de type* `unsigned int` *et* `_donnees` *de type* `char *` *ainsi qu'une méthode public* `taille` *fournissant la valeur de l'attribut* `_taille` *(l'implémentation de cette méthode se fera dans le fichier* `Chaine.cpp`*). On ne mettre pas de constructeur dans la classe pour l'instant (prochain exercice).*

Dans un fichier `Programme.cpp`*, faire une inclusion du fichier* `Chaine.hpp` *et écrire une fonction* `main` *qui :*

— *Déclare une variable de type* `Chaine`*.*

— *Déclare une pointeur sur un objet de type* `Chaine`*.*

— *Alloue dynamiquement une instance de* `Chaine` *dont on place l'adresse dans le pointeur.*

— *Appelle la méthode* `taille` *sur ces deux instances, la valeur renvoyée sera affichée avec* `printf`*.*

Pour tester votre programme, compiler à l'aide de la commande :

```
g++ -Wall Programme.cpp Chaine.cpp -o Programme
```

Identifier clairement par des commentaires les lignes où des instances sont créées ainsi que les allocations en précisant leur nature.

Est-ce que toutes les instances créées sont libérées à la fin du programme (vérifier cela avec `valgrind` *sur une machine sous linux,* `meunier, morbier, cantal, maroilles, cabecou` *par exemple) ? Justifier et corriger au besoin.*

▸**Exercice 2.** *Constructeur par défaut et destructeur*

Le constructeur en C++ n'a pas de type de retour et porte le même nom que la classe. Celui-ci est appelé après l'étape de réservation des ressources nécessaires à la création de l'instance.

Dans le fichier `Chaine.cpp` *ajouter un constructeur par défaut (sans argument) dans la classe* `Chaine`*. Ce constructeur se contentera d'afficher une message tel que :*

```
printf("%s(%d): %s\n",__FILE__,__LINE__,__func__);
```

Compiler :

```
CXXFLAGS="-Wall" make Chaine.o
CXXFLAGS="-Wall" make Programme.o
g++ -Wall Programme.o Chaine.o -o Programme
```

Vérifier que les instances sont bien créées.

En C++ le destructeur est une méthode de la classe appelée lors de la destruction de l'objet, avant la libération des ressources associées à l'instance. Le destructeur n'a pas de type de retour et a pour nom le nom de classe préfixé du caractère `~`*.*

De même que pour le constructeur par défaut, ajouter un destructeur qui trace les appels à celui-ci. Tester votre programme.

▸**Exercice 3.** *Allocation/Libération*
Écrire le constructeur qui prend en argument une chaîne de caractères (`char *`*) se terminant par un* `'\0'` *et qui recopie cette chaîne dans* `_donnees` *avec* `strcpy`*.*

Modifier le programme pour tester ce nouveau constructeur. Compiler et executer avec `valgrind`*. Corriger si nécessaire.*

▸**Exercice 4.** *état par défaut*
Effectuer l'initialisation de l'instance dans le constructeur par défaut et vérifier que tout ce passe bien (compilation, exécution, valgrind).

▸**Exercice 5.** *Par copie*
Ajouter dans le programme principal la fonction suivante :

```
void annexe(Chaine s){}
```

Appeler cette fonction dans le `main`*, compiler, tester avec* `valgrind`*. Analyser l'exécution de votre programme à l'aide de* `gdb`*.*

Expliquer le comportement observé, pensez à indiquer clairement la succession d'allocations et d'instanciations effectuées.

▸**Exercice 6.** *getter :*
Écrire une méthode `char get(unsigned int i)` *permettant d'obtenir le* $i^{\text{ème}}$ *caractère de la chaîne.*

2 Travail personnel : les méthodes virtuelles

Voici un petit exercice vous permettant d'expérimenter par vous même les mécanismes du polymorphisme en C++.

▸**Exercice 7.** *Polymorphisme choisi ?*
Récupérer le fichier `Exemple_td1.cpp`

Etudier le code, écrire **sur une feuille** *le diagramme de classe, compiler et executer.*

Pour chacune des modifications suivantes, recompiler et executer le programme :

— *ajouter le mot clé* `virtual` *uniquement devant* `Mere::message`
— *ajouter le mot clé* `virtual` *uniquement devant* `Fille::message`
— *ajouter le mot clé* `virtual` *devant* `Mere::message` *et* `Fille::message`*.*

Trouver le sens de ce mot clé (cela a un rapport avec le titre de l'exercice).

*Essayer de reproduire cette exemple en utilisant l'***allocation automatique***. Pouvez-vous en tirer une conclusion sur l'utilisation du polymorphisme en C++ ?*

Utiliser la fonction `main` *commentée en bas du fichier et re-tester avec ou sans* `virtual` *sur* `Mere::message`*. Expliquer le résultat observé.*

3 Bibliographie :

— Le Language C++, Bjarne Stroustrup, CompusPress.
— http ://www.cplusplus.com/
— http ://www.developpez.com
— Effective C++, Scott Meyers, Addison Wesley.
— More Effective C++, Scott Meyers, Addison Wesley.