Programmation Orienté Objet

2022-2023
Travaux Pratiques 5

Site du cours : `https://defelice.up8.site/poo.html`
Exemple : `g++ mon_fichier.cpp -std=c++17 -Wall -Wextra -g -fsanitize=address -o mon_programme`
Les exercices marqués d'un @ sont à faire dans un second temps.

Dans tous les exercices de ce TP (et des tp suivants) il faut, si possible, penser à déclarer les méthodes constantes. Pensez également à éviter toute erreur de mémoire (fuite de mémoire, dépassement d'indice,...).

Dans ce tp : il n'est pas permis de s'aider des outils de bibliotèques (comme `std::string`) sauf pour afficher ou lire dans des fichiers ou les flux standards (comme `iostream`) ou pour la construction (en `std::string` par exemple).

Exercice 1. *vecteur*

Construire la classe `vector_t`. Exemple d'utilisation de cette classe :

```
#include<cassert>
vector_t v(5); // tableau de 5 cases allouée (par exemple toute à 0)
vector_t r(500000); // tableau de 5000000 cases allouées
v[3]=2; // met la valeur 2 dans la case d'indice 3
int a=v[3]; // initialise a avec le contenu de la 4 ième case
v[6]; // doit produire une erreur  (exemple throw 1; ou exit(1))
v.max() // renvoie l'entier maximum
v.min() // renvoie l'entier minimum
vector_t b=v*3; // multiplie par 3 chaque élément, puis initialise b
assert(b[3]!=v[3]); // b et v sont différent
r=4*v; // multiplie par 3 chaque élément, produit un nouveau vecteur
// pas de fuite de mémoire ni de double libération
```

Comment construire (dans `main`) un objet constant `c` de la classe `vector_t` de longueur `6` cases qui contient `2,3,5,7,11,13` Sachant qu'on ne pourra pas utiliser une instruction comme `c[5]=3` ?

Exercice 2. *Chaine*

On veut implanter une classe `chaine_t` qui se comporte un peu comme la classe `std::string`.

Exemple d'utilisation :

```
chaine_t a;
chaine_t b{"Bonjour"};
chaine_t const c="Salut";
chaine_t m("Ave");
chaine_t n(45); // n contiendra l'écriture de 45 en base dix, "45" donc
chaine_t* d=new chaine_t{"Au revoir"};
a.append(b); // ajoute le contenu de b à la fin de a (sans modifier b)
a=b+c; // a contient la concatenation de b et c
b[4]='u'; // on modifie le 4 ime  octets
//   c[3]='o'; erreur à la compilation
char u=c[3]; ok  (penser le qualificatif method(...) const {...}  fait parti de la signature)
// c.append(a) ; erreur c est const
// b[66]='u'; // erreur execution indice trop grand

c*3 // renvoie une chaine qui est a dupliquée 3 fois : "SalutSalutSalut"
```

```
3*a // pareil
std::cout << c.toString() << std::endl; // la fonction membre toString() renvoie un std::string
c==a; // egale vraie ou faux
c!=a; // egale le contraire
delete d;
void mfonction(chaine_t a){a.append("coucou");}
mfonction(a);
// ici a est inchangé car mfonction modifie une variable  copiée
```

On ne s'inquètera pas des performances seulement du bon fonctionnement de la classe.
Penser :

— à bien appliquer la règle des trois.
— à bien implanter les méthodes constantes et non constantes.