Programmation orientée objet

LIV 2024-2025

Travaux Pratiques 10 - interface, classe abstraite

Site du cours : `https://defelice.up8.site/poo.html`
Les exercices marqués de (@) sont optionnels.
Un fichier écrit en langage C++ se termine conventionnellement par .cpp.
Une commande de compilation est `g++ fichier_source1.cpp fichier_source2.cpp fichier_source3.cpp`.
Voici des options de cette commande.

- `-o nom_sortie` pour donner un nom au fichier de sortie (par défaut `a.out`).
- `-Wall` pour demander au compilateur d'afficher plus d'avertissements (warnings)
- `-Wextra` pour demander au compilateur d'afficher plus d'avertissements (warnings)
- `-std=c++20` pour compiler selon la norme c++20
- `-g -fsanitize=address` pour que l'exécutable puisse detecter certaines erreurs de mémoire (adresse inacessible, fuite mémoire) (l'inonveniant et que l'executable est plus lent, -g pour avoir plus de précision).
- `-g` pour ajouter des informations de débugage, utile avec debug (executable plus lent).

Exemple : `g++ -Wall -Wextra fichier1.cpp -o monprogramme`

Exercice 1. *Test d'efficacité*

Le fichier suivant : `annexe04.cpp` contient une fonction `testEnsemble` écrite pour tester le fonctionnement d'une structure de donnée qui implante le comportement d'un ensemble d'entier : `InterfaceEnsemble`. (en l'occurence on ne souhaite que les trois comportements suivants : vider l'ensemble, ajouter un élément, tester si un élément est présent)

Le but de l'exercice est de construire trois classes qui implantent l'interface d'un ensemble (donc qui hérite de `InterfaceEnsemble` et ensuite d'expérimenter la fonction `InterfaceEnsemble` sur ces trois classes. Les noms des trois méthodes de `InterfaceEnsemble` sont à déduire de la fonction `testEnsemble`.

Les trois classes héritant de `InterfaceEnsemble` doivent être :

- Une classe `VectorSTL` (à hériter de la classe `vector` de la STL) (chaque valeur est placée à la fin).
- Une classe `SetSTL` (à hériter de `set` de la STL).
- Une liste chainée `ListeChainee`. A vous de construire la classe (chaque valeur est placée en tête de liste).

annexe04.cpp

```
1  #include<cstdlib>
2  #include<ctime>
3  #include"interfaceEnsemble.hpp" // à écrire
4  double testEnsemble(InterfaceEnsemble& ensemble,double& mesure){
5  /*
6     Vérifie quelques aspects du bon fonctionnement de
7     "ensemble" et effectue des mesures d'efficacité.
8     Renvoie 1 si aucun disfonctionnement
9     n'a été detecté 0 sinon. Affecte à "mesure"
10    le temps d'accès moyen d'accès à un élément,
11    mesuré en secondes.
12 */
13    const int maxI=10000000;
14    clock_t chrono; // chronomètre
15    unsigned long temps_passe;
16    ensemble.vider(); // vide tout
17    for(int i=0;i<maxI;i++)
18       ensemble.ajouter(i);
19    chrono=clock();
20    for(int i=0;i<maxI;i++)
21    {
22       if(not ensemble.estDans(i))
```

```
23                return 0; // cas d'erreur
24             if( ensemble.estDans(i+maxI))
25                return 0; //cas d'erreur
26          }
27          chrono=clock()-chrono;
28          temps_passe=chrono;
29
30          ensemble.vider();
31          srand(271828);
32          for(int i=0;i<maxI;i++)
33             ensemble.ajouter(rand());
34          srand(271828);
35          chrono=clock();
36          for(int i=0;i<maxI;i++)
37             if(not ensemble.estDans(rand()))
38                return 0;
39          chrono=clock()-chrono;
40          temps_passe+=chrono;
41          mesure=(double)temps_passe/maxI/3/CLOCKS_PER_SEC;
42          return 1;
43       }
44
```