

TP n°1.2

Introduction à C++

Note : ces exercices necessitent la lecture des supports des deux premiers cours que l'on peut retrouver a l'adresse <http://www.liafa.univ-paris-diderot.fr/~yunes/cours/cpp/>

Exercice 1 (Surcharge)

Ecrivez un nouveau module de fonctions toutes appelees `plus` permettant de calculer : pour l'une la somme de deux ints passes en parametres et renvoyant un int, pour l'autre la somme de deux doubles et renvoyant un double. Appelez la fonction plus en passant : deux ints, un int et un short, deux floats, deux doubles, un int et un double. Que dit le compilateur ? Pourquoi ?

Exercice 2 (Passage d'un tableau)

Ecrivez un nouveau module de fonctions toutes appelees `somme` et qui permettent de calculer la somme des elements d'un tableau d'ints pour l'une et de double pour l'autre. Appelez la fonction `somme` en passant : un tableau d'ints, un tableau de shorts, un tableau de doubles. Que dit le compilateur ? Pourquoi ?

Exercice 3 (Arguments constants)

Ajoutez `const` aux arguments des fonctions de l'exercice 2 apres les avoir recopiees. Essayez dans le corps des fonctions de modifier les arguments. Que dit le compilateur ? Pourquoi ?

Exercice 4 (Encore des constantes)

Definissez les fonctions `f(char)`, `g(char &)`, `h(const char &)` et appelez les avec les arguments `'a'`, `49`, `3300`, `c`, `uc` ou `c` est un `char` et `uc` un `unsigned char`. Qu'est-ce qui se passe ? Dans quels cas le compilateur doit-il creer une variable temporaire ?

Exercice 5 Testez le programme suivant :

```
int main()
{
    int a=1, b=2, c=0;
    ajouter(a,b,c);
    cout <<a<<' '<<b<<' '<<c<<endl;
}
```

Avec ces 3 fonctions que vous definirez successivement :

```
void ajouter(int a, int b, int c){c=a+b;}
void ajouter(int a, int b, int *c){*c=a+b;}
void ajouter(int a, int b, int &c){c=a+b;}
```

Exercice 6 (Tableau, Namespace)

Dans un module approprie, ecrivez un ensemble de fonctions permettant d'accéder de facon securisee a un tableau (creez une structure tableau avec un champ longueur, etc) : ces fonctions verifieront que l'on utilise le tableau correctement (debordements). Indication : on veut pouvoir (entre autres) ecrire `elementAt(tableau,i) = 12`.

Exercice 7 (tri) L'algorithme trifusion fonctionne comme suit :

1. On coupe en deux parties a peu pres egales les donnees a trier
2. On trie recursivement les donnees de chaque partie
3. On fusionne les deux parties

Creez une fonction C++ qui trie un tableau en utilisant cet algorithme.

Exercice 8 (Une premiere classe)

Ecrivez une classe `Compte` qui represente un compte de banque. Le constructeur prend le montant initial du compte. Ajoutez des methodes pour retirer de l'argent, pour deposer de l'argent et pour afficher le montant actuel. Dans la fonction `main` creez un objet `c1` de type `Compte` de facon statique et un objet `c2` de type `Compte` de facon dynamique. Appelez les methodes des objets en utilisant `.` et `->`.