

Tp 2 C++. Master 1 Physique

1 Compilation séparée

Lorsque un projet dépasse une certaine taille on préfère fragmenter le code source en plusieurs modules (fichiers `.cpp`). Avec cette approche :

- on gagne en lisibilité,
- plusieurs personnes peuvent travailler sur le même projet, chacun travaille sur un module sans interférer avec ses collègues,
- après une modification, on n'a pas besoin de recompiler le programme entier, il suffit de recompiler le module modifié,
- on peut utiliser le même module dans deux projets indépendants.

On veut appliquer cette approche à

```
#include <iostream>
using namespace std;

int main()
{
    int func ();
    func();
    return 0;
}

int func ()
{
    cout<< "compile correctement" <<endl;
    return 0;
}
```

Exercice 1

1. Séparez le programme en deux fichiers de sorte que l'un contienne la définition de la fonction `main()` et l'autre celle de `func()`. Recompilez tous les modules avec les options de compilation qui signalent les prototypes (et autres déclarations) manquants. Faites de sorte qu'il y ait toutes les déclarations nécessaires, pour que le compilateur n'émette pas d'erreurs ni d'avertissements (*warnings*). Si tout va bien, vous avez deux fois la même déclaration (une fois dans chaque fichier `.cpp`).

2. Dans un grand projet il peut y avoir des dizaines de milliers de déclarations, il n'est pas question d'en mettre une copie dans chaque fichier source `.cpp`. On met toutes les déclarations dans des fichiers en-tête (*header*) `.hpp` et on utilise la directive `#include` pour les inclure dans les sources `.cpp`.
Mettez les déclarations de votre programme dans un fichier `decl.hpp`, et utilisez `#include` dans vos fichiers sources `.cpp` (un pour la fonction et l'autre pour le `main`). Assurez vous que tout marche bien.

2 Transmission des arguments

2.1 Arguments simples

Exercice 2

Les trois fonctions incrémenteront la valeur de la donnée passée en argument.

1. **Transmission par valeur.**

Faites un programme contenant une fonction qui prend un `double` comme argument : `void test_par_valeur(double);`

- a) Vérifiez que les arguments reçus par la fonction sont des copies des variables passées en argument.
- b) Est-ce que l'adresse de l'argument reçu peut être la même que celle de la variable passée en argument ?

2. **Transmission par pointeur.**

Complétez le programme avec une fonction qui prend en argument un pointeur vers un `double` : `void test_par_adresse(double*);`

- a) Vérifiez que lors d'un appel, le pointeur reçu par la fonction contient bien l'adresse de la variable passée en argument.

3. **Transmission par référence.**

Complétez à nouveau le programme avec une fonction qui cette fois-ci utilise le passage par référence en argument, à la façon C++ : `void test_par_reference(double &);`

4. Comparer les trois façons de transmettre un `double` à une fonction.
5. Dans le programme principal suivant, (voir fichier `tp2.exo4.cpp`) quelles sont les lignes de code erronées ?

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    double z=0;
    test_par_valeur(z);
    cout << "apres test_par_valeur " << z << "\n";
```

```

cout << "adresse de z " << &z << "\n";
test_par_adresse(&z);
cout << "apres test_par_adresse " << z << "\n";
test_par_reference(z);
cout << "apres test_par_reference " << z << "\n";
test_par_valeur(3*sqrt(z));
test_par_adresse(&3*sqrt(z));
test_par_reference(3*sqrt(z));
return 0;
}

```

6. Parmi les trois façons de transmettre un double à la fonction, laquelle (lesquelles) choisirez-vous pour :
 - incrémenter l'argument
 - afficher l'argument
 Faites les programmes correspondants.

2.2 Transmission de tableaux

Exercice 3

Quand on passe un tableau en argument à une fonction, c'est uniquement l'adresse du premier élément du tableau qui est transmise. En effet, en langage C++ les tableaux et les pointeurs ont un rôle très similaire, on peut presque toujours utiliser un pointeur à la place d'un tableau. Autrement dit le tableau entier n'est pas copié dans la pile, c'est uniquement son adresse qui y est copiée.

1. Il y a-t-il une différence entre les trois fonctions suivantes :
 - `void func1(unsigned *tab);`
 - `void func2(unsigned tab[]);`
 - `void func3(unsigned tab[10]);`
2. Faites une fonction qui prend en argument un tableau d'entiers et qui le remplit avec les nombres $0, 1, 2, \dots, N$, la valeur de N est à spécifier par l'utilisateur. Comment transmet-on la dimension du tableau à la fonction ?

Exercice 4

1. Écrivez un programme dans lequel on alloue dynamiquement un tableau de type `int` à une dimension. N'oubliez pas de libérer correctement la mémoire à la fin du programme. Pour ces allocations dynamiques, on utilisera les opérateurs `new` et `delete`.
2. Testez votre programme dans le cas d'un tableau dynamique de 10 entiers dont les valeurs seront de 0 à 9.