Université Paris-XII — Faculté de Sciences économiques et de Gestion
Master Professionnel *Méthodes Appliquées de la Statistique et de l'Économétrie pour la Recherche, l'Analyse et le Traitement de l'Information*
Cours « Programmation en langage C++ » — Année universitaire 2008-2009
François Legendre — Francois.Legendre@cee.enpc.fr
http://fj.legendre.free.fr

Mes premiers programmes en C++

Listing 1 – table_francs_euros.cpp

```
1  // table_francs_euros.cpp -- coding: utf-8
2  #include <iostream>   // Utilisation des flux cin, cout et cerr.
3  #include <iomanip>    // Pour setw(...) dans les opérations de sorties.
4
5
6  using namespace std ; // Importer l'espace des noms de la bibliothèque standard.
7
8  // Définition d'une variable constante de type double.
9  const double VALEUR_EURO_EN_FRANCS = 6.55957 ;
10
11 // Déclaration des fonctions éventuellement utilisées par la suite.
12 double franc2euro(const double & somme_en_francs) ;
13 double euro2franc(const double & somme_en_euros) ;
14 void tirets(size_t nombre) ;
15
16 int main() { // Point d'entrée du programme.
17
18   const double tableau[] = { .1, .5, 1, 5, 10, 50, 100, } ;
19   const size_t nombre = sizeof(tableau) / sizeof(tableau[0]) ;
20
21   const size_t nombre_tirets = 2+6+3+10+2 ;
22   tirets(nombre_tirets) ;
23   cout << "|␣" << setw(6) << "Francs" << "␣|␣" << setw(10) <<
24      "Euros" << "␣|\n" ;
25   tirets(nombre_tirets) ;
26   for ( size_t i = 0 ; i < nombre ; ++ i ) {
27     cout << "|␣" << setw(6) << tableau[i] << "␣|␣" << setw(10) <<
28        franc2euro(tableau[i]) << "␣|\n" ;
29   }
30   tirets(nombre_tirets) ;
31   return 0 ; // Le code de retour du programme est égal à 0.
32 }
33 // Cette fonction calcule la contrevaleur en euros d'une somme exprimée en francs.
34 double franc2euro(const double & arg) {
35   return arg / VALEUR_EURO_EN_FRANCS ;
36 }
37 // Cette fonction insère une ligne de n caractères '-' sur le flux de sortie cout.
38 void tirets(size_t n) {
39   while ( n -- > 0 )
40     cout << '-' ;
41   cout << endl ;
42 }
```

Compilation du programme : ***g++ -Wall table_francs_euros.cpp***
Exécution du programme : ***a.exe***

g++ est le nom du programme exécutable du compilateur C++ *open source* GNU développé initialement par la *Free Software Foundation*. ***-Wall*** (pour *Warning all*) est l'option de compilation qui permet d'obtenir le plus grand nombre de messages d'avertissement : je vous conseille de toujours utiliser cette option. Par défaut, le nom de programme exécutable engendré par le compilateur est ***a.exe*** (pour le système d'exploitation Window de Microsoft).

Pour donner un autre nom à ce programme exécutable, il faudrait utiliser par exemple la commande suivante :

g++ -Wall table_francs_euros.cpp --output table

Listing 2 – adresse_variables.cpp

```
1  // adresse_variables.cpp -- coding: utf-8
2  #include <iostream>   // Utilisation des flux cin, cout et cerr.
3  #include <iomanip>    // Pour setw(...) dans les opérations de sorties.
4
5  using namespace std ; // Importer l'espace des noms de la bibliothèque standard.
6
7  double var_globale_1 = 1 ; // Une première variable statique et globale.
8  double var_globale_2 = 2 ; // Une seconde variable statique et globale.
9
10 // Déclaration des fonctions utilisées par la suite.
11 void afficher_adresse_de(const string & nom, const double & variable) ;
12 void une_fonction() ;
13
14 int main() {
15
16   double var_locale_1 = 1 ;            // Une variable automatique et locale.
17   const double const_var_locale = 2 ;  // Une variable automatique et locale.
18   static double static_var = 3 ;       // Une variable statique et locale.
19
20   cout << "Adresse des variables\n" ;
21   afficher_adresse_de("var_globale_1", var_globale_1) ;
22   afficher_adresse_de("var_globale_2", var_globale_2) ;
23   afficher_adresse_de("var_locale_1", var_locale_1) ;
24   afficher_adresse_de("const_var_locale", const_var_locale) ;
25   afficher_adresse_de("static_var", static_var) ;
26
27   if ( true ) {
28     double var_bloc_1 = 1 ;            // Une première variable automatique.
29     afficher_adresse_de("var_bloc_1", var_bloc_1) ;
30     double var_bloc_2 = 2 ;            // Une seconde variable automatique.
31     afficher_adresse_de("var_bloc_2", var_bloc_2) ;
32   }
33   if ( true ) {
34     double var_autre_bloc = 1 ;        // Itou, dans un autre bloc.
35     afficher_adresse_de("var_autre_bloc", var_autre_bloc) ;
36   }
37   une_fonction() ;
38 }
39 void afficher_adresse_de(const string & nom, const double & variable) {
40   cout << setw(20) << nom << ' ' << size_t(&variable) << endl ;
41 }
42 void une_fonction() {
43   double var_fonction = 1 ;            // Une variable automatique et locale.
44   afficher_adresse_de("var_fonction", var_fonction) ;
45 }
```