



Département d'Informatique

Info 2 PC

Licence S2 - SI

2012-2013

TP1

Enregistrements – Programmation modulaire – Compilation séparée

1 – Compilation d'un fichier

Recopiez le programme C++ ci-dessous, déjà étudié en cours, dans un fichier nommé *gestion_records.cpp*.

Après la compilation et l'édition des liens (`g++ gestion_records.cpp -o gestion`), exécutez le programme (`./gestion`).

2 - Compilation séparée

Séparez *gestion_records.cpp* en différents fichiers (voir ci-dessous) correspondants aux actions de saisie, de tri et d'affichage. Ces fichiers seront compilés séparément avant l'édition des liens.

Chaque action sera implémentée par une fonction. Leurs profils (entêtes) sont :

```
void saisie( athlete part[], int nbp );  
void tri( athlete part[], int nbp );  
void affichage( athlete part[], int nbp );
```

Pour une **compilation globale** avec plusieurs fichiers, on utilisera la commande :

```
g++ Source1.cpp Source2.cpp Source3.cpp Source4.cpp -o executable
```

Pour une **compilation séparée**, on utilisera la commande :

```
g++ -c Source1.cpp
```

Cela génère un fichier binaire `Source1.o`

Une fois tous les fichiers compilés, on fera l'édition de liens avec

```
g++ Source1.o Source2.o Source3.o Source4.o -o executable
```

Attention : Que vous utilisiez la compilation globale ou la compilation séparée, seul un (et un seul) de vos fichiers sources doit contenir une fonction nommée `main`. Il est conseillé de mettre cette fonction seule dans un fichier qu'on nommera `main.cpp`.

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

```
// Données caractéristiques d'un athlète
// -----
struct athlete
{
    int    numero;
    float  record;
};
```

```
// Nombre de participants
// -----
const int nbParticipants = 10;

// Tableau d'athlètes
// -----
athlete participants[ nbParticipants ];
```

gestion_types.h

```
int main( void )
```

```
{
    athlete tmp;
    int    i, j, indiceMin, rang;
    float  min;
```

gestion_saisie.cpp

```
// Saisie du tableau
// -----
for( i = 0; i < nbParticipants; i++ )
{
    cout << "N° et record ?" << endl;
    cin >> participants[i].numero >> participants[i].record;
}
```

```
// Tri du tableau
// -----
for( j = 0; j < nbParticipants - 1; j++ )
{
    min = participants[ j ].record;
    indiceMin = j;

    for( i = j+1; i < nbParticipants; i++ )
        if( participants[ i ].record < min )
        {
            min = participants[ i ].record;
            indiceMin = i;
        }

    if( indiceMin != j )
    {
        tmp = participants[ j ];
        participants[ j ] = participants[ indiceMin ];
        participants[ indiceMin ] = tmp;
    }
}
```

gestion_tri.cpp

gestion_affichage.cpp

```
// Affichage des résultats
// -----
cout << "Rang Participant Record" << endl;
cout << "1 " << participants[ 0 ].numero << " " << participants[ 0 ].record << endl;

rang = 1;
for( i = 1; i < nbParticipants; i++ )
{
    if( participants[ i ].record > participants[ i - 1 ].record )
        rang++; // On n'incrmente le rang que dans le cas ou il n'y a pas égalité

    cout << rang << " " << participants[ i ].numero << " " << participants[ i ].record << endl;
}
```

```
return 0;
```

```
}
```

Les avantages de ce découpage(une fonction par fichier C++) :

- cela permet de se concentrer sur le développement de cette fonction
- il est plus simple de trouver les erreurs à la compilation
- si un programme externe souhaite utiliser une fonction, c'est plus simple.
- si a posteriori on trouve une solution pour optimiser une fonction, il suffit de la réécrire une fois et tous les programmes qui l'utilise pourront en bénéficier (pour peu que l'on recompile)
- cela offre plus de modularité et il serait même envisageable d'avoir deux fichiers qui proposent la même fonction. Il suffirait au niveau de l'étape d'édition de liens de dire lequel des deux fichiers on veut utiliser.
- nous avons vu les prémisses pour constituer une bibliothèque. Il suffit d'inclure les définitions (le .h) et de faire le lien avec le code lors de la compilation. Avec Arduino, on peut utiliser de nombreuses bibliothèques plutôt que de tout redévelopper.