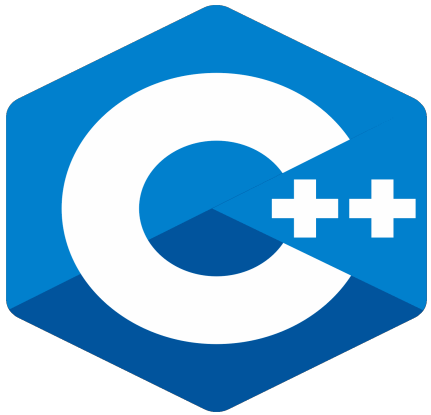
C++ : fichiers

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

C++

Les fichiers ?

- Un des premiers supports de stockage
- Dans ce cours, on va détailler
  - les fichiers de code source
  - les fichiers de données

C++

Les fichiers de code, pourquoi ?

- Permettant de mieux structurer le code (les fonctions)
- Deux types de fichier avec C++
  - `.h` : pour déclarer les prototypes de nos fonctions
  - `.cpp` : pour implémenter les prototypes déclarés dans le fichier `.h`

C++

Les fichiers, comment (avec CodeBlocks) ?

- Aller dans `File > New > File`
- Choisir `C/C++ header` ensuite cliquer sur `Go` puis sur `Next >`
- Saisir `calcul` dans `Filename with full path` ensuite cocher les deux cases `Debug` et `Release`
- Valider en cliquant sur `Finish`
- Refaire la même chose en choisissant cette fois `C/C++ source`

C++

Commençons par inclure `calcul.h` dans `calcul.cpp`

```
#include "calcul.h"
```

C++

Commençons par inclure `calcul.h` dans `calcul.cpp`

```
#include "calcul.h"
```

Pour les fichiers d'entête définis par le développeur, on utilise les " " à la place de < >

C++

Code généré dans `calcul.h`

```
#ifndef CALCUL_H_INCLUDED
#define CALCUL_H_INCLUDED




#endif // CALCUL_H_INCLUDED
```

C++

Code généré dans `calcul.h`

```
#ifndef CALCUL_H_INCLUDED
#define CALCUL_H_INCLUDED




#endif // CALCUL_H_INCLUDED
```


Remarques

- Le code généré dans `calcul.h` permet d'éviter les inclusions multiples de ce fichier.
- Les prototypes doivent être situés entre avant la dernière ligne `#endif // CALCUL_H_INCLUDED`

C++

Déplaçons maintenant la fonction somme dans `calcul.cpp`

```
#include "calcul.h"

int somme (int a, int b)
{
  return a + b;
}
```

C++

Déplaçons maintenant la fonction somme dans `calcul.cpp`

```
#include "calcul.h"

int somme (int a, int b)
{
  return a + b;
}
```

Déclarons son prototype dans `calcul.h`

```
#ifndef CALCUL_H_INCLUDED
#define CALCUL_H_INCLUDED

int somme (int a, int b);

#endif // CALCUL_H_INCLUDED
```

C++

Nouveau contenu de la fonction principale

```
#include <iostream>
#include "calcul.h"

using namespace std;

int main()
{
    int x = 2, y = 3;
    int resultat = somme (x, y);
    cout << "Le resultat est " << resultat << endl;
    return 0;
}
```

C++

Exemple avec un type avancé (string, vector...) : code de `calcul.cpp`

```
#include "calcul.h"

int somme (int a, int b)
{
  return a + b;
}

int nbrVoyelle(std::string str)
{
  int nbr = 0;
  for(char c: str){
    if (c == 'a' || c == 'o' || c == 'e' || c == 'u' || c == 'i
       ' || c == 'y')
         nbr++;
  }
  return nbr;
}
```

C++

Contenu de `calcul.h`

```
#ifndef CALCUL_H_INCLUDED
#define CALCUL_H_INCLUDED

#include <string>

int somme (int a, int b);
int nbrVoyelle(std::string str);

#endif // CALCUL_H_INCLUDED
```

C++

Contenu de la fonction principale

```
#include <iostream>
#include "calcul.h"

using namespace std;

int main()
{
    int x = 2, y = 3;
    int resultat = somme (x, y);
    cout << "Le resultat est " << resultat << endl;
    cout << "Le nombre de voyelle dans bonjour est "
        << nbrVoyelle("bonjour") << endl;
    return 0;
}
```

C++

Les fichiers de données, pourquoi ?

- Permettant de lire et écrire des données utilisées dans un programme
- Plusieurs étapes pour l'utilisation d'un fichier
  - inclusion de la librairie `fstream` (pour file stream)
  - déclaration d'un fichier en lecture ou en écriture
  - Ouverture
  - utilisation
  - fermeture

C++

Pour inclure la librairie `fstream`

```
#include <fstream>
```

C++

Pour inclure la librairie `fstream`

```
#include <fstream>
```

Pour déclarer un fichier d'écriture

```
ofstream file;
```

C++

Pour inclure la librairie `fstream`

```
#include <fstream>
```

Pour déclarer un fichier d'écriture

```
ofstream file;
```

Pour ouvrir un fichier

```
file.open("fichier.txt");
```

C++

Pour inclure la librairie `fstream`

```
#include <fstream>
```

Pour déclarer un fichier d'écriture

```
ofstream file;
```

Pour ouvrir un fichier

```
file.open("fichier.txt");
```

Explication

- `fichier.txt` est le nom de notre fichier sur le disque
- Il se trouve dans le même dossier que nos fichiers sources, s'il n'existe pas il sera créé et s'il existe il sera écrasé
- Il est possible d'indiquer un chemin différent : par exemple `file.open("C:/Users/elmou/fichier.txt");`

C++

Avant d'utiliser le fichier, on teste si le fichier a été ouvert avec succès

```
if(file)
{
    // Ouverture réussie, on peut utiliser le fichier
}
else
{
    cout << "Problème d'ouverture de fichier" << endl;
}
```

C++

Avant d'utiliser le fichier, on teste si le fichier a été ouvert avec succès

```
if(file)
{
    // Ouverture réussie, on peut utiliser le fichier
}
else
{
    cout << "Problème d'ouverture de fichier" << endl;
}
```

Pour écrire dans le fichier

```
if (file)
{
    file << "bonjour" << endl;
    file << 'a' << endl;
    file << 85 << endl;
}
else
{
    cout << "Problème d'ouverture de fichier" << endl;
}
```

C++

Avant d'utiliser le fichier, on teste si le fichier a été ouvert avec succès

```
if(file)
{
    // Ouverture réussie, on peut utiliser le fichier
}
else
{
    cout << "Problème d'ouverture de fichier" << endl;
}
```

Pour écrire dans le fichier

```
if (file)
{
    file << "bonjour" << endl;
    file << 'a' << endl;
    file << 85 << endl;
}
else
{
    cout << "Problème d'ouverture de fichier" << endl;
}
```

C++

Pour écrire à la fin du fichier, on ajoute le paramètre `app` (pour append)

```
file.open("fichier.txt", ios::app);
```

C++

Pour écrire à la fin du fichier, on ajoute le paramètre `app` (pour append)

```
file.open("fichier.txt", ios::app);
```

Pour fermer

```
if (file)
{
    file << "bonjour" << endl;
    file << 'a' << endl;
    file << 85 << endl;
    file.close();
}
else
{
    cout << "Problème d'ouverture de fichier" << endl;
}
```

C++

Pour déclarer un fichier de lecture (`input file stream`)

```
ifstream file;
```

C++

Pour déclarer un fichier de lecture (`input file stream`)

```
ifstream file;
```

Pour ouvrir un fichier

```
file.open("fichier.txt");
```

C++

Pour déclarer un fichier de lecture (`input file stream`)

```
ifstream file;
```

Pour ouvrir un fichier

```
file.open("fichier.txt");
```


Trois fonctions/opérateurs pour lire le contenu

- `get()` permet de lire un caractère
- `getline()` permet de lire une ligne
- `>>` permet de lire un mot

C++

Pour lire le fichier ligne par ligne

```
if(file)
{
    string line;
    while(get(file, line))
    {
        cout << line << endl;
    }
}
else
{
    cout << "Problème d'ouverture de file" << endl;
}
```

C++

Pour lire le fichier ligne par ligne

```
if(file)
{
    string line;
    while(get(file, line))
    {
        cout << line << endl;
    }
}
else
{
    cout << "Problème d'ouverture de file" << endl;
}
```

`getline()` : place la ligne suivante dans `line` et retourne `true`. S'il n'y a plus de ligne à lire, elle retourne `false`.

C++

Pour lire le fichier caractère par caractère

```
if(file)
{
    char c;
    while(file.get(c))
    {
        cout << c << endl;
    }
}
else
{
    cout << "Problème d'ouverture de file" << endl;
}
```

C++

Pour lire le fichier caractère par caractère

```
if(file)
{
    char c;
    while(file.get(c))
    {
        cout << c << endl;
    }
}
else
{
    cout << "Problème d'ouverture de file" << endl;
}
```

`get()` : place le caractère suivant dans `line` et retourne `true`. S'il n'y a plus de caractère à lire, elle retourne `false`.

C++

Pour lire le fichier mot par mot

```
if(file)
{
    string word;
    while (file >> word)
    {
        cout << word << endl;
    }
}
else
{
    cout << "Problème d'ouverture de file" << endl;
}
```

C++

Pour lire le fichier mot par mot

```
if(file)
{
    string word;
    while (file >> word)
    {
        cout << word << endl;
    }
}
else
{
    cout << "Problème d'ouverture de file" << endl;
}
```

Les séparateurs considérés sont l'espace et le retour à la ligne

C++

Remarque

Il est aussi possible de lire un nombre entier ou réel.

C++

Remarque

Il est aussi possible de lire un nombre entier ou réel.

Exemple

```
double x;
file >> x;
```

C++

Pour supprimer un fichier

```
if (remove("fichier.txt") == 0)
{
    cout << "Fichier supprimé avec succès" << endl;
}
else
{
    cout << "Suppression impossible" << endl;
}
```