Fichiers

R.Champagnat, E. Carnovali, J.Bohé, **M.Hamdi**

IUT de La Rochelle / département Informatique

4 décembre 2022

Au cours précédent

- Les fonctions
- La portée des variables
- Les références
- Les pointeurs

Plan

Sommaire

Entrées/sorties Fichiers I

Définition

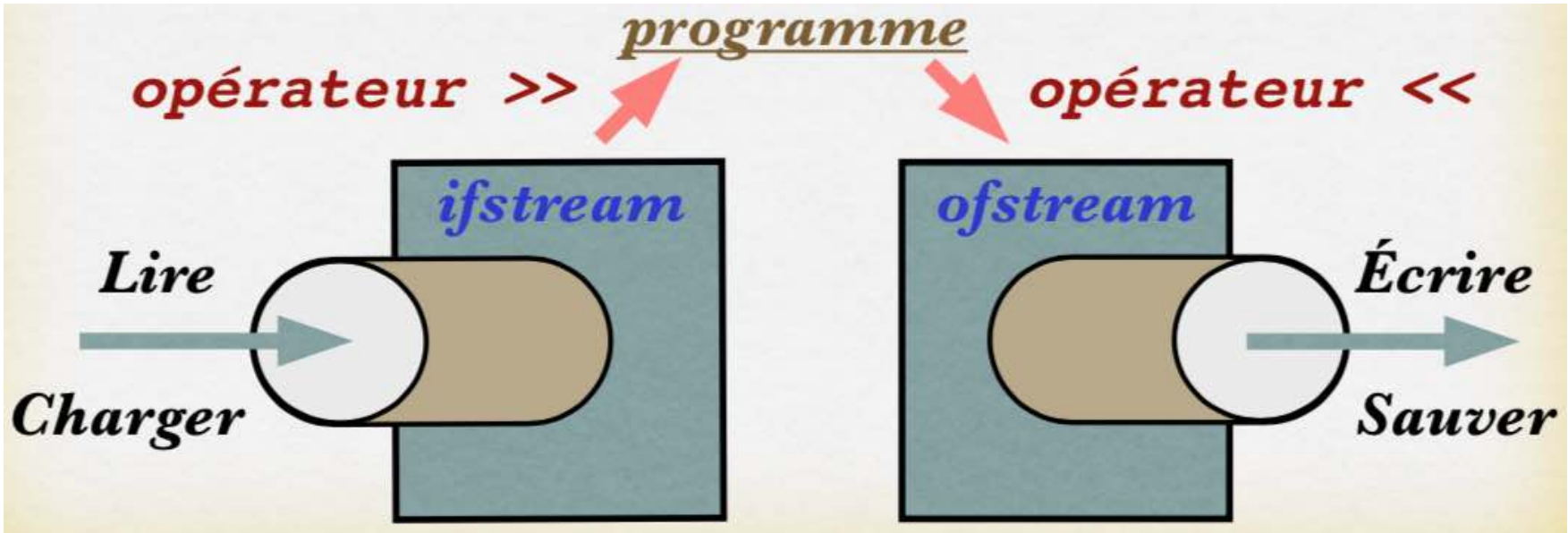
Un fichier est un ensemble de données structurées suivant un format spécifique à une application donnée. Il porte un nom unique destiné à l'identifier.

- Utilisation : pour charger/sauvegarder de manière persistantes les variables (données) lors de l'exécution d'un programme
- Format des données
  - Texte (.txt, .cpp)
  - Texte encodé (.doc, .xls, .csv)
  - Texte marqué (.xml, .html)

Déclaration d'une variable pour manipuler un fichier I

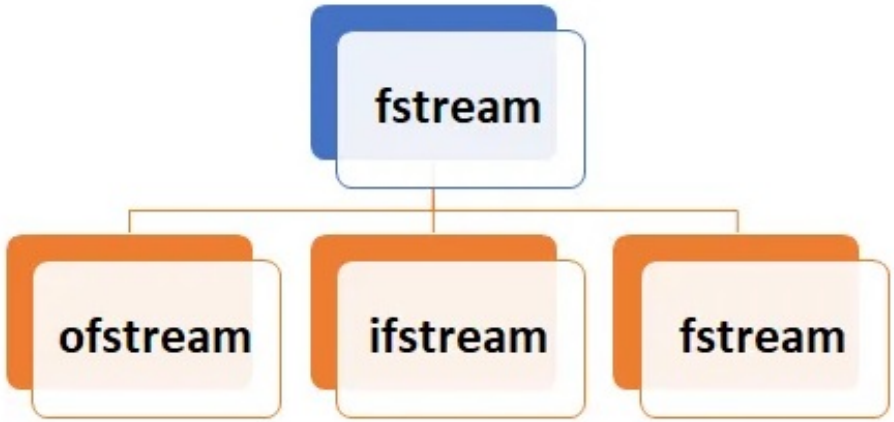
Déclaration

Il faut associer une variable (au niveau du programme) à un descripteur de fichier (sur un support en terme de localisation physique).



Déclaration d'une variable pour manipuler un fichier II

- Pour lire ou écrire dans un fichier, on doit inclure le fichier d'en-tête **fstream** ("file stream" ou "flux vers les fichiers", en français).
- On doit créer un objet de type **ofstream** pour ouvrir un fichier en écriture, **ifstream** pour l'ouvrir en lecture et **fstream** pour les deux.



Déclaration d'une variable pour manipuler un fichier III

- **ofstream (Input FileStream)** : Cette classe représente un flux de sortie. Elle est utilisée pour créer des fichiers et écrire des données dans des fichiers.
- **ifstream (Output File Stream)** : Cette classe représente un flux d'entrée. Elle est utilisée pour lire les données des fichiers.

```
2  #include <fstream>
3  using namespace std;
4
5  int main(){
6      ifstream inputFile("Dev1_load.txt");
7      ofstream outputFile("Dev1_save.txt");
8      ...
9  }
```

Ouverture d'un fichier I : Mode d'ouverture

- Lors de l'ouverture d'un fichier, on peut spécifier le mode d'ouverture en ajoutant un deuxième paramètre à la création du flux.

ios::in	Fichier ouvert en lecture.
ios::out	Fichier ouvert en écriture.
ios::binary	Fichier binaire, ne faire aucun formatage.
ios::ate	Aller à la fin du fichier à l'ouverture (au lieu de rester au début).
ios::app	Ajoute les données, en écrivant toujours à la fin.
ios::trunc	Supprime le contenu du fichier, s'il existe déjà.

Ouverture d'un fichier II : Mode d'ouverture

- Un exemple d'ouverture de fichier en écriture avec **fstream**.

```
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5  int main() {
6      fstream outputFile;
7      outputFile.open("Dev1_save", ios::out);
8      }
9      return 0;
10 }
```

- Pour pouvoir par exemple écrire à la fin d'un fichier, il faut le spécifier lors de l'ouverture : **ofstream monFlux(nomFichier, ios : :app) ;**.

Ouverture d'un fichier III

- Des problèmes peuvent survenir lors de l'ouverture d'un fichier, si le fichier ne vous appartient pas ou si le disque dur est plein, par exemple.

```
2  #include <fstream>
3  using namespace std;
4
5  int main() {
6      ifstream inputFile("Dev1_load.txt");  //Ouverture d'un fichier en lecture
7
8      if(inputFile)  //On teste si tout est OK
9      {
10         //Tout est OK, on peut utiliser le fichier
11     }
12     else
13     {
14         cout << "ERREUR: Impossible d'ouvrir le fichier." << endl;
15     }
16 }
```

Fermeture et ouverture d'un fichier I

- La règle générale pour créer un fichier est la suivante :
  - il faut l'ouvrir en écriture.
  - on écrit des données dans le fichier.
  - on ferme le fichier.
- Pour lire des données écrites dans un fichier :
  - on l'ouvre en lecture.
  - on lit les données en provenance du fichier.
  - on ferme le fichier.

Fermeture et ouverture d'un fichier II

- Exemple :

```
2  #include <fstream>
3  using namespace std;
4  int main() {
5      ofstream  outputFile("Dev1-save.txt");
6      ...
7      outputFile.close();
8      outputFile.open("Dev1-save.txt");
9      ...
10     outputFile.close();
11 }
```

Écrire dans un fichier

- Afin d'écrire dans un fichier, il faut l'ouvrir en écriture et écrire le message souhaité avec l'opérateur **monFlux** << **"Texte" ;**

```
2  #include <fstream>
3  using namespace std;
4
5  int main() {
6      ofstream  outputFile("Dev1-save.txt");
7      outputFile << "Un message" << endl;
8  }
9
```

Lire dans un fichier I

- Caractère par caractère, en utilisant get()

```
2  char a;
3  inputFile.get(a); // lit une seule lettre et la stocke dans la variable a
```

- Mot par mot (ou nombre), en utilisant les chevrons >>.

```
2  double nombre;
3  inputFile >> nombre; //Lit un nombre à virgule depuis le fichier
4  string mot;
5  inputFile >> mot;    //Lit un mot depuis le fichier
```

- Ligne par ligne, en utilisant getline().

```
2  string ligne;
3  getline(inputFile, ligne); //On lit une ligne complète
```

Lire dans un fichier II

- Changement de mode.

```
2  #include <fstream>
3  #include <string>
4  using namespace std;
5  int main() {
6      ifstream inputFile("Dev1-load.txt");
7      int number;
8      inputFile >> number;
9      inputFile.ignore(); // changement de mode
10     string text;
11     getline(inputFile, text);
12 }
```

Sommaire

Parcours séquentiel I

- Objectif : parcourir tous les éléments d'un fichier ligne par ligne
- Principe : tant que l'on a pas atteint la fin du fichier, lire une ligne
- Si on ne connaît pas *a priori* le nombre de lignes contenues dans le fichier, il faut utiliser la « fonction » `eof()` et une boucle
- Exemple

Parcours séquentiel II

```
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5
6  int main() {
7      ifstream inputFile
8      inputFile.open("Dev1_Load.txt");
9      string s;
10     int i;
11     while(!inputFile.eof()) // tant que l'on a pas atteint la fin du fichier
12     {
13         inputFile >> s;
14         inputFile.ignore();
15         cout << s << endl;
16         // ou
17         getline(inputFile, s);
18         cout << s << endl;
19         inputFile >> i;
20         cout << i << endl;
21     }
22 return 0;
23 }
```

Recherche Maximum I

- Objectif : dans un fichier contenant un nombre par ligne, trouver le nombre le plus grand
- Principe : utiliser une variable intermédiaire initialisée avec le premier nombre puis parcourir le fichier, si un nombre est plus grand le mémoriser
- Exemple

Recherche Maximum II

```
 2  #include <iostream>
 3  #include <fstream>
 4  using namespace std;
 5
 6  int main() {
 7      ifstream inputFile("Dev1_Load.txt");
 8      int max;
 9      int nb;
10      inputFile >> max;
11      while(!inputFile.eof())
12      {
13          inputFile >> nb;
14          if (nb > max)
15          {
16              max = nb;
17          }
18      }
19      cout << max << endl;
20  return 0;
```

Accumuler I

- Objectif : accumuler tous les éléments d'un fichier (somme des éléments par exemple)
- Principe : tant que l'on a pas atteint la fin du fichier, lire une ligne et ajouter son contenu à une somme.
- Exemple

Accumuler II

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    ifstream inputFile
    inputFile.open("Dev1_Load.txt");
    int sum;
    sum = 0;
    int i;
    while(!inputFile.eof())
    {
        inputFile >> i;
        sum = sum + i;
    cout << sum << endl;
return 0;
}
```

Sommaire

Résumé I

- Pour lire ou écrire dans un fichier, on doit inclure le fichier d'en-tête **fstream** .
- On doit créer un objet de type **ofstream** pour ouvrir un fichier en écriture, et **ifstream** pour l'ouvrir en lecture.
- L'écriture se fait comme avec **cout : monFlux** << **"Texte" ;** , tandis que la lecture se fait comme avec **cin : monFlux** >> **variable ;**.
- On peut lire un fichier ligne par ligne avec **getline()**.