Introduction à la programmation
Travaux pratiques: séance 11
INFO0201-1

X. Baumans
(xavier.baumans@ulg.ac.be)

04/04/2014

Programme de la séance

- Lecture/Ecriture dans un fichier

  $\rightarrow$ Chargement de plus de données qu'au clavier

  $\rightarrow$ (ré-)Utilisation de données plus complexes

  $\rightarrow$ Sauvegarde de données

Lire/Ecrire dans un fichier

Rappel : Pour la saisie au clavier et l'affichage à l'écran, il existe deux objets **cin** et **cout** :

- Ce sont en fait des variables un peu particulières : des "objets", qui représentent la console et le clavier ;
- On utilise les opérateurs "«" et "»" pour lire et écrire dans ces objets.

Lire/Ecrire dans un fichier

Rappel : Pour la saisie au clavier et l'affichage à l'écran, il existe deux objets **cin** et **cout** :

- Ce sont en fait des variables un peu particulières : des "objets", qui représentent la console et le clavier ;
- On utilise les opérateurs "«" et "»" pour lire et écrire dans ces objets.

Pour écrire et lire dans des fichiers, il existe des types d'objets similaires, **ifstream** et **ofstream**. Pour les utiliser, il est nécessaire d'inclure la librairie **fstream** :

```
1 #include <fstream>
```

- **ifstream** : "*input-file-stream*" est un "flux de fichier en entrée" qui permet de lire un fichier et d'en placer le contenu dans des variables du programme ;
- **ofstream** : "*output-file-stream*" est un "flux de fichier en sortie" qui permet d'écrire des données dans un fichier.

Ecrire dans un fichier (1/4)

Etapes pour l'écriture dans un fichier :

1. Ouvrir le fichier
   $\rightarrow$ Permet de spécifier dans quel fichier écrire et de quelle manière.

Ecrire dans un fichier (1/4)

Etapes pour l'écriture dans un fichier :

1. Ouvrir le fichier
   $\rightarrow$ Permet de spécifier dans quel fichier écrire et de quelle manière.
2. Vérifier que le fichier est correctement ouvert
   $\rightarrow$ Si un problème est survenu lors de l'ouverture du fichier, on ne pourra pas écrire dedans. Il faut donc vérifier d'abord.

Ecrire dans un fichier (1/4)

Etapes pour l'écriture dans un fichier :

1. Ouvrir le fichier
   $\rightarrow$ Permet de spécifier dans quel fichier écrire et de quelle manière.
2. Vérifier que le fichier est correctement ouvert
   $\rightarrow$ Si un problème est survenu lors de l'ouverture du fichier, on ne pourra pas écrire dedans. Il faut donc vérifier d'abord.
3. Ecrire dans le fichier
   $\rightarrow$ Ecrire toutes les données souhaitées dans le fichier.

Ecrire dans un fichier (1/4)

Etapes pour l'écriture dans un fichier :

1. Ouvrir le fichier
   → Permet de spécifier dans quel fichier écrire et de quelle manière.
2. Vérifier que le fichier est correctement ouvert
   → Si un problème est survenu lors de l'ouverture du fichier, on ne pourra pas écrire dedans. Il faut donc vérifier d'abord.
3. Ecrire dans le fichier
   → Ecrire toutes les données souhaitées dans le fichier.
4. Fermer le fichier
   → Lorsque les opérations d'écriture sont terminées, il est nécessaire de fermer le fichier pour signaler au système qu'on n'y accèdera plus. Si l'écriture des données n'était pas terminée, cela force également le système à le faire à ce moment-là.

Ecrire dans un fichier (2/4)

La première chose à faire est d'ouvrir le fichier. Pour cela, il faut déclarer un objet **ofstream** avec les paramètres suivants :

ofstream nom_variable("*chemin_vers_le_fichier*", *mode*) ;

- "*chemin_vers_le_fichier*" : une chaîne de caractères qui indique le nom du fichier à ouvrir ;
- *mode* : paramètres qui indiquent la manière d'ouvrir le fichier :
  - **ios**::**trunc** (truncate / tronquer) : spécifie que si le fichier existe déjà, son contenu précédent doit être écrasé ;
  - **ios**::**app** (append / ajouter) : si le fichier existe déjà, ajouter les nouvelles données à la suite sans effacer les précécentes.

  Dans les 2 cas, si le fichier n'existe pas encore, un nouveau fichier vide est créé automatiquement.

```
1 // Ouvre un fichier "data.txt" en écriture, auquel on
     pourra accéder par l'objet mon_fichier
2 ofstream mon_fichier("data.txt", ios::trunc);
```

Ecrire dans un fichier (3/4)

Vérifier l'ouverture du fichier : il faut ensuite vérifier que le fichier est correctement ouvert. Il existe pour cela une fonction **is_open()** qui retourne une valeur

- **true** si le fichier est correctement ouvert ;
- **false** si une erreur s'est produite lors de l'ouvertue du fichier.

Cette fonction ne prend pas d'argument, et elle s'utilise avec un objet fichier précédemment déclaré de la manière suivante :

```
1 mon_fichier.is_open()
```

Ecrire dans un fichier (4/4)

Ecrire dans le fichier : l'écriture dans le fichier s'effectue au moyen de l'opérateur "«", comme pour cout.

```
1 mon_fichier << "Hello" << endl;
2 mon_fichier << "5 + 4 = " << 5 + 4 << endl;
```

De la même manière qu'avec cout, on peut fixer le nombre de chiffres significatifs à afficher avec

```
1 mon_fichier.precision(valeur);
```

Fermer le fichier : pour fermer le fichier on utilise la fonction **close()**, qui ne prend pas d'argument :

```
1 mon_fichier.close();
```

Ecrire dans un fichier : exemple

```
1  #include <fstream>
2
3  int main()
4  {
5    int age = 19;
6    ofstream mon_fichier("data.txt", ios::trunc);
7    if(mon_fichier.is_open()) // fichier bien ouvert ?
8    {
9      // Le fichier est bien ouvert, on peut écrire
10     mon_fichier << "J'ai " << age << " ans." << endl;
11     mon_fichier.close(); // fermer le fichier
12   }
13   else
14   {
15     cout << "ERREUR: Impossible d'ouvrir le fichier";
16   }
17   return 0;
18 }
```

Lire dans un fichier

Pour lire un fichier, la méthode est similaire.
Un objet de type **ifstream** est cette fois-ci utilisé, et le sens “»” des chevrons utilisés change, comme pour l'utilisation de cout et cin.
Les étapes pour lire dans un fichier sont

1. Ouvrir le fichier
2. Vérifier que le fichier est correctement ouvert
3. Lire les données du fichier
4. Fermer le fichier

Lire dans un fichier

Pour ouvrir un fichier en lecture, on déclare un objet **ifstream** avec les paramètres suivants :

ifstream nom_variable(“*chemin_vers_le_fichier*”) ;

- “*chemin_vers_le_fichier*” : une chaîne de caractères qui indique le nom du fichier à ouvrir ;

```
1 // Ouvre un fichier "data_input.txt" en lecture,
      auquel on pourra accéder par l'objet mon_fichier
2 ifstream mon_fichier("data_input.txt");
```

Lire dans un fichier

Pour ouvrir un fichier en lecture, on déclare un objet **ifstream** avec les paramètres suivants :

ifstream nom_variable(“*chemin_vers_le_fichier*”) ;

- “*chemin_vers_le_fichier*” : une chaîne de caractères qui indique le nom du fichier à ouvrir ;

```
1 // Ouvre un fichier "data_input.txt" en lecture,
      auquel on pourra accéder par l'objet mon_fichier
2 ifstream mon_fichier("data_input.txt");
```

Il faut ensuite vérifier que le fichier est correctement ouvert, grâce à la même fonction **is_open()** que pour les fichiers en écriture.

```
1 mon_fichier.is_open()
```

Lire dans un fichier

Avant chaque opération de lecture, il faut ensuite vérifier que le fichier est toujours prêt à être lu grâce à la fonction **good()**. Cette fonction retourne une valeur

- **true** : si on peut encore lire dans le fichier ;
- **false** : si on ne peut plus lire dans le fichier, que ce soit parce qu'une erreur s'est produite, parce que la fin du fichier a été atteinte, etc...

```
1 mon_fichier.good()
```

Lire dans un fichier

Avant chaque opération de lecture, il faut ensuite vérifier que le fichier est toujours prêt à être lu grâce à la fonction **good()**. Cette fonction retourne une valeur

- **true** : si on peut encore lire dans le fichier ;
- **false** : si on ne peut plus lire dans le fichier, que ce soit parce qu'une erreur s'est produite, parce que la fin du fichier a été atteinte, etc...

```
1 mon_fichier.good()
```

On peut ensuite lire les données au moyen de l'opérateur "»" comme avec cin :

```
1 int a;
2 mon_fichier >> a;
```

Lire dans un fichier

Avant chaque opération de lecture, il faut ensuite vérifier que le fichier est toujours prêt à être lu grâce à la fonction **good()**. Cette fonction retourne une valeur

- **true** : si on peut encore lire dans le fichier ;
- **false** : si on ne peut plus lire dans le fichier, que ce soit parce qu'une erreur s'est produite, parce que la fin du fichier a été atteinte, etc...

```
1 mon_fichier.good()
```

On peut ensuite lire les données au moyen de l'opérateur "»" comme avec cin :

```
1 int a;
2 mon_fichier >> a;
```

Et pour finir, le fichier doit être fermé avec la fonction **close()** :

```
1 mon_fichier.close();
```

Lire dans un fichier : exemple

```
1  #include <fstream>
2  int main()
3  {
4    ifstream mon_fichier("data_input.txt"); //Ouverture du fichier
5    if(mon_fichier.is_open()) // fichier ouvert ?
6    {
7      while(mon_fichier.good()) // fichier lisible ?
8      {
9        int a = 0;
10       mon_fichier >> a;
11       cout << "Le fichier contient " << a << endl;
12     }
13     mon_fichier.close(); // fermer le fichier
14   }
15   else{
16     cout << "ERREUR: Impossible d'ouvrir le fichier" << endl;
17   }
18   return 0;
19 }
```

Exercices (1/2)

1. Écriture dans un fichier
   - Écrire un petit programme qui demande à l'utilisateur de rentrer au clavier une dizaine de pays européens assortis de leur capitale ;
   - Écrire les données ainsi obtenues dans un fichier comprenant deux colonnes : l'une intitulée "Pays" et l'autre "Capitales".
2. Lecture d'un fichier
   - Commencer par générer un petit fichier data.txt contenant une centaine de nombres aléatoires entiers compris entre 1 et 1000 ;
   - Charger les données du petit fichier data.txt ;
   - Les stocker dans un tableau et le trier au moyen d'une fonction tri ;
   - Copier les données nouvellement triées à la suite des anciennes dans le fichier de départ.

Exercices (2/2)

3. Évaluation de polynômes : un programme évalue un polynôme à intervalles réguliers et écrit les données dans un fichier pour pouvoir en tracer un graphique.
   - Écrire une fonction qui prend en paramètre un tableau qui contient les coefficients d'un polynôme P, une valeur x et qui retourne la valeur de $P(x)$;
   - Permettre à l'utilisateur d'évaluer un polynôme de son choix, dans l'intervalle de son choix avec le pas de son choix ;
   - Écrire les couples $(x, P(x))$ dans un fichier ;
   - Tester votre programme avec le polynôme $2x^4 - 3x^2 + x - 2$ dans l'intervalle $[-3, 3]$ avec un pas de 10^{-3}.