Cours sur les fichiers

Ecrire dans un fichier nécessite de bien comprendre comment les variables sont stockés dans l'ordinateur. Ca va donc être un bon prétexte pour quelques rappels, et quelques tests.

Test 1 : écrire des nombres dans un fichier texte

Testez le programme suivant, et modifiez-le pour qu'il produise un fichier texte un peu plus lisible.

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

int main()
{
	int nombre;
	ofstream fichier("monFichier.txt",ios_base::out);
	for(int i=0;i<10;i++){
		cout<<"saisissez un nombre (encore "<<10-i<<") : ";
		cin>>nombre;
		fichier<<nombre;
	}
	fichier.close();
	return 0;
}
```

Test 2 : écrire le contenu d'un tableau dans un fichier

Testez le programme suivant, et modifiez-le pour qu'il mette dans le fichier les valeurs de toutes les cases.

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

int main()
{
	int nombre[10];
	ofstream fichier("monFichier.txt",ios_base::out);
	for(int i=0;i<10;i++){
		cout<<"saisissez un nombre (encore "<<10-i<<") : ";
		cin>>nombre[i];
	}
	fichier<<nombre;
	fichier.close();
	return 0;
}
```

Test 3 : écrire une structure dans un fichier

Comme vous l'avez constaté au test précédent, il n'est pas possible de mettre directement tout le tableau dans le fichier, mais il faut le faire case par case. C'est un peu le même principe avec une structure, donc modifiez ce programme pour qu'il fonctionne correctement.

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

struct voiture{
	int nbKilometre;
	string modele;
	string couleur;
}

int main()
{
	voiture maVoiture;
	ofstream fichier("monFichier.txt",ios_base::out);
	cout<<"saisissez le modele, la couleur et les Km de la voiture :";
	cin>>maVoiture.modele>>maVoiture.couleur>>maVoiture.nbKilometre;
	//fichier<<maVoiture;
	fichier.close();
	return 0;
}
```

Test 4 : écrire un tableau de structure dans un fichier

Maintenant, compliquons un peu. Faites un programme qui demande à l'utilisateur de saisir 5 voitures (que vous stockerez dans un tableau), puis que vous enregistrez dans un fichier.

Test 5 : Lire un fichier texte et l'afficher à l'écran

Pour pouvoir utiliser un fichier en lecture, il faut changer le type de la variable **`fichier`** : au lieu d'être une variable de type **`ofstream`** c'est une variable de type **`ifstream`** :

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
	ifstream fichier("monFichier.txt",ios_base::in);
	string mot;
	fichier>>mot;
	while(!fichier.eof()){
		cout<<mot;
		fichier>>mot;
	}
	fichier.close();
	return 0;
}
```

Quelques remarques pour bien comprendre le code précédent :

Pour afficher tout le fichier, il faut lire, mot à mot, le contenu du fichier. Mais une fois qu'on arrive à la fin du fichier, il faut s'arrêter de lire… C'est pour cela que l'on met comme condition d'arrêt de la boucle **`while`** : **`(!fichier.eof())`**. La fonction **`eof()`** renvoie **`vrai`** quand on est arrivé à la fin du fichier. C'est pour ça que l'on doit inverser ce résultat pour rester dans la boucle grâce au point d'exclamation : **`!`**. La condition est donc à **`vrai`**, tant qu'on n'est pas à la fin du fichier…

REMARQUE : La fonction eof() n'est pas bien faite : elle détecte la fin du fichier que si on a dépassé la fin du fichier. C'est pour ça qu'il faut lire le fichier à la fin de la boucle pour que le test soit fait juste après.

Test 6 : Lire un fichier ligne par ligne

Vous l'aurez probablement constaté, le gros problème de cette première solution, c'est qu'il n'est pas possible d'afficher le fichier en gardant les retours à la ligne… Du coup, il faut changer de méthode :

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
	ifstream fichier("monFichier.txt",ios_base::in);
	string ligne;
	getline(fichier,ligne);
	while(!fichier.eof()){
		cout<<ligne<<endl;
		getline(fichier,ligne);
	}
	fichier.close();
	return 0;
}
```

Test 7 : lire autre chose que des chaines de caractères

Le gros problème qui se pose avec les fichiers textes, c'est que ce qu'on peut lire, c'est des mots… Mais si on veut lire des nombres ou même des structures plus complexes, ce n'est pas possible ! Donc il faut utiliser une autre fonction. Téléchargez le fichier `fonctions.h` sur mon site internet et ajoutez-le dans votre projet. Il faut aussi rajouter **`#include "fonctions.h"`** au début du fichier. Maintenant, vous pouvez utiliser la fonction `convertir()` que j'ai créée. Elle s'utilise en mettant comme premier paramètre une variable de type string qui contient un nombre et en deuxième paramètre la variable qui recevra la valeur. Par exemple, si on a que des nombres à virgules dans le fichier, on peut faire un truc de ce style :

```
string mot;
float monChiffre;
fichier>>mot;
while(!fichier.eof()){
	convertir(mot, monChiffre);
	cout<< monChiffre<<endl;
	fichier>>mot;
}
```

Conclusion et mini TP

Maintenant, vous savez comment utiliser les fichiers textes. Mais la réalité devient un peu plus compliquée… C'est ce que l'on va découvrir au cours de ce petit TP :

Une DVDtheque dans un fichier…

En vous servant du TP 17, vous devrez faire deux fonctions : une pour charger une DVDthèque, et une pour enregistrer une DVDthèque dans des fichiers.

Faites bien attention à l'organisation de votre fichier : réfléchissez bien à l'ordre des informations que vous écrirez dedans ! Voila un exemple de fichier :

```
Lelore
comedie aventure tragedie policier science_fiction historique
2
Persepolis Satrapi 95 francais tragedie
Scarface De_Palma 165 americain policier
```

Il faut donc à l'avance savoir que vous aurez en premier le propriétaire de la DVDthèque, ensuite les 6 genres de film, ensuite le nombre de DVDs et enfin les infos des différents DVDs. Pour vous aider, voila quelques conseils :

Les espaces dans les noms…

Si vous mettez un nom de film avec un espace, vous risquez de mettre le premier mot dans une variable et le deuxième mot dans une autre variable… Je vous conseille donc de mettre un nom par ligne et d'utiliser `getline()` pour récupérer chacune des lignes. L'exemple de fichier n'est donc pas conseillé !

Utiliser des fonctions…

Si vous ne découpez pas en sous-fonctions, vous risquez de vite vous embrouiller. Je vous conseille donc de faire ces deux fonctions :

- `DVD lireDVD(ifstream fichier)` : lit les 5 infos d'un DVD (titre, réalisateur, durée, nationalité et genre) à partir du fichier passé en paramètre et renvoie une variable de type DVD.
- `void ecrireDVD(ofstream fichier, DVD d)` : écrit les 5 infos du DVD dans le fichier passé en paramètre.

Une fois que vous avez fait ces deux fonctions, il faut les utiliser au bon moment… C'est-à-dire qu'il faut utiliser la fonction `lireDVD()` quand on est rendu dans le fichier au bon endroit (quand on est rendu à lire un DVD). Ça parait simple et logique, mais il faut y penser…

De plus, il faudra utiliser `ecrireDVD()` autant de fois que vous aurez de DVD. Donc à vous d'utiliser une boucle bien faite pour que la fonction marche comme il faut ;-)