Le langage C++ : Les fichiers

Les variables n'existent que dans la mémoire vive. Une fois votre programme arrêté, toutes vos variables sont supprimées de la mémoire et il n'est pas possible de retrouver ensuite leur valeur.

Heureusement, on peut lire et écrire dans des fichiers en langage C/C++. Ces fichiers seront écrits sur le disque dur de votre ordinateur : l'avantage est donc qu'ils restent là, même si vous arrêtez le programme ou l'ordinateur.
Pour lire et écrire dans des fichiers, nous allons avoir besoin de réutiliser tout ce que nous avons appris jusqu'ici : pointeurs, structures, chaînes de caractères, etc.

Généralité sur les fichiers

- 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.

Fichiers textes ou binaires

Il existe 2 types de fichiers :

- les fichiers textes qui sont des fichiers lisibles par un simple éditeur de texte.
- les fichiers binaires dont les données correspondent en général à une copie bit à bit du contenu de la RAM. Ils ne sont pas lisibles avec un éditeur de texte.

cstdio ou fstream

Il existe principalement 2 bibliothèques standard pour écrire des fichiers :

- cstdio qui provient en fait du langage C.
- fstream qui est typiquement C++.

Utilisation de cstdio

La fonction FILE * fopen(const char * filepath, char * mode)

Cette fonction permet d'ouvrir un fichier en lecture ou en écriture. Le paramètre filepath est un tableau de char contenant le chemin du fichier sur lequel on souhaite travailler. Le paramètre mode indique le mode d'ouverture de filepath : lecture ou écriture, texte ou binaire.

Le mode peut avoir l'une des valeurs suivantes :

- "r" (read) : lecture,
- "w" (write) : écriture, fichier créé ou écrasé s'il existait déjà,
- "a" (append) : écriture en fin de fichier existant (ajout de données).

Sur certaines plateformes (Windows par exemple), on peut y ajouter le type d'écriture (texte ou binaire), sur les autres (Linux par exemple) le type est ignoré :

- "b" : mode binaire,
- "t" : mode texte.

Enfin, on peut ajouter le signe "+" afin d'ouvrir le fichier en lecture et écriture à la fois.
Exemples :

- "r+" : ouverture en lecture et écriture,
- "wb" : ouverture en écriture binaire.

La fonction fopen retourne le pointeur NULL si l'ouverture du fichier a échouée. Dans le cas contraire, elle retourne un pointeur vers une structure FILE. Ce pointeur servira à écrire ou lire dans le fichier, ainsi qu'à le fermer.

La fonction fclose(FILE *)

Cette fonction permet de fermer un fichier, qu'il soit ouvert en lecture ou en écriture. On passe en paramètre à cette fonction le pointeur FILE * fourni par la fonction fopen(...).

Les fichiers binaires

La fonction fwrite(const void * buffer, int size,int nb, FILE * f) :

Cette fonction écrit nb élements de size octets (soit nb*size octets) à partir du pointeur buffer (dans la RAM) vers le fichier f qui doit être ouvert en écriture. Il s'agit donc d'un transfert d'octets de la RAM dans un fichier.

La fonction fread(const void * buffer, int size,int nb, FILE * f) :

Cette fonction lit nb élements de size octets (soit nb*size octets) à partir du fichier f (qui doit être ouvert en lecture) vers le pointeur buffer (dans la RAM). Il s'agit donc d'un transferts d'octets d'un fichier vers la RAM.

Exemple : écriture du fichier

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

int main (void)
{
  FILE * f;
  int a = 78, i, t1 [6];
  double b = 9.87;
  char c = 'W', t2 [10];

  for(i = 0; i < 6; i++)
   {
         t1 [i] = 10000 + i;
   }

  strcpy (t2, "AZERTYUIO");
  cout << t2 << endl;

  f = fopen ("toto.xyz", "wb");
  if (f == NULL)
  {
    cout << "Impossible d'ouvrir le fichier en écriture !" << endl;
  }
  else
  {

    fwrite (&a,sizeof(int),1,f);
    fwrite (&b,sizeof(double),1,f);
    fwrite (&c,sizeof(char),1,f);
    fwrite (t1,sizeof(int),6,f);
    fwrite (t2,sizeof(char),10,f);
    fclose (f);
  }
  return 0;
}
```

Dans ce programme, on ouvre le fichier binaire nommé toto.xyz en écriture. Si on a réussi à ouvrir le fichier, on y écrit un entier, un double, un char, puis un tableau de 6 entiers et finalement un tableau de 10 char.

On remarquera que pour écrire un entier il faut écrire &a pour obtenir un pointeur vers cet entier. Pour copier le tableau t1 on écrire juste t1 car t1 est déjà un pointeur vers le premier élément du tableau.

Exemple : lecture du fichier

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

int main (void)
{
  FILE * f;

  int a, t1 [6], i;
  double b;
```

```
    char c, t2[10];

    f = fopen("toto.xyz", "rb");

    if (f == NULL)
       {
      cout << "Impossible d'ouvrir le fichier en lecture !" << endl;
       }
    else
    {
      fread (&a,sizeof(int),1,f);
      fread (&b,sizeof(double),1,f);
      fread (&c,sizeof(char),1,f);
      fread (t1,sizeof(int),6,f);
      fread (t2,sizeof(char),10,f);
      fclose (f);
    }
       cout << "a=" << a << endl
       cout  << "b=" << b << endl
    cout << "c=" << c << endl;

    for (i = 0; i < 6; i++)
    {
            cout << t1 [i] << endl;
    }
    cout << t2 << endl;
    return 0;
  }
```

Dans ce programme, on ouvre le fichier binaire nommé toto.xyz en écriture. Si on a réussi à ouvrir le fichier, on lit un entier, un double un char, puis un tableau de 6 entiers et finalement un tableau de 10 char.

Les fichiers textes

la fonction fprintf(FILE *f, const char * format,...)

La fonction fprintf permet d'écrire en mode texte dans un fichier différentes données. On n'oubliera pas de laisser un espace entre les données pour pouvoir les relire (ou un passage à la ligne).
Le paramètre format permet de spécifier la nature des données et des caractéristiques sur leur écriture dans le fichier (le nombre de caractères par exemple).

Exemples de formats :
"%d" ==> indique un entier
"%lf" ==> indique un double
"%3.7lf" ==> indique un double avec 3 chiffres avant la virgule et 7 après.
"%s" ==> indique une chaine de caractères sans espace.

la fonction fscanf(FILE * f, const char * format,...)

La fonction scanf permet de lire les données à partir d'un fichier texte en utilisant le format de données indiqué (qui est identique à printf).

- **Exemple : écriture du fichier**

```
int main (void)
    {

      FILE * f;
      int a = 78, t1[6], i;
      double b = 9.87;
      char c = 'W', t2[10];

      for (i = 0; i < 6; i++)
         {
               t1 [i] = 10000+i;
         }

      strcpy (t2, "AZERTYUIO");
      f = fopen ("toto.txt", "wt");
      if (f == NULL)
       {
        cout << "Impossible d'ouvrir le fichier en écriture !" << endl;
       }

      else
      {
        fprintf (f, "%d %lf %c ", a, b, c);
        for (i=0;i<6;i++)
          {
            fprintf (f, "%d ", t1[i]);
          }
        fprintf (f, "%s ", t2);
        fclose (f);
      }

      return 0;

    }
```

Dans ce programme, on ouvre le fichier texte nommé toto.txt en écriture. Si on a réussi à ouvrir le fichier, on y écrit un entier, un double, un char, puis un tableau de 6 entiers et finalement une chaîne de caractères sans espace contenu dans un tableau de char.

Exemple : lecture du fichier

```
int main (void)
    {

      FILE * f;
      double b;
      int a, t1 [6], i;
      char c, t2 [10];

      f = fopen ("toto.txt", "rt");
      if (f == NULL)
         {
               cout << "Impossible d'ouvrir le fichier en lecture !" << endl;
         }
      else
      {

        fscanf (f, "%d %lf %c ", &a, &b, &c);
        for (i = 0; i < 6; i++)
```

```
            {
                    fscanf (f, "%d ", &t1 [i]);
            }
          fscanf (f, "%s ", t2);
       fclose (f);
    }

    cout << "a=" << a << endl
    cout << "b=" << b << endl
    cout << "c=" << c << endl;

    for (i = 0; i < 6; i++)
        {
            cout << t1 [i] << endl;
        }
    cout << t2 << endl;
       return 0;
}
```

Dans ce programme, on ouvre le fichier binaire nommé toto.txt en lecture. Si on a réussi à ouvrir le fichier, on lit un entier, un double, un char, puis un tableau de 6 entiers et finalement une chaîne de caractères.

Utilisation de fstream

Les fichiers textes

La classe ofstream :

Il s'agit d'un fichier ouvert en écriture : pour créer un tel fichier il suffit d'appeler le constructeur qui a en paramètre le nom du fichier : par exemple **ofstream f("toto.txt");**.
Pour savoir si le fichier a bien été ouvert en écriture la méthode is_open() renvoie true si le fichier est effectivement ouvert.
Pour écrire dans le fichier on utilise l'opérateur << sans oublier d'écrire des séparateurs dans le fichier texte.

La classe ifstream :

Il s'agit d'un fichier ouvert en lecture : pour créer un tel fichier il suffit d'appeler le constructeur qui a en paramètre le nom du fichier : par exemple **ifstream f("toto.txt");**.
Pour savoir si le fichier a bien été ouvert en lecture la méthode is_open() renvoie true si le fichier est effectivement ouvert.

Pour lire dans le fichier on utilise l'opérateur >>.

- **Exemple : écriture d'un fichier texte**

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

 int main(void)

 {
   int a = 78, t1 [6], i;
   double b = 9.87;
```

```
            char c = 'W';
            char s[256]= "AZERTYUIO";


            for (i = 0; i < 6; i++)
             {
                      t1 [i] = 10000+i;
             }

            ofstream f ("toto.txt");
            if (!f.is_open())
               {
                    cout << "Impossible d'ouvrir le fichier en écriture !" << endl;
               }
            else
            {
              f << a << " " << b << " " << c << endl;
              for (i = 0; i < 6; i++)
               {
                f << t1 [i] << " ";
               }
              f << s;
            }
            f.close();
            return 0;
          }
```

Exemple : lecture d'un fichier texte

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

 int main (void)
 {
   int a, t1 [6], i;
   double b;
   char c;
   string s;
   ifstream f ("toto.txt");

   if (!f.is_open())
      {
     cout << "Impossible d'ouvrir le fichier en lecture !" << endl;
      }
   else
   {
     f >> a >> b >> c;
     for (i = 0; i < 6; i++)
          {
                f >> t1 [i];
          }
     f >> s;
   }

   f.close();
   cout << "a=" << a << endl
   cout << "b=" << b <<endl
   cout << "c=" << c <<endl;
```

```
    for (i = 0; i < 6; i++)
     {
              cout << t1 [i] << endl;
     }
    cout << s << endl;
    return 0;

}
```

Les fichiers binaires

- **La classe ofstream :**

Pour ouvrir en écriture un fichier binaire, il suffit d'appeler le constructeur qui a en paramètre le nom du fichier et le mode d'ouverture et fixer ce deuxième paramètre à ios::out | ios::binary: par exemple **ofstream f("toto.xyz",ios::out | ios::binary);**.
Pour savoir si le fichier a bien été ouvert en écriture la méthode is_open() renvoie true si le fichier est effectivement ouvert.
Pour écrire dans le fichier on utilise la méthode write((char *)buffer , int nb) pour écrire nb octets dans ce fichier.

- **La classe ifstream :**

Pour ouvrir en lecture un fichier binaire, il suffit d'appeler le constructeur qui a en paramètre le nom du fichier et le mode d'ouverture et fixer ce deuxième paramètre à ios::in | ios::binary: par exemple **ifstream f("toto.xyz",ios::in | ios::binary);**.
Pour savoir si le fichier a bien été ouvert en écriture la méthode is_open() renvoie true si le fichier est effectivement ouvert.
Pour lire dans le fichier on utilise la méthode read((char *)buffer , int nb) pour lire nb octets de ce fichier.

- **Ecriture d'un fichier binaire**

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

 int main (void)

 {
   int a = 78, t1 [6], i;
   double b = 9.87;
   char c = 'W';

   for (i = 0; i < 6; i++)
      {
           t1 [i] = 10000+i;
      }

   ofstream f ("toto.xyz", ios::out | ios::binary);
   if(!f.is_open())
      {
     cout << "Impossible d'ouvrir le fichier en écriture !" << endl;
      }
```

```
   else
   {

            f.write ((char *)&a, sizeof(int));
            f.write ((char *)&b, sizeof(double));
            f.write ((char *)&c, sizeof(char));
            for (i = 0; i < 6; i++)
                  {
                        f.write ((char *)&t1[i], sizeof(int));
                  }
     }
     f.close();
     return 0;
}
```

- **lecture d'un fichier binaire**

```
int main (void)
    {
      int a, t1 [6], i;
      double b;
      char c;
      char s[256];

      ifstream f ("toto.xyz", ios::in | ios::binary);
      if(!f.is_open())
      {
         cout << "Impossible d'ouvrir le fichier en écriture !" << endl;
      }
       else
       {

         f.read ((char *)&a, sizeof(int));
         f.read ((char *)&b, sizeof(double));
         f.read ((char *)&c, sizeof(char));
         for (i = 0; i < 6; i++)
           {
           f.read ((char *)&t1[i], sizeof(int));
           }

       }
       f.close();
       cout << "a=" << a << endl
       cout << "b=" << b << endl
       cout << "c=" << c << endl;

       for (i = 0; i < 6; i++)
      {
         cout << t1 [i] << endl;
      }
       return 0;
    }
```