Chapitre 14:

Les Fichiers en C++

Introduction

1- Définition et propriétés

- **Un fichier est une suite de données homogènes conservées en permanence sur un support externe (disquette, disque dur,...).**

- **En C++, les fichiers sont considérés comme une suite d'octets (1 octet=1 caractère).**

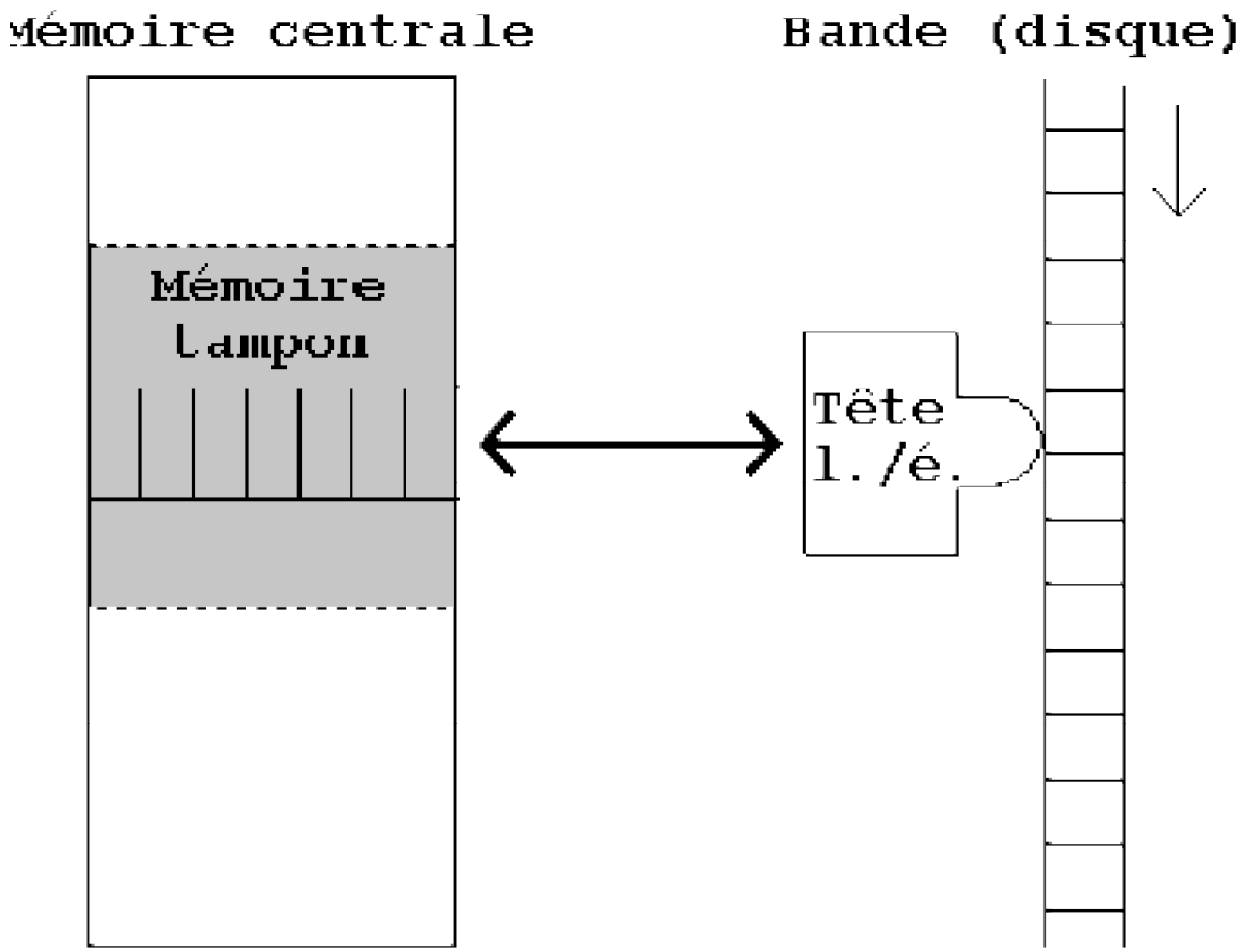
- **Principe de manipulation d'un fichier:**
  1- Ouverture du fichier,
  2- Lecture, écriture, et déplacement dans le fichier,
  3- Fermeture du fichier.

2- Mémoire tampon

- Les accès à un fichier (en vue d'une lecture ou écriture d'information) se font par l'intermédiaire d'une mémoire tampon (buffer).

- Il s'agit d'une zone de la mémoire centrale qui stocke une quantité, assez importante de données du fichier.

- L'utilisation de la mémoire tampon a l'effet de réduire le nombre d'accès au support externe d'une part et le nombre des mouvements de la tête de lecture/écriture d'autre part.


Mémoire centrale
Bande (disque)
Mémoire
Tampon
Tête
l./é.

3- Flux entrées-Sorties :

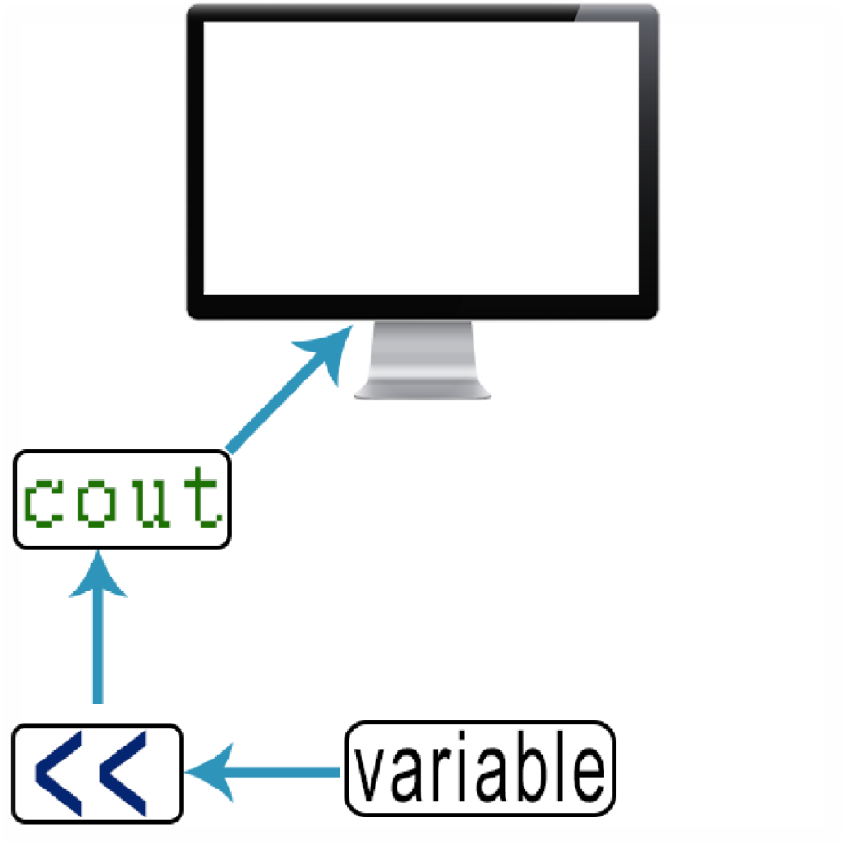
- ✓ On parle de flux pour désigner les moyens de communication d'un programme avec l'extérieur. Les flux (anglais : stream) correspondent aux échanges réalisés entre un programme et les périphériques : clavier, écran, disque dur, etc
- ✓ Les entrées-sorties sont toujours réalisées comme des flots de données (flux) qu'on peut symboliser comme des canaux entre le programme en mémoire et les différents périphériques :

  *Clavier -programme : flux d'entrée cin, associé à l'opérateur >>

  * Programme écran : flux de sortie cout, associé à l'opérateur <<

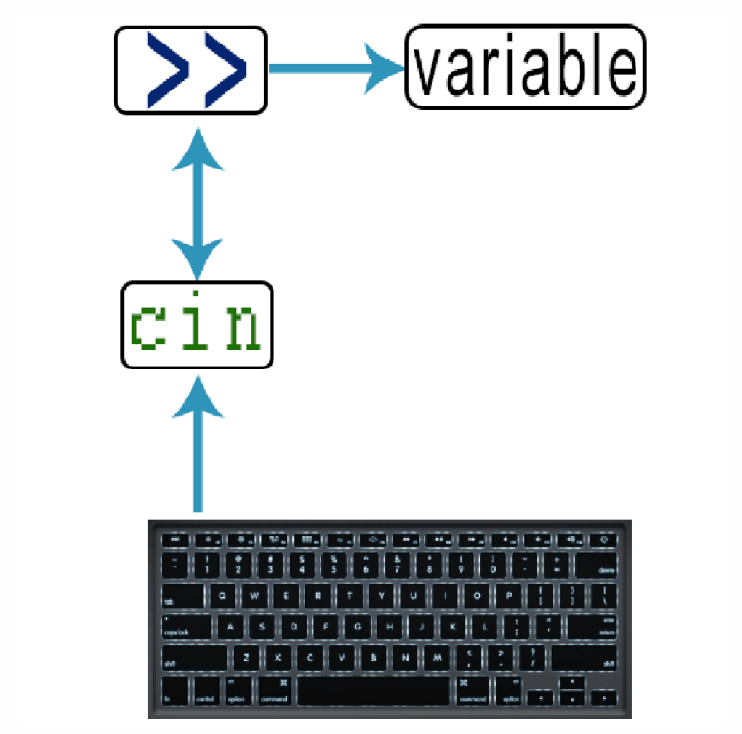
Exemple Flux de sortie

Flux cout :



Exemple Flux d’entrée

Flux cin :



Librairie (Bibliothèque)

L’utilisation d’un fichier comporte en général 3 phases :

- ✓ **L’ouverture d’un flux selon un certain mode (lecture, écriture, etc)**
- ✓ **La lecture ou l’écriture selon le mode**
- ✓ **La fermeture du flux.**

Pour utiliser les fichiers en C++, il est nécessaire d'inclure le fichier d'en-tête <fstream >

Les Modes d’ouverture d’un fichier

1- Ouvrir un fichier en lecture: ifstream

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
        ifstream fichier("test.txt", ios::in);  // on ouvre le fichier en lecture

        if(fichier)  // si l'ouverture a réussi
        {
            // instructions
                fichier.close();  // on ferme le fichier
        }
        else  // sinon
                cerr << "Impossible d'ouvrir le fichier !" << endl;

        return 0;
}
```

2-Ouvrir un fichier en écriture: ofstream

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
        ofstream fichier("test.txt", ios::out | ios::trunc);  //déclaration du flux

        if(fichier)  // si l'ouverture a réussi
        {
            // instructions
                fichier.close();  // on referme le fichier
        }
        else  // sinon
                cerr << "Erreur à l'ouverture !" << endl;

        return 0;
}
```

- Pour ouvrir le fichier il faut un flux agissant sur le fichier.
- Pour chaque fichier et/ou mode d'ouverture, il faut un flux différent

3- Mode d’ouverture

Pour l'écriture, il y a différents modes d'ouverture :

- **ios::out (output) : spécifie qu'on ouvre le fichier en écriture -par défaut - quand on utilise un objet ofstream .**
- **ios::app (append = ajouter à la suite) : lorsqu'on ouvre le fichier en écriture, on se trouve à la fin pour écrire des données à la suite du fichier (sans effacer le contenu, s'il y en a un). Avec ce mode d'ouverture, à chaque écriture, on est placé à la fin du fichier.**

- **ios::trunc (truncate = tronquer) : lorsqu'on ouvre le fichier en écriture, spécifie qu'il doit être effacé s'il existe déjà, pour laisser un fichier vide.**

- **ios::ate (at end) : ouvre le fichier en écriture et positionne le curseur à la fin de celui-ci. La différence avec ios::app est que, pour cette dernière, si on se repositionne dans le fichier, l'écriture ne se fera pas forcément**

- **Pour spécifier plusieurs modes d'ouverture, on utilise l'opérateur "ou" : | (pipe en anglais).**

```
8       ofstream fichier("test.txt", ios::out | ios::trunc);  //déclaration du flux
```

- **En utilisant fstream, on ouvre en lecture et en écriture un fichier. Le fonctionnement est le même que pour ifstream ou ofstream.**

fstream flux("fichier.extention", ios::in | ios::out | [ios::trunc | ios::ate]);

Lire le contenu d’un fichier

Lire le contenu d’un fichier

Pour lire un fichier, il y a différentes méthodes qu'il faut employer en fonction de ce que l'on veut faire :

- **getline(flux, chaineDeCaractères)** : pour lire une ligne complète .
- **flux.get(caractère) :** pour lire un caractère .
- **flux >> variable :** pour récupérer à partir du fichier jusqu'à un délimiteur (espace, saut à la ligne, ...).

A- La fonction getline()

```cpp
#include <iostream>
#include <string>
#include <fstream>

using namespace std;

int main()
{
	ifstream fichier("test.txt", ios::in);  // on ouvre en lecture

	if(fichier)  // si l'ouverture a fonctionné
	{
		string contenu;  // déclaration d'une chaîne qui contiendra la ligne
		getline(fichier, contenu);  // on met dans "contenu" la ligne
		cout << contenu;  // on affiche la ligne

		fichier.close();
	}
	else
		cerr << "Impossible d'ouvrir le fichier !" << endl;

	return 0;
}
```

A- La fonction getline()

Lire un fichier en entier :

```
1 if(fichier)
2 {
3         string ligne;
4         while(getline(fichier, ligne))
5         {
6                 cout << ligne << endl;
7         }
8 }
```

B- La fonction get()

```
1  #include <iostream>
2  #include <fstream>
3
4  using namespace std;
5
6  int main()
7  {
8          ifstream fichier("test.txt", ios::in);  // on ouvre
9
10         if(fichier)
11         {
12                 char caractere;  // notre variable où sera stocké le caractère
13                 fichier.get(caractere);  // on lit un caractère et on le stocke
14                 cout << caractere;  // on l'affiche
15
16                 fichier.close();
17         }
18         else
19                 cerr << "Impossible d'ouvrir le fichier !" << endl;
20
21         return 0;
22 }
```

Déclaration d'un caractère

C- Utiliser l'operateur « >> »

```cpp
#include <iostream>
#include <fstream>
using namespace std;
int main()
{

    ifstream fichier("fichier.txt");
if(fichier){

        int a=0;
        fichier>>a;

        cout<<endl<<"a="<<a<<endl;
        char chaine[100];
        fichier>>chaine;
        cout<<endl<<"chaine="<<chaine<<endl;
        }else{
cout<<"erreur de lecture"<<endl;
}
return 0;
}
```

Contenu Du fichier : fichier.txt

```
123 Bonjour
Bonjour
Bonjour
Bonjour

```

Résultat:

```
a=123

chaine=Bonjour
```

D- Détection de fin de fichier: eof()

Syntaxe:

int fichier.feof() ;

- **Consulte un « identificateur de fin de fichier » sur lequel agissent les différentes fonctions de manipulation de fichier.**

- **Retourne:**

  - Une valeur égale à zéro si la fin de fichier (EOF) n'a pas été détectée.
  - Une valeur différente de zéro sinon.

23

- **Exemple:**

Pour lire dans un fichier un texte caractère par caractère il suffit d'utiliser la boucle suivante:

```
Char c;
while(fichier.eof()==0)
        { fichier.get(c);
          cout<<c;
        }
```

Ecrire dans un fichier

Ecrire dans un fichier

Pour l'écriture, nous allons voir deux méthodes :

- **flux <<élémentQuelconque : écrit dans le fichier un élément quelconque (string, int, ...) .**

- **flux.put(caractère) : écrit un seul caractère dans le fichier.**

A- Utiliser l'operateur « << »

```cpp
#include <iostream>
#include <string>
#include <fstream>

using namespace std;

int main()
{
    ofstream fichier("test.txt", ios::out | ios::trunc);  // ouverture en écritu

    if(fichier)
    {
        string nom = "Xav57";
        int age = 19;
        fichier << "Date de naissance : " << 24 << '/' << 3 << '/' << 1988 <
        fichier << "Bonjour, " << nom << ". Vous avez " << age << " ans.";

        fichier.close();
    }
    else
        cerr << "Impossible d'ouvrir le fichier !" << endl;

    return 0;
}
```

B- La fonction put()

```
1 if(fichier)
2 {
3         char car = 'S';
4         fichier.put(car);
5
6
7 }
```

Positionnement Du curseur

A- Connaitre la position du curseur

Il existe une fonction permettant de savoir à quel octet du fichier on se trouve. Autrement dit, elle permet de savoir à quel caractère du fichier on se situe.

- Pour ifstream : tellg()
- Pour ofstream : tellp()

Elles s'utilisent toutes les deux de la même manière.

B- La fonction tellp()

```
6
7       ofstream fichier("fichier.txt",ios::app);
8   if(fichier){
9         int position=-1;
10        position=fichier.tellp();
11        cout<<"Je suis a la "<<position<<" eme position"<<endl;
12  }else{
13  cout<<"erreur de lecture"<<endl;
14  }
15
```

Le Résultat est donc le suivant

```
Je suis a la 0 eme position
```

C- Se déplacer

Là encore, il existe deux fonctions, une pour chaque type de flux.

- Pour ifstream : seekg()
- Pour ofstream : seekp()

Elles s'utilisent de la même manière :

```
flux.seekp(nombreCaracteres, position);
```

C- Se déplacer

Les trois positions possibles sont :

- **le début du fichier : ios::beg**
- **la fin du fichier : ios::end**
- **la position actuelle : ios::cur**

Exemple 1

```
6
7       ofstream fichier("fichier.txt",ios::app);
8   if(fichier){
9           int position=-1;
10          position=fichier.tellp();
11          cout<<endl<<endl<<"Je suis a la "<<position<<" eme position"<<endl;
12          fichier.seekp(4,ios::beg);
13          position=fichier.tellp();
14          cout<<endl<<endl<<"Je suis a la "<<position<<" eme position"<<endl;
15  }else{
16  cout<<"erreur de lecture"<<endl;
17  }
18
```

Exemple 2

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

int main()
{
    ifstream fichier("C:/Nanoc/meilleursScores.txt");  //On ouvre le fichier
    fichier.seekg(0, ios::end);  //On se déplace à la fin du fichier

    int taille;
    taille = fichier.tellg();
    //On récupère la position qui correspond donc a la taille du fichier !

    cout << "Taille du fichier : " << taille << " octets." << endl;

    return 0;
}
```