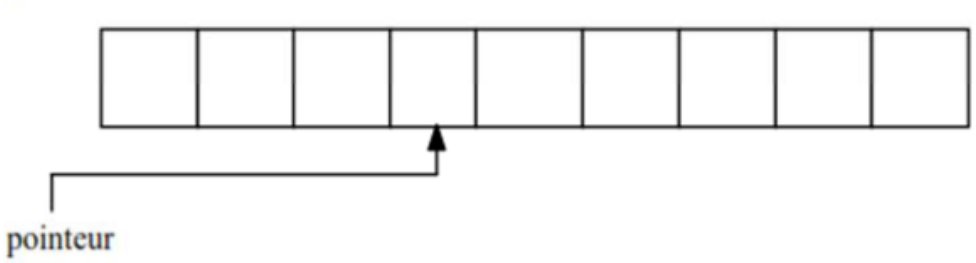
Chapter 6 : *-2nd part-*

The Files

1- Introduction

A file is a collection of information stored on a mass storage device (hard drive, floppy disk, magnetic tape, CD-ROM). This information is saved one after the other and is not necessarily of the same type (a char, an int, a structure...) A pointer allows you to navigate within the file. We access information by moving the pointer to its position.



On the storage medium, the file has a name. This name is composed of two parts: the actual name and the extension. The extension provides information about the type of stored information (provided that the extensions associated with the file type are respected).

1. Introduction to file management in C++

File management is a fundamental aspect of C++ programming. This allows a program to read and write data to files, thereby facilitating the persistence of information between different program executions. In C++, this is accomplished using the fstream, ifstream (for reading), and ofstream (for writing) libraries.

1.1 Types of flow

- **Ofstream(Output file stream):** This stream is used for writing to a file.

- **Ifstream(Input file stream)** : This stream is used for reading from a file.

Context:

- **Fstream(file stream):** This stream can be used for both reading and writing to a file.

2. Opening and Closing Files

To work with files in C++, it is necessary to open them before being able to perform operations on them (reading or writing). When you're done, you should always close the files to free up resources.

2.1 Open a file

1-

```
ofstream flux("fichier.txt"); // Ouvre ou crée "fichier.txt" en mode écriture
```

2-

Or withe the method open()

```
ofstream flux;

flux.open("fichier.txt");  // Ouvre ou crée "fichier.txt"
```

Example : Direct opening during the declaration

In this example, the file is opened in write mode at the time of the object declaration.

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

int main() {
    ofstream flux("fichier1.txt"); // Ouvre ou crée "fichier1.txt" en mode écriture

    if (flux) {  // Vérifie si le fichier est bien ouvert
        flux << "Bonjour, ceci est écrit avec une ouverture directe." << endl;
        cout << "Données ecrites dans fichier1.txt !" << endl;
    } else {
        cout << "Erreur : Impossible d'ouvrir fichier1.txt !" << endl;
    }

    flux.close(); // Ferme le fichier
       system("PAUSE");
    return 0;
}
```

Explication :

- The file **`fichier1.txt`** is created or opened directly when the stream object is initialized.
- A check is performed to ensure that the opening was successful.
- The data is written to the file, and the file is then closed.

Example 2 : Opening with the method `open()`

In this example, the file is opened after declaration using the method **`open()`**.

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

int main() {
    ofstream flux; // Déclaration de l'objet flux
    flux.open("fichier2.txt"); // Ouverture du fichier avec la méthode open()

    if (flux.is_open()) {  // Vérifie si le fichier est ouvert correctement
        flux << "Ceci est ecrit avec la méthode open()." << endl;
        cout << "Données ecrites dans fichier2.txt !" << endl;
    } else {
        cout << "Erreur : Impossible d'ouvrir fichier2.txt !" << endl;
    }

    flux.close(); // Ferme le fichier


       system("PAUSE");
    return 0;
}
```

3. Writing to a file

3.1 Use of ofstream

To write data to a file, we use the `ofstream` object. Here is a simple example where we write a string to a file.

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

int main() {
    ofstream flux("fichier_sortie.txt");  // Création ou ouverture du fichier en mode
écriture
    if (flux) {
        flux << "Bonjour tout le monde" << endl;  // Écriture dans le fichier
        flux.close();  // Fermeture du fichier
    } else {
        cout << "Erreur d'ouverture du fichier!" << endl;
    }
       system("PAUSE");
    return 0;
}
```

3.2 Management of writing errors

If the write fails, it is important to handle the errors. This can be done by checking if the file is opened correctly, as shown in the example above.

4. Reading from a file

4.1 Use of `ifstream`

To read data from a file, we use the **ifstream** object. For example, we can read a file line by line with the **getline()** function.

Here's an example of a program that reads a file line by line:

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

int main() {
    ifstream flux("fichier_sortie.txt");  // Ouverture du fichier en mode lecture
    if (flux) {
        string ligne;
        while (getline(flux, ligne)) {  // Lecture ligne par ligne
            cout << ligne << endl;  // Affichage de la ligne lue
        }
        flux.close();  // Fermeture du fichier
    } else {
        cout << "Erreur d'ouverture du fichier!" << endl;
    }


       system("PAUSE");
    return 0;
}
```

4.2 Management of reading errors

As with writing, it is necessary to check whether the file is open before reading. In addition, it is recommended to use a loop that checks for the end of the file with getline().

5. File manipulation on different disks

It is possible to specify absolute paths to save or read files in different directories or hard drives. For example, if you want to create a file on the D: drive, you specify the full path:

```
ofstream flux("D:\\fichier_sortie.txt");  // Fichier créé sur le disque D
```

Similarly, reading from a specific file on another disk can be done by specifying its absolute path:

```
ifstream flux2("D:\\ fichier_sortie.txt");  // Lecture d'un fichier depuis le disque D
```

---

6. Practical examples

Exercise 1: Creating and writing to a file

Objective: Create a text file and write data to it.

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

using namespace std;

int main() {
    ofstream flux("fichier.txt");
    if (flux) {
        flux << "Bonjour !" << endl;
        flux.close();
    } else {
        cout << "Erreur" << endl;
    }

    ifstream flux2("fichier.txt");
    if (flux2) {
        string ligne;
        while (getline(flux2, ligne)) {
            cout << ligne << endl;
        }
        flux2.close();
    } else {
        cout << "Erreur" << endl;
    }

       system("PAUSE");
    return 0;
}
```

Exercise 2: Creating a file on another disk

Objective: Create a file and save it to the D: drive

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

using namespace std;

int main() {
    ofstream flux("D:\\fichierL3.txt");
    if (flux) {
        flux << "Hello !" << endl;
        flux.close();
    } else {
        cout << "Erreur" << endl;
    }
       system("PAUSE");
    return 0;
}
```

Exercise 3: Reading a file from a specific disk

Objective: Read a file located on disk D:.

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

using namespace std;

int main() {
    ifstream flux("D:\\fichierL3.txt ");
    if (flux) {
        string ligne;
        while (getline(flux, ligne)) {
            cout << ligne << endl;
        }
        flux.close();
    } else {
        cout << "Erreur" << endl;
    }
       system("PAUSE");

    return 0;
}
```

7. Conclusion :

File manipulation in C++ is essential for creating robust applications that interact with persistent data. Using the ifstream and ofstream classes, we can efficiently read and write to files. These examples demonstrate basic but important operations that every programmer must master to work with files in their C++ programs.