

U n i t a t 1 0

Entrada i sortida per arxiu

Introducció.

10.1 Declaració d'un arxiu.

10.2 Obertura d'un arxiu.

10.3 Tancar un arxiu.

10.4 Funcions amb arxius.

10.5 Escriure i llegir en arxius de text.

10.6 Escriure i llegir en arxius binaris.

Introducció.

En la majoria d'aplicacions de mitjana entitat necessitem que la informació s'emmagatzemi de forma persistent, és a dir, que no s'esborri una vegada acabada l'execució del programa. A més moltes aplicacions treballen amb gran quantitat d'informació, que pot omplir i bloquejar la memòria principal. Aquestes necessitats queden cobertes amb sistemes d'emmagatzematge en el disc dur, amb la utilització d'arxius.

El sistema d'entrada i sortida de C++ proporciona un sistema independent del dispositiu físic en el qual, o des del qual, es realitzen les operacions. Aquest sistema independent és una *abstracció* de dispositiu que rep el nom de *flux* o *corrent* (en anglès *stream*). De fet, només necessitem considerar un arxiu com un *flux* de dades que circulen en un o altre sentit.

Existeixen dos tipus d'arxius segons les dades que conté:

- **Arxius de text:** s'interpreten els caràcters de control. Tradueix en l'entrada les seqüències de retorn de carro/alimentació de línia en caràcters de nova línia i a la sortida fa la traducció a l'inrevés.
- **Arxius binaris:** no s'interpreten els caràcters de control. Emmagatzemen blocs de dades.

Les operacions bàsiques dels arxius són:

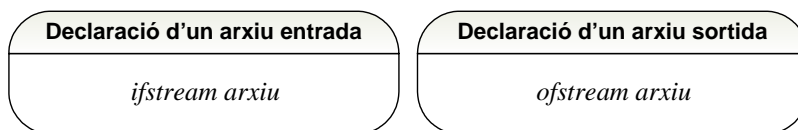
- **Obrir.**
- **Escriure.**
- **Llegir.**
- **Tancar.**

Hi ha una altra classificació dels arxius segons el tipus d'accés:

- **Arxius d'entrada.** Són aquells que només els volem llegir. (Flux d'entrada al programa).
- **Arxius de sortida.** Són aquells que només els volem escriure. Flux de sortida del programa).
- **Arxius de entrada/sortida.**

10.1 Declaració d'un arxiu.

Com qualsevol variable, una variable arxiu ha de ser declarada abans d'utilitzar-la. Així, hi ha dues sentències de declaració, si l'arxiu és per llegir (entrada) o és per escriure (sortida).



Declaració d'una variable d'arxiu anomenada *arxiu*. Serà del tipus entrada si utilitzem *ifstream* i de sortida si utilitzem *ofstream*. Per totes les operacions amb arxius necessitem la biblioteca *fstream.h*, per tant, és necessari incloure la directiva **#include** `<fstream.h>` en qualsevol programa en què s'utilitzin arxius.

10.2 Obertura d'un arxiu.

Declarar una variable d'arxiu no és suficient per poder treballar amb ell. És necessari precisar dues coses:

- D'una banda, és imprescindible assenyalar de quin *arxiu físic es tracta*. És necessari donar un nom a l'arxiu que serà el que restarà al suport magnètic.
- D'altra banda, farà falta precisar el tipus *de treball* que volem fer: llegir, escriure,...

Aquestes característiques es donen amb l'ajuda de la instrucció **open** que té la següent sintaxi:

Obertura d'un arxiu de text.

```

Obertura d'un arxiu de text entrada

#include <fstream.h>
...
ifstream arxiu;

arxiu.open("dades.txt");           // obre per llegir
...
arxiu.open("dades.txt", ios::in | ios::out); //obre per llegir i escriure
```

Obertura d'un arxiu de text sortida

```
#include <fstream.h>
...
ofstream arxiu;

arxiu.open("dades.txt");           // obre per escriure i esborra
                                   // el contingut si estava creat

arxiu.open("dades.txt", ios::app);  // obre per afegir dades al final

arxiu.open("dades.txt", ios::in | ios::out); // obre per llegir i escriure
```

Obertura d'un arxiu binari.**Obertura d'un arxiu binari entrada**

```
#include <fstream.h>
...
ifstream arxiu;

arxiu.open("dades.txt", ios::binary);           // obre per llegir
...
arxiu.open("dades.txt", ios::binary | ios::in | ios::out); // obre per llegir i escriure
```

Obertura d'un arxiu binari sortida

```
#include <fstream.h>
...
ofstream arxiu;

arxiu.open("dades.txt", ios::binary);           // obre per escriure i esborra
                                                // el contingut si estava creat

arxiu.open("dades.txt", ios::binary | ios::app); // obre per afegir dades al final

arxiu.open("dades.txt", ios::binary | ios::in | ios::out); // obre per llegir i escriure
```

En l'operació d'obrir, el primer paràmetre és el nom de l'arxiu que tindrà en el suport magnètic (amb cometes dobles). Pot ser també una variable que contingui una cadena de caràcters amb el nom de l'arxiu.

Els altres paràmetres són els modes d'obertura de l'arxiu.

10.3 Tancar un arxiu.

Al final del treball sobre l'arxiu, o si més no abans de sortir del programa, és necessari tancar l'arxiu. Aquest tancament té diferents objectius.

- El primer consisteix en alliberar el canal atribuït a l'arxiu i permetre obrir un altre arxiu sense superar el màxim nombre possible d'arxius oberts al mateix temps.
- El segon, i gairebé sempre el més important, és buidar el buffer (zona de memòria) associat a l'arxiu.

Per tancar ho farem amb la instrucció **close()** que té la següent sintaxi:

| Tancar un arxiu | |
|---|--------------------------------|
| <code>#include <fstream.h></code> | |
| <code>...</code> | |
| <code>ifstream arxiu;</code> | |
| <code>...</code> | |
| <code>arxiu.close();</code> | <code>// Tancar l'arxiu</code> |

10.4 Funcions amb arxius.

Abans de començar a llegir o escriure en un arxiu s'ha de verificar que l'operació d'obertura s'ha realitzat amb èxit i el flux de dades pot iniciar-se.

Per aquestes comprovacions i d'altres hi ha les següents funcions:

bad(). Aquesta funció torna un valor *true* si el flux de dades està corrupte.

good(). Aquesta funció torna un valor *true* si NO existeix error en l'escriptura o lectura anterior i *false* en cas contrari.

eof(). Aquesta funció torna un valor *true* si estem al final de l'arxiu i *false* en qualsevol altre cas. Perque aquesta funció ens doni valors correctes s'ha de fer una lectura avançada.

fail(). Aquesta funció torna *true* si hi ha un error en l'operació de flux associada a l'arxiu.

| Comprovar el flux de dades |
|---|
| <code>#include <fstream.h></code> |
| <code>...</code> |
| <code>ifstream arxiu;</code> |
| <code>arxiu.open("noms.txt");</code> |
| |
| <code>if (arxiu.bad())</code> |
| <code>{</code> |
| <code> cout << "Error per obrir l'arxiu";</code> |
| <code>}</code> |
| <code>else</code> |
| <code>{</code> |
| <code> ... // Podem treballar amb l'arxiu</code> |
| <code>}</code> |

| Llegir un arxiu fins al final |
|---|
| <code>#include <fstream.h></code> |
| <code>...</code> |
| <code>ifstream arxiu;</code> |
| <code>arxiu.open("noms.txt");</code> |
| |
| <code>arxiu>>..... // lectura avançada</code> |
| <code>while (!arxiu.eof())</code> |
| <code>{</code> |
| <code> // Llegir l'arxiu</code> |
| <code>}</code> |

10.5 Escriure i llegir en arxius de text.

L'escriptura i lectura en arxius de text es pot realitzar directament amb els operadors << i >> com es fa amb els fluxos estàndards de pantalla *cin* i *cout*.

Escriure en un arxiu de text

```
#include <fstream> // Biblioteca per treballar amb arxius
#include <stdio> // Biblioteca estàndar de C

using namespace std;

int main()
{
    ofstream arxiu;
    arxiu.open("direccionsIP.txt");
    if (arxiu.bad())
    {
        cout << "Error per crear o obrir l'arxiu" << endl;
    }
    else
    {
        // Escriure en l'arxiu
        arxiu << "Clase A" << " " << 8 << " Red " << endl;
        arxiu << "Clase B" << " " << 16 << " Red " << endl;
        arxiu << "Clase C" << " " << 24 << " Red " << endl;
        arxiu.close();
    }
}
```

Llegir d'un arxiu de text

```
#include <fstream> // Biblioteca per treballar amb arxius
#include <stdio> // Biblioteca estàndar de C

using namespace std;

int main()
{
    ifstream arxiu; // declaració de l'arxiu
    int bits;
    char classe[10];
    char tipus[5];

    arxiu.open("direccionsIP.txt");
    if (arxiu.bad())
    {
        cout << "Error per crear o obrir l'arxiu" << endl;
    }
    else
    {
        arxiu >> classe; // Lectura avançada
        while (!arxiu.eof())
        {
            cout << classe << " "; // Lectura de valors en l'arxiu
            arxiu >> bits;
            cout << bits << " ";
            arxiu >> tipus;
            cout << tipus << endl;
            arxiu >> classe;
        }
        arxiu.close();
    }
}
```

Per la lectura de l'arxiu s'ha de tenir en compte la lectura avançada, per després consultar si estem al final de l'arxiu.

10.6 Escriure i llegir en arxius binaris.

També hi ha la possibilitat d'escriure o llegir tot un bloc de dades binàries d'un arxiu. Vectors, estructures, vectors d'estructures i altres dades. Les funcions que fan aquesta feina són ***write()*** i ***read()***. Els arxius binaris contenen la informació tal qual està a la memòria, és a dir, sense convertir-la en text.

Tenen la següent sintaxi:

Escriure dades binàries

```
arxiu.write (&car , num);
```

Llegir dades binàries

```
arxiu.read (&car , num);
```

On *car* pot ser una variable de qualsevol tipus, però s'ha de convertir a tipus *char* (*char **), i *num* és una variable enter. Aquestes funcions fan una escriptura/lectura (a partir de la posició actual del cursor de l'arxiu de la variable *arxiu*) de *num* bytes de l'arxiu. Normalment, per expressar el número de bytes que volem escriure/llegir utilitzarem la funció *sizeof*.

Escriure en un arxiu binari

```
#include <fstream> // Biblioteca per treballar amb arxius
#include <stdio> // Biblioteca estàndar de C

using namespace std;

int main()
{
    int taula[2][2]={1, 2, 3, 4};

    ofstream arxiu;
    arxiu.open("numeros.txt");
    if (arxiu.bad())
    {
        cout << "Error per crear o obrir l'arxiu"<<endl;
    }
    else
    {
        // Escriure en l'arxiu
        arxiu.write((char*)&taula, sizeof(int));
        arxiu.close();
    }
}
```

Llegir d'un arxiu binari

```
#include <fstream> // Biblioteca per treballar amb arxius
#include <stdio> // Biblioteca estàndar de C

using namespace std;

int main()
{
    int taula[2][2];

    ifstream arxiu;
    arxiu.open("numeros.txt");
    if (arxiu.bad())
    {
        cout << "Error per crear o obrir l'arxiu"<<endl;
    }
    else
    {
        // Llegir de l'arxiu
        arxiu.read((char*)&taula, sizeof(taula));
        while(!arxiu.eof())
        {
            arxiu.read((char*)&taula, sizeof(taula));
        }
        arxiu.close();
    }
}
```

Exercicis amb arxius

1. Realitzar un programa per llegir i escriure en un arxiu cadenes de text. Es mostrarà un menú principal amb les opcions d'escriure en un arxiu, llegir d'un arxiu i sortir. La informació que guardarem en l'arxiu serà l'estat de vendes del trimestre. Es a dir, s'ha de guardar el mes de l'any, la facturació en euros que s'ha fet, i el venedor que més ha facturat mitjançant un número de venedor (1,2..). Totes aquestes dades ens les proporcionarà l'usuari. Comprovar que la informació que s'ha guardat en l'arxiu, és la mateixa que la que l'usuari ha donat. *[Solució: 10_1_arxiu_cadenes]*.
2. Realitzar un programa per llegir i escriure amb format diferents variables. Es mostrarà un menú principal amb les opcions de guardar vehicles, llegir vehicles i sortir. Les dades que guardarem seran les característiques d'un cotxe: marca, model, cilindrada i preu.

L'opció de guardar vehicles crearà una base de dades en diferents arxius, per marques.

L'usuari haurà d'indicar el nom més l'extensió "txt" d'aquest arxiu.

L'opció de llegir vehicle, primer ens preguntarà quin arxiu volem recuperar i ens mostrarà per pantalla totes les seves característiques. *[Solució: 10_2_arxiu_cotxe]*.