Objectius de la pràctica:
- Saber obrir, tancar i tractar la informació continguda en fitxers.

Ús de fitxers

Totes les estructures de dades que hem vist en pràctiques anteriors estaven emmagatzemades en la memòria principal. Açò fa que les dades es perden quan acaba el programa. Els fitxers són una estructura especial que fa servir la memòria secundària. Les dades s'emmagatzemaran en memòria no volàtil i, per aquest motiu, no es perdrà el seu contingut quan finalitze el programa.

Per a qualsevol científic, aquest tipus de dades és molt important ja que els fitxers s'empraran per a guardar una gran quantitat de mesures i resultats de proves que, posteriorment, podran sen processades i analitzades per a extraure conclusions. Així, doncs, mentre que en pràctiques anteriors el resultat dels nostres exercicis era un únic fitxer amb el codi font del programa (*.cpp), ara tindrem a més a més un o més fitxers de dades (normalment, *.dat o *.txt).

Tipus d'accés a la informació i fitxers de text

Per a accedir a les dades contingudes dins d'un fitxer, hom pot seguir dues opcions: l'accés seqüencial i l'accés directe.

- Accés seqüencial: L'accés a les dades del fitxer només pot fer-se cap endavant, des del primer element i sempre d'un en un.

- Accés directe: L'accés a les dades es pot fer directament a qualsevol posició del fitxer. És un accés similar a aquell que es fa amb els arrays mitjançant els índexs.

En aquestes pràctiques només treballarem amb l'accés seqüencial. A més a més, ens limitarem a treballar amb fitxers de text, açò és, fitxers que contenen una seqüència de caràcters separats per salts de línia.

Fitxers lògics i fitxers físics

En C++ els fitxers són un tipus de dades més. Llavors, un fitxer concret es referencia a través d'una variable de tipus fitxer. Aquesta variable rep el nom de fitxer lògic. En C++ existeixen dos tipus de dades per a declarar fitxers:

`ifstream` per a declarar fitxers d'entrada (d'on llegirem)
`ofstream` per a declarar fitxers d'eixida (on escriurem)

Exemple de declaració d'una variable de tipus fitxer d'eixida:

```
ofstream f;    // f és una variable de tipus fitxer d'eixida
```

Per a utilitzar aquests tipus caldrà incloure la següent capçalera:

```
#include <fstream>
```

La variable anterior (`f`), perquè siga d'utilitat, haurà d'estar associada a un fitxer "real" (p. ex. dades.txt). Aquest fitxer rep el nom de fitxer físic.

Operacions amb fitxers

Per a relacionar els fitxers lògics amb els físics farem servir la següent operació d'obertura del fitxer:

```
f.open("dades.txt");     // Ara ja podem usar el fitxer
```

Així, doncs, abans de poder fer cap operació amb el fitxer caldrà obrir-lo i, respectivament, en acabant de treballar serà necessari tancar-lo amb la següent instrucció:

```
f.close();     // Tancament del fitxer f
```

Quan s'obri un fitxer, el programa es posiciona sobre el seu primer element. Si el fitxer que s'ha obert és per a llegir dades (`ifstream`), cal que aquest existisca amb anterioritat. Del contrari, l'operació generarà un error, ja que no es pot obrir un fitxer que no existeix per a lectura. Si el fitxer és d'eixida (`ofstream`) es crearà un fitxer nou i, si ja existia, el seu contingut anterior serà esborrat completament.

El mode d'obertura d'un fitxer es pot modificar de la següent manera:

```
f.open("dades.txt", ios:app);
```

La instrucció anterior obri el fitxer i es posiciona després del darrer element per a permetre afegir dades sense esborrar el seu contingut previ. Si el fitxer no existeix, quan es tracta d'obrir, generarà un error.

Per a controlar els errors que potencialment es poden produir quan tractem d'obrir un fitxer caldrà emprar estructures condicionals com ara:

```
if (!f)
    cout << "Error al tractar d'obrir el fitxer " << endl;
```

Quan no hi haja cap errada (else) es podrà continuar amb la resta del codi del programa.

Nota: Si volem obrir un fitxer el nom del qual està emmagatzemat en una variable de tipus string, haurem d'emprar la següent instrucció;

```
string nom = "dades.txt";
f.open(nom.c_str());
```

Operacions d'entrada i d'exida de dades

Són exactament les mateixes que es poden fer amb l'entrada estàndard (`cin`) i l'eixida estàndard (`cout`). Açò és, per a guardar el contingut d'una variable (de qualsevol tipus simple) en un fitxer farem:

```
f << x;
```

D'altra banda, per a llegir alguna dada des d'un fitxer i emmagatzemar-la en una variable farem:

```
f >> x;
```

Exemple: Programa que escriu els nombres de l'1 al 10 en el fitxer "dades.txt" (fent servir un `while` en comptes d'un `for`, que és com ho hem vist a classe de teoria).

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream f;         // Fitxer d'eixida
    int i;

    f.open("dades.txt");
    if (!f)
        cout << "Error obrint fitxer" << endl;
    else
    {
        i = 1;
        while (i <= 10)
        {
            f << i << endl;  // Escriu el contingut de i i bota de línia
            i++;
        }
        f.close();
    }

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

Exemple: Programa que llig els 10 nombres enters i els mostra per pantalla (com succeia abans, també es podria fer amb un `while` en comptes de fer-ho amb un `for`).

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream f;         // Fitxer d'entrada
    int i, dada;

    f.open("dades.txt");
    if (!f)
        cout << "Error obrint fitxer" << endl;
    else
    {
        for (i = 0; i <= 10; i++)
        {
            f >> dada;
            cout << dada << endl;
        }
        f.close();
    }

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

Tanmateix, allò més habitual és que no coneguem quantes dades anem a llegir i que, per tant, vulguem llegir fins que arribem al final del fitxer. En aquest cas caldrà usar un bucle `while` com el que es mostra a continuació:

```
int main()
{
    ifstream f;
    int dada;

    f.open(“dades.txt”);
    if (!f)
        cout << “Error obrint fitxer” << endl;
    else
    {
        while (f >> dada)
            cout << dada << endl;
        f.close();
    }

    system(“pause”);
    return 0;
}
```

Quan vulguem llegir el fitxer caràcter a caràcter, allò més habitual serà emprar el mètode **get()**. Tanmateix, aquest mètode no retorna cert o fals per a saber si s'ha pogut llegir amb èxit (com succeeix amb l'operador >>), ja que ha de retornar el caràcter que ha llegir. En conseqüència, la manera de llegir un fitxer amb get() serà com es mostra a continuació:

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream f;         // Fitxer d'entrada
    int i, dada;

    f.open(“dades.txt”);
    if (!f)
        cout << “Error obrint fitxer” << endl;
    else
    {
        dada = f.get();
        while (!f.eof())
        {
            cout << dada << endl;
            dada = f.get();
        }
        f.close();
    }

    system(“pause”);
    return 0;
}
```

El **mètode eof()** retorna cert si la dada que hem llegit és el final del fitxer i fals en cas contrari. Per aquesta raó cal llegir primer el caràcter i després comprovar si hem arribat al final del fitxer.

Lectura d'estructures

Quan volem llegir un conjunt homogeni de dades simples des d'un fitxer, podem col·locar la instrucció de lectura dins de la condició d'un bucle `while`. Tanmateix, per a llegir una estructura de dades heterogènia hem de llegir primer cadascun dels camps abans de considerar que la lectura s'ha efectuat correctament. Per tant, per a poder mantindre la lectura com a una instrucció dins de la condició del `while`, caldrà definir una funció que faça aquesta lectura i retorne **true** (si s'ha pogut llegir sense errades) o **false** (en el cas contrari).

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
using namespace std;

struct Telefon
{
    string nom;
    string telefon;
};

bool llegirTel(ifstream & f, Telefon & tel); // Prototip

int main()
{
    ifstream guia;
    Telefon tel;

    guia.open("guia.dat");
    if (!guia)
        cout << "Error obrint fitxer" << endl;
    else
    {
        while (llegirTel(guia, tel))
        {
            cout << tel.nom << endl;
            cout << tel.telefon << endl << endl;
        }
        guia.close();
    }

    system("pause");
    return 0;
}

/*********************************************************************/
/* llegir Tel    Llig una estructura composta per un nom i un telèfon */
/*                                                                   */
/* f        fitxer del qual vull llegir                              */
/* tel      estructura on vaig a col·locar la informació llegida     */
/*********************************************************************/
bool llegirTel(ifstream & f, Telefon & tel)
{
    getline(f, tel.nom);
    getline(f, tel.telefon);
    return (!f.eof());
}
```

Convé observar que els fitxers s'han de passar sempre com paràmetres per referència, tant si els anem a modificar com si no. A més a més, en l'exemple anterior hem usat la funció **getline**, que

permet llegir una línia completa des del fitxer. És evident que, per a que aquest codi funcione, el fitxer de dades ha de tindre cada dada en una línia distinta, açò és, noms i telèfons entrellaçats.

Nota important per a entendre l'exemple anterior: el fitxer del quan llegim haurà de tindre un darrer salt de línia després de l'última dada. En cas contrari, no es llegirà l'última estructura.

Resum

1. No oblideu afegir la llibreria: `#include <fstream>`

2. Només practicarem amb fitxers seqüencials de tipus text:
   - Fitxers que guarden únicament caràcters.
   - L'accés a la informació serà seqüencial, açò és, per a arribar a un cert lloc del fitxer cal passar (llegir) per totes les dades anteriors.

3. Els fitxers es poden obrir per a llegir el seu contingut o per a escriure en ells. Mai llegirem i escriurem simultàniament (nota: el fitxer és sols el nom de la variable):
   - Declaració d'una variable fitxer d'escriptura: `ofstream fitxer;`
   - Declaració d'una variable fitxer de lectura: `ifstream fitxer;`
   - Obertura del fitxer: `fitxer.open("nom_fitxer.extensió");`
   - Tancament del fitxer: `fitxer.close();`
   - Nota: Per defecte, Dev C++ deixa el fitxer en el mateix directori on estiga el programa. Si volem indicar una altra ruta o directori, haurem d'especificar-la:
     ```
     fitxer.open("C:\\tmp\\hola.txt");   // Noteu la doble \
     ```

4. Escriptura i lectura. La lectura i l'escriptura de la informació es duu a terme a través dels operadors << i >> (com en el cas de cin i cout):
   - `fitxer << dada;  // Escriptura en fitxer`
   - `fitxer >> dada;  // Lectura des del fitxer`

5. Per a llegir totes les dades d'un fitxer fins al final usarem la sentència:
   - ```
     // La condició s'avaluarà a fals quan arribe al final
     while (fitxer >> dada)
     ```

6. Estructures. No es pot llegir ni emmagatzemar una estructura directament en un fitxer sinó que cal procedir per separat amb cadascun dels camps. Per exemple:
   - ```
     struct Complex {
     {
         int real;
         int imag;
     }
     Complex c;
     fitxer << c.real << " " << c.imag;   // Correcte
     fitxer << c;                         // Incorrecte!
     ```