Tema 5
Arxius o fitxers

Informàtica
Grau en Física
Universitat de València

Ariadna.Fuertes@uv.es
Francisco.Grimaldo@uv.es

Guió del tema

- Introducció:
  - Concepte de fitxer.
  - Tipus d'accés a fitxers.
  - Tipus de fitxers: tipus text i tipus binari.
  - Fitxers lògics i fitxers físics.
- Operacions sobre fitxers:
  - Obertura del fitxer.
  - Tancament del fitxer.
  - Escriptura en fitxer.
  - Lectura de fitxer.
  - Altres instruccions.
- Processament d'un fitxer.
- Operacions de lectura i d'escriptura en fitxers de text:
  - Escriptura mitjançant <<.
  - Lectura mitjançant >>.
  - Lectura mitjançant `get()`.
  - Lectura d'estructures.
  - Pas de fitxers com a paràmetres de funcions.
- Operacions de lectura i d'escriptura en fitxers binaris.
  - Mètode d'accés directe.
  - Lectura i escriptura de fitxers binaris.

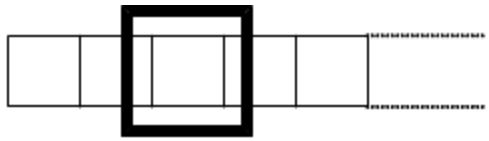
Introducció (1/6)

- **Concepte de fitxer.**
- Totes les estructures de dades que hem vist fins ara desen la informació a la memòria principal.
- Açò comporta dues limitacions importants:
  - 1. Les dades desapareixen quan el programa acaba.
  - 2. La quantitat de dades no pot ser molt gran a causa de la grandària reduïda de la memòria principal.
- Afortunadament, hi ha estructures especials que fan servir la memòria secundària: els fitxers.
- Un fitxer és, a més a més, una estructura dinàmica, ja que la grandària en pot variar al llarg de l'execució del programa depenent de la quantitat de dades que continga.

Introducció (2/6)

- **Tipus d'accés a fitxer**

- Com que són a la memòria secundària, no tots els elements del fitxer són accessibles de manera immediata. Solament podrem accedir a un únic element del fitxer, que anomenarem *finestra del fitx*er.



- Depenent de com es desplace la finestra pel fitxer, distingim dos tipus d'accés:
  - Accés seqüencial: La finestra del fitxer solament es pot moure cap endavant, a partir del primer element i sempre d'un en un.
  - Accés directe: La finestra del fitxer es pot situar directament en qualsevol posició del fitxer. És un accés similar al que es fa en els arrays.

Introducció (3/6)

- L'accés directe sovint és més eficient, perquè per a llegir una dada no cal llegir abans totes les anteriors.

- Tanmateix, l'accés seqüencial és a vegades necessari ja que hi ha dispositius de memòria secundària que solament admeten l'accés seqüencial (com ara les cintes). A més a més, l'accés seqüencial s'usa en dispositius que sí que admeten l'accés directe quan volem llegir els elements d'una manera seqüencial, ja que aquest accés és més senzill.

- Per a treballar amb fitxers (memòria secundària) caldrà un conjunt d'operacions bàsiques:

  - per a poder escriure la informació en el fitxer
  - i per a poder recuperar-la i posar-la en la memòria (llegir),
  - també caldran funcions per a obrir i tancar fitxers.

Introducció (4/6)

- **Tipus de fitxers.**
- Hi ha dos tipus principals de fitxers (segons com desem la informació en l'interior):
- Fitxers de tipus text.
- Emmagatzemen la informació en forma de seqüència de caràcters (açò és, com es mostren per pantalla). Per exemple, el valor enter '25' es transforma en els caràcters '2' i '5'.
- Per tant, les operacions d'escriptura i de lectura de fitxers d'aquest tipus seran similars a les que emprem per a escriure en pantalla i llegir del teclat.
- Fitxers de tipus binari.
- Contenen seqüències d'elements d'un tipus de dades determinat. És a dir, la informació es desa en el fitxer com es troba a la memòria principal: un patró de zeros i uns. Per exemple, el valor enter '25' es desarà com '00011001'.
- Un fitxer binari és, per tant, similar a un vector. Per a accedir-ne a les dades, però, farem servir unes funcions específiques que seran diferents de les dels fitxers de text.

Introducció (5/6)

- **Fitxers lògics i fitxers físics.**
- En un llenguatge de programació, els fitxers són un tipus de dades més. Un fitxer concret es referencia mitjançant una variable de tipus fitxer. Açò és el que anomenem fitxer lògic.
- En C++ hi ha dos tipus de dades bàsics per a declarar fitxers:
  - ifstream // Per a declarar fitxers d'entrada (in) (des d'on llegir)
  - ofstream // Per a declarar fitxers d'eixida (out) (on escriure)
- Per a poder usar aquests tipus, cal incloure abans el fitxer de capçalera <fstream>

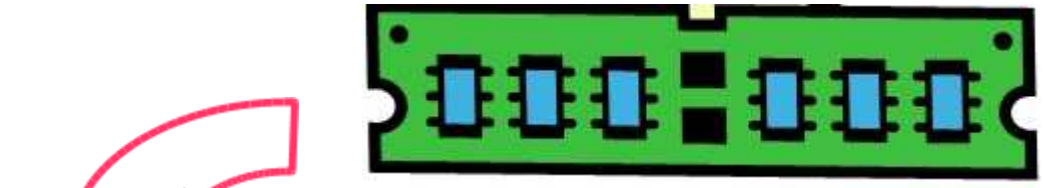
- **Exemple** de sentència que declara una variable (fitxer lògic) de tipus fitxer d'eixida:

```
ofstream f;
```

- Perquè aquesta variable siga útil ha d'estar associada amb un fitxer "real", açò és, que siga reconegut pel sistema operatiu (p. ex. dades.txt), ja que serà finalment el sistema operatiu l'encarregat de fer l'escriptura o la lectura des del fitxer. A aquest fitxer l'anomenem fitxer físic.
- Per a relacionar el fitxer lògic amb el fitxer físic cal realitzar una operació d'*obertura del fitxer*.

Introducció (6/6)

- **Resum**



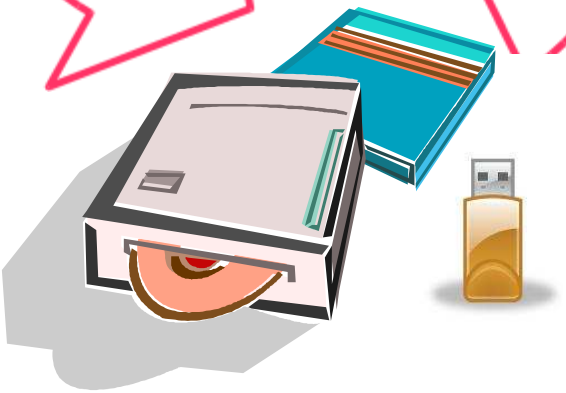
cout

cin

ifstream

Fitxer lògic

ofstream



Fitxer físic

Operacions sobre fitxers (1/5)

- **Obertura del fitxer:**
- *Físicament*, un fitxer és un conjunt de bits desats en un suport determinat (generalment magnètic) al qual s'accedeix mitjançant una referència al suport (unitat) i una localització sobre el suport (p. ex.: Sector, cilindre, cara,... en el cas del suport magnètic de disc.)
- La *definició lògica* del fitxer és un conjunt d'informació útil per a l'usuari, desada amb un nom (al qual podem accedir si travessem un cert camí o 'path').
- La manera de relacionar ambdós conceptes és mitjançant el sistema operatiu i en els programes realitzats pels usuaris a través de l'operació d'obertura del fitxer.
- L'operació d'obertura d'un fitxer utilitza la informació que coneix l'usuari per a relacionar-la amb la descripció física i preparar un descriptor que serà usat pel programa per a accedir a la informació continguda dins del fitxer.
- En C++ aquesta operació es fa amb la instrucció open:
  nom_fitxer_logic.open (nom_fitxer_fisic);

Operacions sobre fitxers (2/5)

- Exemple:
  f.open(“dades.txt”);

- D’aquest moment endavant podem accedir al fitxer.
- Cal adonar-se que, per defecte, els fitxers d’entrada (ifstream) s’obrin només per a lectura i situen la finestra del fitxer sobre el primer element. A més a més, el fitxer ha d’existir, si no, es produïrà una errada.
- Al contrari, els fitxers d’eixida (ofstream) s’obrin per defecte només per a escriptura i creen un fitxer nou quan no existeix o esborren el que existia prèviament.
- Per a modificar el mode d’obertura, hom pot afegir un paràmetre més a la instrucció open, tot i indicant-hi el mode d’obertura. Només veurem el mode ios::app que serveix per a obrir un fitxer d’eixida en mode afegir. Llavors, l’obertura no esborra el fitxer anterior, sinó que la finestra se situa després del darrer element. Si el fitxer no existeix, dóna un error.
- nom_fitxer_logic.open (nom_fitxer_fisic, ios::app);

Operacions sobre fitxers (3/5)

- Exemple:
  f.open(“dades.txt” , ios::app);

- Per saber si l’obertura d’un fitxer ha donat algun error, s’aplica l’operador ! sobre el fitxer:

- Exemple:
  if (!f)
  cout << “Error quan s’ha obert el fitxer” << endl;

- Cal comprovar aquesta condició sempre que obrim un fitxer ja que, si s’ha produït un error, no podrem fer-hi cap operació.

Operacions sobre fitxers (4/5)

- **Tancament del fitxer:**
- Quan s'ha acabat de fer totes les operacions, SEMPRE, cal tancar el fitxer. Aquesta operació destrueix les estructures que s'han creat per a obrir-lo (tant dins del programa com del sistema operatiu). També actualitza totalment el fitxer i escriu tota la informació que poguera estar encara al *buffer* (ja que sovint la informació passa a través d'un *buffer*).

- Per a tancar el fitxer, s'empra la instrucció close:
  nom_fitxer_logic.close ();

- Exemple:
  f.close();

Operacions sobre fitxers (5/5)

- **Escriptura en fitxer:**
- Les funcions d'escriptura en fitxer s'empren per a desar informació que es troba a la memòria de l'ordinador en un fitxer i poder utilitzar-la amb posterioritat. Com escriure en un fitxer dependrà del tipus de fitxer en què volem emmagatzemar la informació: text o binari.
- Més endavant, tractarem ambdós casos.
- **Lectura des de fitxer:**
- Les funcions de lectura des de fitxers s'usen per a recuperar informació emmagatzemada en un fitxer i posar-la en la memòria per a poder treballar-hi directament.
- Segons el tipus de fitxer (text o binari), les funcions d'escriptura també seran unes o unes altres.
- En tot cas, per a poder llegir informació des d'un fitxer, cal conèixer exactament com ha estat desada la informació a l'interior d'aquest.
- **Altres instruccions:**
- El caràcter EOF és el caràcter de final de fitxer. Per a saber si s'ha arribat a llegir aquest caràcter, hi ha la funció eof() que detecta si s'ha arribat al final del fitxer.
- eof () torna vertader si la finestra es troba al final del fitxer i, fals en qualsevol altre cas.

Processament d'un fitxer

- Sempre que treballem amb fitxers, la primera tasca que haurem de realitzar serà obrir el fitxer i relacionar-lo amb un descriptor, el qual serà, des d'aquest moment endavant, la referència que emprarem per a treballar amb el fitxer.

- Llavors, ja podrem realitzar qualsevol operació amb el fitxer i, en acabant, el tancarem.

- En resum, per a treballar amb fitxers, hem de seguir l'esquema següent:

Obertura del fitxer → Operacions de lectura o escriptura → Tancament del fitxer

Operacions amb fitxers de text (1/15)

- Les operacions que podem realitzar sobre un fitxer de text són exactament les mateixes que podem realitzar sobre cin (per a fitxers d'entrada) i cout (per a fitxers d'eixida). De fet, cin i cout són dos fitxers predefinits que s'associen amb l'entrada i l'eixida estàndar del sistema operatiu, les quals corresponen sovint amb el teclat i la pantalla, respectivament.

- Exemple: Per a escriure el nombre 10 en el fitxer f:

```
f << 10;
```

Per a escriure un string:

```
f << "Hola";
```

- Escriptura mitjançant <<
- S'usa per a escriure el contingut d'una variable x (de qualsevol tipus simple) en un fitxer:

```
f << x;
```

- Lectura mitjançant >>
- Per a llegir des d'un fitxer i emmagatzemar el que s'haja llegit dins de x:

```
f >> x;
```

Operacions amb fitxers de text (2/15)

- **Exemple:** Programa per a escriure els nombres des de l'1 al 10 en el fitxer dades.txt

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

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

    f.open("dades.txt");  // Obertura del fitxer
    if ( !f )
        cout << "Error obrint el fitxer" << endl;
    else        // Operacions sobre el fitxer
    {
        for(i = 1; i <= 10; i++)
            f << i << endl;     //escriu el contingut de i i salta a la línia següent
        f.close();  // Tancament del fitxer
    }
    return 0;
```

Operacions amb fitxers de text (3/15)

- **Exemple**: Programa per a llegir els 10 primers nombres enters del fitxer i mostrar-los per pantalla.

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

int main()
{
    ifstream f;
    int i, dada;

    f.open("dades.txt");
    if ( !f )
        cout << "Error obrint el fitxer" << endl;
    else
    {
        for(i = 1; i <= 10; i++)
        {
            f >> dada;
            cout << dada << endl;
        }
        f.close();
    }
    return 0;
}
```

Operacions amb fitxers de text (4/15)

- Tanmateix, normalment no sabem quants elements llegirem, sinó que volem llegir fins que arribem al final del fitxer. Llavors, podem fer servir un bucle while de l'estil següent:

```
while (f >> dada)
    cout << dada << endl;
```

- Quan una instrucció per a llegir des d'un fitxer acaba amb èxit, torna cert i quan es produeix algun error (entre els que podem incloure l'arribada al final del fitxer), torna fals. Per tant, la instrucció anterior llegirà, mentre siga possible, tots els nombres del fitxer.

- Aquesta manera de llegir des d'un fitxer es pot emprar amb qualsevol tipus de lectura. Açò és, amb l'operador >> i amb 'getline' si volem llegir fins al final de línia:

```
getline (variable_ifstream , variable_tipus_string );

while ( getline ( f , dadaTipusString)
    cout << dadaTipusString << endl;
```

- Veiem ara el programa anterior modificat perquè incloga el bucle while que acabem d'introduir:

Operacions amb fitxers de text (5/15)

- **Exemple:** Programa per a llegir els nombres enters d'un fitxer i mostrar-los per pantalla.

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

int main()
{
    ifstream f;
    int dada;

    f.open("dades.txt");
    if ( !f )
        cout << "Error obrint el fitxer" << endl;
    else
    {
        while(f >> dada)
            cout << dada << endl;

      f.close();
    }
    return 0;
}
```

Operacions amb fitxers de text (6/15)

- Lectura mitjançant get()

- Si volem llegir el fitxer caràcter a caràcter, allò més habitual serà emprar el mètode get( ). Tanmateix, aquesta funció no torna cert o fals per a saber si s'ha pogut llegir amb èxit, ja que ha de tornar el caràcter que ha llegit.

- La manera de llegir un fitxer amb get() serà, per tant, lleugerament diferent de la que hem vist abans. Caldrà usar el mètode *eof().*

- El mètode eof() és cert si la dada que hem llegit era un final de fitxer i fals en cas contrari.

- Per aquesta raó, cal llegir primer el caràcter i després comprovar si hem arribat al final del fitxer.

Operacions amb fitxers de text (7/15)

- **Exemple**: Programa per a llegir els caràcters d'un fitxer i mostrar-los per pantalla.

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

int main()
{
    ifstream f;
    char dada;

    f.open("dades.txt");
    if ( !f )
        cout << "Error obrint el fitxer" << endl;
    else
    {
        dada = f.get();
        while( ! f.eof() )
        {
            cout << dada << endl;
            dada = f.get();
        }
        f.close();
    }
    return 0;
}
```

Operacions amb fitxers de text (8/15)

- Lectura d'estructures

- La lectura de dades simples des d'un fitxer és dur a terme d'una manera senzilla si introduïm la lectura dins de la condició d'un bucle while.

- Nogensmenys, per a llegir una estructura primer, cal llegir-ne cadascun dels camps abans de considerar-ne correcta la lectura. Per tant, per a fer la lectura dins de la condició del while serà necessari definir una funció que s'encarregue de la lectura i torne 'true' si s'ha pogut realitzar sense errades o, 'false' en cas contrari.

- No obstant això, si el fitxer ha estat escrit per la nostra aplicació i hem comprovat que l'estructura és correcta abans d'escriure'l, podem assumir que "si existeix el primer camp de l'estructura, també existiran els següents", amb la qual cosa podem posar el primer camp dins del bucle while i no usar el mètode eof().

- Veiem primer com seria la lectura sense usar funcions i en l'apartat següent modificarem el programa perquè incloga funcions, amb la qual cosa fem una implementació més correcta.

Operacions amb fitxers de text (9/15)

- Exemple amb mètode eof() i sense funcions:

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

struct Telefon
{
    string nom;
    int telefon;
};

int main(void)
{
    Telefon tel;
    ifstream f;

    f.open("guia.dat");
    if( !f )
        cout << "Error obrint el fitxer" << endl;
    else
    {
                getline(f, tel.nom); //Llig del fitxer una línia de text
                f >> tel.telefon; //Llig l'enter que correspon al núm.
                f.ignore(); //Ignora un caràcter del buffer que és el '\n'

                while( !f.eof() )
                {
                        cout << tel.nom << endl << tel.telefon << endl;

                        getline(f, tel.nom); //Llig del fitxer
                        f >> tel.telefon; //Llig l'enter
                        f.ignore(); //Ignorar el '\n'
                }
                f.close();
        }
        return 0;
}
```

- Exemple sabent que l'escriptura en fitxer ha estat correcta però sense funcions:

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

struct Telèfon
{
    string nom;
    int telèfon;
};

int main(void)
{
    Telefon tel;
    ifstream f;

    f.open("guia.dat");
    if( !f )
        cout << "Error obrint el fitxer" << endl;
    else
    {
                //Llig des del fitxer una línia de text (si existeix)
                while( getline(f, tel.nom) )
                {
                        //Si entra en el bucle, hi ha un registre complet
                         //Llig l'enter que correspon amb el número.
                        f >> tel.telfon;
                        f.ignore(); //Ignorar el '\n'
                        cout << tel.nom << endl << tel.telefon << endl;
                }
                f.close();
        }
        return 0;
}
```

Operacions amb fitxers de text (10/15)

- Pas de fitxers com a paràmetres de funcions.

- Els fitxers s'han de passar sempre com a paràmtres per referència, no importa si els modificarem o no.

- Convé destacar que hem emprat la funció *getline*, que ens permet llegir una línia completa del fitxer. És evident que perquè el codi anterior funcione, el fitxer de dades ha d'incloure cada dada en una línia diferent (en una línia el nom, en la següent, el telèfon associat i així successivament).

- Si modifiquem el programa anterior i hi incloem una funció F_IntroTel que s'encarregarà de llegir la informació del fitxer i tornar un booleà que diga si hem arribat al final del fitxer o no,...

Operacions amb fitxers de text (11/15)

- Exemple amb mètode eof() i funcions:

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

struct Telefon
{
    string nom;
    int telefon;
};

//Prototips de les funcions que cal usar
void EscriuTel (Telefon  ) ;
bool F_IntroTel (ifstream & , Telefon &  ) ;

int main(void)
{
    Telefon tel;
    ifstream guia;

    guia.open("guia.dat");
    if(!guia)
        cout << "Error obrint el fitxer" << endl;
    else
    {
        while(F_IntroTel(guia, tel))
        {
            EscriuTel(tel);
            cout << endl;
        }
        guia.close();
    }
    return 0;
```

```
void EscriuTel (Telefon tel)
{
      cout << tel.nom << endl << tel.telefon ;
        cout << endl << endl;
}

bool F_IntroTel(ifstream & f, Telefon & tel)
{
        //Llig des del fitxer una línia de text
      getline( f , tel.nom);

      //Llig l'enter que correspon amb el número
        f >> tel.telefon;

        //Ignora un caràcter del buffer que és el '\n'
      f.ignore();

        return ( !f.eof() );
}
```

Operacions amb fitxers de text (12/15)

- Exemple de programa que llig des d'un fitxer:
- Tenim un fitxer on s'emmagatzema informació sobre components mecànics d'automòbils. La fitxa de cada component mecànic té la informació següent:

- el nom de la peça (poden ser moltes paraules),
- unitats disponibles (és un nombre enter),
- preu de la peça (pot tenir decimals).

L'estructura del fitxer ("autos.txt") és la següent:

- En la primera línia del fitxer desem totes les peces que hi ha en el fitxer.
- En la línia següent, el nom de la primer peça i després
- en l'altra línia trobarem les unitats disponibles i el preu d'aquestes.
- Exemple:
  - 3
  - Corretja de distribució ford mondeo 1.4
  - 24 123.85
  - Disc de fre audi a3
  - 12 12.4
  - Disc de fre opel corsa
  - 30 3.70

- Feu un programa que llija la informació del fitxer, la carregue en memòria i la mostre a continuació.
- El programa podrà emmagatzemar fins a un màxim de 500 peces. (El fitxer no tindrà mai més d'aquesta quantitat i estarà escrit correctament.)

Operacions amb fitxers de text (13/15)

- Solució:

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

#define TAM 500 //nombre màxim de peces
struct peca
{
        string nom;
        int unitats;
        float preu;
};
void LlegirFitxer (ifstream &f,peca vector[TAM] , int & num );
void MostrarMemoria  (peca vector[TAM] , int  num);

void LeerFichero  (ifstream &f, peca vector[TAM] , int & num )
{
        int i;
        f >> num;
        f.ignore();
        for ( i=0 ; i < num ; i++)
        {
                getline( f , vector[i].nom);
                f >> vector[i].unitats  >> vector[i].preu ;
                f.ignore();
        }
}
void MostrarMemoria  (peca vector[TAM] , int  num)
{
        int i;
        for ( i=0 ; i < num ; i++)
        {
                cout << vector[i].nom << endl;
                cout << vector[i].unitats <<“ ” << vector[i].preu ;
}
```

```
int main ( void )
{
   peca llista[TAM];
   int quants; //nombre de peces que tenim
   string nom;
   ifstream fitx;

   cout << “Programa que llig el fitxer de peces.\n” ;
   cout << “Dona’m el nom del fitxer \n“ ;
   cin >> nom;

   fitx.open( nom.c_str());

  if( !fitx )
        cout << “Error obrint el fitxer\n”;
   else
   {
      LlegirFitxer( fitx, llista , quants );
      fitx.close();

      MostrarMemoria (llista , quants );
   }

   system(“pause”;
   return 0;
}
```

Operacions amb fitxers de text (14/15)

- Exemple de programa que llig de fitxer i escriu en fitxer:

- Anem a fer una nova versió del programa anterior en què afegirem una altra funció que permeta recórrer tota la informació que hi ha a la memòria i la dese en el fitxer, és a dir, que sobreescriga el fitxer anterior.

- Consideracions:

- Cal tenir en compte que hem d'escriure el fitxer d'acord amb el format que haviem decidir anteriorment. Si no, després no podrem llegir-lo correctament. (Si ens fixem en la funció MostrarMemoria, la funció per escriure és tan senzilla com cambiar cin pel nombre de la variable d'accés al fitxer d'eixida).

- Quan volem tant llegir com escriure sobre un mateix fixter, per evitar problemes amb els temps d'accés, és recomanable obrir el fitxer dins de la pròpia funció de lectura/escriptura i que siguen elles mateixes les que tornen si han estat capaces d'obrir-lo o no.

- Si refem el programa anterior, atenent tot açò...

Operacions amb fitxers de text (15/15)

- Solució:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
#define TAM 500 //nombre màxim de peces
struct peca
{
        string nom;
        int unitats;
        float preu;
};
bool LlegirFitxer ( string , peca [TAM] , int &  );
bool EscriuFitxer  ( string , peca [TAM] , int  );
bool LlegirFitxer (  string nom, peca vector[TAM] , int & num )
{
        int i;
        bool error = false;
        ifstream f;

        f.open( nom.c_str());
        if( ! f )
                error = true ;
        else
        {
                f >> num;
                f.ignore();
                for ( i=0 ; i < num ; i++)
                {
                        getline( f , vector[i].nom);
                        f >> vector[i].unitats  >> vector[i].preu ;
                        f.ignore();
                }
                fich.close();
        }
        return error;
}
```

```
bool EscriuFitxer(string nom, peca vector[TAM], int num)
{
        int i;
        bool error = false;
        ofstream g;

        g.open( nom.c_str());
        if( ! g )
                error = true ;
        else
        {
                g << num << endl;
                for ( i=0 ; i < num ; i++)
                {
                        g << vector[i].nom << endl ;
                        g << vector[i].unitats <<" " ;
                        g << vector[i].preu << endl ;
                }
                g.close();
        }
        return error;
}
int main ( void )
{
  peca llista[TAM];
  int quants;  //nombre de peces que tenim
  string nom;

  cout << "Dona'm el nom del fitxer \n" ;
  cin >> nom;

  if( LlegirFitxer ( nom, llista , quants ) == false )
  {
     if( EscriuFitxer ( nom, llista , quants ) )
        cout << "Error obrint fitxer per a escriure\n";
  }
  else
        cout << "Error obrint fitxer per a llegir\n";
  return 0;
}
```

Operacions en fitxers binaris (1/4)

- Abans de descriure les operacions de lectura i escriptura en fitxers binaris, recordem que una característica important d'aquest tipus de fitxers és que podem accedir directament a una dada sense haver de llegir totes les anteriors.

- Per a poder fer aquest accés directe, necessitarem unes funcions específiques.

- Mètode d'accés directe
- L'accés directe consisteix a poder moure la finestra del fitxer a la posició del fitxer que volem, sense haver de llegir totes les anteriors.
- En C++, la finestra del fitxer correspon únicament amb un caràcter i es mou caràcter a caràcter. Per tant, la posició correspondrà amb el nombre de caràcters anteriors i NO al nombre de dades. La primera posició del fitxer és sempre la posició 0.

- Exemple: Suposem un fitxer f amb el contingut següent:

Fitxer f
```
HOLA\n
PERE\n
```

- Caràcter en la posició 0: H
- Caràcter en la posició 3: A
- Caràcter en la posició 6: P (en DOS o WINDOWS) En el sistema operatiu Windows, el salt de línia s'escriu als fitxers mitjançant 2 caràcters, concretament els corresponents als codis 13 i 10.

Operacions en fitxers binaris (2/4)

- Els mètodes usats per a l'accés directe són els següents:

- Per a ifstream:
  nom_fitxer_logic.seekg(pos)
  nom_fitxer_logic.tellg()

- Per a ofstream:
  nom_fitxer_logic.seekp(pos)
  nom_fitxer_logic.tellp()

- seekp(pos) o seekg(pos) col·loca la finestra del fitxer en la posició pos. tellg() o tellp() tornen un enter que indica la posició actual de la finestra del fitxer.

- **Exemple**:

```
// Col·locar la finestra del fitxer al principi del fitxer
f.seekg(0);
// Anar a la posició 6 (7è caràcter) de f
f.seekg(6);
cout << f.get();        ->  P
cout << f.tellg();      ->  7
```

  El mètode tellg ha tornat 7 perquè, quan s'ha llegit el caràcter, la finestra s'ha mogut al caràcter següent.

Operacions en fitxers binaris (3/4)

- Lectura i escriptura de fitxers binaris
- En C++, la lectura en binari es realitza mitjançant el mètode read i l'escriptura, mitjançant el mètode write. En ambdós casos, cal passar com a primer paràmetre un punter de tipus char a la variable on volem llegir o escriure i com a segon el nombre de bytes que volem llegir o escriure.

- **Exemple: Lectura d'enters en binari**.

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

    f.open("dades.bin");
    if(!f)
        cout << "Error obrint el fitxer" << endl;
    else
    {
        while(f.read((char *)(& dada), sizeof(dada) ) )
            cout << dada << endl;
        f.close();
    }
    return 0;
}
```

Operacions en fitxers binaris (4/4)

- **Exemple: Escriptura de 10 enters en binari.**

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

int main()
{
    ofstream f;
    int i;

    f.open(“dades.bin”);

    if(!f)
        cout << “Error obrint el fitxer” << endl;
    else
    {
        for(i = 1; i <= 10; i++)
            f.write((char *)(& i), sizeof(i) );
        f.close();
    }
    return 0;
}
```