Práctica 10: Ficheros en C++

Todos los programas vistos hasta ahora trabajaban con información almacenada en memoria principal, no obstante, hay situaciones en que esto no es apropiado. Algunas de esas situaciones podrían ser:

- Los datos con los que necesita trabajar el programa son demasiado grandes (ocupan mucha memoria) para que entren en la memoria principal.
- Nos interesa mantener la información después de cada ejecución, necesitamos utilizar datos procedentes de otros programas (editores, etc.), o generar datos para que puedan ser utilizados por otros programas.

En dichos casos se utilizarán ficheros para contener la información en memoria secundaria (disco duro, disquetes, etc.).

Definición de Fichero:
Es una colección de elementos lógicamente relacionados y almacenados en memoria secundaria. A más bajo nivel, un fichero es una secuencia de bits almacenado en algún dispositivo externo (como por ejemplo uno de memoria secundaria).

En C++ un fichero es simplemente un flujo externo que se puede abrir para entrada (dando lugar a un *flujo de archivo de entrada* que, para simplificar, llamaremos simplemente *archivo o fichero de entrada*), para salida (dando lugar a un *flujo de archivo de salida* que, para simplificar, llamaremos simplemente *archivo o fichero de salida*) o para entrada-salida (archivo o fichero de entrada-salida o archivo de E/S).

C++ soporta dos tipos de archivos: de texto y binarios. Los primeros almacenan datos como códigos ASCII. Los valores simples, tales como números y caracteres están separados por espacios o retornos de carro. Los segundos almacenan bits de forma directa (por lo que no se necesitan separadores) y se necesita usar la dirección de una posición de almacenamiento.

Una biblioteca en C++ que proporciona "funciones" y operadores para el manejo de ficheros es la biblioteca *fstream.*

```
#include <fstream.h>
```

Definición, Apertura y Cierre de Ficheros.

Declaración de Variables de tipo "Fichero":

```
ifstream descriptor;                  // Para ficheros de entrada
ofstream descriptor;                  // Para ficheros de salida
```

Apertura de Ficheros:

```
ifstream in;                          // Para ficheros de entrada
ofstream out;                         // Para ficheros de salida

in.open("nombre.extensión");          /* Apertura de Fichero de Texto o Binario
                                         para Lectura
                                      */
out.open("nombre.extensión");         /* Apertura de Fichero de Texto o Binario
                                         para Escritura. (borra el contenido si
                                         lo hubiera)
                                      */
out.open("nombre.extensión",ios::app); /* Apertura de Fichero de Texto o Binario
                                         para añadir datos al final
                                      */
in.open("nombre.extensión",           /* Apertura de Fichero de Texto o Binario
        ios::in | ios:: out);            para Lectura y Escritura
                                      */
out.open("nombre.extensión",          /* Apertura de Fichero de Texto o Binario
        ios::in | ios:: out);            para Lectura y Escritura
                                      */
```

Ejemplo 1:

Las dos líneas siguientes abren el fichero "mio.txt" como fichero de entrada (para lectura) y lo asocian al descriptor *in*.

```
ifstream in;            // descriptor del fichero a abrir
in.open("mio.txt");     // Apertura del fichero;
```

Ejemplo 2:

Para abrir el fichero "salida.dat" en modo salida (si el fichero no existe lo crea, y si existe borra su contenido) asociándolo al descriptor *out* podemos usar la siguiente sentencia;

```
ofstream out;
out.open("salida.dat");
```

Ejemplo 3:

```
// Abrimos el fichero "F2.txt" como fichero de salida en modo AÑADIR.
     fstream inout;
     inout.open("F2.txt", ios::app);
```

Cierre de ficheros.
Un fichero anteriormente abierto y con un descriptor asociado a él debe ser cerrado con el fin de liberar los recursos asociado a él de la siguiente forma:

```
descriptor.close()
```

Detección de fin de fichero y otras funciones.

- La función `eof()` que devuelve **`true`** si se ha alcanzado el final del fichero y falso en cualquier otro caso. REQUIERE LECTURA ADELANTADA: Para que la función `eof()` devuelva un valor de verdad (actualizado).
- La funcioón `fail()` devuelve **`true`** si existe un error en una operación de flujo asociada al fichero
- La función `bad()` devuelve **`true`** si el flujo está corrupto.
- La función `good()` que devuelve **`true`** si no existe un error en una operación de flujo y **`false`** en caso contrario.

Antes de empezar a leer o escribir en un fichero es siempre conveniente verificar que la operación de apertura se realizó con éxito.

Ejemplo 4:

```
ifstream in;

in.open("F1.dat");
if (in.bad() )
{
  cout << "\n Incapaz de crear o abrir el fichero ";   // salida estándar
  cout << " para salida " << endl;
}
else
{
    // Se opera sobre el fichero
}
```

Ficheros de texto

La lectura y la escritura en un archivo de texto se puede realizar directamente con los operadores << y >> al igual que se realiza sobre los flujos estándares *cin* y *cout*.

Ejemplo 5:

El siguiente programa escribe tres líneas en un fichero llamado "EJEMPLO5.TXT" que se crea en el programa (si ya existe borramos su contenido). Cada línea consta de un entero, un real y una cadena de caracteres. Los datos en cada línea están separados por espacios en blanco.

```
#include <fstream.h>   // Biblioteca para el manejo de ficheros
#include <iostream.h>  // Biblioteca para la entrada-salida estándar

int main()
{
  ofstream fichout;

  fichout.open("EJEMPLO5.TXT");
  if (fichou.bad()t)
  {
    cout << "\n Incapaz de crear este o abrir el fichero \n";
  }
  else
  { // Escritura en el fichero
    fichout << "Alumno 1" << " " << 5.0 << " APROBADO" << endl;
    fichout << "Alumno 2" << " " << 1.1 << " SUSPENSO" << endl;
    fichout << "Alumno 3" << " " << 8.0 << " NOTABLE " << endl;
    fichout.close();
  }
}  // Fin del main
```

Comentario:
El operador >> omite los espacios en blanco al igual que ocurría en la entrada estándar.

Ejemplo 6:

El siguiente programa lee el fichero de texto "EJEMPLO5.TXT", creado en el ejemplo anterior, y visualiza su contenido en el monitor.

```
#include <fstream.h>   // Libreria para el manejo de ficheros
#include <iostream.h>
typedef char TCadena[30];

int main()
{
  ifstream fichin;   // declaracion del fichero
  int i;
  float r;
  TCadena cad;

 fichin.open("EJEMPLO5.TXT");
 if (fichin.bad())
 { cout << "Incapaz de crear o abrir el fichero " << endl;
 }
 else
 { fichin >> i;             // Observese la lectura adelantada!!!
   while (!fichin.eof())
```

```
    {  cout << i << " ";      // Lectura de valores en el fichero
       fichin >> r;
       cout << r << " ";      // Lectura de valores en el fichero
       fichin >> cad;
       cout << cad << endl;   // Lectura de valores en el fichero
       fichin >> i;
     }
     fichin.close();
  }  // Fin del main
```

Comentarios:
(1) Observe en el ejemplo anterior que ha sido necesario realizar una lectura adelantada previamente al chequeo del fin de fichero. Si no se realiza de esta forma podríamos tener problemas.
(2) Observe también que no es necesario realizar una lectura para "saltarse" los espacios en blanco que fueron introducidos en el fichero "EJEMPLO5.TXT" en el ejemplo 5. Esto es debido a que, como ya se comentó anteriormente, el operador >> omite los espacios en blanco
(3) No olvide cerrar los ficheros!!

Ficheros Binarios

Los ficheros binarios tiene la información tal cual está en memoria, es decir, sin convertirla a texto, La manera de leer y escribir en ficheros binarios consiste en utilizar las funciones *read()* y *write()*.

```
descriptor.read(&c, num);
```

```
Descriptor.write(&c, num);
```

donde `c` es una variable de cualquier tipo, pasada por referencia, y `num` es un entero o una variable de tipo entero, ejecuta una lectura/escritura (a partir de la posición actual del cursor del fichero asociado a la variable *descriptor*) de *num* bytes del fichero. Normalmente, para expresar el tamaño o número de bytes a leer/escribir se usa la función `sizeof` que nos retorna el número de bytes que ocupa una variable o tipo. Por ejemplo, para leer/ escribir un número entero en binario, usaremos:

```
int x;
...
descriptor.read(&x,sizeof(int));
descriptor.write(&x,sizeof(int));
```

Ejemplo de Lectura/Escritura de Ficheros: Una Agenda de Teléfonos

```
#include <iostream.h>
#include <stdlib.h>
#include <fstream.h>
#include <ctype.h>

// Ejemplo de una agenda de teléfonos con toda la información en disco.
//CONSTANTES
const char FINCAD = char(0);
const int MAXCAD  = 80;
const int ENTER  = '\n';
const char SP = ' ';

//TIPOS
typedef char TCadena[MAXCAD+1]; // MAXCAD caracteres + FINCAD

struct TPersona
{
  TCadena nombre;
  TCadena apellido1;
  TCadena apellido2;
  int telefono;
};
```

```
// CABECERA DE PROCEDIMIENTOS Y FUNCIONES
char menu();
void pausa();
void borrar_pantalla();
bool confirmar_salida();
void LeerPersona(TPersona &p);
void EscribirPersona(TPersona p);

// Algoritmos de Manejo de Ficheros de Texto
void insertarPersonaTXT(TCadena nombreFichero, TPersona p);
void listarAgendaTXT(TCadena nombreFichero);
void LeePersonaFicheroTXT(ifstream &fichero, TPersona &p);
void EscribePersonaFicheroTXT(ofstream &fichero, TPersona p);

// Algoritmos de Manejo de Ficheros Binarios
void insertarPersonaBIN(TCadena nombreFichero, TPersona p);
void listarAgendaBIN(TCadena nombreFichero);
void LeePersonaFicheroBIN(ifstream &fichero, TPersona &p);
void EscribePersonaFicheroBIN(ofstream &fichero, TPersona p);

// PROGRAMA PRINCIPAL
int main()
{
  TPersona p;;
  char opcion;
  int num;
  bool fin,encontrado;
  TCadena nomFich;

  fin = false;

  do
  {
    borrar_pantalla();
    opcion = menu();

    switch(opcion)
    {
      case 'A': cout << "Nombre del Fichero: ";
                cin >> nomFich;
                LeerPersona(p);
                insertarPersonaTXT(nomFich,p);
                break;
      case 'B': cout << "Nombre del Fichero: ";
                cin >> nomFich;
                LeerPersona(p);
                insertarPersonaBIN(nomFich,p);
                break;
      case 'C': cout << "Nombre del Fichero: ";
                cin >> nomFich;
                listarAgendaTXT(nomFich);
                break;
      case 'D': cout << "Nombre del Fichero: ";
                cin >> nomFich;
                listarAgendaBIN(nomFich);
                break;
      case 'X': fin = confirmar_salida();
                break;
    }

    if (!fin)
    {
      pausa();
    }
```

```
  } while (!fin);

  pausa();
  return 0;
}

// IMPLEMENTACIÓN DE PROCEDIMIENTOS Y FUNCIONES

void pausa()
{
  system("PAUSE");
}

void borrar_pantalla()
{
  system("CLS");
}
char menu()
{
  char op;

  cout << "AGENDA EN FICHERO" << endl;
  cout << "-----------------" << endl;
  cout << "Autor: <Apellidos> <Nombre>" << endl;

  cout << "A. Insertar Persona TXT" << endl;
  cout << "B. Insertar Persona BIN" << endl;
  cout << "C. Listar TXT" << endl;
  cout << "D. Listar BIN" << endl;
  cout << "X. Salir del Programa." << endl;
  cout << endl;
  cout << "Introduzca Opción:";

  cin >> op;
  op = toupper(op);
  while( ((op<'A') || (op>'D')) && (op!='X') )
  {
    cout << "... OPción Incorrecta ..." << endl;
    cout << endl;
    cout << "Introduzca Opción:";
    cin >> op;
    op = toupper(op);
  }
  return op;
}

void LeerPersona(TPersona &p)
{
  cout << "Nombre    : " ;
  cin  >> p.nombre;
  cout << "Apellidos1: " ;
  cin  >> p.apellido1;
  cout << "Apellidos2: " ;
  cin  >> p.apellido2;
  cout << "Teléfono  : ";
  cin  >> p.telefono;
}

void EscribirPersona(TPersona p)
{
  cout << "Nombre             : " << p.nombre << endl;
  cout << "Apellidos          : " << p.apellido1
       << " " << p.apellido2 << endl;
  cout << "Teléfono           : " << p.telefono << endl;
}
```

```
bool confirmar_salida()
{
  char car;

  cout << "¿Está seguro de salir (S/N)?";
  cin >> car;
  car = toupper(car);

  return (car=='S');
}
```

```
// Algoritmos de Manejo de Ficheros de Texto
void insertarPersonaTXT(TCadena nombreFichero, TPersona p)
{
  ofstream out;

  out.open(nombreFichero,ios::app);
  // Abro el fichero para añadir

  if (out.bad())
  { // El fichero no existe ... lo creo
    out.open(nombreFichero);
  }

  EscribePersonaFicheroTXT(out,p);
  out.close();
}

void listarAgendaTXT(TCadena nombreFichero)
{ ifstream in;
  TPersona persona;

  in.open(nombreFichero);

  if (in.bad())
  {
    cout <<  "Error al abrir el fichero: "
         << nombreFichero << endl;
  }
  else
  {
    LeePersonaFicheroTXT(in,persona);
    while (!in.eof())
    {
      EscribirPersona(persona);
      LeePersonaFicheroTXT(in,persona);
      pausa();
    }
    in.close();
  }
}

void LeePersonaFicheroTXT(ifstream &fichero, TPersona &p)
{
  fichero  >> p.nombre;
  fichero  >> p.apellido1;
  fichero  >> p.apellido2;
  fichero  >> p.telefono;
```

```
}

void EscribePersonaFicheroTXT(ofstream &fichero, TPersona p)
{
  fichero << p.nombre    << SP;
  fichero << p.apellido1 << SP;
  fichero << p.apellido2 << SP;
  fichero << p.telefono  << endl;
}









// Algoritmos de Manejo de Ficheros Binarios
void insertarPersonaBIN(TCadena nombreFichero, TPersona p)
{
  ofstream out;

  out.open(nombreFichero,ios::app); // Abro el fichero para añadir

  if (out.bad())
  { // El fichero no existe ... lo creo
    out.open(nombreFichero);
  }
  EscribePersonaFicheroBIN(out,p);
  out.close();
}


void listarAgendaBIN(TCadena nombreFichero)
{
  ifstream in;
  TPersona persona;

  in.open(nombreFichero);

  if (in.bad())
  {
    cout <<  "Error al abrir el fichero: " << nombreFichero << endl;
  }
  else
  {
    LeePersonaFicheroBIN(in,persona);
    while (!in.eof())
    {
      EscribirPersona(persona);
      LeePersonaFicheroBIN(in,persona);
      pausa();
    }
    in.close();
  }
}

void LeePersonaFicheroBIN(ifstream &fichero, TPersona &p)
{
  fichero.read(&p,sizeof(TPersona));
}
```

```
void EscribePersonaFicheroBIN(ofstream &fichero, TPersona p)
{
  fichero.write(&p,sizeof(TPersona));
}
```

Enunciado.

Un banco almacena en ficheros de texto toda la información de sus clientes y e sus cuentas. En un fichero de texto llamado "clientes.txt", se tiene la información de los clientes (nif, nombre, apellido1, apellido2, dirección y teléfono, todos ellos cadenas de un máximo de 40 caracteres). Y en otro fichero de texto llamado "cuentas.txt" se tiene la información de las cuentas(nif del titular que es una cadena de un máximo de 40 caracteres, número de cuenta que es un número natural y saldo en euros que es un número real).
Por otra parte, se tiene un fichero binario llamado "movimientos.dat" que almacena registros tipo TMovimiento y que tiene los diferentes movimientos de todas las cuentas. Se pide hacer un programa con las siguientes opciones de menú.

```
MENU BANCO
==========

A. Insertar Cliente.
B. Insertar Cuenta
C. Insertar movimiento.
D. Listado Clientes
E. Listado Movimientos Persona.
X. Salir Insertar Cliente.
```

Descripciónde Opciones:

- **A. Insertar Clinetes.** Se solicita el NIF del cliente. Si ya se encuentra en el fichero, se informa de ello y se muestran todos los datos del cliente. Si no existía, se añade al fichero y se da un mensaje diciendo que se ha insertado.
- **B. Insertar Cuenta.** Se solicita el NIF del cliente y se busca en el fichero de clientes. Si no está en el fichero de clientes se informa del error y se termina la operación. Si está, se muestran sus datos, se le asigna como número de cuenta el máximo más 1 de los números de cuenta del fichero de cuentas y se coloca su saldo a 0. Se mostrará por pantalla el número de cuenta asignado y se insertará la información en el fichero de cuentas.
- **C. Insertar movimiento.** Se solicita el NIF, el número de cuenta y el importe del movimiento. tras comprobar que la cuenta existe se insertará el movimiento. Si la cuenta y/o el cliente no existe, se informará de ello y no se insertará.
- **D. Listado Clientes.** Se mostrará por pantalla un listado con los datos de todos los clientes.
- **E. Listado Movimientos Persona.** Se pedirá el NIF de una persona. Tras buscarla en el fichero de clientes, si no se encuentra se avisará y si se encuentra, se mostrará toda la información de la persona y todos los movimientos de su cuenta (suponer que sólo hay una cuenta por persona) y su saldo final.
- **X. Salir.** Se pedirá confirmación y en caso afirmativa se saldrá del programa

Formato del Fichero `Clientes.txt`

```
<NIF>#<NOMBRE>#<APELLIDO1>#<APELLIDO2>#<DIRECCIÓN>#<TELEFONO>

Ejemplo: "CLIENTES.TXT"
11111111A#JOSE MARIA#GARCIA#MUÑOZ#SU CASA#555 111 111
22222222B#ALMA#SANCHEZ#PEREZ#OTRA CASA#555 222 222
33333333C#MARIA#PEREZ#GARCIA#MAS LEJOS#555 333 333
```

Formato del Fichero `Cuentas.txt`

```
<Nº CUENTA> <NIF> <SALDO>

Ejemplo: "CUENTAS.TXT"
2 11111111A 1250.30
3 22222222B 130.25
1 33333333C -5.01
```

Definición del tipo TMovimiento.

```
struct TMovimiento
{
  TCadena fecha;
  int    num_cuenta;
  double valor;
};
```