Lectura y escritura en archivos en C++

Este documento se refiere a archivos de acceso secuencial. En el archivo de cabecera `fstream` están las herramientas para la lectura y escritura en archivos.

Los archivos se abren creando objetos de las clases de flujo `ifstream`, `ofstream` o `fstream`. Estos nombres vienen de *input file stream* para lectura, *output file stream* para escritura y *file stream* que agrupa lectura y escritura.

Más que dar las reglas, se presentan más adelante dos ejemplos. El primero es un programa muy sencillo donde en un archivo se leen dos números y se escribe su suma en otro archivo. El segundo es un programa en el que hay dos archivos de lectura, cada uno tiene la información sobre un vector; el programa lee los dos vectores, si son del mismo tamaño los escribe en un tercer archivo y allí escribe el valor del producto interior. En este último programa el "archivo" se pasa como parámetro de una función. Además en la escritura de los elementos (entradas) del vector se utiliza un formato diferente del formato por defecto de C++. Los dos programas tiene varias líneas de comentarios con explicaciones sobre el manejo de archivos.

Para un objeto `ifstream` el modo de apertura del archivo generalmente es `ios::in`. Para un objeto `ofstream` el modo de apertura del archivo puede ser `ios::out`. Un objeto (archivo) se cierra mediante la orden *objeto*`.close();`.

Primer ejemplo

```
// Lectura y escritura en archivos en C++

#include <iostream>
using namespace std;

#include <fstream>
#include <stdlib.h>

int main(void)
{
  // Este programa utiliza 2 archivos, uno de datos
  // y otro para escribir los  resultados.
  // El nombre externo del archivo de datos sera
```

```
  // dado por el usuario. En este ejemplo, el nombre externo
  // del archivo de resultados es fijo: result

  // Dentro del programa hay dos objetos:
  //    aDat para lectura  y
  //    aRes para escritura.

  // En el archivo de datos lee dos numeros y escribe la
  // suma en el archivo  result.

  double a, b;
  char nombre[80];

  cout<<"\n\nNombre del archivo de datos: ";
  cin>>nombre;

  ifstream aDat(nombre, ios::in);
  if( !aDat ){
    cout<<"\nNombre de archivo erroneo.\n\n";
    exit(1);
  }

  ofstream aRes("result", ios::out);

  aDat>>a;
  aDat>>b;

  // Tambien se puede en una sola orden.
  // aDat>>a>>b;

  aRes<<a<<" + "<<b<<" = "<<a+b<<endl;

  aDat.close();
  aRes.close();

  return 0;
}
```

Segundo ejemplo

```
// Lectura y escritura en archivos en C++

#include <iostream>
using namespace std;

#include <fstream>
#include <stdlib.h>

//-----------------------------------------------------------
void lectVect(istream &archi, double *x, int n);
void escrVect(ostream &archi, double *x, int n);
double prodInt(double *x, double *y, int n);
//-----------------------------------------------------------
int main(void)
{
  // Este programa utiliza 3 archivos, dos de datos
  // y otro para escribir los  resultados.
  // Los nombres de los archivos seran
  // dado por el usuario.

  // Dentro del programa hay tres objetos:
  //    dat1 y dat2 para lectura  y
  //    aRes para escritura.

  // En cada archivo de datos esta la informacion sobre
  // un vector: el numero de elementos (entradas) y los elementos
  // del vector.

  // Si los dos vectores son del mismo tama~no, el programa calcula
  // el producto escalar (o producto punto o producto interior)
  // y escribe el resultado.

  double *x, *y;
  char nombre[80];
  int m, n;
```

```
cout<<"\n\nNombre del  primer archivo de datos: ";
cin>>nombre;
ifstream dat1(nombre, ios::in);
if( !dat1 ){
  cout<<"\nNombre de archivo erroneo.\n\n";
  exit(1);
}

cout<<"Nombre del segundo archivo de datos: ";
cin>>nombre;
ifstream dat2(nombre, ios::in);
if( !dat2 ){
  cout<<"\nNombre de archivo erroneo.\n\n";
  exit(1);
}

cout<<"Nombre del  archivo para resultados: ";
cin>>nombre;
ofstream aRes(nombre, ios::out);
if( !aRes ){
  cout<<"\nNombre de archivo erroneo.\n\n";
  exit(1);
}

dat1>>n;
dat2>>m;

if( n != m ){
  cout<<"Vectores de taman~o diferente.\n";
  exit(1);
}

aRes<<"n = "<<n<<endl;

x = new double[n];
y = new double[n];

lectVect(dat1, x, n);
```

```
  aRes<<"x: \n";
  escrVect(aRes, x, n);

  lectVect(dat2, y, n);
  aRes<<"y: \n";
  escrVect(aRes, y, n);
  aRes<<" producto interior = "<<prodInt(x, y, n)<<endl;

  dat1.close();
  dat2.close();
  aRes.close();

  return 0;
}
//-----------------------------------------------------------
void lectVect(istream &archi, double *v, int n)
{
  // lectura del vector  v  en  archi

  int i;

  for( i = 0; i < n; i ++ ) archi>>v[i];
}
//-----------------------------------------------------------
void escrVect(ostream &archi, double *v, int n)
{
  // escritura del vector v  en archi

  int i;
  int numNumLin = 5;
  // Cuando el vector es grande, es preferible escribir en
  // varias lineas. El valor anterior indica cuantos numeros
  // se escriben en cada linea.

  // para fijar el numero de cifras decimales en la escritura
  archi.setf(ios_base::fixed, ios_base::floatfield);
  archi.precision(6);
```

```
  for( i = 0; i < n; i++ ){
    archi.width(15);
    archi<<v[i];
    if( (i+1)%numNumLin == 0 || i == n-1 ) archi<<endl;
  }

  // para volver a los parametros por defecto
  archi.precision(0);
  archi.unsetf(ios_base::floatfield);
}
//-------------------------------------------------------------
double prodInt(double *x, double *y, int n)
{
  // producto interior (producto punto) de dos vectores

  int i;
  double s = 0.0;

  for( i = 0; i < n; i++) s += x[i]*y[i];
  return s;
}
```