Laboratorio de Programación

Parte 3: Lectura y escritura de archivos en C++

Algoritmos y Estructuras de Datos I

Departamento de Computación, FCEyN, Universidad de Buenos Aires.

Entrada - salida

Ya vimos ejemplos de entrada - salida por consola.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int a;
6    cout << "Ingrese un entero: "; // Salida
7    cin >> a; // Entrada
8    cout << "El numero ingresado es: " << a; // Salida
9    return 0;
10 }
```

- El `cout` (console out) es un *stream* (flujo), que imprime por pantalla.
- El `cin` (console in) es un *stream* (flujo), que lee un dato del teclado.

Escribir un archivo

Escribir en un **archivo de texto plano** en C++ es similar a escribir texto por consola.

- Necesitamos incluir la biblioteca `fstream`.

```
1 #include <fstream>
```

- Hay que declarar el tipo de iostream:
  - `ofstream`: para escribir (**of** por *out file*)
  - `ifstream`: para leer (**if** por *in file*)

```
1 int a;
2 ofstream archivoSalida;
3 ifstream archivoEntrada;
```

Escribir un archivo (ofstream)

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4
5  int main() {
6    int a = 5;
7    ofstream fout;
8    fout.open("archivo.txt"); // crea el archivo
9    fout << "Hola, archivo!" << endl;
10   fout << "Ahora, un entero: " << a << endl;
11   fout << "Y un valor de verdad: " << ((a+2) == 10) << endl;
12   fout.close(); // guarda y cierra el archivo
13   return 0;
14 }
```

Escribir un archivo

Contenido final de `archivo.txt`:

```
1 Hola, archivo!
2 Ahora, un entero: 5
3 Y un valor de verdad: 0
```

Otra visualización (con el caracter “salto de linea”)

```
1 Hola, archivo!\nAhora, un entero: 5\nY un valor de verdad: 0\n
```

Escribir un archivo (al final de uno existente)

¿Qué hace `ofstream.open("archivo.txt")`
si `archivo.txt` ya existe?

- Si no existe, crea el archivo
- Si existe, lo sobreescribe (¡borra lo que había antes!)

¿Cómo podemos hacer para que
el contenido anterior sea respetado?

- Hay que abrirlo en modo **append**

```
1 ofstream.open("archivo.txt",ios_base::app)
```

Escribir un archivo (CSV)

Escribir todos los elementos de una matriz separados por coma

```
1  void escribirVectorCSV(const vector<int> & v) {
2    ofstream fout;
3    fout.open("salida.csv",ios_base::app);
4    for(int i=0; i < v.size(); i = i + 1){
5      fout << v[i] << ',';
6    }
7    fout << endl;
8    fout.close();
9  }
10 void escribirMatrizCSV(const vector<vector<int>> & m){
11   for(int i=0; i < m.size(); i = i + 1){
12     escribirVectorCSV(m[i]);
13   }
14 }
```

Leer un archivo

Leer de un **archivo de texto plano** en C++
es similar a leer texto por consola.

Debemos seguir el siguiente protocolo:

```
1 ifstream archivoIn; // Declara el archivo
2 archivoIn.open("archivo.txt",ios::in); // abre modo lectura
3 archivoIn >> ... ; // lee (1 o varias veces)
4 archivoIn.close(); // Cierra el archivo
```

Para leer es necesario saber cómo está escrito el archivo
(formato) y el tipo de datos que contiene.

Leer un archivo

entrada.txt:

```
15 20 25
```

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

int main() {
  vector<int> v(3); // <0, 0, 0>
  ifstream archivoIn;
  archivoIn.open("entrada.txt", ios::in);
  archivoIn >> v[0]; // <15, 0, 0>
  archivoIn >> v[1] >> v[2]; // <15, 20, 25>
  archivoIn.close(); //
  return 0;
}
```

Leer un archivo (CSV)

matrizIntFloatBool.csv:

```
10,1.7,0
13,0.33,1
7,0.75,1
```

```
void matrizIntFloatBool(vector<int>& a, vector<float>& b,
vector<bool>& c){
    ifstream fin; char coma; bool z;
    fin.open("matrizIntFloatBool.csv", ios::in);
    for(int i=0; i<3; i=i+1){
        fin >> a[i] >> coma >> b[i] >> coma >> z; c[i] = z;
    }
    fin.close();
}
int main() {
    vector<int> a(3); vector<float> b(3); vector<bool> c(3);
    matrizIntFloatBool(a,b,c);
    return 0;
}
```

Leer un archivo (.eof())

¿Y si el tamaño del archivo es variable?
La función .eof() retorna true si no hay más contenido.

```
1  void matrizIntFloatBool(vector<int>& a, vector<float>& b,
2  vector<bool>& c){
3      ifstream fin; char coma; int n; float f; bool z;
4      fin.open("matrizIntFloatBool.csv", ios::in);
5      while(!fin.eof()){
6          fin >> n >> coma >> f >> coma >> z;
7          a.push_back(n); b.push_back(f); c.push_back(z);
8      }
9      fin.close();
10 }
11 int main() {
12     vector<int> a; vector<float> b; vector<bool> c;
13     matrizIntFloatBool(a,b,c);
14     return 0;
15 }
```

Manejo de errores (`.fail()`)

- ¿Qué pasa si queremos leer un archivo que no existe?
- ¿Qué pasa si no tenemos permisos para leer un archivo?
- ¿Qué pasa si no tenemos permisos para escribir un archivo?

La función `.fail()` retorna `true` si hubo una falla al intentar ejecutar una operación (por ejemplo: `.open()`, `.close()`)

Manejo de errores

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

int main() {
    ifstream fin;
    fin.open("archivoNoExiste.txt", ifstream::in);
    if (fin.fail()) { // true si hubo error al abrir
        cout << "Error" << endl;
    } else {
        cout << "Abierto" << endl;
    }
    fin.close();
    return 0;
}
```

Resumen: E/S con archivos en C++

- ifstream: stream de lectura de archivos
- ofstream: stream para escritura de archivos
- open(): abre un archivo para escritura o lectura dependiendo del tipo de stream
- close(): cierra un archivo
- $<<$ (escribe un valor) y $>>$ (lee un valor)
- eof(): retorna `true` si la lectura del archivo llegó al final
- fail(): retorna `true` si la última operación falló

Bibliografía

- B. Stroustrup. The C++ Programming Language.
  - 31.4.1: El STL container *vector*
  - 38.2.1: File Streams