Archivos

Archivos de texto

Tomás Peiretti

Archivos de texto

Puntos clave:

- La información se almacena como una secuencia de caracteres.
- Son secuenciales: El orden de acceso a los datos está determinado, primero se accede al primer elemento y luego se puede ir accediendo a los siguientes, de uno en uno.
- Hay 3 tipos:
  - De entrada: ifstream
  - De salida: ofstream
  - De entrada/salida: fstream

Archivos de texto: implementación

Para utilizar un archivo de texto en nuestro programa debemos:

1. Incluir la librería fstream
2. Declarar la variable que actuará como manejador de fichero
3. Abrir el flujo de datos, vinculando la variable correspondiente con el fichero especificado.
4. Comprobar que la apertura del fichero se realizó correctamente
5. Realizar la transferencia de información
6. Finalmente, cerrar el flujo para liberar la variable manejador de su vinculación con el fichero.

Archivos de texto: implementación

```
1  #include <iostream>
2  #include <fstream>  //1
3  using namespace std;
4
5  int main(int argc, char *argv[]) {
6    ifstream archivo; // 2
7    archivo.open("text.txt"); // 3
8
9    if (archivo.is_open()) { // 4
10     int x;
11     archivo >> x; // 5
12     cout << "Valor en el archivo = " << x << endl;
13   }
14   else {
15     cout << "Error al abrir el archivo" << endl;
16   }
17
18   archivo.close(); // 6
19
20   return 0;
21 }
```

Archivos de texto: modos de apertura

Para abrir un archivo siempre utilizaremos el método open, el cual recibe 2 parámetros:

- El nombre del archivo a abrir
- El modo de apertura

member constant	stands for	access
`in`	**in**put	File open for reading: the ***internal stream buffer*** supports input operations.
`out`	**out**put	File open for writing: the ***internal stream buffer*** supports output operations.
`binary`	**binary**	Operations are performed in binary mode rather than text.
`ate`	**at e**nd	The ***output position*** starts at the end of the file.
`app`	**app**end	All output operations happen at the end of the file, appending to its existing contents.
`trunc`	**trunc**ate	Any contents that existed in the file before it is open are discarded.

These flags can be combined with the bitwise OR operator (`|`).

Archivos de texto: ejemplo

```
1  // Ejemplo: de un archivo que almacena numeros enteros,
2  // se debe procesar e imprimir la suma de todos ellos
3  int main() {
4      fstream arch;
5      arch.open("numeros.txt", ios::in);
6
7      if (arch.is_open()) {
8          int x;
9          int sum = 0;
10         while (arch >> x) {
11             sum += x;
12         }
13         cout << sum << endl;
14     }
15     else {
16         cout << "Error al abrir el archivo" << endl;
17     }
18
19     arch.close();
20
21     return 0;
22 }
```

Archivos de texto: ejemplo

```
1  // Ejemplo: de un archivo que almacena numeros enteros,
2  // se debe procesar e imprimir la suma de todos ellos
3  int main() {
4    fstream arch;
5    arch.open("numeros.txt", ios::in);
6
7    if (arch.is_open()) {
8      int x;
9      int sum = 0;
10     while (arch >> x) {
11       sum += x;
12     }
13     cout << sum << endl;
14   }
15   else {
16     cout << "Error al abrir el archivo" << endl;
17   }
18
19   arch.close();
20
21   return 0;
22 }
```

- Un archivo de texto de entrada se comporta igual que cin
- Un archivo de texto de salida se comporta igual que cout