Archivos

Archivos binarios

Tomás Peiretti

Archivos binarios

Puntos clave:

- C++ trabaja cada archivo como un flujo secuencial de bytes (podemos pensarlo como un arreglo de bytes)
- Permiten almacenar estructuras (struct) completas
- Es posible utilizar acceso directo para leer/escribir
- Hay 3 tipos:
  - De entrada: ifstream
  - De salida: ofstream
  - De entrada/salida: fstream

Archivos binarios: implementación

Al igual que los archivos de texto, para utilizar un archivo binario 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 binarios: implementación

```
1  #include <iostream>
2  #include <fstream>  //1
3  using namespace std;
4
5  int main() {
6    ifstream archivo; // 2
7    archivo.open("text.data", ios::binary); // 3
8
9    if (archivo.is_open()) { // 4
10     int x;
11     archivo.read((char *) &x, sizeof(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 }
```

Leer de un archivo binario

Para leer un archivo binario usaremos el método read de la siguiente manera:

```
.read((char *) & variable, sizeof(variable))
```

Esto leerá un bloque de datos (bytes) del tamaño de la variable y lo transformará al tipo de dato de la variable

Leer de un archivo binario: ejemplo

```
1  struct Figura {
2    int cantLados;
3    int perimetro;
4    double area;
5  };
6
7  int main() {
8    ifstream archFiguras;
9    ifstream archNumeros;
10   archFiguras.open("figuras.data", ios::binary);
11   archNumeros.open("numeros.bin", ios::binary);
12
13
14   if (archFiguras.is_open() && archNumeros.is_open()) {
15     Figura f;
16     int num;
17     archFiguras.read((char *) &f, sizeof(f));
18     archNumeros.read((char *) &num, sizeof(num));
19     cout << "Numero en el archivo = " << num << endl;
20     cout << "Perimetro de la figura en archivo = " << f.perimetro << endl;
21   }
22   else {
23     cout << "Error al abrir los archivos" << endl;
24   }
25
26   archFiguras.close();
27   archNumeros.close();
28
29   return 0;
30 }
```

Escribir en un archivo binario

Para escribir un archivo binario usaremos el método write de la siguiente manera:

.write((char *) & variable, sizeof(variable))

Esto grabará un bloque de datos (bytes) del tamaño de la variable con los bytes correspondientes

Escribir en un archivo binario: ejemplo

```
1  struct Figura {
2    int cantLados;
3    int perimetro;
4    double area;
5  };
6
7  int main(int argc, char *argv[]) {
8    ofstream archFiguras;
9    ofstream archNumeros;
10   archFiguras.open("figuras.data", ios::binary);
11   archNumeros.open("numeros.bin", ios::binary);
12
13   if (archFiguras.is_open() && archNumeros.is_open()) {
14     Figura f = {4, 20, 25};
15     int num = 96;
16     archFiguras.write((char *) &f, sizeof(f));
17     archNumeros.write((char *) &num, sizeof(num));
18   }
19   else {
20     cout << "Error al abrir los archivos" << endl;
21   }
22
23   archFiguras.close();
24   archNumeros.close();
25
26   return 0;
27 }
```

Decidir donde leer/escribir

```
1  struct Figura {
2      int cantLados;
3      int perimetro;
4      double area;
5  };
6  // Actualizar el perimetro y area de la 4ta figura (que es un cuadrado)
7  // para que se corresponda a un cuadrado de lado = 10
8  int main() {
9      fstream archFiguras;
10     archFiguras.open("figuras.data", ios::binary | ios::in | ios::out);
11     // suponiendo que se pudo abrir el archivo,
12     // calculo la posicion de inicio de los bytes
13     //      que corresponden a la 4ta figura
14     int posFig4 = sizeof(Figura) * 3;
15     Figura figura4;
16     // muevo el puntero de lectura del archivo
17     archFiguras.seekg(posFig4);
18     // leo los datos de la figura persistida
19     archFiguras.read((char *) &figura4, sizeof(figura4));
20     // actualizo el perimetro y el area
21     figura4.perimetro = 40;
22     figura4.area = 100;
23     // muevo el puntero de escritura del archivo
24     archFiguras.seekp(posFig4);
25     // guardo los cambios sobre el archivo
26     archFiguras.write((char *) &figura4, sizeof(figura4));
27
28     archFiguras.close();
29     return 0;
30 }
```

Yapa: Leer un arreglo completo

```
1  struct Figura {
2      int cantLados;
3      int perimetro;
4      double area;
5  };
6
7  int main() {
8      ifstream archFiguras;
9      archFiguras.open("figuras.data", ios::binary);
10
11     Figura figuras[100];
12     if (archFiguras.is_open()) {
13         archFiguras.read((char *) &figuras, sizeof(figuras));
14   }
15    else {
16     cout << "Error al abrir los archivos" << endl;
17   }
18
19     archFiguras.close();
20     return 0;
21 }
```

Desafío

Desafío: desarrollar una función que permita encontrar una figura a partir de la cantidad de lados provista.

La búsqueda debe realizarse sobre un archivo binario de figuras, el cual contiene las figuras ordenadas por su cantidad de lados.