

The program begins by opening an input file called "edipo.txt" that contains the text from which the entropy will be calculated. The contents of the input file are then read and stored in a variable of type `std::string`.

The `calculateEntropyShannon` function takes the text as a parameter and calculates the Shannon entropy per word. To do this, it counts the frequency of each word in the text, calculates the probability of occurrence of each word and finally calculates the entropy according to Shannon's formula.

After calculating the entropy, the program also counts the frequency of each word in the text and saves these results in an output file called "entropia.txt".

Finally, the program prints the Shannon entropy per word in the text and a message informing that the result has been saved to the output file.

In short, this program reads a text from a file, calculates the Shannon entropy per word in that text, counts the frequency of each word, and saves these results to another file.]**Entropy:** This program calculates the Shannon entropy per word in a given text. Shannon entropy is a measure of the uncertainty or disorder in a data set, in this case, a text.

The program begins by opening an input file called "edipo.txt" that contains the text from which the entropy will be calculated. The contents of the input file are then read and stored in a variable of type `std::string`.

The `calculateEntropyShannon` function takes the text as a parameter and calculates the Shannon entropy per word. To do this, it counts the frequency of each word in the text, calculates the probability of occurrence of each word and finally calculates the entropy according to Shannon's formula.

After calculating the entropy, the program also counts the frequency of each word in the text and saves these results in an output file called "entropia.txt".

Finally, the program prints the Shannon entropy per word in the text and a message informing that the result has been saved to the output file.

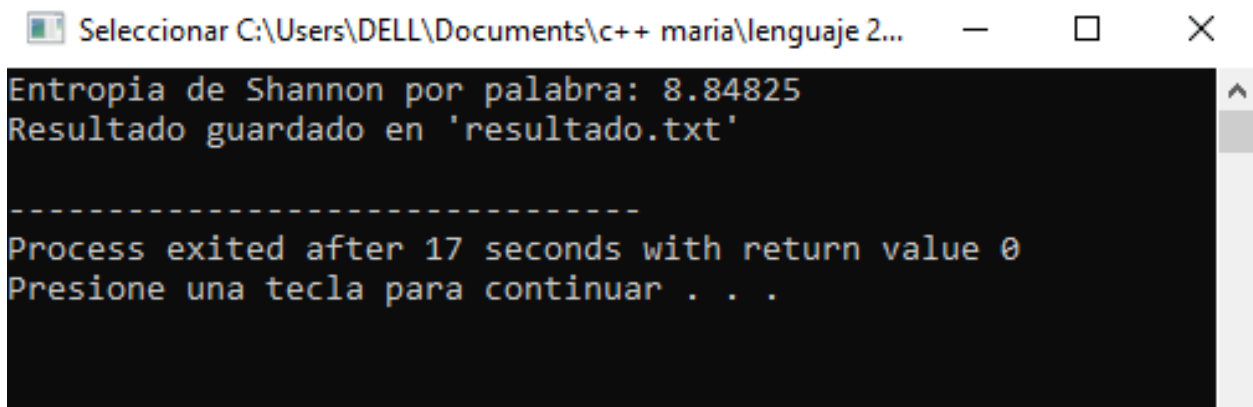
In short, this program reads a text from a file, calculates the Shannon entropy per word in that text, counts the frequency of each word, and saves these results to another file.

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <unordered_map>
5  #include <cmath>
6
7  double calcularEntropiaShannon(const std::string& texto) {
8      std::unordered_map<std::string, int> frecuenciaPalabras;
9      int totalPalabras = 0;
10
11      std::string palabraActual = "";
12      for (char c : texto) {
13          if (std::isalpha(c)) {
14              palabraActual += std::tolower(c);
15          } else if (!palabraActual.empty()) {
16              frecuenciaPalabras[palabraActual]++;
17              totalPalabras++;
18              palabraActual = "";
19          }
20      }
21
22      double entropia = 0.0;
23      for (const auto& par : frecuenciaPalabras) {
24          double probabilidad = static_cast<double>(par.second) / totalPalabras;
25          entropia -= probabilidad * std::log2(probabilidad);
26      }
27
28      return entropia;
29  }
30
31  int main() {
32      std::ifstream archivoEntrada("edipo.txt");
33      if (archivoEntrada.is_open()) {
34          std::string texto((std::istreambuf_iterator<char>(archivoEntrada)), std::istreambuf_iterator<char>());
35          archivoEntrada.close();
36
37          double entropia = calcularEntropiaShannon(texto);
38          std::cout << "Entropia de Shannon por palabra: " << entropia << std::endl;
39      }
```

Figure 1: Code in c++.

```
29 }
30
31 int main() {
32     std::ifstream archivoEntrada("edipo.txt");
33     if (archivoEntrada.is_open()) {
34         std::string texto((std::istreambuf_iterator<char>(archivoEntrada)), std::istreambuf_iterator<char>());
35         archivoEntrada.close();
36
37         double entropia = calcularEntropiaShannon(texto);
38         std::cout << "Entropia de Shannon por palabra: " << entropia << std::endl;
39
40         std::unordered_map<std::string, int> frecuenciaPalabras;
41         std::string palabraActual = "";
42         for (char c : texto) {
43             if (std::isalpha(c)) {
44                 palabraActual += std::tolower(c);
45             } else if (!palabraActual.empty()) {
46                 frecuenciaPalabras[palabraActual]++;
47                 palabraActual = "";
48             }
49         }
50
51         std::ofstream archivoSalida("entropia.txt");
52         if (archivoSalida.is_open()) {
53             for (const auto& par : frecuenciaPalabras) {
54                 archivoSalida << par.first << ": " << par.second << std::endl;
55             }
56             archivoSalida.close();
57             std::cout << "Resultado guardado en 'resultado.txt'" << std::endl;
58         } else {
59             std::cerr << "Error al abrir el archivo de salida." << std::endl;
60         }
61     } else {
62         std::cerr << "Error al abrir el archivo de entrada." << std::endl;
63     }
64
65     return 0;
66 }
```

Figure 2: Code in c++.



```
Seleccionar C:\Users\DELL\Documents\c++ maria\lenguaje 2...
Entropia de Shannon por palabra: 8.84825
Resultado guardado en 'resultado.txt'

-----
Process exited after 17 seconds with return value 0
Presione una tecla para continuar . . .
```

Figure 3: Shannon Entropy.

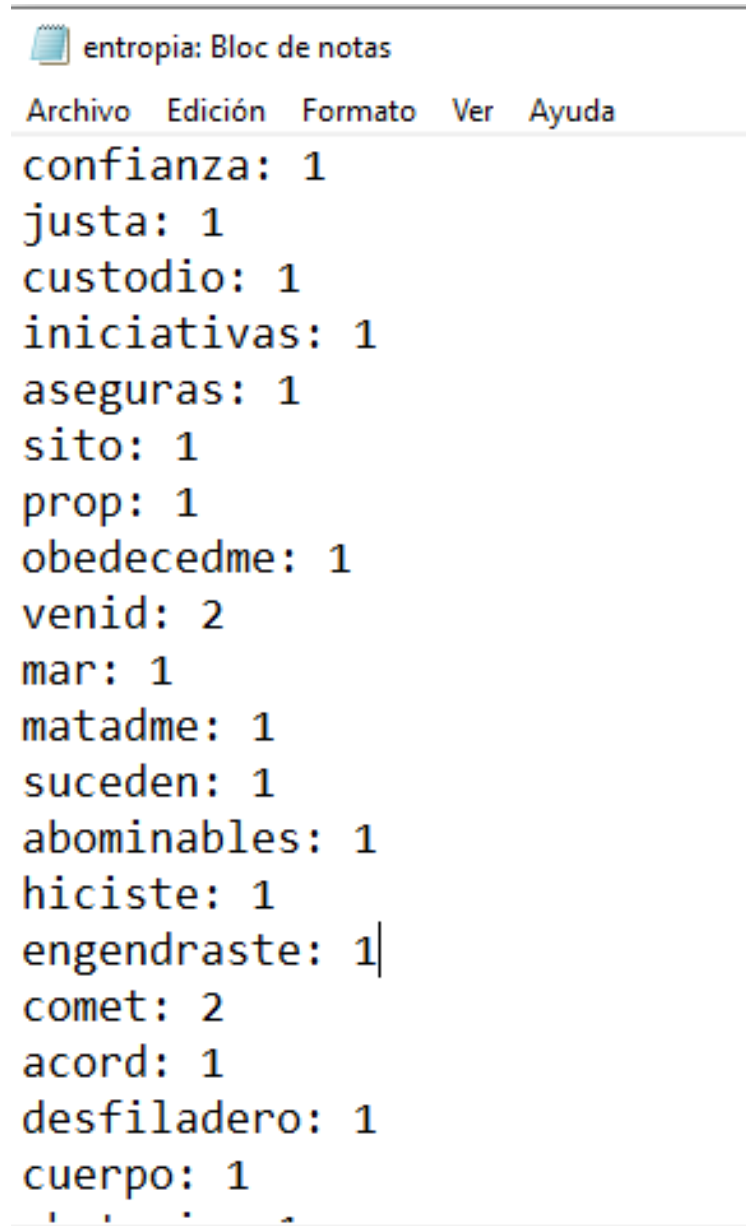


Figure 4: txt File.