



**Universidade do Porto**

**Faculdade de Engenharia**

**FEUP**

**Mestrado Integrado em Engenharia Informática e Computação**

**2º Ano – 2º Semestre**

**2011/2012**

**Relatório do Projecto – Tema 2**

**Compressão de Ficheiros**

**Autores:**

Cristiano Carvalheiro – 100509041 – [ei10041@fe.up.pt](mailto:ei10041@fe.up.pt)  
Daniel Santos Teixeira – 100509067 – [ei10067@fe.up.pt](mailto:ei10067@fe.up.pt)  
Luís Guilherme Martins – 100509105 – [ei10105@fe.up.pt](mailto:ei10105@fe.up.pt)

**Grupo 03 Turma 02**

**Porto, 28 de Maio de 2012**

## 1.Introdução

Este projecto foi realizado no âmbito da unidade curricular de Concepção e Análise de Algoritmos incluída no 2º semestre do 2º ano do Curso Mestrado Integrado em Engenharia Informática e Computação e consiste no desenvolvimento de uma aplicação em C++, que nos permita fazer a compressão de ficheiros de texto (T2).

Neste trabalho, pretendeu-se desenvolver uma aplicação em C++ que seja capaz de fazer a compressão de ficheiros de uma forma eficiente. Assim sendo foram utilizados os algoritmos aconselhados, sendo eles o algoritmo de Huffman e o algoritmo de Lempel-Ziv-Welch(LZW). Estes dois algoritmos servem para comprimir qualquer tipo de ficheiro média, no entanto neste projecto são limitados, como pedido, à compressão de ficheiros de texto.

A aplicação permite carregar um ficheiro de texto e depois comprimi-lo, utilizando os dois algoritmos e no final mostra os resultados de compressão, avaliando o tamanho do ficheiro carregado e o tamanho dos ficheiros comprimidos e calculando também os respectivos rácios de compressão para cada algoritmo. Os algoritmos são capazes também de proceder à descompressão de forma a avaliar se o conteúdo final é igual ao inicial.



## 2. Descrição da Solução

Esta aplicação é capaz de ler qualquer ficheiro de texto (.txt) e comprimi-la de forma a ocupar menos espaço no disco e fazer o inverso, ou seja, a descompressão do ficheiro comprimido.

Ao contrário do que era pedido no enunciado do projecto para a definição do ficheiro de leitura, decidiu-se apenas poder ser este definido em tempo de execução no primeiro menu, uma vez que se achou bastante mais intuitivo do ponto de vista do utilizador e também mais fácil ao nível de implementação. Assim o utilizador pode carregar um ficheiro de texto e apenas quando existir um ficheiro de texto válido carregado é possível proceder à compressão do mesmo.

### Carregar Ficheiro

A definição do ficheiro a ser comprimido dá-se em tempo de execução. O utilizador escreve o nome do ficheiro de texto, e a aplicação verifica se o ficheiro existe e se é possível ser aberto. Só depois de existir um ficheiro válido carregado é possível iniciar a compressão.

### Iniciar Compressão

A compressão do ficheiro carregado é feita através de dois algoritmos, o algoritmo de Huffman e o algoritmo LZW. Além das diferenças normais por serem dois algoritmos diferentes, nós decidimos também testar outra maneira de comprimir o ficheiro, comprimindo-o para um ficheiro de texto.

O algoritmo de Huffman começa por fazer uma tabela de frequência de todos os caracteres existentes, de seguida constrói uma árvore binária ordenando os caracteres e as respectivas frequências sendo as folhas os caracteres com menos frequência e o nós superiores os caracteres com mais frequência até à raiz que contem apenas a soma de todas as frequências. Depois de construída a árvore binária obtêm-se a codificação de cada carácter em binário. Assim sendo escrevemos isto no ficheiro a comprimir. Aqui que decidimos ver os resultados se comprimindo em ficheiro para um ficheiro de texto. A descompressão dá-se invertendo-se os passos até chegarmos ao estado inicial.

O algoritmo LZW funciona tanto melhor quanto maior for o ficheiro e quando mais repetitivo for o texto nele contido, no entanto quando maior for maior é o tempo necessário para proceder à compressão. Isto porque, o algoritmo funciona principalmente analisando a repetição de sequências, isto é, a cada carácter diferente que apa-



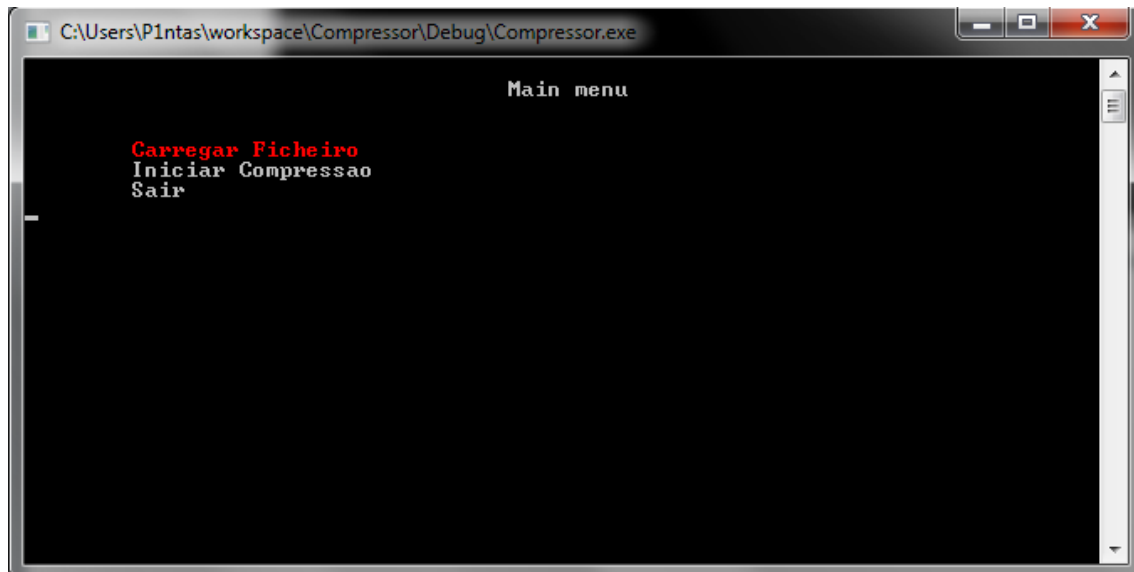
rece é associado um valor de 0 a 255 que corresponde aos 8 bits do carácter e este é adicionado ao dicionário de caracteres/sequências já encontradas, os valores de 256 a 4095 servem para as sequências seguintes que vão sendo encontradas à medida que o ficheiro vai sendo encriptado. A cada passo na compressão, os caracteres são agrupados e desta forma obtemos uma sequência, se essa sequência ainda não tiver aparecido é adicionada ao dicionário e associada a um código. Depois de ter todo o texto codificado, envia-se para um ficheiro, desta vez em formato .bin que é o ficheiro comprimido. Para descomprimir, lê-se o ficheiro e através do dicionário que já é dado constrói-se o texto.



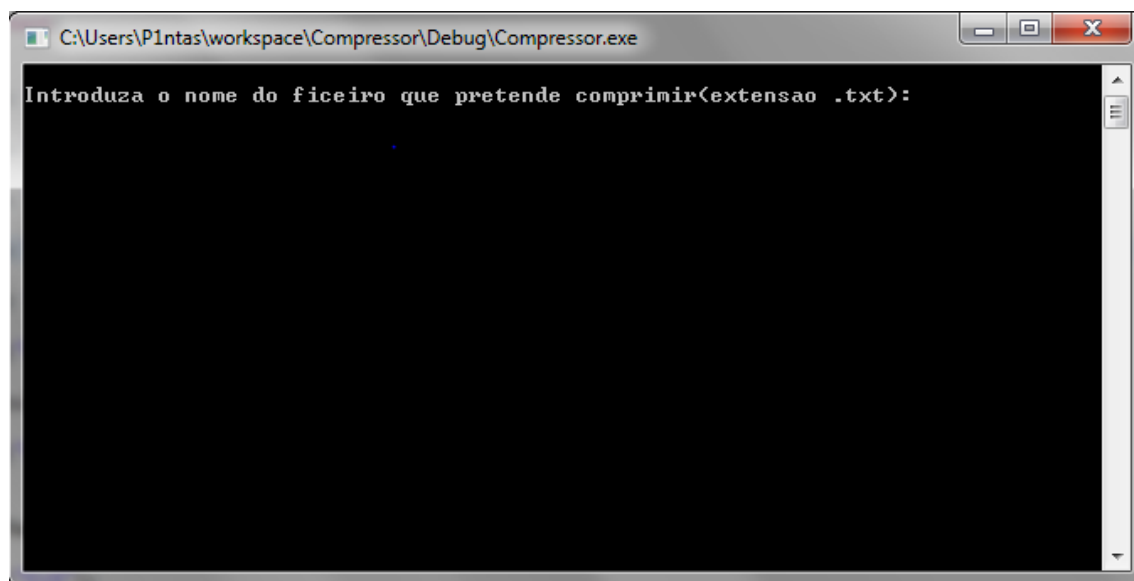
### 3.Casos de Utilização

Assim que o programa é iniciado aparece o menu principal. Antes de iniciar a compressão é obrigatório carregar um ficheiro válido, caso contrário a aplicação não vai deixar comprimir.

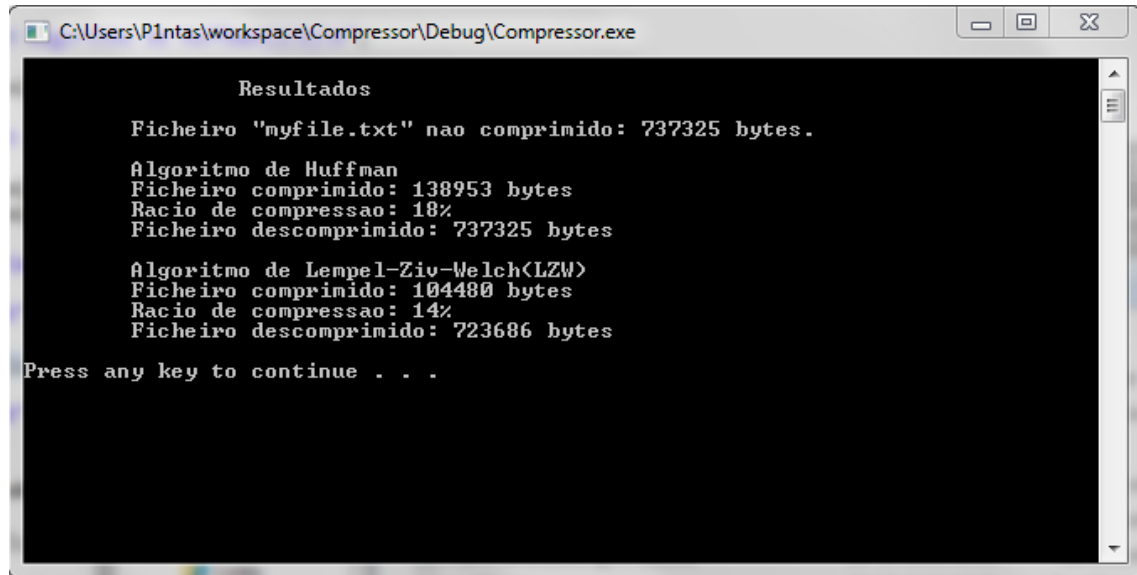
O utilizador tem três opções no menu principal:



1º - Carregar ficheiro – ao escolher a primeira opção carregar o ficheiro a comprimir. Se o ficheiro não existir ou não for válido. Mostra uma mensagem de erro e retorna ao menu principal.



2º - Iniciar Compressão – Inicia a compressão do ficheiro, cria 4 novos ficheiros 2 de compressão e 2 de descompressão para cada algoritmo. E no final de realizada esta tarefa mostra os resultados, mostrando os tamanhos dos vários ficheiros e os rácios de compressão para cada algoritmo.



```
Resultados

Ficheiro "myfile.txt" nao comprimido: 737325 bytes.

Algoritmo de Huffman
Ficheiro comprimido: 138953 bytes
Racio de compressao: 18%
Ficheiro descomprimido: 737325 bytes

Algoritmo de Lempel-Ziv-Welch(LZW)
Ficheiro comprimido: 104480 bytes
Racio de compressao: 14%
Ficheiro descomprimido: 723686 bytes

Press any key to continue . . .
```

3º - Sair – Sair da aplicação



#### 4. Dificuldades encontradas no desenvolvimento do trabalho

Uma das principais dificuldades deste projecto, foi a implementação dos algoritmos, embora tivessem sido fáceis de perceber, a sua implementação era um pouco mais rebuscada. Também a utilização de leitura e escrita em binário foi um pouco mais difícil na medida em que nunca nenhum dos elementos do grupo tinha trabalhado com esse tipo de ficheiros.



## 5. Conclusão

Uma vez concluído o projecto, podemos constatar que os objectivos foram cumpridos na sua maioria. O grupo está satisfeito com a sua prestação uma vez que, conseguimos chegar à solução, conseguimos analisar casos não pedidos e produzimos resultados eficientes e resultados menos normais de forma que podemos discutir o porquê, como foi o caso de usarmos compressão em ficheiro de texto no algoritmo de Huffman.

Como é possível observar na tabela 1 o algoritmo LZW produz resultados mais eficientes do que o algoritmo de Huffman quando os ficheiros são maiores e tem texto e quando o texto se tende a repetir.

O algoritmo de Huffman é um pouco mais limitado quando o ficheiro a comprimir é muito grande, no entanto os resultados obtidos com este algoritmo são resultado da compressão para ficheiro de texto, contudo conseguimos mostrar que utilizando este algoritmo é possível comprimir ficheiros mantendo o mesmo formato. Este método também faz com que para ficheiros de tamanho menor produza resultados extremamente ineficientes, chegando mesmo a produzir um ficheiro de compressão que ocupa mais espaço que o ficheiro original.

Concluimos também que o tempo de execução do algoritmo LZW é bastante influenciado pelo tamanho do ficheiro a comprimir.

Tamanho	Algoritmo de compressão:			
	Lempel-Ziv-Welch		Huffman	
65 B	53 B	81%	1052 B	1618%
804 B	636 B	79%	1367 B	170%
14802 B	11710 B	79%	7322 B	49%
561600 B	21583 B	3%	231875 B	41%
1123200 B	31078 B	2%	462725 B	41%
8985600 B	90354 B	1%	3694625 B	41%

Tabela 1 - Tabela de comparação dos algoritmos de compressão





## 7. Bibliografia

Translore, "My Huffman Solution! Critique My Code Please!", 2009

<http://www.cplusplus.com/forum/general/10291/> – acedido em 24-05-2012

WikiPedia, "LZW", 2011

<http://pt.wikipedia.org/wiki/LZW> – acedido em 24-05-2012

Codecogs, "LZW", 2010

<http://www.codecogs.com/code/computing/io/compression/lzw.php> – acedido em 24-05-2012

Rosettacode.org, "LZW compression", 2012

[http://rosettacode.org/wiki/LZW\\_compression#C.2B.2B](http://rosettacode.org/wiki/LZW_compression#C.2B.2B) – acedido em 24-05-2012

Literateprograms.org, "Huffman coding (C Plus Plus)", 2009

[http://en.literateprograms.org/Huffman\\_coding\\_%28C\\_Plus\\_Plus%29](http://en.literateprograms.org/Huffman_coding_%28C_Plus_Plus%29) – acedido em 26-05-2012

FB36, "Huffman Data Compression (C++ recipe) ", 2010

<http://code.activestate.com/recipes/577480-huffman-data-compression/> – acedido em 27-05-2012

