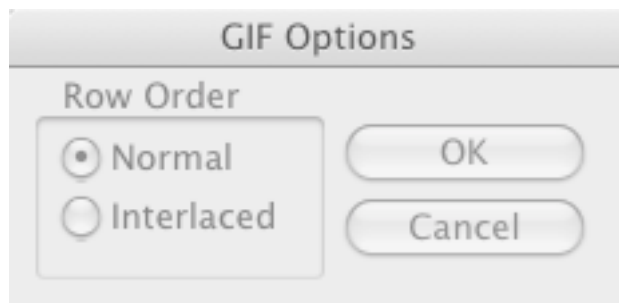


Teoria da Informação

Trabalho Prático nº 2

Codificação LZW e Formato de Imagem GIF



Introdução

Período de execução: 4 aulas práticas laboratoriais

Formato de Entrega:

Meta 1: entrega do código referente à implementação até ao Image Block Header (ver descrição);

Meta 2: entrega final (código completo – não é necessário relatório)

Prazo de Entrega:

Meta 1: 7 de Novembro, 23h59

Meta2: 28 de Novembro, 23h59

Esforço extra aulas previsto: 15h/aluno

Objectivo: Pretende-se que o aluno adquira sensibilidade para a codificação por dicionário dinâmico e para um dos formatos de imagem mais explorados em Web – o formato GIF.

Trabalho Prático

Hoje em dia os formatos de imagem lossless, como sejam os formatos GIF e PNG, de maior aplicabilidade em contextos Web baseiam-se em codificadores por dicionários dinâmicos. O presente trabalho pretende sedimentar os conceitos teóricos relacionados com esta metodologia de codificação, recorrendo para o efeito a um dos algoritmo de dicionário adaptativo mais evoluídos – o algoritmo Lempel-Ziv-Welch (**LZW**). Em particular, é objectivo deste trabalho a implementação de um codificador **GIF** na sua versão **87a**. Para o efeito, o aluno deverá desenvolver um programa na **linguagem C** que permita receba um ficheiro de imagem no formato JPG ou PNG e o converta em formato GIF 87a, armazenando-o em disco.

A. Preparação

1. Leitura dos documentos de apoio ao trabalho prático:

- **(Doc1):** Slides de explicação do formato GIF.
- **(Doc2):** Especificação do formato GIF.
- **(Doc3):** exemplificação da criação do ficheiro .gif a gerar para a imagem Mónica.png

2. Ser-lhe-á fornecido código fonte como base de trabalho, o qual permite efectuar a abertura de imagens e a geração dos header do GIF 87ª. Deverá estudar as funcionalidades implementadas. É objectivo do trabalho que estenda este código por forma a incluir as restantes etapas necessárias à implementação do codificador GIF..

B. Implementação do codificador:

1ª meta:

Implementar a função **writelnImageBlockHeader** (ver ficheiro GIFencoder.c), para escrita do image block header em ficheiro.

2ª meta:

Implementar a função **LZWCompress** (ver ficheiro GIFencoder.c), para compressão LZW dos dados da imagem.

C. Testes

Teste o seu programa com vários ficheiros de imagem diferentes (jpg, bmp, png, etc.). Tenha em atenção de os ficheiros originais não podem conter mais do que 256 cores distintas (número máximo suportado pelo GIF).

D. Descrição Geral

O algoritmo LZW padrão (na verdade no LZW do GIF são usadas algumas pequenas variantes) é o algoritmo de compressão aplicado no formato de imagem GIF que, embora proprietário, continua a ser o formato de imagem Web por excelência.

No formato GIF não existe uma estrutura geral de um data stream, mas antes várias estruturas de data stream alternativas que dependem não só da versão do formato, mas também da sua instanciação concreta. Contudo, os diversos formatos podem ser agrupados nas seguintes famílias:

GIF87a:

[GIF Header](#)
[Image Block](#)
Trailer

GIF89a:

[GIF Header](#)
[Graphic Control Extension](#)
[Image Block](#)
Trailer

GIF Animation

[GIF Header](#)
[Application Extension](#)
[
 [Graphic Control Extension](#)
 [Image Block](#)
]*
Trailer

GIF Header

Offset	Length	Contents
0	3 bytes	"GIF"
3	3 bytes	"87a" or "89a"
6	2 bytes	<Logical Screen Width>
8	2 bytes	<Logical Screen Height>
10	1 byte	bit 0: Global Color Table Flag (GCTF) bit 1..3: Color Resolution bit 4: Sort Flag to Global Color Table bit 5..7: Size of Global Color Table: $2^{(1+n)}$
11	1 byte	<Background Color Index>
12	1 byte	<Pixel Aspect Ratio>
13	? bytes	<Global Color Table(0..255 x 3 bytes) if GCTF is one>
	? bytes	< Blocks >
	1 bytes	<Trailer> (0x3b)

Image Block

Offset	Length	Contents
0	1 byte	Image Separator (0x2c)
1	2 bytes	Image Left Position
3	2 bytes	Image Top Position
5	2 bytes	Image Width
7	2 bytes	Image Height
8	1 byte	bit 0: Local Color Table Flag (LCTF) bit 1: Interlace Flag bit 2: Sort Flag bit 2..3: Reserved bit 4..7: Size of Local Color Table: $2^{(1+n)}$
	? bytes	Local Color Table(0..255 x 3 bytes) if LCTF is one
	1 byte	LZW Minimum Code Size

```
[ // Blocks
    1 byte    Block Size (s)
    (s)bytes  Image Data
]*
    1 byte    Block Terminator(0x00)
```

Graphic Control Extension Block

Offset	Length	Contents
0	1 byte	Extension Introducer (0x21)
1	1 byte	Graphic Control Label (0xf9)
2	1 byte	Block Size (0x04)
3	1 byte	bit 0..2: Reserved bit 3..5: Disposal Method bit 6: User Input Flag bit 7: Transparent Color Flag
4	2 bytes	Delay Time (1/100ths of a second)
6	1 byte	Transparent Color Index
7	1 byte	Block Terminator(0x00)

Comment Extension Block

Offset	Length	Contents
0	1 byte	Extension Introducer (0x21)
1	1 byte	Comment Label (0xfe)
[
	1 byte	Block Size (s)
	(s)bytes	Comment Data
]*		
	1 byte	Block Terminator(0x00)

Plain Text Extension Block

Offset	Length	Contents
0	1 byte	Extension Introducer (0x21)
1	1 byte	Plain Text Label (0x01)
2	1 byte	Block Size (0x0c)
3	2 bytes	Text Grid Left Position
5	2 bytes	Text Grid Top Position
7	2 bytes	Text Grid Width
9	2 bytes	Text Grid Height
10	1 byte	Character Cell Width(
11	1 byte	Character Cell Height
12	1 byte	Text Foreground Color Index(
13	1 byte	Text Background Color Index(
[
	1 byte	Block Size (s)
	(s)bytes	Plain Text Data
]*		
	1 byte	Block Terminator(0x00)

Application Extension Block

Offset	Length	Contents
0	1 byte	Extension Introducer (0x21)
1	1 byte	Application Label (0xff)
2	1 byte	Block Size (0x0b)
3	8 bytes	Application Identifier
[
	1 byte	Block Size (s)
	(s)bytes	Application Data
]*		
	1 byte	Block Terminator(0x00)

Chama-se a atenção para o facto da codificação de uma data stream GIF ser sempre realizada em little-endian.

Relativamente à formatação dos campos e à sua interpretação remete-se o leitor para a norma do formato GIF que se anexa ao presente enunciado e que poderá ser encontrada em <http://astronomy.swin.edu.au/~pbourke/dataformats/gif/>. Nesse documento é particularmente útil o anexo F em que são introduzidas as diferenças do algoritmo LZW implementado.