CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76168 VICTOR BRONDANI DOS SANTOS (76168);
LUIZ MIGUEL OLINEK (76106);
24FRO302 - 1 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage01.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

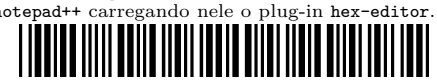
O tamanho da sub-imagem número 3 é

largura	altura

O tamanho da sub-imagem número 5 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



302-76168 - ga/ a

CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76087 LAURA PACHECO DELATTRE (76087);
CARLOS EDUARDO ANTAO (76001);
24FRO302 - 2 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage02.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

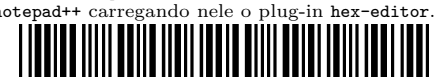
O tamanho da sub-imagem número 1 é

largura	altura

O tamanho da sub-imagem número 7 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



302-76087 - ga/ a

CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76175 VICTOR HUGO DOS SANTOS DE C(76175);
FELIPE EIJI KANO (76032);
24FRO302 - 3 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage03.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

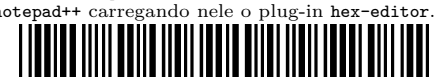
O tamanho da sub-imagem número 5 é

largura	altura

O tamanho da sub-imagem número 9 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



302-76175 - ga/ a

CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76056 GABRIELA BERNARDO PASQUALOT(76056);
DANILO DEPETRIS SOARES (76199);
24FRO302 - 4 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage04.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

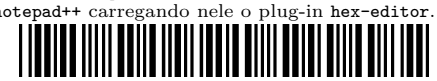
O tamanho da sub-imagem número 2 é

largura	altura

O tamanho da sub-imagem número 6 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



302-76056 - ga/ a

CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76063 GUILHERME VITOR DA SILVA (76063);
EDUARDO CAVALHEIRO DIAS (76018);
24FRO302 - 5 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage05.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

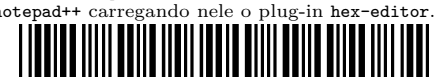
O tamanho da sub-imagem número 1 é

largura	altura

O tamanho da sub-imagem número 3 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



302-76063 - ga/ a

CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76113 NICOLE DA SILVA DE AZEVEDO (76113);
VITOR EDUARDO DE GOES FONTE(76182);
24FRO302 - 6 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage06.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

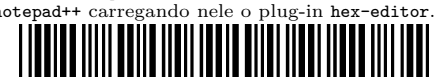
O tamanho da sub-imagem número 1 é

largura	altura

O tamanho da sub-imagem número 2 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



302-76113 - ga/ a

CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76049 FELIPE JESMIEL LEITE (76049);
ERIC CORDEIRO (76025);
24FRO302 - 7 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage07.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

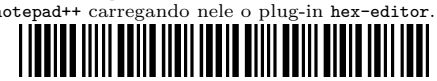
O tamanho da sub-imagem número 4 é

largura	altura

O tamanho da sub-imagem número 7 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76137 PEDRO HENRIQUE PIEKARSKI DE(76137);
RAUL KNAPKI CUNHA (76144);
24FRO302 - 8 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage08.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

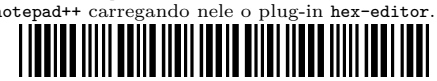
O tamanho da sub-imagem número 1 é

largura	altura

O tamanho da sub-imagem número 2 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.



302-76137 - ga/ a

CEP-UFPR-UTFPR-PUCPr-UP Desenvolvimento de Sistemas 17/06/2024 - 17:08:52.7
Imagens Digitais Prof Dr P Kantek (pkantek@gmail.com)
Analisando um arquivo imagem GIF vivxh81a V: 3.04
76120 PAULO JOSE FREIRE CORREA (76120);
24FRO302 - 9 entregar ate 4/julho _____ / _____ / _____

Uma imagem GIF

Primeiro uma correção: o padrão GIF não é exatamente uma imagem, ou não é apenas uma imagem. Ele, na verdade, descreve um fluxo de informações gráficas entre dois dispositivos e como tais informações deverão ser interpretadas no destino. Este formato foi um copyright da Compuserve e é apenas esta entidade que tinha autoridade para alterar o padrão.

Em 1995 expirou a patente do algoritmo LZW, tornando o padrão GIF livre para uso. Com isto o padrão se popularizou e é amplamente suportado.

Baseado nele, nasceu o padrão PNG, aberto desde 1994. Embora parecidos têm diferenças: o PNG não tem perdas e o GIF pode ter. Com isto, o tamanho do PNG pode ser bem maior. O PNG não tem suporte para animação.

Talvez a principal vantagem do GIF seja a de comprimir imagens. A consequência imediata é que os arquivos reduzem de tamanho em termos de espaço, mas sobretudo em termos de largura de banda durante sua transmissão. Este é o grande calcanhar de Aquiles deste mundo tão baseado na Internet. Note que o que se faz aqui é barganhar processamento (que é maior) versus espaço durante a transmissão, que é menor.

O algoritmo de compressão é o LZW (Lempel-Ziv e Welch). É um algoritmo muito bom, que usa um dicionário dos strings – de tamanho variável – que aparecem no objeto binário sendo comprimido. Quando estes strings aparecem e reaparecem, o compressor envia apenas sua identificação no dicionário, o que de fato comprime o fluxo. O dicionário é construído e atualizado em tempo real pelos dois parceiros (o compressor e o descompressor) o que aumenta a necessidade de processamento, mas diminui o fluxo.

Este algoritmo tem uma característica interessante: ele nasceu como um trabalho de dois cientistas: o Lempel e o Ziv. Eles publicaram o algoritmo LZ. Uns dois anos depois, o Welch descobriu uma pequena inexatidão no algoritmo original e propôs uma correção: foi criado aí o LZW.

O padrão GIF

O padrão GIF é definido em termos de blocos e subblocos que compõe o fluxo. Espera-se que todas as imagens que estão em um único fluxo compartilhem algum tipo de característica, caso contrário é mais negócio definí-las em fluxos separados.

Existem versões do padrão (a 87 e a 89). Tal referência aparece no cabeçalho do fluxo e determina o conjunto mínimo de capacidades requeridas pelo programa decodificador para que este possa mostrar a imagem.

Sub-blocos de dados

Os sub blocos de dados são unidades contendo dados. Eles não tem um label, começam sempre por um byte de tamanho (que exclui este byte de tamanho) e que pode variar entre 0 e 255. Terminam por um terminador contendo zero binário: 0x00.

Header

Formado por 6 bytes, sendo os 3 primeiros iguais a GIF e os outros 3 contendo a versão usada para formatar o fluxo. O padrão recomenda que não se assuma que um arquivo que tenha um GIF nestas posições seja de fato um GIF.

```
Header: 6 bytes (obrigatorio)
0-2: GIF
3-5: 87a ou 89a
```

Descritor de tela logica: 7 bytes (obrigatório, imediatamente após o header)

```
0-1: largura da imagem em pixels (max=65536)
2-3: altura da imagem (max=idem)
4,5 e 6: Não interessam para este exercício. Se tiver curiosidade
procure o padrão GIF, em ...
```

Descritor de imagem

Tem um tamanho de 10 bytes (obrigatorio ao menos 1) e pode ser localizado buscando seu identificador que é `2C`. É claro que os desenhadores GIF não fazem esta busca aleatória, eles caminham pelos blocos suportados pelos tamanhos de cada coisa no arquivo. Mas para simplificar este exercício, pode procurar o `2C`. Mas, tenha em conta que ao mostrar uma imagem comprimida, pela própria essência da compressão (que é diminuir a redundância), pode muito bem aparecer o byte `2C` dentro dos dados comprimidos. Para resolver este problema, ou se trabalha rigorosamente com os tamanhos, ou se exerce alguma criatividade na análise do bloco de controle. Nós aqui, vamos seguir esta segunda linha. Não é muito difícil, é só ter algum jogo de cintura. A seguir a descrição do bloco da imagem:

```
byte 0: 2C
1-2: deslocamento lateral desta imagem
3-4: deslocamento vertical ...
5-6: largura desta imagem
7-8: altura desta imagem
byte 9: específico para lidar com a tabela de cores
        Não interessa para este exercício. A seguir vêm blocos
        dados, de tamanho variável (mas grande).
```

Outros Blocos

Também devem ser localizados pelos seus descritores:

```
21F9 = bloco de controle
21FE = bloco de comentario
2101 = bloco de texto plano
21FF = bloco de aplicacao
3B = terminador de fluxo
```

O que nos interessa para este exercício são os blocos de controle. Fundamentalmente são usados em animações (é o caso aqui).

```
21F9 - Bloco de controle:
byte 0: tamanho
byte 1-bits 0-2: reservado (zeros)
       bits 3-5: metodo de substituicao (0=nada, 1=nao ha substituicao;
       2=area retornada a cor de background, 3=voltar ao que havia antes;
       4 a 7:uso futuro)
       bit 6: 1=aguardar acao do operador, 0=nao aguardar
       bit 7: 0=nao ha cor transparente 1=ha cor
byte 2-3: retardo em centesimos de segundo
byte 4: cor transparente (quando ha)
byte 5: 00 (terminador)
```

☞ Para você fazer

Examine o arquivo GIF gimage09.gif

Localize os blocos de controle e responda: 4 primeiros números do vetor de retardos:

O tamanho da sub-imagem número 1 é

largura	altura

O tamanho da sub-imagem número 2 é

largura	altura

Uma boa ferramenta para examinar o conteúdo hexadecimal do arquivo é o site `http://www.onlinehexeditor.com/`. Outra possibilidade é usar o `notepad++` carregando nele o plug-in `hex-editor`.