Princípios de Programação

Projecto

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2018/2019

A edição de imagem é uma das utilizações mais antigas dos computadores. Por exemplo, o Adobe Photoshop já existe há 28 anos! Neste trabalho vamos fazer um programa para manipular imagens no formato PPM.

O formato PPM

PPM (*portable pixmap format*) é um formato para guardar imagens. Podem encontrar uma descrição mais completa do formato PPM na Wikipédia[1]. Neste trabalho consideramos apenas a variante P3, aquela em que a imagem é representada no formato exemplificado abaixo.

```
P3
# Comentario
3 1 255 255 0 0 0 255
0   0 0 255
```

Eis as caraterísticas principais do formato PPM:

- As linhas que começam por cardinal devem ser ignoradas porque são comentários.
- O ficheiro começa com a string `P3`. Apesar de existirem outras variantes, vamos considerar apenas a P3 neste trabalho.
- O resto do ficheiro tem apenas números inteiros separados por espaços ou mudanças de linha. Não há diferença prática entre mudanças de linha e espaços (excepto para os comentários), bem como não há diferença entre um ou mais espaços ou mudanças de linha.

[1] `https://en.wikipedia.org/wiki/Netpbm_format`.

- Os dois primeiros números inteiros (`3` e `1` no exemplo) representam a largura e altura da imagem em pixeis.
- O terceiro inteiro (`255` no exemplo) representa o valor máximo para cada cor.
- O restante do ficheiro descreve os pixeis da imagem. Neste caso temos um pixel vermelho (`255 0 0`), um verde (`0 255 0`) e um azul (`0 0 255`).

Manipulação de imagem

Na sua utilização mais simples, o programa deve ler um ficheiro PPM e escrever outro igual. Os nomes dos ficheiros de entrada e de saída são lidos da consola. Por exemplo, se `lena.ppm` for o ficheiro na figura 1a, o programa deverá deixar no ficheiro `lena2.ppm` uma cópia de `lena.ppm`.[2]
Eis um exemplo de utilização do vosso programa:

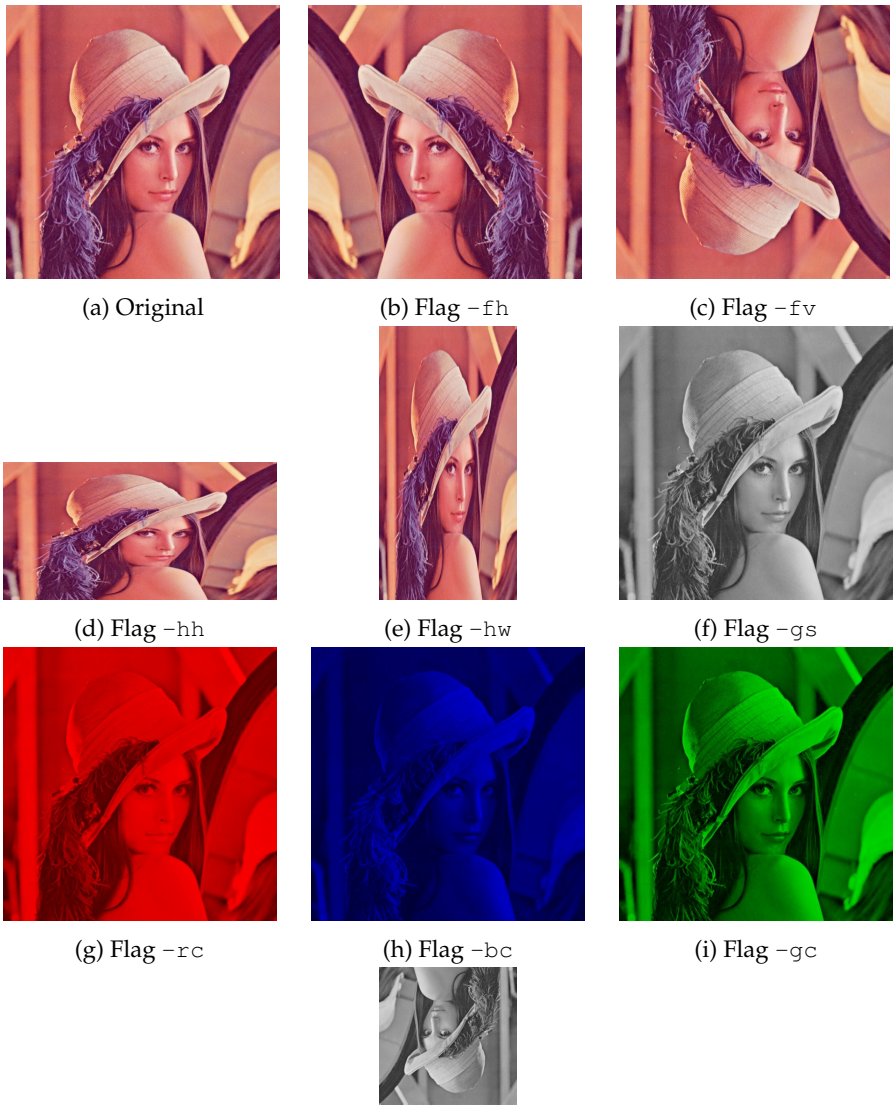
```
$ ghc --make p_fc00001_fc00002.hs -o photochop
[1 of 1] Compiling Main ( p_fc00001_fc00002.hs, p_fc00001_fc00002.o )
Linking photochop ...
$ ./photochop lena.ppm lena2.ppm
```

O uso acima é, fundamentalmente, uma simples cópia de ficheiro. O objectivo deste trabalho é bem mais ambicioso! O vosso programa deverá suportar os seguintes modificadores (*flags*), resultando nas imagens constantes na figura 1.

- Inversão (*flip*) horizontal:
  `$ ./photochop lena.ppm lena2.ppm -fh`
- Inversão (*flip*) vertical: `$ ./photochop lena.ppm lena2.ppm -fv`
- Metade da largura: `$ ./photochop lena.ppm lena2.ppm -hw`
- Metade da altura: `$ ./photochop lena.ppm lena2.ppm -hh`
- Escala de cinzentos: `$ ./photochop lena.ppm lena2.ppm -gs`
- Vermelhos apenas (R): `$ ./photochop lena.ppm lena2.ppm -rc`
- Verdes apenas (G): `$ ./photochop lena.ppm lena2.ppm -gc`
- Azuis apenas (B): `$ ./photochop lena.ppm lena2.ppm -bc`

Nas operações em que se perdem pixeis (metade da largura e metade da altura) deve ser feita a média dos elementos a combinar. As operações podem ser combinadas ou repetidas. Por exemplo,

[2] A Lena é a imagem padrão usada em processamento de imagem, como que um Hello World. Mais informação em `https://en.wikipedia.org/wiki/Lenna`.



(a) Original (b) Flag `-fh` (c) Flag `-fv`

(d) Flag `-hh` (e) Flag `-hw` (f) Flag `-gs`

(g) Flag `-rc` (h) Flag `-bc` (i) Flag `-gc`

(j) Flags `-fh -fv -hw -hh -gs`

Figura 1: Resultados da utilização de diferentes flags.

```
$ ./photochop lena.ppm lena2.ppm -hh -hw -hw -hh
```

deverá resultar numa imagem com $1/4$ da largura e $1/4$ da altura da imagem original. Já a utilização dos modificadores `-fh -fv -hh -hw -gs` deverá devolver a imagem com $1/4$ da área, invertida horizontal e verticalmente, bem como em escala de cinzentos (ver figura 1j). De notar que algumas

operações anulam-se. Por exemplo, a sequência de modificadores `-fh -fh -fv -fh -fv -fh` produz a imagem original.

Testes

Seguindo as boas práticas da linguagem Haskell, deverá separar a parte **`IO`** da parte pura. A parte **`IO`** deverá estar restrita ao menor número possível de funções. Esta estratégia permite facilmente testar as operações mais complexas do seu programa, operações essas que não devem usar **`IO`**. Para tal, deverá usar o módulo `QuickCheck` para escrever propriedades. Deverá escrever 3 propriedades que considere importantes. Eis algumas sugestões:

- Uma imagem invertida duas vezes é a imagem original.
- Qualquer imagem tem um número de pixeis igual ao produto das dimensões no cabeçalho da imagem.
- Nenhum dos valores dos pixeis são superiores ao valor máximo apresentado no cabeçalho.
- Uma operação de redução de largura seguida de uma de redução de altura mantem o rácio largura/altura inalterado.

Utilize a função `quickCheck` para preparar pelo menos três testes. Os testes são acionados através do modificador `-t`. Eis um exemplo:

```
$ ./photochop -t
+++ OK, passed 100 tests.
+++ OK, passed 100 tests.
+++ OK, passed 100 tests.
```

Notas

1. Prepare um novo tipo de dados para representar um ficheiro PPM em memória. Torne este tipo de dados instância da classe `Arbitrary`, de modo conseguir utilizar o módulo `QuickCheck`.
2. Assuma que o ficheiro de entrada está bem formado, isto é, que representa uma imagem PPM. O vosso trabalho não precisa de verificar a boa formação do ficheiro, bem como não precisa de "resistir" a ficheiros mal formados.
3. Os trabalhos serão avaliados automaticamente. Respeite o uso das flags.
4. Não se esqueça de juntar uma assinatura para cada função que escrever.
5. Lembre-se que as boas práticas de programação Haskell apontam para a utilização de várias funções simples em lugar de uma função única mas complicada.

Entrega. Este é um trabalho de resolução em grupos de dois ou três alunos. Os trabalhos devem ser entregues no Moodle até às 23:55 do dia 12 de dezembro de 2018.

Ética Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. Lembre-se: "Alunos detetados em situação de fraude ou plágio, plagiadores e plagiados, ficam reprovados à disciplina (sem prejuízo de ser acionado processo disciplinar concomitante)".