INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados 2014/2015 - 1º Semestre

Enunciado do 1º Projecto

Data de entrega: 3 de Novembro de 2014 às 22h00

1 Introdução

O formato de imagem **PPM** (Portable Pixel Map[1]) foi concebido com o fim de facilitar a portabilidade de imagens entre plataformas. Permite representar imagens a cores de forma muito simples, ainda que de maneira assumidamente ineficiente devido por exemplo a não usar qualquer tipo de compressão. Na sua versão denominada "Plain PPM", esta simplicidade é levada ao extremo, sendo toda a informação representada por meio de caracteres ASCII. Em contrapartida, a escrita de programas que processa este formato é muito facilitada, enquanto que a representação das imagens propriamente ditas é de fácil análise uma vez que se tratam de ficheiros de texto (até podem ser enviadas por SMS!).

O formato PPM Baseia-se na representação de imagens digitais como uma matriz de pontos, denominados *píxeis* (do inglês pixel = picture element), cada um contendo informação relacionada com a visualização de uma unidade elementar da imagem numa dada localização. Neste formato, as componentes da informação associada a cada píxel consistem nas dimensões vermelha, verde e azul da cor do píxel no modelo de cores RGB. A Figura 1 ilustra a representação de uma imagem como uma matriz de píxeis, sendo a cor RGB de cada um deles representada por três inteiros dentro de uma gama.[2] Nesta imagem, três píxeis são ampliados, e são apresentadas as componentes RGB que definem a cor de cada um deles. Estas componentes estão sob a

[1] Um dos vários formatos de imagem usados e definidos pelo projecto Netpbm. Mais informações em: http://netpbm.sourceforge.net/

[2] Créditos da imagem: "Rgb-raster-image"by Gringer - Own work. Licensed under Creative Commons Zero, Public Domain Dedication via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Rgb-raster-image.svg#mediaviewer/File:Rgb-raster-image.svg

forma de uma percentagem; se o valor máximo da gama for por exemplo 255, os valores RGB seriam $(237, 237, 237)$, $(89, 89, 40)$ e $(229, 229, 0)$.

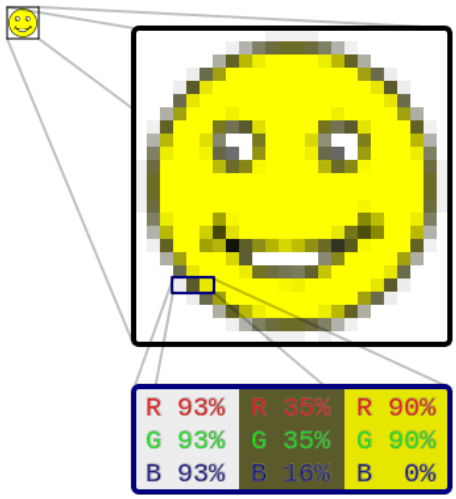
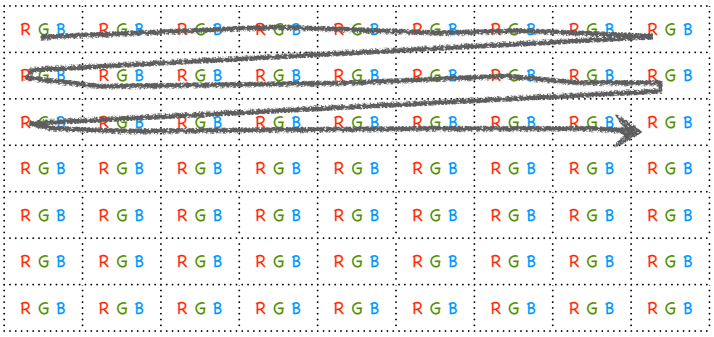


Figura 1: Representação de uma imagem por uma matriz de píxeis.

Na versão Plain PPM, cada ficheiro representa uma única imagem. O ficheiro contém puramente texto em ASCII, sendo todos os valores numéricos inteiros positivos representados em base decimal. A informação é estruturada em duas partes: *cabeçalho* e *matriz de píxeis*. No cabeçalho, todas as linhas que se iniciam com um caractere "#" até uma mudança de linha (inclusive) são comentários e são ignorados. Para além dos comentários, o cabeçalho tem o seguinte formato:

- A primeira linha inicia-se com a cadeia de caracteres "`P3`". Trata-se do designado "número mágico" que especifica o formato em questão.
- Seguem-se um ou mais caracteres brancos (espaços, tabs, mudanças de linha).
- Dois números separados por caracteres brancos. Indicam a largura e a altura da imagem, respectivamente o número de colunas seguido do de linhas de píxeis na imagem, que designaremos, respectivamente, por **C** e **L** (inteiros positivos).
- Caracteres brancos.
- O valor máximo de cada componente RGB, inferior a 65536 e superior a 0, que designaremos por **MAXcor** (inteiro positivo). Neste projecto consideraremos apenas valores para **MAXcor** inferiores a 256.
- Um único caractere branco (tipicamente uma mudança de linha), que assinala o fim do cabeçalho.

Segue-se imediatamente a matriz de píxeis, que consiste em e $3 \times \mathbf{C} \times \mathbf{L}$ números separados por caracteres brancos. Agrupados três a três, cada número representa uma das componentes RGB de um píxel da matriz, pela ordem vermelho, verde, azul. Os píxeis aparecem agrupados linha a linha da matriz, sendo as linhas ordenadas de cima para baixo, enquanto que os píxeis de cada linha são ordenados da esquerda para a direita, da seguinte forma: [3]



Consideremos o exemplo de um ficheiro "arcoiris.ppm", em formato Plain PPM:

```
P3
# arcoiris.ppm
5 3
255
255   0   0   255 127   0   255 255   0     0 255   0     0   0 255
255 127   0   255 255   0     0 255   0     0   0 255    75   0 130
255 255   0     0 255   0     0   0 255    75   0 130   127   0 255
```

A imagem representada encontra-se na Figura 2, ampliada de forma a que cada quadrado colorido corresponda a um único píxel. Os três valores RGB que definem a cor de cada quadrado aparecem sobrepostos na imagem.

2 Especificação do Programa

Neste projecto pretende-se desenvolver um programa que permita manipular imagens no formato Plain PPM. O programa deverá actuar como um filtro, recebendo como entrada um ficheiro PPM, e produzindo um novo ficheiro contendo uma representação da imagem processada, também em formato PPM.

O programa será chamado com uma de três opções (ver Tabela 1) como argumento na linha de comandos, especificando o tipo de processamento que ele deve efectuar. À medida que o programa lê o texto que representa uma imagem, deve processar essa informação de acordo com a opção escolhida, imprimir o texto correspondente à imagem modificada. Em cada execução do programa, ele recebe apenas uma imagem, e produz apenas uma imagem, terminando de seguida.

[3] Há uma regra adicional que **não** iremos ter em conta neste projecto por uma questão de simplicidade. No formato Plain PPM, o tamanho das linhas do ficheiro (i.e., sequência de caracteres terminada por uma mudança de linha ou fim de ficheiro) não pode exceder os 70 caracteres de comprimento.

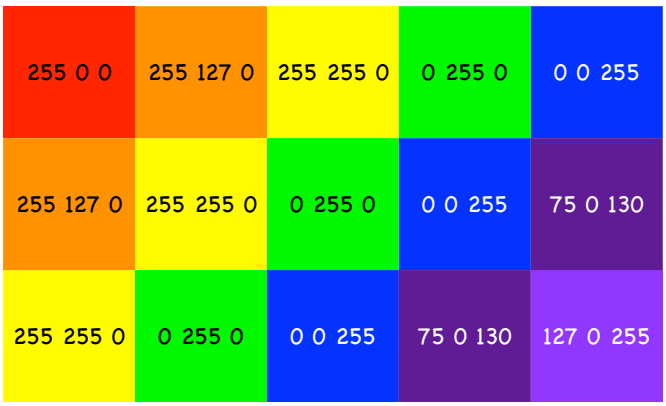


Figura 2: Imagem representada no ficheiro arcoiris.ppm.

`n`	obter a imagem com o negativo das cores
`e`	obter o espelho da imagem, segundo o eixo vertical mediano
`p M N`	obter padrão com `M` linhas e `N` colunas a partir da imagem dada

Tabela 1: Opções de funcionamento do programa.

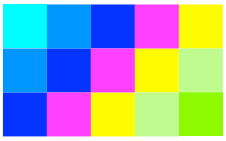
O programa deverá ler os dados de entrada a partir do *standard input*. A opções da Tabela 1 têm o seguinte significado:

Opção: `n`**.** A imagem é modificada de modo a que a cor de cada píxel na imagem de entrada seja substituída pelo seu negativo. Mais precisamente, o triplo (R, G, B) que é associado a cada píxel da imagem de entrada, deverá passar a ser (**MAXcor** $- R$, **MAXcor** $- G$, **MAXcor** $- B$). Os restantes dados (valores **C**, **L** e **MAXcor**) devem permanecer inalterados.

Por exemplo, aplicado ao ficheiro arcoiris.ppm este filtro produziria o seguinte texto de saída:

```
P3
# negativo
# arcoiris.ppm
5 3
255
  0 255 255     0 128 255     0   0 255   255   0 255   255 255   0
  0 128 255     0   0 255   255   0 255   255 255   0   180 255 125
  0   0 255   255   0 255   255 255   0   180 255 125   128 255   0
```

A imagem correspondente a este texto de saída seria então:



Opção: `e`**.** A imagem é modificada de modo a que a cor de cada píxel na imagem de entrada seja substituída pela cor do píxel na posição simétrica (horizontalmente) da mesma imagem (de entrada). Mais precisamente, a cor de cada píxel que se encontra na coluna i de uma dada linha da imagem de saída deverá passar a ser a cor do píxel que se encontrava na coluna $\mathbf{C} - i$ da mesma linha da imagem de entrada. Os restantes dados (valores **C**, **L** e **MAXcor**) devem permanecer inalterados.

Por exemplo, aplicado ao ficheiro arcoiris.ppm este filtro produziria o seguinte texto de saída:

```
P3
# espelho
# arcoiris.ppm
5 3
255
  0   0 255     0 255   0   255 255   0   255 127    0   255   0   0
 75   0 130     0   0 255     0 255   0   255 255    0   255 127   0
127   0 255    75   0 130     0   0 255     0 255    0   255 255   0
```

Neste caso, a imagem associada seria:

Opção: `p M N`**.** A imagem é construída formando `M` linhas de `N` repetições da imagem de entrada. Mais precisamente, a nova imagem terá `M` $\times$ **L** linhas e `N` $\times$ **C** colunas, sendo as componentes RGB de cada píxel na posição (i, j) da imagem nova dadas pelos valores RGB do píxel na posição (i mod `M`, j mod `N`) da imagem inicial. Os restantes dados (**MAXcor**) devem permanecer inalterados.

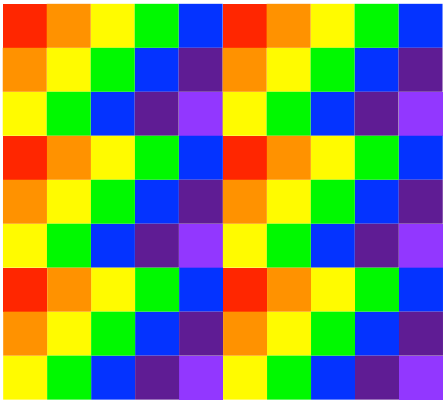
Por exemplo, aplicando o comando `p 3 2` ao ficheiro arcoiris.ppm este filtro produziria o seguinte texto de saída:[4]

```
P3
# padrao 3x2
# arcoiris.ppm
10 9
255
255   0   0   255 127   0   255 255   0     0 255   0     0   0 255↙
   255   0   0   255 127   0   255 255   0     0 255   0     0   0 255
255 127   0   255 255   0     0 255   0     0   0 255    75   0 130↙
```

[4]Note que, devido ao comprimento das linhas exceder o do papel, elas aparecem cortadas no ponto assinalado pela seta ↙ (este símbolo **não** pertence ao texto de saída), continuando na linha seguinte. Cada linha do ficheiro ocupa, portanto, duas linhas nesta representação do texto de saída.

```
  255 127   0   255 255   0     0 255   0     0   0 255    75   0 130
255 255   0     0 255   0     0   0 255    75   0 130   127   0 255↙
  255 255   0     0 255   0     0   0 255    75   0 130   127   0 255
255   0   0   255 127   0   255 255   0     0 255   0     0   0 255↙
  255   0   0   255 127   0   255 255   0     0 255   0     0   0 255
255 127   0   255 255   0     0 255   0     0   0 255    75   0 130↙
  255 127   0   255 255   0     0 255   0     0   0 255    75   0 130
255 255   0     0 255   0     0   0 255    75   0 130   127   0 255↙
  255 255   0     0 255   0     0   0 255    75   0 130   127   0 255
255   0   0   255 127   0   255 255   0     0 255   0     0   0 255↙
  255   0   0   255 127   0   255 255   0     0 255   0     0   0 255
255 127   0   255 255   0     0 255   0     0   0 255    75   0 130↙
  255 127   0   255 255   0     0 255   0     0   0 255    75   0 130
255 255   0     0 255   0     0   0 255    75   0 130   127   0 255↙
  255 255   0     0 255   0     0   0 255    75   0 130   127   0 255
```

A imagem correspondente seria:



O programa deve ser capaz de processar qualquer imagem que satisfaça as regras de formatação descritas acima nesta enunciado, com dimensões não superiores a 1400 linhas e 1400 colunas no caso da opção *padrão*, e com dimensões não superiores a 1400 colunas no caso da opção *espelho*. (No caso da opção *negativo*, a imagem poderá ter qualquer dimensão.) Pode, no entanto, assumir que os dados de entrada obedecem sempre a essas regras. Neste contexto, o programa não precisa de fazer validação de dados de entrada nem produzir qualquer mensagem de erro.

Note que, dada a indefinição em relação ao número e tipo de caracteres brancos que existem no ficheiro, bem como à possível existência de comentários, a mesma imagem pode ter múltiplas representações em formato PPM. No entanto, com o intuito de facilitar a automatização de testes aos projectos, o formato dos dados de saída será univocamente determinado pelos dados de entrada.

3 Dados de Saída

O programa deverá escrever no *standard output* o texto que define a nova imagem no formato PPM, usando as seguintes regras:

- O cabeçalho do ficheiro de saída deve preservar todos os caractres não brancos, com duas excepções:
  - Deve ser introduzida uma nova segunda linha contendo um comentário que indica qual a opção utilizada. Esta linha deve consistir precisamente numa das seguintes cadeias de caracteres: "`# negativo`", "`# espelho`" ou "`# padrao MxN`" (sendo `M` e `N` substituídos pelos valores correspondentes).
  - Os caracteres (não brancos) que definem as dimensões da imagem (valores **C** e **L**) devem ser substituídos pelos novos valores, diferindo apenas quando é escolhida a opção `p M N` com $\texttt{M}, \texttt{N} > 1$.

  Os únicos caracteres brancos existentes no cabeçalho devem ser os seguintes:
  - Uma mudança de linha a seguir à cadeia de caracteres `P3`.
  - Uma mudança de linha a seguir ao comentário que foi introduzido na segunda linha.
  - Aqueles existentes nos comentários que faziam parte do ficheiro de entrada.
  - Um espaço entre os caracteres (não brancos) que definem as dimensões da nova imagem (os novos valores **C** e **L**), e uma mudança de linha depois deles.
  - Uma mudança de linha depois dos caracteres (não brancos) que definem o valor máximo da gama de cores RGB (o valor **MAXcor**).
- As linhas que contêm os valores RGB da matriz de píxeis devem conter apenas os seguintes caracteres brancos:
  - Os valores RGB devem ocupar sempre o mesmo número de caracteres que o valor **MAXcor**, sendo preenchidos à esquerda por espaços caso necessário. Este efeito pode ser obtido utilizando o formato de escrita de inteiros "$\%*\texttt{d}$" da função `printf`.
  - Cada uma das três componentes RGB de um píxel devem ser separadas por um único espaço (para além dos necessários para satisfazer a regra do ponto anterior).
  - Os valores associados a cada píxel devem ser separados por 3 espaços.
  - Deve haver uma única mudança de linha no final dos valores que definem cada linha da matriz.
- Não deve existir mais nenhum caractere a seguir à última linha da matriz (para além da mudança de linha mencionada no ponto anterior).

4 Exemplos e Material de Apoio

Na página da disciplina, secção "Material de apoio – Projectos" são fornecidos a título de exemplo ficheiros PPM contendo diferentes imagens, bem como os correspondentes ficheiros de saída produzidos por cada uma das possíveis opções de funcionamento do programa. Estarão disponíveis ainda outros materiais de apoio na resolução do projecto, incluindo sugestões de aplicações para produzir e visualizar ficheiros PPM.

5 Compilação e Execução do Programa

O compilador a utilizar é o `gcc` com as seguintes opções de compilação: `-ansi -Wall -pedantic`.

Para compilar o programa deve executar o seguinte comando:

```
$ gcc -ansi -Wall -pedantic -o proj1 *.c
```

o qual deve ter como resultado a geração do ficheiro executável `proj1`, caso não haja erros de compilação. A execução deste comando não deverá escrever qualquer resultado no terminal.

Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso. Por exemplo, durante a compilação do programa, o compilador não deve escrever mensagens de aviso (*warnings*).

O programa deve ser executado da forma seguinte:

```
$ ./proj1 opção < in.ppm > out.ppm
```

Em que a palavra `opção` é substituída por uma das opções do programa especificadas no enunciado: `n`, `e` ou `p M N` (sendo `M` e `N` inteiros positivos).

6 Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. Instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.
- Efectue o upload de um ficheiro arquivo com extensão `.zip` que inclua os ficheiros fonte (`.c`) e (caso existam) cabeçalho (`.h`) que constituem o programa.

  Para criar um ficheiro arquivo com a extensão `.zip` deve executar o seguinte comando na directoria onde se encontram os ficheiros com extensão `.c` e `.h` (se for o caso), criados durante o desenvolvimento do projecto:

  ```
  $ zip proj1.zip *.c *.h
  ```

- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste.
- **O sistema não permite submissões com menos de 30 minutos de intervalo para o mesmo grupo.** Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projecto: **22h00 do dia 3 de Novembro de 2014**. Até à data limite poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a **última** entrega efectuada. Deverá portanto verificar cuidadosamente que a última entrega realizada corresponda à versão do projecto que pretende que seja avaliada.

7 Avaliação do Projecto

7.1 Componentes da Avaliação

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho da funcionalidade do programa realizado. Inclui-se a verificação de que o uso de memória e o tempo de execução não são excessivos. Esta componente é avaliada entre 0 e 16 valores.
2. A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: rigor dos comentários, indentação, estruturação, modularidade e abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída na discussão final do projecto.

Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado, sendo a respectiva classificação ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. **Elementos do grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.**

7.2 Atribuição Automática da Classificação

A classificação da primeira componente da avaliação do projecto é obtida automaticamente através de um conjunto de testes executados num computador com o sistema operativo **GNU/Linux**. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação podem incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que

pode utilizar e no tempo total disponível para execução, sendo estes limites distintos para cada teste.

Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.

Não será disponibilizada informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.

7.3 Detecção de Cópias

A avaliação dos projectos inclui a utilização de um sistema para detecção de situações de cópia entre projectos. A submissão de um projecto pressupõe o compromisso de honra que o trabalho incluso foi realizado única e exclusivamente pelos alunos referenciados nos ficheiros submetidos para avaliação. A quebra deste compromisso, i.e. qualquer situação de fraude ou facilitação de fraude, terá como consequência a reprovação à disciplina de IAED neste semestre, assim como a comunicação da ocorrência ao respectivo Coordenador de curso e ao Conselho Pedagógico do IST para sanções adicionais de acordo com as regras aprovadas pela UTL e publicadas em Diário da República.

7.4 Considerações adicionais

Todos os programas são avaliados do modo seguinte:

```
$ ./proj1 [opção] < in.ppm > out.ppm; diff out.ppm exp.ppm
```

em que o ficheiro (de texto ASCII) `exp.ppm` representa o resultado esperado da execução do programa para os dados de entrada definidos pelo ficheiro (de texto ASCII) `in.ppm`. A impossibilidade de verificar automaticamente o resultado da execução de um dado programa implica uma penalização de **100%** na componente de avaliação automática. Considera-se que um programa passou um teste com sucesso se o resultado produzido por esse programa for exactamente igual ao resultado esperado, o que significa que o comando `diff` não deverá encontrar diferenças entre o resultado produzido pelo programa submetido e o esperado. Para poder ser avaliado, um projecto deverá compilar correctamente num computador com o sistema operativo **GNU/Linux**, sendo utilizado o compilador **gcc** da **GNU** com as opções especificadas no enunciado. A entrega de código não compilável, ou a não inclusão de qualquer dos ficheiros requeridos, ou a utilização de nomes diferentes para o ficheiro executável conduz a uma classificação de 0 (zero) valores.