UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS - ICEX
ALGORITMOS E ESTRUTURAS DE DADOS II

Ingrid Rosselis Sant’Ana da Cunha

TP0

Belo Horizonte
2016

Introdução:

O primeiro trabalho da disciplina de AEDS II teve como objetivo simular o processo usado por um detector de objetos. O programa procura um objeto específico em um espaço, baseado em duas imagens em tons de cinza de formato .pgm que contém, além de informações sobre o número de colunas e de linhas e o valor máximo de um pixel, uma matriz de inteiros que forma as imagens. O procedimento se resume em obter as duas imagens, calcular valores em pontos e encontrar o maior deste, por fim, devolver o ponto do maior em um arquivo.

Implementação:

Para a implementação do programa foram utilizados as duas estruturas definidas no roteiro do trabalho - a estrutura PGM para armazenar as informações das imagens e a estrutura Ponto que armazena pontos - e vetores simples para auxiliar quando necessário.

LePGM: A função recebe como parâmetro o nome do arquivo de entrada e devolve uma estrutura do tipo PGM* criada na própria função e preenchida com informações retiradas do arquivo. Ela pode ser dividida em três partes: a primeira é a abertura do arquivo e um teste para saber se o processo obteve sucesso.

Na segunda, o P2 padrão de cada imagem é pulado com um fseek(f, 3, 0) - onde f é o ponteiro para o arquivo, 3 é o número de caracteres pulados ('P', '2' e '\n') e 0 é o valor que representa o início do arquivo - porque não há necessidade de salvá-lo, e as 3 informações principais da imagem que se situam logo em seguida são coletadas com fscanf. A estratégia para criar a "matriz de pixels" baseada no **dados da estrutura PGM foi alocar um vetor de ponteiros do tipo unsigned char (tipo da matriz definido na estrutura) com o tamanho do número de linhas e depois criar vetores para cada ponteiro do vetor inicial com o tamanho do número de colunas do mesmo tipo, formando assim uma matriz que pode ser acessada com [i][j], sendo i e j os índices de linha e coluna.

Por último, os valores restantes foram tirados do arquivo e colocados diretamente dentro da matriz da forma convencional (for aninhados), o f foi fechado e o ponteiro PGM* já preenchido foi devolvido, por isso não sendo desalocado com free.

JanelaDeslizante: Essa função aplica o processo de "deslizar a janela" sobre a matriz cena. Ela foi dividida **JanelaDeslizante**, **CorrelacaoCruzada** e **AchaValor** para facilitar e tornar mais clara a implementação. O algoritmo se baseia em colocar os valores e os pontos em dois vetores distintos, mas mantendo o mesmo índice para cada valor relacionado a cada ponto. Para isso, primeiro os limites de x e y (pontos) foram usados em dois for aninhados para saber quantas "janelas" do tamanho da matriz objeto a matriz cena permite.

Depois os vetores auxiliares foram alocados com tamanho necessário e outro laço igual ao primeiro foi utilizado para igualar a posição i do vetor de inteiros com a chamada função de correlação cruzada, que tem como um dos parâmetros a posição i do vetor de pontos. Por fim, a função **AchaValo**r é chamada (ver função mais abaixo), os vetores auxiliares são desalocados e a função retorna o ponto p onde a correlação tem o maior valor.

CorrelacaoCruzada: Função que calcula e devolve o valor da correlação cruzada entre duas imagens para determinado ponto. A função funciona de acordo com a sua versão matemática passado no roteiro: são dois for aninhados para trabalhar com o tamanho da matriz da imagem objeto e cada valor da matriz é multiplicado pela sua respectiva posição na matriz da imagem cena. A diferença é que toda vez que a correlação cruzada é chamada, a janela do tamanho da matriz

objeto muda por conta da posição contida na estrutura Ponto passada como parâmetro e que é somada ao índice da matriz cena no cálculo. A função retorna o valor calculado para o ponto.

AchaValor: É uma função auxiliar usada para encontrar o maior valor num vetor de inteiros passado como parâmetro juntamente com o seu tamanho. A única diferença entre sua versão convencional é que esta retorna o índice da posição do maior no vetor.

EscreveArquivo: Função que recebe o ponto que se deseja escrever e o nome do arquivo de saída. A função cria o arquivo com fopen, testa se obteve sucesso, e usa fprint para escrever o ponto no formato "x y" e fecha o arquivo. Devolve 0 se teve erro ao criar o arquivo e 1 se a função executou corretamente.

TestaExtensao: Função que testa se a extensão do nome do arquivo de saída passado como parâmetro está correta. Retorna 1 se não estiver e 0 se estiver.

Por fim, a main tem a única finalidade de testar o número de parâmetros passados pelo cmd, chamar as funções **LePGM** (uma para PGM* cena e uma para PGM* obj), igualar **JanelaDeslizante** a um ponto p, testar se a função **EscreveArquivo** obteve sucesso e desalocar os ponteiros do tipo PGM.

Resultados:

Por decisão durante o processo de montagem, foi escolhido priorizar a clareza do código e evitar erros desnecessários em detrimento da velocidade de execução do algoritmo, ou seja, o tempo de execução de imagens grandes é um pouco mais alto do que as do resto pela escolha de como trabalhar a ideia apresentada para este trabalho - a exemplo, a imagem Tsukuba apresenta um tempo em média de 4 segundos (medidos em um computador pessoal) em comparação à imagem de teste que não demora para gravar o arquivo. Contudo, essa velocidade não interfere no código e os resultados obtidos são exatamente como o esperado, da forma que está no roteiro do trabalho e que foi disponibilizado na pasta output.

Já no cmd, o trabalho apresenta uma mensagem para cada reação esperada, apenas para se ter uma noção do ocorrido no programa:

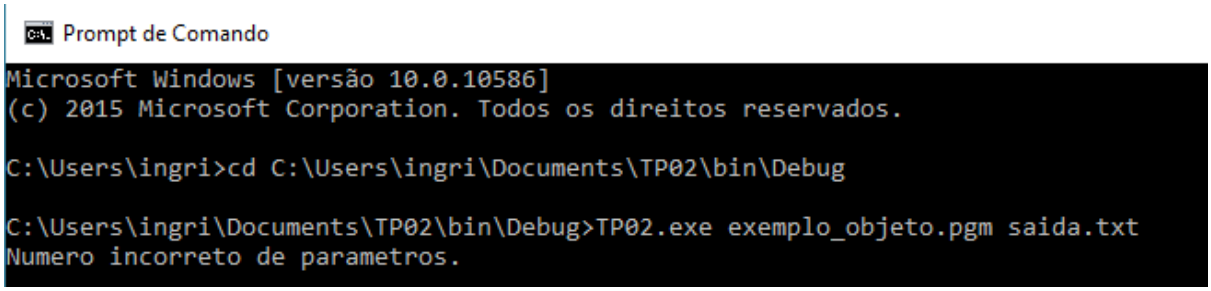
Programa foi executado corretamente:

```
Microsoft Windows [versão 10.0.10586]
(c) 2015 Microsoft Corporation. Todos os direitos reservados.

C:\Users\ingri>cd C:\Users\ingri\Documents\TP02\bin\Debug

C:\Users\ingri\Documents\TP02\bin\Debug>TP02.exe teste_cena.pgm teste_objeto.pgm saida.txt
Arquivo gravado com sucesso!

C:\Users\ingri\Documents\TP02\bin\Debug>
```

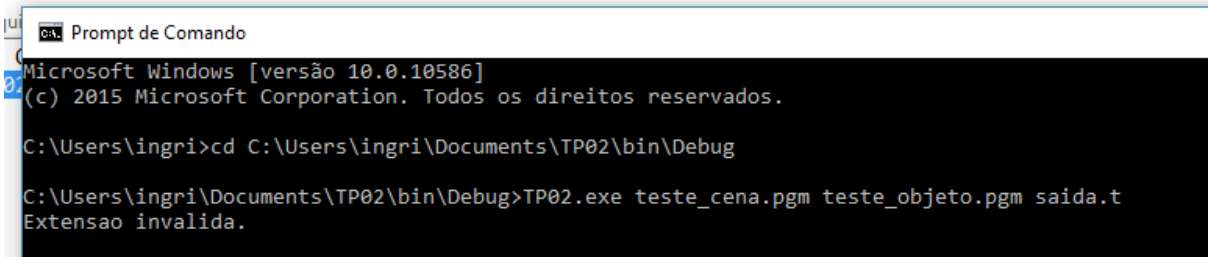
Não conseguiu abrir os arquivos das imagens:

```
Prompt de Comando
Microsoft Windows [versão 10.0.10586]
(c) 2015 Microsoft Corporation. Todos os direitos reservados.

C:\Users\ingri>cd C:\Users\ingri\Documents\TP02\bin\Debug

C:\Users\ingri\Documents\TP02\bin\Debug>TP02.exe exemplo_cena exemplo_objeto.pgm saida.txt
exemplo_cena: No such file or directory
```

Número de parâmetros está incorreto:



Extensão do arquivo de saída for diferente de “txt”:



Conclusão:

Poucas dificuldades foram encontradas durante o desenvolvimento do código, a maioria relacionada com o tratamento de matrizes, vetores e arquivos. Todas as dúvidas foram tiradas com pesquisa na internet e nada prejudicou ou dificultou a resolução do algoritmo. Alguns poréns devem ser ressaltados, uns foram resolvidos e os outros não, e que felizmente não alteram a execução do programa:

No Windows, para abrir pelo cmd, o programa não aceita por o diretório com o .exe juntos e os parâmetros da main, ou seja, abrir como “C:\Users\ingri\Documents\TP02\bin\Debug\TP02.exe teste_cena.pgm teste_objeto.pgm saida.txt”, por exemplo.Neste caso, os arquivos não são encntrados dentro do diretório e é apresentado dois erros, ambos vindos do teste de abertura na LePGM. É necessário usar “cd diretório” e depois colocar o .exe, os .pgm e o .txt para funcionar corretamente.

Outro problema que o trabalho apresentava era, algumas vezes e de forma aleatória, erro de ponteiro logo depois do fprint na função **EscreveArquivo**. Com pesquisa, foi decoberto que o problema poderia ser o buffer do arquivo, sendo assim foi colocado um fflush depois do fprintf, mas o problema não havia sido todo corrigido. Então, foi colocado um fflush antes também e o problema não voltou a ocorrer.

Um terceiro problema não foi resolvido, se refere à extensão do arquivo: a mensagem de “extensão inválida” funciona em casos como “saida.t”, “saida.” e qualquer outro que não seja “.txt”, porém isso não se aplica ao caso “saida”, onde o programa grava um arquivo sem extensão que não pode ser aberto. Vários testes e mudanças na função **TestaExtensao** foram feitos, mas nenhum deles surtiu efeito nesse caso.

Referências:

Nesse trabalho não foram usadas referências extras além da tirada de dúvidas referentes à linguagem C no livro do Luís Damas (Linguagem C, 10º ed.) e no Stack Overflow.

- **Pontos extras:**

O ponto extra traz mais um desafio: entender o porquê a correlação cruzada não funciona corretamente usando uma determinada imagem objeto, isto é, a resposta do arquivo não condiz com o que deveria. O algoritmo não funciona no caso porque, provavelmente devido aos números serem pequenos, o maior valor não representa o ponto de correlação cruzada das imagens. Exemplificando:

A imagem cena é:

1	**5**	0	8
0	**7**	10	5
6	2	4	7

E a imagem do objeto passada é:

1	5
0	7

Fazendo as contas de acordo com o algoritmo original, o valor resultante do ponto onde a imagem objeto se encontra na imagem cena (0, 0) é 75, porém no ponto (1, 1) o valor da correlação é 85.

Para corrigir esse problema foi implementado um algoritmo que funciona independente do valor de multiplicação dos inteiros. A função **CorralacaoModificada** recebe as duas imagens e assume o papel da **JanelaDeslizante**, isto é, define o ponto para ser passado para a função **Compara**. Esta função faz algo semelhante à **CorrelacaoCruzada**, mas ela compara se os números da matriz objeto são iguais à janela formada pela matriz cena em cada ponto e retorna 0 se toda matriz for igual a janela do ponto e 1 se não. A partir do momento em que a **CorrelacaoModificada** recebe 0 como retorno função de comparação, ela quebra o laço e devolve o ponto onde a imagem objeto foi encontrada. O interessante é que a execução do programa utilizando a nova função se mantém constante para todos os casos, inclusive Tsukuba, e é mais rápida que a primeira.

Os parâmetros passados pelo cmd podem ser tanto 4 quanto 5, e o quinto - usado para determinar se será utilizada a nova correlação ou a antiga, só aceita os parâmetros 0 ou 1.