UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciência da Computação

Disciplina	Turmas
Algoritmos e Estruturas de Dados II	

Professores

Erickson Rangel do Nascimento e Leonardo Barbosa e Oliveira

Data de entrega: 18/05/2017 **até as** 23:55

Trabalho Prático em Linguagem C (TP1) - 14 pontos

O objetivo deste trabalho é implementar uma busca recursiva por um valor específico em imagens digitais no formato PGM, que foi descrito no TP0. Para o funcionamento correto da busca, um algoritmo recursivo e uma pilha deverão ser corretamente implementados.

Estruturas de Dados

Para o desenvolvimento do trabalho serão necessárias algumas estruturas de dados, que deverão seguir as especificações a seguir.

Para armazenar as imagens, a estrutura usada no TP0 será mantida:

```
typedef struct {
   int c;                    // número de colunas na imagem
   int l;                    // número de linhas na imagem
   unsigned char maximo;     // valor máximo para cada pixel
   unsigned char **imagem;   // variável para armazenar os pixels da imagem (matriz)
} PGM;
```

Para armazenar o caminho até o ponto encontrado, será necessário implementar uma pilha da seguinte maneira:

```
typedef struct {
Celula *topo;     // Armazena somente a célula no topo da pilha
} Pilha;
```

A pilha, por sua vez é composta por células, que terão a seguinte estrutura:

```
typedef struct C{
int l;              // Linha de um ponto do caminho até a primeira célula de valor 0
int c;              // Coluna de um ponto do caminho até a primeira célula de valor 0
struct C *proximo;  // Ponteiro para a próxima célula da pilha
} Celula;
```

Busca Recursiva

A busca recursiva a ser implementada nesse trabalho tem como objetivo encontrar o pixel (posição na imagem PGM) com valor 0 mais próximo da origem (pixel 0,0 – canto superior esquerdo da imagem) seguindo as seguintes regras de caminhamento na imagem:

1. A busca será recursiva, portanto, **não será permitido nenhuma iteração (for e while) sobre os pixels da imagem**.
2. Se o valor da posição em avaliação não for 0, escolha entre os vizinhos esquerdo, direito, superior e inferior (caso existam) aquele de menor valor. Cuidado para não visitar nenhum pixel já visitado, seu programa entrará em *loop* infinito.
3. Caso dois ou mais vizinhos tenham o menor valor, escolha segundo a seguinte ordem de precedência: direito → inferior → esquerdo → superior.
4. Use uma pilha para armazenar as posições já avaliadas e a posição a ser avaliada.
5. Depois de encontrar o **primeiro** pixel de valor 0, escreva o caminho que leva da origem até esse pixel segundo o exemplo a seguir usando a mesma pilha que armazenou as posições.

Abaixo está um exemplo de imagem com o caminho que deve ser encontrado entre a origem e o primeiro valor 0 encontrado de acordo com as regras e uma representação do estado da pilha ao final da execução da busca.

Entrada

92	80	205	0
170	32	164	43
25	164	0	36
75	0	87	26

Caminho

92	0,0
80	0,1
32	1,1
164	1,2
0	2,2

Pilha

2,2
1,2
1,1
0,1
0,0

Para esse exemplo, a saída esperada é `(0,0), (0,1), (1,1), (1,2), (2,2)`.

O que deve ser feito

1. Elaborar um programa capaz de ler uma imagem no formato PGM e retornar o caminho do pixel (0,0) até o primeiro pixel com valor 0 encontrado através das regras listadas anteriormente.
2. O nome do arquivos de entrada será fornecido por argumentos para o programa.

   ```
   .\exec entrada.pgm
   ```

   onde *exec* é o nome do executável, *entrada.pgm* representa o nome do arquivo de entrada (imagem no formato PGM).
3. A saída do programa deverá ser feita para a saída padrão (`printf`) e os programas deverão imprimir **somente** o caminho da busca. O formato da saída deverá seguir o padrão do exemplo.

4. O programa deve ter a função `PGM *LerPGM(char* entrada)` que recebe o nome do arquivo como entrada, realiza a leitura da imagem e retorna uma instância alocada de forma *dinâmica* da estrutura de dados PGM descrita acima. É importante ressaltar que a variável imagem dentro da TAD também deverá ser alocada de forma dinâmica no tamanho exato da imagem.

5. O programa deve ter a função `void Busca(PGM *img, Pilha *pilha)` que realiza a busca recursiva na imagem e armazena o caminho na pilha.

6. O programa deverá ter a função `void ImprimeCaminho(Pilha *pilha)` que imprime o caminho da origem até o ponto encontrado.

7. Todas as funções implementadas devem possuir um cabeçalho conforme o exemplo a seguir:

```
/*-------------------------------------------------------------------
Protótipo: void Busca(PGM *img, Pilha *pilha)
Função: Realiza a busca ...
Entrada: estrutura PGM e pilha contendo ...
Saída: ...
-------------------------------------------------------------------*/
```

8. Seu programa não deverá ter nenhum *leak* de memória, ou seja, tudo que for alocado de forma dinâmica, deverá ser desalocado com a função "free()"antes do término da execução.

9. O programa deverá ser possível de ser compilado no Linux, portanto ele não deverá conter nenhuma biblioteca que seja específica do sistema operacional Windows.

10. Você deverá implementar o trabalho na IDE Code::Blocks (*disponível em: http://www.codeblocks.org/*). Deve ser submetido ao moodle o diretório do projeto criado no Code::Blocks em um arquivo compactado contendo os arquivos ".h"(com as declarações das funções para ler e salvar imagens PGM) e ".c"(um com as implementações das funções descritas nesse documento e outro com a função *main* (este deverá ter o nome *"main.c"*). Além disso, a submissão também deverá conter o relatório do trabalho em formato PDF.

11. O trabalho deverá ser entregue via *moodle* até as 23:55 horas do dia 18/05/2017. **Trabalhos que forem entregues fora do prazo não serão aceitos em nenhuma hipótese.**

Adicionalmente, alguns métodos facilitarão o desenvolvimento deste trabalho e serão vistos como boas práticas. São eles:

1. Um método `Desempilha` que retira a célula do topo da pilha e devolve um apontador para essa célula, ao invés de desalocar seu espaço de memória.

2. Um método `Busca` que busca na pilha uma célula com um determinado par de coordenadas e retorna `1` se existe tal célula e `0` caso contrário.

Documentação

Escreva um documento explicando o seu código e avaliando o desempenho de sua implementação. Separe-o em cinco seções: introdução, implementação, resultados, conclusão e referências. Seja claro e objetivo.

- **Introdução**: Faça uma breve introdução o problema, definindo-o com as *suas* palavras.
- **Implementação**: Explique quais foram as estratégias adotadas para a leitura das imagens em formato PGM, realização da operação de convolução e a escrita da imagem PGM em disco. Descreva qual o papel de cada função, seus parâmetros de entrada e de saída e a sua **análise de complexidade**.
- **Resultados**: Faça testes com seu programa utilizando diversas imagens. Apresente o resultado obitido e faça uma breve discussão sobre eles. (Serão disponibilizadas duas imagens para teste)

- **Conclusão**: Explique quais foram as dificuldades encontradas durante o desenvolvimento e as conclusões obtidas.
- **Referências**: Se você utilizou informações adicionais além das especificadas neste trabalho, cite as fontes.

Avaliação

- **60% dos pontos** pela implementação, onde serão avaliados, além do funcionamento adequado do programa: identação correta do código, comentários das funções, alocações de desalocações dinâmicas bem feitas e modularização.
- **40% dos pontos** pela documentação, onde serão avaliados a clareza das seções e a discussão dos resultados obtidos.
- ***Lembramos que será utilizado um software de detecção de cópias e qualquer tipo de plágio detectado resultará em nota 0 para ambos trabalhos!***