UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciência da Computação

Disciplina	Curso	Turmas	Período
Algoritmos e Estruturas de Dados II			2°

Professores
Camilo Oliveira, Gisele Pappa, Ítalo Cunha, Loïc Cerf, William Schwartz

Trabalho Prático 1 - Labirinto

Data de entrega: 20/04/2012 **(não haverá adiamento)**
Valor: 10 **pontos (+2 pontos extras)**

Considere o seguinte problema: *Um rato tentando sair de um labirinto.*

O rato percorre um labirinto em busca da saída, e no caminho coleta pedaços de queijo. Para cada posição, o rato pode ir em uma dentre quatro posições: direita, esquerda, para cima e para baixo. O labirinto é implementado como uma matriz de caracteres bidimensional, na qual as passagens são marcadas com o caractere numérico 0 (zero), as paredes com o caractere numérico 1 (um), a posição de saída com o caractere maiúsculo S, as posições com queijo com o caractere maiúsculo Q, a localização inicial do rato é indicada pelo caractere minúsculo i. O labirinto é todo cercado por paredes (dígitos 1), exceto na localização da saída.

Os labirintos serão armazenados em arquivos de extensão `.txt` e podem ter tamanhos diferentes. Esses arquivos, quando lidos, devem ser armazenados em matrizes. Dois exemplos de arquivos de entrada são mostrados a seguir. Os dois primeiros números do arquivo contém o número de linhas e colunas da matriz, respectivamente. **Observação**: o caractere de terminação de linha '`\n`' deve ser ignorado.

```
5 5
11111
1i0Q1
11101
SQ0Q1
11111

40 13
1111111111111111111111111111111111111111
1i0000000000000Q0000000000010001000000001
1010101010110101011111101010001010101101
1010111110110101000000101010100000000000S
1010001010111101011110000000110101011101
1011111100000110100000011110110101010101
1011111111111101011111111111110110101110101
1010000000000000Q000000000010000000Q0000001
1011011010101010101010101010111010101101
1010001000100001000010000010101000101101
1010111111111111111111111110101110111101
1000000000000001Q000000100000100000000001
1111111111111111111111111111111111111111
```

O que deve ser feito

1. Implementar um programa em linguagem C capaz de fazer um rato percorrer um labirinto e encontrar a saída.

2. O programa deve receber pelo `prompt` do sistema operacional (linha de comando) o nome do arquivo contendo o labirinto e o nome do arquivo de saída, como a seguir:
   `./executavel <arquivo .txt contendo o labirinto> <arquivo de saída>`

3. A saída do programa deve seguir a seguinte especificação:

   O programa gera um arquivo de saída, com nome especificado pela linha de comando, com o formato mostrado abaixo. *passos* indica o total de passos que foram feitos durante a busca pela saída, *comprimento* indica o número de movimentos no percurso até a saída (número de passos a partir da posição inicial até a saída do labirinto), x_i e y_i denotam as coordenadas que compõem o percurso entre a posição inicial e a saída, e *queijos* indica o número de queijos encontrados no percurso (*apenas* no percurso que leva à saída, *não* contar todos os queijos encontrados durante a busca). Todas essas informações devem estar contidas no arquivo de saída.

   ```
   p passos no total
   c passos ate a saida
   x1 y1
   x2 y2
   ...
   xn yn
   q queijos encontrados
   ```

   Por exemplo, a saida para o primeiro arquivo de entrada mostrado anteriormente deve ser similar à saída abaixo. Note que neste exemplo apenas o primeiro número impresso, o número de passos, depende do método de busca implementado. Todos os outros valores são fixos.

   ```
   7 passos no total
   7 passos ate a saida
   1 1
   1 2
   1 3
   2 3
   3 3
   3 2
   3 1
   3 queijos encontrados
   ```

4. Como no TP0, todas as funções devem ser precedidas por comentários. Neste trabalho, o arquivo `labirinto.h` a ser utilizado será disponibilizado juntamente com a especificação do trabalho. Portanto, as funções, cujos protótipos estão no `labirinto.h`, devem ser criadas e desenvolvidas no arquivo `labirinto.c`.

   Os TADs e as funções a implementar são:

   ```
   struct percurso {
       /* numero total de movimentos durante a busca */
       int passos;
       /* sequencia de posicoes ate a saida */
       struct posicao **posicoes;
       /* numero de elementos em **posicoes */
   ```

```
    int comprimento;
    /* numero de queijos no percurso em **posicoes */
    int queijos;
};

struct labirinto {
    // 'i'=inicio, '0'=livre, '1'=parede, 'Q'=queijo, 'S'=saida
    char *mapa;
    int colunas; /* mapa[colunas*x + y] */
    int linhas;
};

struct posicao {
    int x;
    int y;
};

struct labirinto * carrega_labirinto(char *nome_arquivo);
void libera_labirinto(struct labirinto *lab);
char inspeciona(struct labirinto *lab, struct posicao *pos);

struct posicao * coordenada_inicial(struct labirinto *lab);
struct posicao * cria_posicao(int x, int y);
void libera_posicao(struct posicao *pos);

struct percurso * acha_saida(struct labirinto *lab, struct posicao *pos);
void imprime_percurso(struct percurso *perc, char *nome_arquivo);
void libera_percurso(struct percurso *perc);
```

5. Analise a complexidade de algoritmo proposto e adicione na documentação a ser entregue.

Crédito Extra

O trabalho receberá crédito extra nos seguintes casos.

- Trabalho esteja acompanhado da implementação de interface gráfica para o labirinto utilizando a biblioteca Allegro (`http://alleg.sourceforge.net/`). Detalhes da interface gráfica bem como *screeshots* devem ser apresentados na documentação. **(1pt extra)**
- Soluções utilizando heurísticas que resolvam o problema com custo computacional reduzido ou com menor número de passos (a heurística utilizada deve ser explicada e analisada em detalhes na documentação). **(1pt extra)**

O que deve ser entregue

- Código fonte do programa em C (todos os arquivos .c e .h), bem identada e comentada, conforme especificado acima.
- Documentação sobre o trabalho (em .pdf), incluindo as decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Um exemplo de documentação pode ser encontrado no Moodle. Porém, ela deve conter, entre outras coisas:

    1. Introdução: descrição sucinta do problema a ser resolvido e visão geral sobre o funcionamento do programa.

2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, compilador utilizado, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
3. Estudo de Complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O), considerando conjuntos de tamanho n.
4. Testes: descrição dos testes realizados e listagem da saída (não edite os resultados).
5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.

- O trabalho será entregue através do sistema de submissão de trabalhos práticos: `http://aeds.dcc.ufmg.br`, em um arquivo compactado (.zip) contendo os arquivos fonte (arquivo .c, o arquivo .h, o main.c, e arquivos com a interface gráfica para aqueles que decidirem criar a interface utilizando a biblioteca Allegro). O a documentação deve ser enviada em pdf, no link especial para sua submissão no sistema.

Comentários gerais

- Comece a fazer este trabalho logo, enquanto o prazo para terminá-lo está tão longe quanto jamais poderá estar.
- Clareza, indentação e comentários no programa também vão valer pontos.
- Trabalhos copiados serão penalizados com a nota zero.
- O trabalho é individual.
- Penalização por atraso: (2d - 1) pontos, onde d é o número de dias de atraso.