UNISINOS - UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS – Curso: Informática / Ciência da Computação

LABORATÓRIO II

Disciplina: Linguagem de Programação PASCAL **E-mail:** ***fosorio@ unisinos.br***
Professor responsável: ***Fernando Santos Osório*** **Web:** ***http://www.inf.unisinos.br/~osorio/lab2.html***
Semestre: 2005/2 **Xerox : *Pasta 54* – LAB. II (Xerox do "Alemão")**
Horário: 53

TRABALHO PRÁTICO 2005/2 – GRAU A

(Versão 1.0 – 29/08/2005) – 30% da Nota do GA

Faça um programa para achar a saída de um Labirinto usando um ***algoritmo não recursivo***. O programa deve ler uma descrição de um labirinto (dimensão 20 x 20 posições), contido em um arquivo texto, conforme especificado mais abaixo. Uma vez lido o arquivo, o programa deve ser capaz de "simular" uma pessoa se deslocando neste labirinto, onde esta pessoa irá possuir um sistema GPS(*Global Positioning System*) que lhe indica a coordenada (L,C = Linha, Coluna) onde ela se encontra no labirinto. O labirinto pode não ter saídas, possuir apenas uma saída, ou mesmo possuir diversas saídas. Inicialmente o programa deve solicitar ao usuário uma coordenada inicial (Li, Ci) a partir de onde vai começar a busca pela saída, sendo gerados como resultado da execução deste programa três arquivos texto: (1) arquivo das posições das saídas: "saidas.txt"; (2) arquivo de todo o trajeto percorrido: "log.txt" e (3) arquivo que descreve o trajeto que liga diretamente a posição inicial até uma das saídas: "trajeto.txt" (Trajeto: desafio adicional no trabalho! Vale + 0.5pts na nota!). Estes arquivos devem respeitar exatamente a descrição fornecida logo a seguir.

O programa deverá fazer uso de ***ao menos*** uma pilha (bloco de rascunho usado pela pessoa para anotar posições ainda não exploradas) e uma fila ou deque (bloco de rascunho usado para registrar o trajeto percorrido). A pilha servirá para implementar uma solução não recursiva do problema, substituindo as facilidades de empilhamento de dados implementadas pela própria recursão (você deve implementar uma "simulação" da solução recursiva). Atenção: TODOS os caminhos possíveis devem ser visitados, ou seja, se houver mais de uma saída do labirinto, todas as saídas devem ser encontradas (o labirinto inteiro deve ser explorado).

- Descrição do Arquivo de Entrada: Lab.txt

  Este arquivo é composto de 20 linhas de texto, onde cada uma das linhas contém uma string de 20 caracteres, definindo assim um labirinto 20x20. Cada caractere da string representa uma informação sobre o conteúdo da sua respectiva posição no labirinto, podendo receber os seguintes valores:

  ´*´ - Posição ocupada (parede)
  ´. ´ - Posição livre (caminho livre)
  ´#´ - Saída do labirinto (passagem de saída)

  Como exemplo deste arquivo será apresentado aqui um labirinto de dimensões reduzidas de apenas 6x6, onde deve ser lembrado que o labirinto real irá possuir uma dimensão de 20x20 posições. Exemplos de labirintos serão colocados à disposição dos alunos na Internet.

Labirinto 6x6

1,1	1,2	1,3	1,4	1,5	
	2,2			2,5	
3,1	3,2	3,3		3,5	
		4,3			
5,1	5,2	5,3	5,4		
		6,3			6,6

Arquivo Texto Correspondente:

```
.....*
*.**.*
...*.*
**.***
....**
**#**.
```

Supondo a posição inicial em (1,1), o trajeto até a saída em (6,3) seria:
(1,1) (1,2) (2,2) (3,2) (3,3) (4,3) (5,3) (6,3)

- Descrição do Arquivo Gerado com as Saídas: Saidas.txt

  Este arquivo contém uma lista de pares de coordenadas L e C (linha,coluna) indicando os lugares onde foram encontradas saídas do labirinto. Os pares de valores das coordenadas são gravados, um em cada linha, separados por um espaço em branco, onde se existirem 3 saídas do labirinto teremos 3 linhas no arquivo, com uma saída descrita em cada linha. Exemplo:

  Saída em: (6,3)
  Arquivo: saidas.txt
  6 3

  Saída em: (1,1) (7,4) (20,20)
  Arquivo: saídas.txt
  1 1
  7 4
  20 20

- Descrição do Arquivo Gerado com os Caminhos Explorados: Log.txt

  Este arquivo contém uma lista de pares de coordenadas L e C (linha,coluna) indicando a seqüência exata de todos os lugares por onde a pessoa passou em sua exploração do labirinto, respeitando a mesma ordem que foi usada na exploração (esta ordem pode variar dependendo da implementação de cada aluno). A descrição das posições é feita de forma similar ao arquivo "saida.txt", ou seja, um par L,C em cada linha do arquivo, separados por um espaço em branco. Um exemplo de como ficaria este arquivo, considerando o labirinto 6x6 apresentado acima, é dado abaixo:

  Caminho percorrido: (1,1) (1,2) (1,3) (1,4) (1,5) (2,5) (3,5) (2,2) (3,2) (3,1)
  (3,3) (4,3) (5,3) (5,2) (5,1) (5,4) (6,3)

  Notar que:

  1. A posição (6,6) está isolada do resto do labirinto e, portanto, não é alcançada (visitada).
  2. Quando chegamos em um "beco sem saída", é necessário continuar a explorar as outras posições que ainda ficaram para serem exploradas mais tarde (devem estar no rascunho), isso ocorre no exemplo acima quando passamos do (3,5) de volta ao (2,2).
  3. O exemplo acima considerou uma certa prioridade escolhendo uma direção (para cima, para baixo, para esquerda ou para a direita) que deve ser explorada antes das demais. Você seria capaz de descobrir qual foi esta ordem adotada neste exemplo?
  4. Em função da prioridade escolhida a saída acabou sendo a última posição visitada, mas supondo que a saída estivesse em outra posição, por exemplo, em (3,5), o programa

deveria achar a saída e mesmo assim continuar a busca por outras saídas, obtendo ao final o mesmo caminho total que foi percorrido no exemplo acima.

Arquivo: log.txt

1 1
1 2
1 3
1 4
1 5
2 5
3 5
2 2
3 2
3 1
3 3
4 3
5 3
5 2
5 1
5 4
6 3

- Descrição do Arquivo Gerado com o Trajeto Final: Trajeto.txt (*Questão Desafio!*)

Este arquivo contém uma lista de pares de coordenadas L e C (linha,coluna) indicando a seqüência exata de todos os lugares por onde a pessoa deve passar, para ir do início do labirinto até uma das saídas (qualquer uma), sem desvios, e respeitando a ordem que foi usada na exploração. A descrição das posições é feita de forma similar ao arquivo "saida.txt", ou seja, um par L,C em cada linha do arquivo, separados por um espaço em branco. Um exemplo de como ficaria este arquivo, considerando o labirinto 6x6 apresentado acima, é dado abaixo:

Caminho percorrido: (1,1) (1,2) (2,2) (3,2) (3,3) (4,3) (5,3) (6,3)

Problema a ser resolvido...
Como trasnformar esta seqüência: (1,1) (1,2) (1,3) (1,4) (1,5) (2,5) (3,5) (2,2) (3,2) (3,1) (3,3) (4,3) (5,3) (5,2) (5,1) (5,4) (6,3)
Nesta outra: (1,1) (1,2) (2,2) (3,2) (3,3) (4,3) (5,3) (6,3) ?

Arquivo: trajeto.txt

1 1
1 2
2 2
3 2
3 3
4 3
5 3
6 3

- **Interface do Programa:**

```
>> Labirinto – Autor: Fulano <<

Entre com o nome do arquivo de entrada: labir1.txt
Entre com a posição inicial - Linha: 1
Entre com a posição inicial - Coluna: 1

Gerando: Saídas
Gerando: Caminho Percorrido
Gerando: Trajeto Final

Fim.
```

*** As estruturas de dados usadas neste programa devem ser baseadas (similares) as rotinas genéricas de manipulação de estruturas de dados que estudamos e implementamos na nossa disciplina (LISTA, FILA, PILHA e DEQUE), com alocação estática (ARRAYs).**

*** Se você quiser, também serão aceitas estruturas de dados (LISTA, FILA, PILHA e DEQUE) com alocação dinâmica (Ponteiros), mas você deve assumir a "responsabilidade" por ter optado por este tipo de estruturas.**

*** Faça um programa MODULAR usando procedures, functions e se quiser UNITS e SEM USAR VARIÁVEIS GLOBAIS.**

*** Será valorizado o aluno que desenvolver a implementação usando UNITS. Neste caso o aluno entregará o programa fonte do programa do labirinto e o fonte da Unit também.**

*** Enviar o trabalho por e-mail para o professor (anexado) até a data especificada. NÃO INCLUIR O EXECUTÁVEL (.exe) => ENVIAR APENAS O FONTE (.pas). Caso tenha usado o FreePascal, indicar no texto do e-mail. Só serão considerados entregues os trabalhos que receberem uma resposta confirmando o recebimento do e-mail.**

- ***Erros que tem que ser detectados***:

- O programa do Labirinto deve ter um mínimo de consistência, detectando erros do tipo: lista vazia, lista cheia (estouro da capacidade de armazenamento da estrutura de dados) e labirinto sem saída. Entretanto, nós assumiremos um "sistema bem comportado", ou seja, assumimos que os dados fornecidos pelo usuário (labirinto) não conterão dados inválidos, onde o labirinto irá respeitar a especificação descrita na definição do trabalho (tamanho, valores adotados, etc).

  - Estouro (overflow / underflow) da pilha, fila ou deque.
  - O labirinto terá uma dimensão de 20 x 20 casas. Prever uma lista de até 400 elementos.

BOM TRABALHO!