TRABALHO PRÁTICO 1 AEDS II

Labirinto

Discentes: Eurico Santiago, Miguel Leone, Luiz Barbosa Leite
Docente: Iago Augusto de Carvalho

1. ***Introdução***

O problema do labirinto é um problema a de busca a ser resolvido, onde se busca encontrar um caminho do ponto de entrada ('E') até o ponto de saída ('S') em uma matriz que representa o labirinto. Cada célula do labirinto pode ser um espaço vazio ('0'), uma parede ('x'), a entrada ('E') ou a saída ('S'). O objetivo é percorrer o labirinto, evitando paredes e marcando o caminho percorrido, até encontrar a saída.

2. Estruturas de Dados

O programa utiliza as seguintes estruturas de dados:

- **Matriz (`char lab[10][10]`)**: Representa o labirinto, onde cada célula pode conter um caractere representando o estado (entrada, saída, parede, etc.).

- **Estrutura Pilha**: Implementada na `pilha.h` e `pilha.c`, utilizada para armazenar as coordenadas (linha e coluna) do caminho percorrido. As operações básicas da pilha (empilhar e desempilhar) são usadas para backtracking ao explorar o labirinto.

3. Algoritmos

O algoritmo principal utilizado para encontrar a saída do labirinto é uma abordagem de busca com o uso de pilhas. O fluxo do algoritmo é o seguinte:

- **Carregamento do Labirinto**:
  - O labirinto é carregado de um arquivo para a matriz.

- **Identificação das Coordenadas**:
  - As posições da entrada e saída são identificadas.

- **Exploração do Labirinto**:
  - A posição atual é marcada como visitada.
  - O algoritmo tenta mover para as direções: direita, abaixo, esquerda e acima, **respectivamente.**
  - Se uma posição válida for encontrada (um espaço vazio ou a saída), as coordenadas são empilhadas.
  - Se não houver movimento válido, o algoritmo desempilha até encontrar uma nova posição para explorar.

- **Exibição do Caminho**:
  - O caminho percorrido é exibido ao final da execução
  -

4. Complexidade

- A complexidade do algoritmo é O(**N**), onde **N** é o número de células no labirinto. Cada célula é visitada no máximo uma vez.

5. Makefile

Abaixo está uma explicação das principais partes do Makefile utilizado neste projeto:

- **Variáveis**:
  - `CC`: Define o compilador a ser utilizado. No caso, o GCC (GNU Compiler Collection).
  - `CFLAGS`: Define as opções de compilação, onde `-Wall` ativa todos os avisos e `-g` inclui informações de depuração.
  - `OBJ`: Lista os arquivos objetos gerados a partir dos arquivos de código fonte (`.o`).
- **Regras**:
  - `all`: Regra padrão que compila o executável `labirinto`. Esta regra depende dos arquivos objetos listados na variável `OBJ`.
  - `labirinto`: Regra que vincula os arquivos objetos para gerar o executável. Utiliza o compilador e as opções definidas.
  - `main.o`, `funcs.o`, `pilha.o`: Regras que compilam cada um dos arquivos fonte individuais, gerando os respectivos arquivos objetos. Cada regra especifica as dependências (por exemplo, cabeçalhos que devem ser incluídos).
  - `clean`: Regra que remove arquivos objetos e o executável, limpando o diretório de trabalho. Isso é útil para garantir que não haja conflitos entre compilações anteriores.

Instruções de Compilação

1. Salve o Makefile no diretório do projeto.
2. Use o comando `make` para compilar o programa.
3. Para remover arquivos gerados, use `make clean`.

Essa estrutura torna o processo de compilação mais eficiente e menos propenso a erros, permitindo ao desenvolvedor focar no código-fonte em vez de se preocupar com o gerenciamento de arquivos.