Documentação de Software

Documentação do Software

LogiTrack

Autores:

-Bernardo Demaria Santos

-Marcelo Esteves Bernardi

-Raphael Thierry Coelho

-Stêvão Guadanini Diniz

Belo Horizonte
07/2025.

LogiTrack

Documentação do Software

Introdução

Escopo do software

Nome do sistema e de seus componentes principais

LogiTrack

Missão ou objetivo do software

O programa tem como objetivo simular a gestão de entregas, permitindo o cadastro e controle de locais, veículos e pedidos. O sistema calcula rotas com base na distância entre pontos, atualiza o status dos veículos durante a entrega e realiza backup e restauração dos dados em arquivos binários, aplicando os principais conceitos da disciplina de Algoritmos e Estruturas de Dados I.

Descrição do domínio do Cliente (Regras de Negócio)

Número	Regra de Negócio	Descrição
1	Cadastro de Locais	O sistema deve permitir o cadastro, atualização, listagem e remoção de locais.
2	Cadastro de Veículos	Os veículos devem ser registrados com status (livre ou ocupado) e local atual.
3	Cadastro de Pedidos	O usuário pode cadastrar pedidos com origem e destino válidos (locais existentes).
4	Associação de Pedido a Veículo	Apenas veículos livres podem ser associados a pedidos para realizar entregas.
5		

	Cálculo de Rota	A distância entre origem e destino é calculada pela fórmula da distância euclidiana.
6	Finalização de Entrega	Ao finalizar uma entrega, o status do veículo volta para livre e seu local é atualizado.
7	Backup e Restauração de Dados	Os dados de locais, veículos e pedidos podem ser salvos e carregados a partir de arquivos .dat.

Funcionalidades do produto

Número	Funcionalidade do sistema
1	Cadastro, listagem, atualização e remoção de locais
2	Cadastro, listagem, atualização e remoção de veículos
3	Cadastro, listagem, atualização e remoção de pedidos
4	Associação automática de pedido a veículo com base na disponibilidade e rota
5	Backup e restauração dos dados (locais, veículos e pedidos) via arquivos

LogiTrack

Usuários e sistemas externos

Descrição

Número	Usuários	Definição
1	Administrador do Sistema	Responsável por cadastrar, atualizar e remover locais, veículos e pedidos.
2		

	Operador de Logística	Realiza a associação de pedidos a veículos, acompanha entregas e finaliza rotas.
3	Sistema de Arquivos (externo)	Sistema externo usado para salvar e carregar backups (`.dat`).

Documentação do código

Documentação da Estrutura de dados geral do software

O sistema utiliza **vetores (arrays estáticos)** de structs para armazenar os principais dados:

- Local locais[MAX]: vetor de estruturas representando os pontos de origem/destino.
- Veiculo veiculos[MAX]: vetor de veículos, cada um com um local atual e status (livre/ocupado).
- Pedido pedidos[MAX]: vetor de pedidos, cada um com origem e destino (referências a índices do vetor de locais).

As estruturas (structs) definidas representam os dados persistidos e manipulados durante a execução. O uso de vetores permite acesso direto, ordenado por índice, e é apropriado para a escala do sistema.

Função cadastrarLocal

void cadastrarLocal(Local locais[], int *numLocais)

- **Parâmetros**:
  - locais[] → vetor de structs Local (entrada/saída)
  - *numLocais → ponteiro para inteiro, representa a quantidade de locais cadastrados (entrada/saída)
- **Retorno**: void (sem retorno)

Essa função cadastra um novo local no vetor locais. Solicita ao usuário o nome e as coordenadas X e Y, e armazena as informações na próxima posição disponível. Caso o número máximo de locais seja atingido, uma mensagem é exibida e o cadastro é abortado.

Função listarLocais

void listarLocais(Local locais[], int numLocais)

- **Parâmetros**:
  - locais[] → vetor de structs Local (entrada)
  - numLocais → inteiro, número total de locais cadastrados (entrada)
- **Retorno**: void

Essa função percorre o vetor de locais e imprime no console os dados de cada local (ID, nome e coordenadas).

Função atualizarLocal

void atualizarLocal(Local locais[], int numLocais)

- **Parâmetros**:
  - locais[] → vetor de structs Local (entrada/saída)
  - numLocais → inteiro, total de locais cadastrados (entrada)
- **Retorno**: void

Atualiza as informações de um local com base no ID informado pelo usuário. Se o ID for inválido, exibe uma mensagem de erro.

Função removerLocal

void removerLocal(Local locais[], int *numLocais)

- **Parâmetros**:
  - locais[] → vetor de structs Local (entrada/saída)
  - *numLocais → ponteiro para inteiro, total de locais (entrada/saída)
- **Retorno**: void

Remove um local pelo ID. Realiza o deslocamento dos elementos posteriores no vetor e decrementa o total de locais.

Função cadastrarPedido

void cadastrarPedido(Pedido pedidos[], int *numPedidos)

- **Parâmetros**:
  - pedidos[] → vetor de structs Pedido (entrada/saída)
  - *numPedidos → ponteiro para inteiro, total de pedidos (entrada/saída)
- **Retorno**: void

Cadastra um novo pedido, pedindo ao usuário os IDs de origem, destino e peso. O ID do pedido é atribuído automaticamente.

Função listarPedidos

void listarPedidos(Pedido pedidos[], int numPedidos)

- **Parâmetros**:
  - pedidos[] → vetor de structs Pedido (entrada)

- numPedidos → inteiro (entrada)

- **Retorno**: void

Lista os pedidos cadastrados no console, com ID, origem, destino e peso.

Função atualizarPedido

void atualizarPedido(Pedido pedidos[], int numPedidos)

- **Parâmetros**:
  - pedidos[] → vetor de structs Pedido (entrada/saída)
  - numPedidos → inteiro (entrada)
- **Retorno**: void

Atualiza os dados de um pedido pelo ID. Se o ID for inválido, exibe uma mensagem de erro.

Função removerPedido

void removerPedido(Pedido pedidos[], int *numPedidos)

- **Parâmetros**:
  - pedidos[] → vetor de structs Pedido (entrada/saída)
  - *numPedidos → ponteiro para inteiro (entrada/saída)
- **Retorno**: void

Remove um pedido do vetor. Os elementos seguintes são deslocados e o total de pedidos é decrementado.

Função cadastrarVeiculo

void cadastrarVeiculo(Veiculo veiculos[], int *numVeiculos)

- **Parâmetros**:
  - veiculos[] → vetor de structs Veiculo (entrada/saída)
  - *numVeiculos → ponteiro para inteiro (entrada/saída)
- **Retorno**: void

Cadastra um novo veículo. O status é definido como "disponível" por padrão. Solicita placa, modelo e ID do local atual.

Função listarVeiculos

void listarVeiculos(Veiculo veiculos[], int numVeiculos)

- **Parâmetros**:
  - veiculos[] → vetor de structs Veiculo (entrada)
  - numVeiculos → inteiro (entrada)
- **Retorno**: void

Imprime os dados de todos os veículos cadastrados: ID, placa, modelo, status e local atual.

Função atualizarVeiculo
void atualizarVeiculo(Veiculo veiculos[], int numVeiculos)

- **Parâmetros**:
  - veiculos[] → vetor de structs Veiculo (entrada/saída)
  - numVeiculos → inteiro (entrada)
- **Retorno**: void

Permite alterar os dados de um veículo (placa, modelo, status e local atual) com base no ID informado pelo usuário.

Função removerVeiculo
void removerVeiculo(Veiculo veiculos[], int *numVeiculos)

- **Parâmetros**:
  - veiculos[] → vetor de structs Veiculo (entrada/saída)
  - *numVeiculos → ponteiro para inteiro (entrada/saída)
- **Retorno**: void

Remove um veículo do vetor, deslocando os elementos seguintes e atualizando a contagem total.

Função associarPedidoRotaVeiculo
void associarPedidoRotaVeiculo(Pedido pedidos[], int numPedidos, Local locais[], int numLocais, Veiculo veiculos[], int numVeiculos)

- **Parâmetros**:
  - pedidos[] → vetor de Pedido (entrada)
  - numPedidos → inteiro (entrada)
  - locais[] → vetor de Local (entrada)
  - numLocais → inteiro (entrada)
  - veiculos[] → vetor de Veiculo (entrada/saída)
  - numVeiculos → inteiro (entrada)
- **Retorno**: void

Associa o último pedido cadastrado ao veículo disponível mais próximo do local de origem. Atualiza o status e a posição do veículo.

Função finalizarEntrega
void finalizarEntrega(Veiculo veiculos[], int numVeiculos)

- **Parâmetros**:
  - veiculos[] → vetor de Veiculo (entrada/saída)
  - numVeiculos → inteiro (entrada)
- **Retorno**: void

Finaliza uma entrega alterando o status do veículo para disponível. O usuário deve informar o ID do veículo.

Função salvarLocais

void salvarLocais(Local locais[], int numLocais)

- **Parâmetros**:
  - locais[] → vetor de Local (entrada)
  - numLocais → inteiro (entrada)
- **Retorno**: void

Salva os locais cadastrados em um arquivo binário chamado "locais.dat".

Função carregarLocais

void carregarLocais(Local locais[], int *numLocais)

- **Parâmetros**:
  - locais[] → vetor de Local (saída)
  - *numLocais → ponteiro para inteiro (saída)
- **Retorno**: void

Carrega os dados dos locais a partir do arquivo "locais.dat"

Função salvarVeiculos

void salvarVeiculos(Veiculo veiculos[], int numVeiculos)

- **Parâmetros**:
  - veiculos[] → vetor de Veiculo (entrada)
  - numVeiculos → inteiro (entrada)
- **Retorno**: void

Grava os dados dos veículos no arquivo "veiculos.dat".

Função carregarVeiculos

void carregarVeiculos(Veiculo veiculos[], int *numVeiculos)

- **Parâmetros**:
  - veiculos[] → vetor de Veiculo (saída)
  - *numVeiculos → ponteiro para inteiro (saída)
- **Retorno**: void

Lê os dados do arquivo "veiculos.dat" e carrega no vetor de veículos.

Função salvarPedidos

void salvarPedidos(Pedido pedidos[], int numPedidos, const char* nomeArquivo)

- **Parâmetros**:
  - pedidos[] → vetor de Pedido (entrada)
  - numPedidos → inteiro (entrada)
  - nomeArquivo → string constante (entrada)
- **Retorno**: void

Salva os pedidos no arquivo binário informado por nomeArquivo.

Função carregarPedidos

void carregarPedidos(Pedido pedidos[], int *numPedidos, const char* nomeArquivo)

- **Parâmetros**:
  - pedidos[] → vetor de Pedido (saída)
  - *numPedidos → ponteiro para inteiro (saída)
  - nomeArquivo → string constante (entrada)
- **Retorno**: void

Lê os dados do arquivo binário informado e carrega os pedidos no vetor correspondente.

Testes do software

Casos de testes do software: função associarPedidoRotaVeiculo

Número	Varáveis de Entrada	Valores válidos	Resultado Esperado	Valores inválidos	Resultado Esperado
1	pedidos, locais, veiculos	Pedido com origem 0, destino 1, veículo livre	Veículo alocado, status alterado para ocupado	—	—
2	pedidos, locais, veiculos	Pedido com origem e destino diferentes	Veículo movido para destino, status atualizado	veículo status = 1 (ocupado)	Exibe “Nenhum veículo disponível”

3	pedidos, locais, veiculos	lista de veículos vazia (numVeiculos = 0)	Mensagem de erro exibida	—	—
4	pedidos com idOrigem fora do array	idOrigem = 99, locais[] tem apenas 2 locais	Ignorado com segurança, sem alteração nos dados	—	—
...					

Casos de testes do software: função menu

Número	Varáveis de Entrada	Valores válidos	Resultado Esperado	Valores inválidos	Resultado Esperado
1	opção	0	Encerra o programa com mensagem “Saindo...”	-1, 17, 100	Exibe “Opção inválida!”
2	opção	1 a 16	Executa a função correspondente à opção	letra ('a'), símbolo ('#')	scanf falha, mantém loop do menu
3	opção	13	Associa pedido a veículo e exibe rota	—	—
4	opção	15 ou 16	Salva ou restaura os dados do sistema	—	—

...					