TRABALHO DE ESTRUTURA DE DADOS I - Documentação

INFORMAÇÕES GERAIS

Documentação do código apresentado por Alysson dos Anjos, Cainan de Brito e Letícia Almeida para a avaliação do componente curricular Estrutura de Dados I do curso de Sistemas da Informação da Universidade do Estado da Bahia (UNEB) durante o semestre 2023.2.

Proposta da avaliação:

A criação de um programa completo em C de uma agência de viagens que mantém um cadastro de sítios turísticos por país, organizado em uma lista com dois ponteiros: um que aponte para o próximo país, e outro que aponte para os sítios turísticos do local em questão.

A empresa precisa fornecer serviços para dois perfis de clientes: aqueles que conhecem o destino desejado e os que não têm certeza para onde viajar. Para os clientes de primeiro tipo, a agência solicita nome e país de destino, incrementando o contador de visitas ao país correspondente na lista duplamente encadeada, já para o segundo tipo de cliente, a agência deve oferecer opções de destinos de acordo com uma árvore binária de auxílio para decisões, com o mínimo de quatro níveis. Os clientes que fecham contratos são contabilizados da mesma forma. O sistema final deve:

- Apresentar os menus de atendimento adequados;
- Listar todos os países que foram visitados por clientes decididos;
- Listar todos os países e sítios turísticos que foram visitados para clientes que usaram o sistema de auxílio;
- Lista de países visitados em geral;
- Lista de países não visitados; e
- Informar o país e o sítio mais visitado pelos dois tipos de clientes.

Resumo do programa elaborado

O programa elaborado para o sistema da agência de viagens foi dividido em diferentes módulos, contando com dois arquivos de texto (*LetDestinos.txt* e *LetTexto.txt*), seis bibliotecas de cabeçalho (*arvore.h, destinos.h, clientes.h, tratamentosErro.h, interface.h e cores.h*) e o arquivo principal (*main.c*) que reúne a implementação das bibliotecas anteriormente citadas, além da ponto de entrada para o programa. O repositório do GitHub para os códigos está disponível em: https://github.com/leticosta4/trabalhoED1.git.

Divisão dos Módulos do Programa

main.c

lerArquivoLista

É responsável por fazer a leitura do arquivo de texto *LetDestinos.txt*, que contém os dados que serão utilizados para preenchimento da lista principal do projeto: lista duplamente encadeada de países, e a lista das cidades desses países. O arquivo de texto também possui um índice que indica quando cada função de inserção, invocada dentro do laço de repetição de leitura das linhas do arquivo, deve ser chamada para os ponteiros das listas de países e cidades, definidos no escopo local da função padrão *main*. Além disso, a função retorna uma mensagem caso tenha algum erro na abertura do arquivo.

lerArquivoArvore

É responsável por fazer a leitura do arquivo de texto *LetTexto.txt*, que contém os dados que serão utilizados para preenchimento da árvore de decisões. O arquivo de texto também possui um índice de prioridade que serve para o balanceamento das perguntas na árvore durante seu preenchimento, iniciado a partir do ponteiro inicial da árvore, definido também no escopo local da *main*. Além disso, a função retorna uma mensagem caso tenha algum erro na abertura do arquivo.

main

É a entrada principal do programa e coordena etapas de:

- Inicialização do ponteiro referente aos arquivos de texto que serão lidos e dos ponteiros principais para as estruturas de dados presentes no programa;
- Chamada inicial das funções de leitura dos arquivos, das funções de interface gráfica e um aviso de que o código foi feito em base linux;
- Invocação das outras funções que controlam o programa ao longo do switch-case que representa as opções escolhidas do menu principal da agência de viagens. Esse switch-case é operado dentro de um laço de repetição que interage com o usuário solicitando uma opção.

Por fim, quando o cliente sai do sistema, a mensagem de fim é exibida e como padrão da função *main*, o valor zero é retornado indicando que o programa foi finalizado com sucesso

arvore.h

arvoreRespostas

É a função que percorre a árvore de decisões durante a execução da mesma no programa, confirmada pela opção 2 no menu principal da agência, tendo como parâmetro o ponteiro principal para a árvore, definido no próprio escopo local da função *main*. Ela funciona de maneira recursiva, retornando a resposta do usuário (“sim” ou “não”) às perguntas ou ainda o

local de destino do mesmo, o que ocorre ao atingir as folhas da árvore - condição de parada da função. O retorno do cliente é verificado pela chamada da função *erroResposta*, condição de continuação do laço de repetição, se a resposta for positiva, a árvore continua a ser percorrida pela esquerda, caso contrário, pela direita.

inserir

É responsáveis pelo preenchimento da árvore de decisões da agência de viagens, a partir dos dados que são lidos do arquivo de texto. Ela recebe como parâmetros o ponteiro principal da árvore, um valor de prioridade para balanceamento e um ponteiro para string, que representa a pergunta a ser feita ao usuário ou o nome do seu local de destino; a inserção é feita de maneira recursiva seguindo a ordem dos índices de prioridade definidos no arquivo *LetTexto.txt*, de forma que se o número for maior do que o atual a inserção ocorre pela direita, caso contrário, pela esquerda. A função retorna um ponteiro do seu próprio tipo para o ponteiro inicial que representa a árvore atualizada.

listarArvore

Função de teste para visualização da árvore de decisões, utilizada durante o desenvolvimento do projeto. Ela lista o conteúdo da árvore de maneira recursiva, percorrendo a estrutura de dados com os seus ponteiros esquerdo e direito, em pré-ordem.

destinos.h

inserirPaises e inserirCidade

São responsáveis pelo preenchimento das listas de países e cidades que pertencem aos mesmos, a partir dos dados que são lidos do arquivo de texto. Elas recebem como parâmetros o ponteiro principal da lista e um ponteiro para string, que representa o nome do país ou da cidade; a inserção é feita usando a lógica de uma pilha dinâmica. Ambas as funções retornam um ponteiro do seu respectivo tipo (país ou sítio turístico) para o ponteiro inicial que representa a lista atualizada.

Como a lista de países é duplamente encadeada, com um ponteiro para o próximo país e outro para a lista de cidades, em *inserirCidade*, a função retorna o ponteiro da lista atualizada para o campo da cidade no próprio ponteiro do país, que é especificado na struct desse tipo.

listarLista

É a função que exibe a lista geral de países e sítios disponíveis na agência de viagens. Ela funciona por meio de 2 laços de repetição que percorre a lista duplamente encadeada dos países, que possui um dos ponteiros referente às cidades do mesmo, as quais também são percorridas pelo loop.

listasEspecificas

Função com um switch-case padrão chamada na *main* quando são selecionadas do menu as opções de listas diferentes da lista geral (4 a 7). O switch-case chama a função de lista de acordo com a opção selecionada.

listarSemAuxilio e listarComAuxilio

Funções que listam, respectivamente, os países, e suas respectivas cidades, que foram visitados sem o sistema de auxílio da árvore binária, e os que foram visitados com esse auxílio, sendo chamadas em *listasEspecificas*. A verificação é feita pela checagem dos contadores dos 2 tipos de turistas, definidos dentro da struct de cidade. Para os casos de nenhum país tenha sido visitado sem ou com a árvore de escolhas, mensagens específicas são exibidas.

listarNaoVisitados e listarSoVisitados

Funções que listam, respectivamente, os países, e suas respectivas cidades, que não foram visitados, e os que foram visitados, sendo chamadas em *listasEspecificas*. A verificação é feita pela checagem simultânea dos contadores dos 2 tipos de turistas, definidos dentro da struct de cidade. Para os casos de todos os países ou nenhum país ter sido visitado, mensagens específicas são exibidas.

paisMaisVisitado e cidadeMaisVisitada

São funções chamadas na opção 8 do menu principal da agência, sendo responsáveis por exibir o país e o seu sítio turístico mais visitado. A função *cidadeMaisVisitada* só é invocada se o retorno da função *paisMaisVisitado* for diferente de zero, o que é verificado dentro de uma condicional, indicando que pelo menos um país foi visitado pelo menos uma vez, caso contrário, uma mensagem é exibida ao usuário.

Ambas as funções funcionam com a lógica de percorrer a lista e comparar dois itens (país ou cidade) a partir dos seus contadores de quantidade de turistas, definidos nas structs dos tipos *Pais* e *Sitio*. Quando o(s) local(is) mais visitado(s) é(são) encontrado(s) seu(s) nome(s) é(são) exibido(s) na tela, e os demais destinos são também impressos mas como espaços em branco, o que tinha sido previamente sido definido da função através do preenchimento de uma matriz do tipo char que armazena os nomes dos países e cidades em suas respectivas funções.

clientes.h

addCliente

Lida com a inserção dos clientes no sistema, sendo chamada dentro do switch-case da função *main*. Ela recebe como parâmetros o ponteiro principal para a lista de países, um ponteiro de

string para o nome do local e um identificador do tipo de turista, sendo 1 sem o auxílio da árvore de decisões e 2 com esse auxílio.

Para o caso do cliente tipo 1, o ponteiro de string para o nome do local é inicialmente preenchido com um espaço vazio na invocação da função, já que ainda será adquirido posteriormente. Assim, primeiramente a função *opcao1menu* é chamada, e dependendo da opção escolhida desse segundo menu, ou a lista de países é exibida, ou o usuário volta para o menu principal ou é feita a coleta do local de destino inserido pelo cliente. Já para o cliente tipo 2, o ponteiro de string para o nome do local de destino é preenchido de acordo com a chamada da função *arvoresResposta*.

Dessa forma, para ambos tipos de cliente, com o local da viagem coletado, é feito o processo de separação entre o nome do país e o da cidade, que são enviados, por fim, para a função *compIncrementa*.

compIncrementa

É responsável por incrementar o contador do tipo de turista específico no sistema, tendo como parâmetros as strings de país e de sítio turístico, o ponteiro da lista de países e o identificador do tipo de cliente.

Ela funciona seguindo lógica de percorrer a lista geral de países, e posteriormente a de cidades desse país, e de acordo com o destino identificado (especificado pelas duas strings passadas nos parâmetros), o contador do tipo de turista específico é incrementado em uma unidade.

Além disso, essa função também verifica, para o caso de clientes que não usaram a árvore de decisões, se o local digitado existe na lista geral de destinos da agência de viagens ou não, exibindo uma mensagem quando isso não ocorre e invocando novamente a função *addCliente* para que o usuário possa digitar novamente o local desejado.

pausar

É chamada depois de cada função no switch-case da main. Ela aguarda até que o usuário pressione Enter (‘*n*’) e consome qualquer entrada adicional até que Enter seja pressionado novamente.

tratamentosErro.h

Possui apenas uma função que garante que as respostas do usuário, às perguntas feitas pela árvore de decisão para o segundo tipo de cliente, sejam ‘s’ (sim) ou ‘n’ (não), retornando 1 para estes caracteres certos e zero para caracteres diferentes de ‘s’ ou ‘n’, quando chamada dentro da função *arvoreRespostas* da biblioteca *arvore.h*.

interface.h

Biblioteca de interface gráfica, conta com as seguintes funções:

opcao1Menu

Espécie de “submenu” exibido para o cliente do tipo 1, que tem as opções de digitar diretamente o destino de viagem, exibir a lista geral de destinos e sair.

menu

É o menu inicial do programa, que contém o acesso à lista e à árvore de destinos de acordo com o interesse do usuário e as seis opções requisitadas na proposta do programa, tendo no total 9 alternativas disponíveis ao usuário.

aviao e theEnd

Funções para ilustração da agência de viagens, que mostram um avião com o nome da empresa no início da compilação e uma logo de “fim” no final, respectivamente.

cores.h

Biblioteca que atua em conjunto com a de interface com o usuário. Possui funções que implementam as cores roxo, azul, branco e vermelho, usando sequência de escape ANSI (American National Standards Institute) para cada cor.