Trabalho da disciplina Métodos de Programação

3

Generated by Doxygen 1.8.1.2

Mon Nov 23 2015 16:51:28

Contents

Chapter 1

TrabalhoMP

Projeto da disciplina de Métodos de Programação (2/2015) na Universidade de Brasília

Rede de distribuição

Obs.: Todos os comandos a seguir devem ser executados no diretório raiz do projeto

```
-> Para compilar o programa, execute o comando "make"
-> Para rodar o programa, execute o comando "make rodar"
-> Para compilar o teste, execute o comando "make teste"
-> Para rodar o teste, execute o comando "make testar"
-> Para usar a ferramenta gcov, execute o comando "make gcov"
-> Para usar a ferramenta cppcheck, execute o comando "make cppcheck"
-> Para usar a ferramenta valgrind, execute o comando "make valgrind"
-> Para limpar todos os arquivos gerados pelos comandos acima, execute o comando "make clean"
```

Obs2.: Dependendo do SO utilizado, se faz necessário pular uma linha no arquivo de teste, para o programa poder funcionar corretamente

```
Ex.: elementary OS, versão 0.2.1 "Luna" (64-bit)
```

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 adaptador Struct Reference

`#include <Principal.h>`

Collaboration diagram for adaptador:

Data Fields

- char * nome
- int posicao [2]
- int recursoRecebido
- struct interconexao * saidas
- struct interconexao * entradas
- int quantidadeSaidas
- struct adaptador * proximo

4.1.1 Detailed Description

----------——Adaptadores----------—— Cabecalho do elemento Adaptador

nome: nome do adaptador

posicao: vetor posicao, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

recursoRecebido: quantidade de recurso recebido por segundo pelo Adaptador

entrada: ponteiro que representa a interconexao de entrada do adaptador

saidas: representa as interconexoes realizadas/apontadas pelos adaptadores

quantidadeSaidas: quantidade de conexoes que o adaptador possui

proximo: representa a proxima Adaptador da lista de Adaptadores

4.1.2 Field Documentation

4.1.2.1 struct interconexao* entradas

4.1.2.2 char* nome

4.1.2.3 int posicao[2]

4.1.2.4 struct adaptador* proximo

4.1.2.5 int quantidadeSaidas

4.1.2.6 int recursoRecebido

4.1.2.7 struct interconexao* saidas

The documentation for this struct was generated from the following file:

- /home/eduardo/TrabalhoMP/app/header/Principal.h

4.2 cidade Struct Reference

`#include <Principal.h>`

Collaboration diagram for cidade:

Data Fields

- char * nome
- int posicao [2]
- int recursoNecessario
- int recursoRecebido
- int recursoGasto
- int tagEstado
- int turnosNegativados
- int turnosNoVermelho
- struct cidade * proximo
- struct interconexao * entradas

4.2.1 Detailed Description

----------——Cidades----------—— Cabecalho do elemento Cidade

nome: nome da cidade

posicao: vetor posicao, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

recursoNecessario: quantidade de recurso que a cidade precisa por segundo

recursoRecebido: quantidade de recurso recebido por segundo pela Cidade

recursoGasto: quantidade de recurso que a cidade usou

tagEstado: flag para definir qual eh o estado da cidade (VERMELHO,AMARELO,VERDE)

turnosNegativados: contador para armazenar quantos turnos a cidade ficou fora do VERDE

turnosNoVermelho: contador para armazenar quantos turnos a cidade ficou no VERMELHO

proximo: representa a proxima cidade da rede de cidades

entradas: ponteiro que representa a interconexao de entrada da cidade

4.2.2 Field Documentation

4.2.2.1 struct interconexao* entradas

4.2.2.2 char* nome

4.2.2.3 int posicao[2]

4.2.2.4 struct cidade* proximo

4.2.2.5 int recursoGasto

4.2.2.6 int recursoNecessario

4.2.2.7 int recursoRecebido

4.2.2.8 int tagEstado

4.2.2.9 int turnosNegativados

4.2.2.10 int turnosNoVermelho

The documentation for this struct was generated from the following file:

- /home/eduardo/TrabalhoMP/app/header/Principal.h

4.3 gerador Struct Reference

`#include <Principal.h>`

Collaboration diagram for gerador:

Data Fields

- char * nome
- int posicao [2]
- int taxaProducao
- int recursoProduzido
- int custo
- struct gerador * proximo
- struct interconexao * saida

4.3.1 Detailed Description

----------——Geradores----------—— Cabecalho do elemento Gerador

nome: nome do gerador

posicao: vetor posicao, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

taxaProducao: quantidade de recurso que a cidade precisa por segundo

recursoProduzido: quantidade total de recurso produzido pelo gerador

custo: custo de geração por segundo

proximo: representa a proxima cidade da rede de cidades

saida: representa as interconexoes realizadas/apontadas pelos adaptadores

4.3.2 Field Documentation

4.3.2.1 int custo

4.3.2.2 char* nome

4.3.2.3 int posicao[2]

4.3.2.4 struct gerador* proximo

4.3.2.5 int recursoProduzido

4.3.2.6 struct interconexao* saida

4.3.2.7 int taxaProducao

The documentation for this struct was generated from the following file:

- /home/eduardo/TrabalhoMP/app/header/Principal.h

4.4 interconexao Struct Reference

`#include <Principal.h>`

Collaboration diagram for interconexao:

Data Fields

- char * nome
- int posicaoInicial [2]
- int posicaoFinal [2]
- Destino tagDestino
- float chanceFalha
- int tempoConserto
- int custoConserto
- int contadorTempoConserto
- int numeroFalha
- Falha tagFalha
- int capacidadeMaxima
- int recursoTransportado
- struct interconexao * proximo
- struct adaptador * entradaAdaptador
- struct adaptador * saidaAdaptador
- struct interconexao * entradaInterconexao
- struct interconexao * saidaInterconexao
- struct interconexao * proximoEntradaAdaptador
- struct interconexao * proximoSaidaAdaptador
- struct interconexao * proximoEntradaCidade
- struct gerador * entradaGerador
- struct cidade * saidaCidade

4.4.1 Detailed Description

----------——Interconexoes----------—— Cabecalho do elemento Inerconexao

nome: nome da interconexao

posicaoInicial: vetor posicaoInicial, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

posicaoFinal: vetor posicaoFinal, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

tagDestino: tag para indentificar qual é a ligaçao final de cada conexao, seja cidade ou adaptador

chanceFalha: chance de falha por segundo

tempoConcerto: tempo de concerto em caso de falha em segundos

custoConcerto: custo do concerto em segundos

contadorTempoConserto: contabiliza quanto tempo ja esta no concerto

numeroFalha: total de falhas

tagFalha: indica se houve falha

capacidadeMaxima: capacidade maxima da interconexao

recursoTranstortado: quantidade de recuso que esta sendo transportado pela conexao no turno

proximo: representa a proxima interconexao na lista de interconexoes

entradaAdaptador: aponta, caso a entrada seja um adaptador, para o adaptador cuja saída eh esta interconexao

saidaAdaptador: aponta, caso a saida seja um adaptador, para o adaptador cuja entrada eh esta interconexao

entradaInterconexao: aponta, caso a entrada seja interconexao, para a interconexao cuja saida eh esta interconexao

saidaInterconexao: aponta, caso a saida seja interconexao, para a interconexao cuja entrada eh esta interconexao

proximoEntradaAdaptador: ponteiro para a proxima entrada do adaptador relacionado a interconexao

proximoSaidaAdaptador: ponteiro para a proxima saida do adaptador relacionado a interconexao

proximoEntradaCidade: ponteiro para a proxima cidade de destino

entradaGerador: aponta, caso a entrada seja um gerador, para o gerador cuja saida eh esta interconexao

saidaCidade: aponta, caso a saida seja uma cidade, para a cidade cuja entrada eh esta interconexao

4.4.2 Field Documentation

4.4.2.1 int capacidadeMaxima

4.4.2.2 float chanceFalha

4.4.2.3 int contadorTempoConserto

4.4.2.4 int custoConserto

4.4.2.5 struct adaptador* entradaAdaptador

4.4.2.6 struct gerador* entradaGerador

4.4.2.7 struct interconexao* entradaInterconexao

4.4.2.8 char* nome

4.4.2.9 int numeroFalha

4.4.2.10 int posicaoFinal[2]

4.4.2.11 int posicaoInicial[2]

4.4.2.12 struct interconexao* proximo

4.4.2.13 struct interconexao* proximoEntradaAdaptador

4.4.2.14 struct interconexao* proximoEntradaCidade

4.4.2.15 struct interconexao* proximoSaidaAdaptador

4.4.2.16 int recursoTransportado

4.4.2.17 struct adaptador* saidaAdaptador

4.4.2.18 struct cidade* saidaCidade

4.4.2.19 struct interconexao* saidaInterconexao

4.4.2.20 Destino tagDestino

4.4.2.21 Falha tagFalha

4.4.2.22 int tempoConserto

The documentation for this struct was generated from the following file:

- /home/eduardo/TrabalhoMP/app/header/Principal.h

4.5 relatorio Struct Reference

```
#include <Principal.h>
```

Data Fields

- int tempoTotalSimulacao
- int custoTotalSimulacao
- int totalGeradores
- int energiaTotalGerada
- int totalCidades
- int energiaGastaCidades
- float tamanhoTotalInterconexoes
- int numeroFalhaInterconexoes
- int numeroCidadesNegativadas
- int tempoSemRecurso
- int numeroCidadesNoVermelho
- int tempoCidadesNoVermelho

4.5.1 Detailed Description

----------——Relatorio----------—— Cabecalho onde serao armazenadas as respostas paras as perguntas do relatorio inicial

tempoTotalSimulacao: tempo total da simulacao

custoTotalSimulacao: custo total na simulacao

totalGeradores: total de geradores

energiaTotalGerada: energia total gerada

totalCidades: total de cidades

energiaGastaCidades: energia total gasta pelas cidades

tamanhoTotalInterconexoes: tamanho total das interconexoes

numeroFalhaInterconexoes: numero de falhas nas interconexoes

numeroCidadesNegativadas: numero de cidades que ficaram com menos recurso que o necessário

tempoSemRecurso: tempo que ficaram sem recurso

nomeroCidadesNoVermelho: número de cidades que ficaram com menos de 30% dos recursos

tempoCidadesNoVermelho: tempo que ficaram com menos de 30% de recurso

4.5.2 Field Documentation

4.5.2.1 int custoTotalSimulacao

4.5.2.2 int energiaGastaCidades

4.5.2.3 int energiaTotalGerada

4.5.2.4 int numeroCidadesNegativadas

4.5.2.5 int numeroCidadesNoVermelho

4.5.2.6 int numeroFalhaInterconexoes

4.5.2.7 float tamanhoTotalInterconexoes

4.5.2.8 int tempoCidadesNoVermelho

4.5.2.9 int tempoSemRecurso

4.5.2.10 int tempoTotalSimulacao

4.5.2.11 int totalCidades

4.5.2.12 int totalGeradores

The documentation for this struct was generated from the following file:

- /home/eduardo/TrabalhoMP/app/header/Principal.h

Chapter 5

File Documentation

5.1 /home/eduardo/TrabalhoMP/app/header/Adaptadores.h File Reference

```
#include "Principal.h"
```

Include dependency graph for Adaptadores.h: This graph shows which files directly or indirectly include this file:

Functions

- Adaptador * criaListaAdaptador ()
- Vazio adaptadorVazio (Adaptador *)
- Adaptador * insereAdaptador (char *, Adaptador *)
- void imprimeListaAdaptador (Adaptador *)
- void liberaListaAdaptador (Adaptador *)
- void defineDistribuicao (Adaptador *)
- void mandarRecursoAdaptado (Adaptador *)

5.1.1 Function Documentation

5.1.1.1 Vazio adaptadorVazio (Adaptador * *listaAlvo*)

Funcao: adaptadorVazio

Verifica se a lista de adaptadores esta vazia

Parameters

listaAlvo	ponteiro para o inicio da lista de adaptadores

Returns

variavel do tipo Vazio, indicando se a lista esta vazia

Assertiva de entrada: estrutura do tipo Adaptador

Assertiva de saida: condicao do Adaptador sendo vazio ou nao vazio

Funcao: adaptadorVazio (Iterador)

AssertivaSaida: VAZIO || NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazio, adaptadorVazio, Adaptador ∗listaAlvo

Interfaces implicitas: Vazio - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de adaptadores AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.1.1.2 Adaptador∗ criaListaAdaptador ( )

Funcao: criaListaAdaptador

Inicia um ponteiro que sera para Adaptador

AssertivaSaida: NULL;

Funcao: criaListaAdaptador (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Adaptador

Interfaces explicitas: Adaptador∗, criaListaAdaptador

5.1.1.3 void defineDistribuicao (Adaptador ∗ *listaAlvo*)

Define como sera a distribuicao entre as conexoes que esta ligada a cada adaptador e manda a quantidade que sera passada para as correspondentes conexoes

interconexao->recursoTransportado = interconexao->capacidadeMax ∗ adaptador->recursoRecebido / soma de todas as capacidadeMax

Essa funcao espera que a lista de interconexoes ja foi previamente estabelecida

Parameters

listaAlvo	ponteiro para o inico da lista de adaptadores;

Funcao: defineDistribuicao

AssertivaEntrada: adaptadorVazio(listaAlvo) == NAO_VAZIO; para cada adaptador da lista: adaptador->saidas[i] != null;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: definicao da distribuicao do adaptador

Interfaces explicitas: void, defineDistribuicao, Adaptador ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de adaptadores < conexoes do adaptador

< quantidade de recurso transportado

Assertiva estrutural: aux é a lista nao-nula de adaptadores

AE: aux nao chegou ao fim da lista de adaptadores

Assertiva estrutural: somatorio eh a soma das capacidades maximas do adaptador corrente

AE: o adaptador corrente possui saidas

Assertiva estrutural: conexao eh a lista de saidas do adaptador corrente

Comentarios de argumentacao

Enquanto percorre a lista de conexoes que saem do adaptador somam-se a capaxidade maxima de todas as conexoes

AE: a lista de saidas do adaptador nao chegou ao fim

AS: a lista de saidas do adaptador chegou ao fim

Assertiva estrutural: conexao eh a lista de saidas do adaptador corrente

AE: a lista de saidas do adaptador nao chegou ao fim

AE: a saida corrente do adaptador nao possui falha

Assertiva estrutural: recursoTransportado é a quantidade de recurso que cada conexao vai transportar no turno

Comentarios de argumentacao

Se a a capacidade maxima da saida corrente do adaptador for maior ou igual ao recursoTransportado Entao o recursoTransportado da saida corrente do adaptador recebe o recursoTransportado Senao recebe a capacidade-Maxima da saida corrente do adaptador FimSe

AS: a saida corrente do adaptador possui falha

AS: a lista de saidas do adaptador chegou ao fim

AS: a lista de adaptadores chegou ao fim

5.1.1.4 void imprimeListaAdaptador (Adaptador * *listaAlvo*)

Funcao: imprimeListaAdaptador

Imprime de todas as celulas de lista de adaptador as respectivas caracteristicas: nome posicao x posicao y recurso recebido quantidade de saidas

Parameters

listaAlvo	lista que sera impressa

Assertiva de entrada: A lista nao deve ser vazia

Assertiva de saida: Se a lista de adaptadores a ser imprimida nao eh vazia Entao ela eh imprimida Senao a lista de adaptadores nao eh imprimida FimSe

Funcao: imprimeListaAdaptador (Iterador)

AssertivaEntrada: adaptadorVazio(listaAlvo) == NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: impressao da lista de adaptadores

Interfaces explicitas: void, imprimeListaAdaptador, Adaptador *listaAlvo

Interfaces implicitas: listaAlvo - lista de adaptadores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Imprime os atributos do adaptador corrente

AS: listaAlvo chegou ao fim

5.1.1.5 Adaptador* insereAdaptador (char * *registro,* Adaptador * *listaAlvo*)

Insere uma nova celula de adaptador na lista dos adaptadores

Parameters

registro	String que sera decodificada e inserida
listaAlvo	lista a qual essa nova celula sera inserida

Returns

Adaptador novo ponteiro de referencia para o inico da lista

Assertiva de entrada: registro - eh um vetor contendo o conteudo do txt, deve ser diferente de NULL

Assertiva de saida: A lista recebida pela funcao, deve ser o proximo adaptador apontado pela lista retornada

Funcao: insereAdaptador

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir um novo adaptador na lista de adaptadores

Interfaces explicitas: Adaptador$*$, insereAdaptador, char $*$registro, Adaptador $*$listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de adaptadores $<$ Alocacao de um novo adaptador

$<$ Alocacao de um vetor do tamanho do registro

$<$ Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome do novo adaptador possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos do adaptador

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 4

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome do adaptador Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos da adaptador

Comentarios de argumentacao

Os atributos nao lidos do adaptador inserido recebem o valor nulo, e o proximo adaptador da lista que contem o novo adaptador inserido na cabeca recebe a lista de adaptadores atual

5.1.1.6 void liberaListaAdaptador (Adaptador * *listaAlvo*)

Libera o espaco de memoria alocado para a lista de adaptadores

Parameters

listaAlvo	lista a qual sera desalocada

Assertiva de entrada: A lista nao deve ser vazia

Assertiva de saida: A lista deve estar vazia

Funcao: liberaListaAdaptador (Iterador)

AssertivaEntrada: adaptadorVazio(listaAlvo) == NAO_VAZIO;

AssertivaSaida: adaptadorVazio(aux1) == VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: liberacao da lista de adaptadores

Interfaces explicitas: void, liberaListaAdaptador, Adaptador *listaAlvo

Interfaces implicitas: listaAlvo - lista de adaptadores Asseriva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Adaptador corrente

AS: listaAlvo chegou ao fim

5.1.1.7 void mandarRecursoAdaptado (Adaptador * *listaAlvo*)

Faz o recurso chegar ate as cidades ou adaptadores de destino, faz isso para toda lista de adaptadores

Essa funcao espera que a lista de interconexoes ja foi previamente estabelecida e que a definicao de como sera a distribuicao ja esteja previamente definida

Parameters

listaAlvo	ponteiro para o inico da lista de adaptadores;

Funcao: mandarRecursoAdaptado

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Adaptador

A estrategia de distribuicao ja foi previamente definida e as interconxoes ligadas diretamente cada celula do adaptador ja esta com o recursoTransportado preenchido

Requisitos: alterar a interconexao ligada a cada celula da lista de geradores com o valor que sera transportado Altera o adaptador ou cidade de destino com o recurso que sera recebido

Interfaces explicitas: void, mandarRecursoAdaptado Interfaces implicitas:

listaAlvo - lista de adaptadores AE : percorrendo listaAlvo

AE: precorre a lista de conexoes que saem de cada adaptador

AE: vai ater a ulcima interconexao antes do destino

Assertiva de argumentacao

Verifica se a proxima Interconexao nao esta falha se nao esta define o recurso que ela ira transportar

AE : verifica se a proxima conexao nao eh falha

AE : se nao for falho define quanto de recurso sera transportado

AE : posterior esta falho, recursoTransportado = 0

se a saida eh uma cidade

ou se a saida é um adaptador

5.2 /home/eduardo/TrabalhoMP/app/header/Cidades.h File Reference

```
#include "Adaptadores.h"
```

Include dependency graph for Cidades.h: This graph shows which files directly or indirectly include this file:

Functions

- Cidade * criaListaCidade ()
- Vazia cidadeVazia (Cidade *)
- Cidade * insereCidade (char *, Cidade *)
- void imprimeListaCidade (Cidade *)
- void liberaListaCidade (Cidade *)
- int recursoGastoTotal (Cidade *)
- int numeroCidades (Cidade *)
- void gerenciaRecursoRecebido (Cidade *)
- int numeroCidadesNegativadas (Cidade *)
- int tempoSemRecursoNecessario (Cidade *)
- int numeroCidadesNoVermelho (Cidade *)
- int tempoCidadesNoVermelho (Cidade *)

5.2.1 Function Documentation

5.2.1.1 Vazia cidadeVazia (Cidade * *listaAlvo*)

Funcao: cidadeVazia

Verifica se a lista de cidades esta vazia

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Returns

variavel do tipo Vazia, indicando se a lista esta vazia

Assertiva de entrada: estrutura do tipo Cidade

Assertiva de saida: condicao da Cidade sendo vazia ou nao vazia

Funcao: cidadeVazia (Iterador)

AssertivaSaida: VAZIA || NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazia, cidadeVazia, Cidade *listaAlvo

Interfaces implicitas: Vazia - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de cidades AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.2.1.2 Cidade* criaListaCidade ( )

Funcao: criaListaCidade

Inicia um ponteiro que sera para Cidade

Returns

null

Assertiva de saida: estrutura do tipo Cidade nula

Funcao: criaListaCidade (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Cidade

Interfaces explicitas: Cidade*, criaListaCidade

5.2.1.3 void gerenciaRecursoRecebido (Cidade * *listaAlvo*)

Funcao: gerenciaRecursoRecebido

Altera os atributos das celulas de cidade de acordo com o recurso que foi recebido

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Assertiva de entrada: estrutura do tipo Cidade

Funcao: gerenciaRecursoRecebido

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: gerencia o recurso recebido, aterando o valor do recurso gasto

Interfaces explicitas: int, gerenciaRecursoRecebido, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE : percorrer a lista de cidades

AE : se a porcentagem do recurso Necessario eh positiva

AE: se a porcentagem eh negativa mas nao critica

AE: se a porcentagem eh critica

5.2.1.4 void imprimeListaCidade (Cidade * *listaAlvo*)

Funcao: imprimeListaCidade

Imprime de todas as celulas de lista de cidade as respectivas caracteristicas: nome posicao x posicao y recurso necessario recurso recebido recurso gasto

Parameters

listaAlvo	lista que sera impressa

AssertivaEntrada: A lista nao deve ser vazia

AssertivaSaida: Se a lista de cidades a ser imprimida nao eh vazia Entao ela eh imprimida Senao a lista de cidades nao eh imprimida FimSe

Funcao: imprimeListaCidade (Iterador)

AssertivaEntrada: cidadeVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: impressao da lista de cidades

Interfaces explicitas: void, imprimeListaCidade, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Imprime os atributos da cidade corrente

AS: listaAlvo chegou ao fim

5.2.1.5 Cidade* insereCidade (char * *registro,* Cidade * *listaAlvo*)

Funcao: insereCidade

Inserir uma nova celula na lista de cidades a insercao se da pelo inicio da lista e é retornadado o novo ponterio para lista.

Parameters

registro	string que sera lida do arquivo representando Cidade
listaAlvo	lista de cidades onde a nova celula sera inserida

Returns

novo pontero para a o inicio da lista de cidades

Assertiva de entrada: registro - eh um vetor contendo o conteudo do txt, deve ser diferente de NULL

Assertiva de saida: A lista recebida pela funcao, deve ser a proxima cidade apontada pela lista retornada

Funcao: insereCidade

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir uma nova cidade na lista de cidades

Interfaces explicitas: Cidade*, insereCidade, char *registro, Cidade *listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de cidades < Alocacao da nova cidade

< Alocacao de um vetor do tamanho do registro

< Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome da nova cidade possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos da cidade

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 4

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome da cidade Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos da cidade

Comentarios de argumentacao

Os atributos nao lidos da cidade inserida recebem o valor nulo, e a proxima cidade da lista que contem a nova cidade inserido na cabeca recebe a lista de cidades atual

5.2.1.6 void liberaListaCidade (Cidade * *listaAlvo*)

Funcao: liberaListaCidade

Desaloca a memoria reservada para toda celula pertecente a lista de cidades

Parameters

listaAlvo	lista a ser desalocada

Assertiva de entrada: A lista nao deve ser vazia

Assertiva de saida: A lista deve estar vazia

Funcao: liberaListaCidade (Iterador)

AssertivaEntrada: cidadeVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: cidadeVazia(aux1) == VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: liberacao da lista de cidades

Interfaces explicitas: void, liberaListaCidade, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Cidade corrente

AS: listaAlvo chegou ao fim

5.2.1.7 int numeroCidades (Cidade * *listaAlvo*)

Funcao: numeroCidades

Calcula o numero total de cidades na lista de cidades

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Returns

Numero total de cidades

Assertiva de entrada: estrutura do tipo Cidade

Assertiva de saida: numero total de cidades na lista de cidades

Funcao: numeroCidades (Iterador)

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: conta o numero total de cidades em uma lista de cidades

Interfaces explicitas: int, numeroCidades, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Contagem do numero de total de cidades na listaAlvo

AS: listaAlvo chegou ao fim

5.2.1.8 int numeroCidadesNegativadas (Cidade * *listaAlvo*)

Funcao: numeroCidadesNegativadas

Resultado da contagem de quantas cidades ficaram sem recurso necessario

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Returns

total de cidades que ficaram com menos recurso que o necessario

Assertiva de entrada: estrutura do tipo Cidade

Funcao: numeroCidadesNegativadas

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar quantas cidades tiveram seus recursos abaixo do necessario

Interfaces explicitas: int, numeroCidadesNegativadas, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer a lista de cidade

AE : teve algum turno negativado

5.2.1.9 int numeroCidadesNoVermelho (Cidade * *listaAlvo*)

Funcao: numeroCidadesNoVermelho

Resultado da contagem de quantas cidades receberam menos de 30% do recurso necessario

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Returns

total de cidades que receberam menos de 30% do recurso necessario

Assertiva de entrada: estrutura do tipo Cidade

Funcao: numeroCidadesNoVermelho

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar quantas cidades tiveram seus recursos abaixo de 30% do necessario

Interfaces explicitas: int, numeroCidades, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer a lista de cidade

AE : teve algum turno no vermelho

5.2.1.10 int recursoGastoTotal (Cidade * *listaAlvo*)

Funcao: recursoGastoTotal

Resultado da soma de todos recursos gasto pelas cidades

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Returns

total de recursos gasto pelas cidades

Assertiva de entrada: estrutura do tipo Cidade

Assertiva de saida:

Funcao: recursoGastoTotal

AssertivaEntrada: cidadeVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: total >= 0;

Hipotese: cidade - ponteiro para o inicio da lista de cidades

Requisitos: Somar os recursos gatos por todas as cidades

Interfaces explicitas: int, recursoGastoTotal, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: lista nao chegou ao fim

Comentarios de argumentacao

Incrementa o somatorio com o valor do recurso gasto pela celula atual muda a referencia de cidade para a proxima celula

AS: lista chegou ao fim

5.2.1.11 int tempoCidadesNoVermelho (Cidade * *listaAlvo*)

Funcao: tempoCidadesNoVermelho

Resultado da soma do tempo que as cidades passaram com menos de 30% do recurso necessario

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Returns

total da soma dos tempos que as cidades ficaram com menos de 30% do recurso necessario

Assertiva de entrada: estrutura do tipo Cidade

Funcao: tempoSemRecursoNecessario

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar a soma dos tempos que as cidades ficaram com menos de 30% do recurso necessario

Interfaces explicitas: int, tempoSemRecursoNecessario, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer lista de cidades

5.2.1.12 int tempoSemRecursoNecessario (Cidade * *listaAlvo*)

Funcao: tempoSemRecursoNecessario

Resultado da soma do tempo de turnos que as Cidades ficaram sem recurso necessario

Parameters

listaAlvo	ponteiro para o inicio da lista de cidades

Returns

total da soma dos tempos que as cidades ficaram sem recurso necessario

Assertiva de entrada: estrutura do tipo Cidade

Funcao: tempoSemRecursoNecessario

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar a soma dos tempos que as cidades ficaram sem recurso

Interfaces explicitas: int, tempoSemRecursoNecessario, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer lista de cidades

5.3 /home/eduardo/TrabalhoMP/app/header/Geradores.h File Reference

```
#include "Cidades.h"
```

Include dependency graph for Geradores.h: This graph shows which files directly or indirectly include this file:

Functions

- Gerador * criaListaGerador ()
- Vazio geradorVazio (Gerador *)

- Gerador ∗ insereGerador (char ∗, Gerador ∗)
- void imprimeListaGerador (Gerador ∗)
- void liberaListaGerador (Gerador ∗)
- int recursoProduzidoTotal (Gerador ∗)
- int custoGeradores (Gerador ∗)
- void mandarRecursoProduzido (Gerador ∗)
- int numeroGeradores (Gerador ∗)

5.3.1 Function Documentation

5.3.1.1 Gerador∗ criaListaGerador ( )

Funcao: criaListaGerador

Inicia um ponteiro que sera para Gerador

Returns

null

Assertiva de saida: estrutura do tipo Gerador nula

Funcao: criaListaGerador (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Gerador

Interfaces explicitas: Gerador∗, criaListaGerador

5.3.1.2 int custoGeradores (Gerador ∗ *listaAlvo*)

Funcao: custoGeradores

Calcula a soma de custo por segundo de todos os geradores da lista

Parameters

listaAlvo	ponteiro para o inicio da lista de geradores

Returns

Soma do custo por segundo de todos os geradores

Assertiva de entrada: estrutura do tipo Gerador

Assertiva de saida: custo total dos geradores

Funcao: custoGeradores

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: realiza a soma dos custos dos geradores

Interfaces explicitas: int, custoGeradores, Gerador ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Somando os custos de cada gerador da lista de geradores

5.3.1.3 Vazio geradorVazio (Gerador * *listaAlvo*)

Funcao: geradorVazio

Verifica se a lista de geradores esta vazia

Parameters

listaAlvo	ponteiro para o inicio da lista de geradores

Returns

variavel do tipo Vazio, indicando se a lista esta vazia

Assertiva de entrada: estrutura do tipo Gerador

Assertiva de saida: condicao do Gerador sendo vazio ou nao vazio

Funcao: geradorVazio (Iterador)

AssertivaSaida: VAZIO || NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazio, geradorVazio, Gerador *listaAlvo

Interfaces implicitas: Vazio - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de geradores AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.3.1.4 void imprimeListaGerador (Gerador * *listaAlvo*)

Funcao: imprimeListaGerador

Imprime de todas as celulas de lista de gerador as respectivas caracteristicas: nome posicao x posicao y taxa de producao recurso produzido custo

Parameters

listaAlvo	lista que sera impressa

AssertivaEntrada: A lista nao deve ser vazia

AssertivaSaida: Se a lista de geradores a ser imprimida nao eh vazia Entao ela eh imprimida Senao a lista de geradores nao eh imprimida FimSe

Funcao: imprimeListaGerador

AssertivaEntrada: geradorVazio(listaAlvo) == NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: impressao da lista de geradores

Interfaces explicitas: void, imprimeListaGerador, Gerador *listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Imprime os atributos do gerador corrente

AS: listaAlvo chegou ao fim

5.3.1.5 Gerador* insereGerador (char * *registro,* Gerador * *listaAlvo*)

Funcao: insereGerador

Inserir uma nova celula na lista de geradores a insercao se da pelo inicio da lista e é retornadado o novo ponterio para lista.

Parameters

registro	string que sera lida do arquivo representando Gerador
listaAlvo	lista de cidade a qual a nova celula sera inserida

Returns

novo pontero para a o inicio da lista de geradores

Assertiva de entrada: registro - eh um vetor contendo o conteudo do txt, deve ser diferente de NULL

Assertiva de saida: A lista recebida pela funcao, deve ser o proximo gerador apontado pela lista retornada

Funcao: insereGerador

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir um novo gerador na lista de geradores

Interfaces explicitas: Gerador*, insereGerador, char *registro, Gerador *listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de geradores < Alocacao de um novo gerador

< Alocacao de um vetor do tamanho do registro

< Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome do novo adaptador possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos do gerador

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 5

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome do gerador Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos do gerador

Comentarios de argumentacao

Os atributos nao lidos do gerador inserido recebem o valor nulo, e o proximo gerador da lista que contem o novo gerador inserido na cabeca recebe a lista de geradores atual

5.3.1.6 void liberaListaGerador (Gerador * *listaAlvo*)

Funcao: liberaListaGerador

Desaloca a memoria reservada para toda celula pertecente a lista de geradores

Parameters

listaAlvo	lista a ser desalocada

AssertivaEntrada: A lista nao deve ser vazia

AssertivaSaida: A lista deve estar vazia

Funcao: liberaListaGerador (Iterador)

AssertivaEntrada: geradorVazio(listaAlvo) == NAO_VAZIO;

AssertivaSaida: geradorVazio(listaAlvo) == VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: liberacao da lista de geradores

Interfaces explicitas: void, liberaListaGerador, Gerador *listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Gerador corrente

AS: listaAlvo chegou ao fim

5.3.1.7 void mandarRecursoProduzido (Gerador * *listaAlvo*)

Funcao: mandarRecursoProduzido

Altera os nos de conexão do grafo com o recurso enviado Altera os adaptadores com o recurso recebido

Parameters

listaAlvo	ponteiro para o inicio da lista de geradores

Funcao: mandarRecursoProduzido

AssertivaEntrada: geradorVazio(listaAlvo) == NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: alterar a interconexao ligada a cada celula da lista de geradores com o valor que sera transportado Altera o adaptador de destino com o recurso que sera recebido

Interfaces explicitas: void, mandarRecursoProduzido, Gerador ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores AE: nao chegou ao final da listaAlvo

AE: o elemento possui saidas

AE: interconexao ligada a essa celula nao teve falha

AE: gerador possui uma saida

AE: laço que cobre os casos de uma interconexao apontar para outra antes

AE: uma interconexao aponta para outra;

Assertiva de argumentacao

Verifica se a proxima Interconexao nao esta falha se nao esta define o recurso que ela ira transportar

AE : verifica se a proxima interconexao nao esta falha

AE: conexao nao falha

AE : conexao Falha transporta 0 de recurso

AE: chegou na ultima interconexao depois do gerador;

AE: a interconexao aponta para o adaptador

AE: interconexao ligada a essa celula falhou

AE: tem saida

AE: percorrrer a lista de inteconexao das saidas

AE: a interconexao aponta para o adaptador

AS: chegou ao final da listaAlvo

5.3.1.8 int numeroGeradores (Gerador ∗ *listaAlvo*)

Funcao: numeroGeradores

Calcula o numero total de geradores na lista de geradores

Parameters

listaAlvo	ponteiro para o inicio da lista de geradores

Returns

Numero total de geradores

Assertiva de entrada: estrutura do tipo Gerador

Assertiva de saida: numero total de geradores na lista de geradores

Funcao: numeroGeradores (Iterador)

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: conta o numero total de geradores em uma lista de geradores

Interfaces explicitas: int, numeroGeradores, Gerador ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Contagem do numero de total de geradores na listaAlvo

5.3.1.9 int recursoProduzidoTotal (Gerador * *listaAlvo*)

Funcao: recursoProduzidoTotal

Calcula a soma do total de recursos produzido pelas celulas.

Parameters

listaAlvo	inicio da lista de geradores;

Returns

soma de todo o recurso produzido pelos geradores

Assertiva de saida: energia total produzida por todos os geradores

Funcao: numeroGeradores (Iterador)

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: realiza o calculo da energia total gerada

Interfaces explicitas: int, recursoProduzidoTotal, Gerador *listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Somando os recursos produzidos para obter a energia total gerada

5.4 /home/eduardo/TrabalhoMP/app/header/Geral.h File Reference

```
#include "Interconexoes.h"
```

Include dependency graph for Geral.h: This graph shows which files directly or indirectly include this file:

Functions

- void conecta (Cidade *, Gerador *, Interconexao *, Adaptador *)
- void verifica (Cidade *, Gerador *, Interconexao *, Adaptador *)

5.4.1 Function Documentation

5.4.1.1 void conecta (Cidade * *cidades*, Gerador * *geradores*, Interconexao * *interconexoes*, Adaptador * *adaptadores*)

Funcao: conecta

Realiza a conexao entre as listas

Parameters

cidade	lista de cidades
gerador	lista de geradores
interconexao	lista de interconexoes
adaptador	lista de adaptadores

Assertiva de entrada:

Assertiva de saida:

Funcao: conecta

Hipóteses: cidades - ponteiro para uma lista do tipo Cidade geradores - ponteiro para uma lista do tipo Gerador interconexoes - ponteiro para uma lista do tipo Interconexao adaptadores - ponteiro para uma lista do tipo Adaptador

Requisitos: realizar a conexao entre as listas, para gerar a rede de distribuicao

Interfaces explicitas: void, conecta, Cidade ∗cidades, Gerador ∗geradores, Interconexao ∗interconexoes, Adaptador ∗adaptadores

Interfaces implicitas: cidades - lista de cidades geradores - lista de geradores interconexoes - lista de interconexoes adaptadores - lista de adaptadores < Ponteiro auxiliar para cidade

< Ponteiro auxiliar para gerador

< Ponteiro auxiliar para interconexao

< Ponteiro auxiliar para interconexao

< Ponteiro auxiliar para adaptador

AE: o auxI1 ainda nao chegou ao fim da lista de interconexoes

AE: o auxA ainda nao chegou ao fim da lista de adaptadores

AE: a posicao inicial da interconexao apontada por auxI1 coincide com a posicao do adaptador apontado por auxA

Comentarios de argumentacao

insere-se o adaptador como entrada da interconexao insere-se a interconexao na lista de saidas do adaptador

AE: a posicao final da interconexao apontada por auxI1 coincide com a posicao do adaptador apontado por auxA

Comentarios de argumentacao

insere-se o adaptador como saida da interconexao insere-se a interconexao na lista de entradas do adaptador

AS: o auxA chegou ao fim da lista de adaptadores

AE: o auxG ainda nao chegou ao fim da lista de geradores

AE: a posicao inicial da interconexao apontada por auxI1 coincide com a posicao do gerador apontado por auxG

Comentarios de argumentacao

insere-se a interconexao como saida do gerador insere-se o gerador como entrada da interconexao

AS: o auxG chegou ao fim da lista de geradores

AE: o auxC ainda nao chegou ao fim da lista de cidades

AE: a posicao final da interconexao apontada por auxI1 coincide com a posicao da cidade apontado por auxC

Comentarios de argumentacao

insere-se a cidade como saida da lista de interconexao insere-se a interconexao na lista de entradas da cidade

AS: o auxG chegou ao fim da lista de geradores

AE: o auxI2 ainda nao chegou ao fim da lista de interconexoes

AE: a posicao inicial da interconexao apontada por auxI1 coincide com a posicao final da interconexao apontada por auxI2

Comentarios de argumentacao

insere-se a interconexao auxI1 como saida da interconexao auxI2 insere-se a interconexao auxI2 como entrada da interconexao auxI1

AE: a posicao final da interconexao apontada por auxI1 coincide com a posicao inicial da interconexao apontada por auxI2

Comentarios de argumentacao

insere-se a interconexao auxI2 como saida da interconexao auxI1 insere-se a interconexao auxI1 como entrada da interconexao auxI2

AS: o auxI2 chegou ao fim da lista de interconexoes

AS: o auxI1 chegou ao fim da lista de interconexoes

5.4.1.2 void verifica (Cidade * *cidades,* Gerador * *geradores,* Interconexao * *interconexoes,* Adaptador * *adaptadores*)

Funcao: verifica

Realiza a verificacao das conexoes, checando se as listas estao vazias

Parameters

cidade	lista de cidades
gerador	lista de geradores
interconexao	lista de interconexoes
adaptador	lista de adaptadores

Assertiva de entrada:

Assertiva de saida:

Funcao: verfica

Hipóteses: cidades - ponteiro para uma lista do tipo Cidade geradores - ponteiro para uma lista do tipo Gerador interconexoes - ponteiro para uma lista do tipo Interconexao adaptadores - ponteiro para uma lista do tipo Adaptador

Requisitos: verificar as conexoes e gerar um arquivo com as inconsistencias encontradas

Interfaces explicitas: void, conecta, Cidade *cidades, Gerador *geradores, Interconexao *interconexoes, Adaptador *adaptadores

Interfaces implicitas: cidades - lista de cidades geradores - lista de geradores interconexoes - lista de interconexoes adaptadores - lista de adaptadores < Ponteiro auxiliar para cidade

< Ponteiro auxiliar para gerador

< Ponteiro auxiliar para interconexao

< Ponteiro auxiliar para adaptador

AE: a lista de cidades nao eh vazia

AE: o auxC ainda nao chegou ao fim da lista de cidades

AE: a cidade corrente nao posssui entradas

AS: o auxC chegou ao fim da lista de cidades

AE: a lista de cidades eh vazia

AE: a lista de geradores nao eh vazia

AE: o auxG ainda nao chegou ao fim da lista de geradores

AE: o gerador corrente nao posssui saidas

AS: o auxG chegou ao fim da lista de geradores

AE: a lista de geradores eh vazia

AE: a lista de interconexoes nao eh vazia

AE: o auxI ainda nao chegou ao fim da lista de interconexoes

AE: a interconexao nao possui entradas em geradores, adaptadores e interconexoes

AE: a interconexao nao possui saidas em geradores, adaptadores e interconexoes

AS: o auxI chegou ao fim da lista de interconexoes

AE: a lista de interconexoes eh vazia

AE: a lista de adaptadores nao eh vazia

AE: o auxA ainda nao chegou ao fim da lista de adaptadores

AE: o adaptador corrente nao posssui entradas

AE: o adaptador corrente nao posssui saidas

AS: o auxA chegou ao fim da lista de adaptadores

AE: a lista de adaptadores eh vazia

5.5 /home/eduardo/TrabalhoMP/app/header/Interconexoes.h File Reference

`#include "Geradores.h"`
Include dependency graph for Interconexoes.h: This graph shows which files directly or indirectly include this file:

Functions

- Interconexao * criaListaInterconexao ()
- Vazia interconexaoVazia (Interconexao *)
- Interconexao * insereInterconexao (char *, Interconexao *)
- void imprimeListaInterconexao (Interconexao *)
- void liberaListaInterconexao (Interconexao *)
- float tamanhoConexao (Interconexao *)
- float tamanhoTotalConexao (Interconexao *)
- int totalGastoConserto (Interconexao *)
- Falha calculaFalha (Interconexao *)
- void mandarRecursoTransportado (Interconexao *)
- int custoGastoComConserto (Interconexao *)
- int numeroTotalFalhas (Interconexao *)
- void gerenciaFalhas (Interconexao *)

5.5.1 Function Documentation

5.5.1.1 Falha calculaFalha (Interconexao * *listaAlvo*)

Funcao: calculaFalha

Calcula a possibilidade de falha

Parameters

listaAlvo	celula a qual vai ser calculada a falha

Returns

variavel do tipo Falha, indicando se houve falha

Assertiva de entrada: a chance de falha deve estar entre 0 e 1

Assertiva de saida: Se a chance de falha for maior que 0 e maior que um numero aleatorio Entao ocorre um falha Senao nao ocorre uma falha FimSe

Funcao: calculaFalha

AssertivaEntrada: conexao->chanceFalha >= 0 && conexao->chanceFalha <= 1;

AssertivaSaida: FALHA || SEM_FALHA;

Requisitos: calculo da chance de falha

Interfaces explicitas: Falha, calculaFalha, Interconexao ∗conexao

Interfaces implicitas: Falha - tipo de dado, indicando se houve falha ou nao listaAlvo - lista de interconexoes Asseriva estrutural: num eh um numero gerado aleatoriamente

5.5.1.2 Interconexao∗ criaListaInterconexao ( )

Funcao: criaListaInterconexao

Inicia um ponteiro que sera para Interconexao

AssertivaSaida: NULL;

Funcao: criaListaInterconexao (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Interconexao

Interfaces explicitas: Interconexao∗, criaListaInterconexao

5.5.1.3 int custoGastoComConserto (Interconexao ∗ *listaAlvo*)

Funcao: custoGastoComConserto

Calcula o custo que foi gasto com o conserto das interconexoes

Parameters

listaAlvo	ponterio de referencia para o inicio da lista de interconexoes

Assertiva de entrada: interconexao - eh uma lista de interconexoes nao vazia

Assertiva de saida: valor gasto com os consertos das interconexoes

Funcao: custoGastoComConserto

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponterio para o inicio da lista do tipo Interconexao

Requisitos: calcular o quanto foi gasto com concerto das interconexoes

Interfaces explicitas: int, custoGastoComConcerto, Interconexao ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE : listaAlvo nao chegou ao fim

Comentarios de argumentacao

Incrementa o total com o resultado da multiplicacao do numero de falhas com o custo gasto com o conserto de cada falha

AS: listaAlvo chegou ao fim

5.5.1.4 void gerenciaFalhas (Interconexao ∗ *listaAlvo*)

Funcao: gerenciaFalhas

Marca as celulas que falharam como falhas e comtabilizam as celulas que estao no concerto o tempo que falta para sairem

Parameters

listaAlvo	ponterio de referencia para o inicio da lista de interconexoes

Assertiva de entrada: interconexao - eh uma lista de interconexoes nao vazia

Assertiva de saida: Se a interconexao corrente nao falhou Entao Se a chance desta interconexao falhar der FALHA Entao zera o contador de tempo de conserto numero de falhas + 1 tagFalha = falha FimSe Senao contador de tempo de conserto + 1 Se contador de tempo de conserto alcançar ou ultrapassar o tempo de conserto zera o contador de tempo de conserto tagFalha = sem falha Fimse

Funcao: gerenciaFalhas

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponterio para o inicio da lista do tipo Interconexao

Requisitos: marcar as celulas que falharam como falhas e contabilizar o concertos das celulas que estao paradas

Interfaces explicitas: int, numeroTotalFalhas, Interconexao ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes AE: listaAlvo nao chegou ao fim

AE: interconexao corrente nao falhou

AE: chance falha da interconexao corrente eh FALHA

AE: interconexao corrente falhou

AE: tempo de conserto foi atingido pelo contador

AS: listaAlvo chegou ao fim

5.5.1.5 void imprimeListaInterconexao (Interconexao ∗ *listaAlvo*)

Funcao: imprimeListaInterconexao

Imprime de todas as celulas de lista de interconexoes as respectivas caracteristicas: nome posicao inicial x posicao inicial y posicao final x posicao final y tag de destino chance de falha tempo de concerto custo de concerto numero de falhas tag de falha capacidade maxima recurso transportado

Parameters

listaAlvo	lista que sera impressa

AssertivaEntrada: A lista nao deve ser vazia

AssertivaSaida: Se a lista de interconexoes a ser imprimida nao eh vazia Entao ela eh imprimida Senao a lista de interconexoes nao eh imprimida FimSe

Funcao: imprimeListaInterconexao (Iterador)

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: impressao da lista de interconexoes

Interfaces explicitas: void, imprimeListaInterconexao, Interconexao ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Imprime os atributos da interconexao corrente

AS: listaAlvo chegou ao fim

5.5.1.6 Interconexao* insereInterconexao (char * *registro,* Interconexao * *listaAlvo*)

Funcao: insereInterconexao

Inserir uma nova celula na lista de interconexoes a insercao se da pelo inicio da lista e é retornadado o novo ponterio para lista.

Parameters

registro	string que sera lida do arquivo representando Inteconexao
listaAlvo	lista de interconexoes onde a nova celula sera inserida

Returns

novo pontero para a o inicio da lista de interconexoes

Assertiva de entrada: registro - eh um vetor contendo o conteudo do txt, deve ser diferente de NULL

Assertiva de saida: A lista recebida pela funcao, deve ser a proxima interconexao apontada pela lista retornada

Funcao: insereInterconexao

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir um novo gerador na lista de interconexoes

Interfaces explicitas: Interconexao*, insereInterconexao, char *registro, Interconexao *listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de interconexoes < Alocacao da nova Interconexao

< Variavel auxiliar para percorrer a lista e inserir o elemento no final

< Alocacao de um vetor do tamanho do registro

< Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome da nova interconexao possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos da interconexao

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 8

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome da interconexao Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos da interconexao

Comentarios de argumentacao

Os atributos nao lidos da inteconexao inserida recebem o valor nulo, e a proxima interconexao da lista que contem a nova inteconexao inserido na cabeca recebe a lista de interconexoes atual

AE: se a listaAlvo nao for vazia

AE: o aux nao chegou ao final da lista de interconexoes

AE: se a listaAlvo for vazia

5.5.1.7 Vazia interconexaoVazia (Interconexao * *listaAlvo*)

Funcao: interconexaoVazia

Verifica se a lista de interconexoes esta vazia

Parameters

listaAlvo	ponteiro para o inicio da lista de interconexoes

Returns

variavel do tipo Vazia, indicando se a lista esta vazia

Assertiva de entrada: estrutura do tipo Interconexao

Assertiva de saida: condicao da Interconexao sendo vazia ou nao vazia

Funcao: interconexaoVazia (Iterador)

AssertivaSaida: VAZIA || NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazia, interconexaoVazia, Interconexao *listaAlvo

Interfaces implicitas: Vazia - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de interconexoes AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.5.1.8 void liberaListaInterconexao (Interconexao * *listaAlvo*)

Funcao: liberaListaInterconexao

Desaloca a memoria reservada para toda celula pertecente a lista de interconexoes

Parameters

listaAlvo	lista a ser desalocada

AssertivaEntrada: A lista nao deve ser vazia

AssertivaSaida: A lista deve estar vazia

Funcao: liberaListaInterconexao (Iterador)

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: interconexaoVazia(listaAlvo) == VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: liberacao da lista de interconexoes

Interfaces explicitas: void, liberaListaInterconexao, Interconexao ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Assertiva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Interconexao corrente

AS: listaAlvo chegou ao fim

5.5.1.9 void mandarRecursoTransportado (**Interconexao** ∗ *listaAlvo*)

Funcao: mandarRecursoTransportado

Muda cada referencia seja para adaptador ou para cidade, dependenodo do em qual esta ligada, alterando o valor do recurso atual.

Parameters

listaAlvo	ponterio de referencia para o inicio da lista de interconexoes

Assertiva de entrada: interconexao - eh uma lista de interconexoes nao vazia

Assertiva de saida: Se o destino da interconexao for um adaptador Entao soma-se o recurso transportado ao Adaptador da lista de interconexoes Senao soma-se o recurso transportado ao Adaptador da lista de interconexoes FimSe

Funcao: mandarRecursoTransportado

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: transporte dos recursos

Interfaces explicitas: void, mandarRecursoTransportado, Interconexao ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a lista nao-nula de interconexoes

AE: aux nao chegou ao fim da lista de inteconexoes

AE: a interconexao corrente nao possui falha

Comentarios de argumentacao

Enqaunto a lista de interconexoes eh percorrida, dependendo do destino da interconexao, soma-se o recurso ao respectivo destino

AE: o destino da interconexao eh um Adaptador

AE: o destino da interconexao eh uma Cidade

AS: a lista de interconexoes chegou ao fim

5.5.1.10 int numeroTotalFalhas (Interconexao * *listaAlvo*)

Funcao: numeroTotalFalhas

Faz a contabilidade de quantas falas teve durante toda a simulaca

Parameters

listaAlvo	ponterio de referencia para o inicio da lista de interconexoes

Assertiva de entrada: interconexao - eh uma lista de interconexoes nao vazia

Assertiva de saida: total de falhas

Funcao: numeroTotalFalhas

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponterio para o inicio da lista do tipo Interconexao

Requisitos: contabilizar a quantidade de falhas

Interfaces explicitas: int, numeroTotalFalhas, Interconexao *listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo ainda nao acabou Comentarios de argumentacao

Incrementa o somatorio com o numero de falhas da celula atual

5.5.1.11 float tamanhoConexao (Interconexao * *listaAlvo*)

Funcao: tamanhoConexao

Calcula o tamanho da celula de conexao que eh passada para a funcao

Parameters

listaAlvo	ponteiro para a celula de interconexao

Returns

tamanho da celula de conexao que foi passada

Assertiva de entrada: interconexao - eh uma lista de interconexoes nao vazia

Assertiva de saida: tamanho de uma conexao

Funcao: tamanhoConexao

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: distancia > 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: calculo da distancia entre a posicao final e inicial

Interfaces explicitas: float, tamanhoConexao, Interconexao *listaAlvo

Interfaces implicitas: float - valor do tamanho listaAlvo - lista de interconexoes < Valor no eixo x da posicao inicial

< Valor no eixo y da posicao inicial

< Valor no eixo x da posicao final

< Valor no eixo y da posicao final

Comentarios de argumentacao

Calculando a distancia utilizando ((xb-xa)^2 + (yb-ya)^2)^(1/2)

5.5.1.12 float tamanhoTotalConexao (Interconexao * *listaAlvo*)

Funcao: tamanhoTotalConexao

Calcula o tamanho total das conexoes da lista que eh passada

Parameters

listaAlvo	ponteiro para o inicio da lista de celulas de conexao

Returns

retorna o tamanho total das conexoes da lista

Assertiva de entrada: interconexao - eh uma lista de interconexoes nao vazia

Assertiva de saida: resultado da soma dos tamanhos de todas as conexoes

Funcao: tamanhoTotalConexao

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: resultado $>$ 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: calculo da soma dos tamanhos das interconexoes

Interfaces explicitas: float, tamanhoTotalConexao, Interconexao *listaAlvo

Interfaces implicitas: float - valor da soma dos tamanhos listaAlvo - lista de interconexoes Asseriva estrutural: aux é a lista nao-nula de interconexoes

AE: aux nao chegou ao fim da lista de inteconexoes

Comentarios de argumentacao

Enquanto a lista de interconexoes eh percorrida, o tamanho das conexoes sao somados e armazenados na variavel resultado

AS: a lista de interconexoes chegou ao fim

5.5.1.13 int totalGastoConserto (Interconexao *)

Funcao: totalGastoConserto

Calcula o custo total gasto com conserto de todas as celulas de conexao

Parameters

listaAlvo	ponteiro para o inicio da lista de interconexao

Returns

total gasto com conserto de todas as conexoes

5.6 /home/eduardo/TrabalhoMP/app/header/Principal.h File Reference

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

Include dependency graph for Principal.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct cidade
- struct adaptador
- struct interconexao
- struct gerador
- struct relatorio

Typedefs

- typedef enum vazia Vazia
- typedef enum vazio Vazio
- typedef enum falha Falha
- typedef enum destino Destino
- typedef enum condicaoCidade CondicaoCidade
- typedef struct cidade Cidade
- typedef struct adaptador Adaptador
- typedef struct interconexao Interconexao
- typedef struct gerador Gerador
- typedef struct relatorio Relatorio

Enumerations

- enum vazia { VAZIA, NAO_VAZIA }
- enum vazio { VAZIO, NAO_VAZIO }
- enum falha { FALHA, SEM_FALHA }
- enum destino { ADAPTADOR, CIDADE }
- enum condicaoCidade { VERDE, AMARELO, VERMELHO }

5.6.1 Typedef Documentation

5.6.1.1 typedef struct adaptador Adaptador

----------——Adaptadores----------—— Cabecalho do elemento Adaptador

nome: nome do adaptador

posicao: vetor posicao, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

recursoRecebido: quantidade de recurso recebido por segundo pelo Adaptador

entrada: ponteiro que representa a interconexao de entrada do adaptador

saidas: representa as interconexoes realizadas/apontadas pelos adaptadores

quantidadeSaidas: quantidade de conexoes que o adaptador possui

proximo: representa a proxima Adaptador da lista de Adaptadores

5.6.1.2 typedef struct cidade Cidade

----------——Cidades----------—— Cabecalho do elemento Cidade

nome: nome da cidade

posicao: vetor posicao, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

recursoNecessario: quantidade de recurso que a cidade precisa por segundo

recursoRecebido: quantidade de recurso recebido por segundo pela Cidade

recursoGasto: quantidade de recurso que a cidade usou

tagEstado: flag para definir qual eh o estado da cidade (VERMELHO,AMARELO,VERDE)

turnosNegativados: contador para armazenar quantos turnos a cidade ficou fora do VERDE

turnosNoVermelho: contador para armazenar quantos turnos a cidade ficou no VERMELHO

proximo: representa a proxima cidade da rede de cidades

entradas: ponteiro que representa a interconexao de entrada da cidade

5.6.1.3 typedef enum condicaoCidade CondicaoCidade

Enumeracao para detectar qual é o estado que a cidade esta de acordo com o recurso rece

5.6.1.4 typedef enum destino Destino

Enumeracao para detectar qual eh o ponto final de ligacao da conexao

5.6.1.5 typedef enum falha Falha

Enumeracao para detectar uma falha

5.6.1.6 typedef struct gerador Gerador

----------——Geradores----------—— Cabecalho do elemento Gerador

nome: nome do gerador

posicao: vetor posicao, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

taxaProducao: quantidade de recurso que a cidade precisa por segundo

recursoProduzido: quantidade total de recurso produzido pelo gerador

custo: custo de geração por segundo

proximo: representa a proxima cidade da rede de cidades

saida: representa as interconexoes realizadas/apontadas pelos adaptadores

5.6.1.7 typedef struct interconexao Interconexao

----------——Interconexoes----------—— Cabecalho do elemento Inerconexao

nome: nome da interconexao

posicaoInicial: vetor posicaoInicial, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

posicaoFinal: vetor posicaoFinal, representando x na posicao[0] e y na posicao[1], ambas em km, representando tambem a posicao do adaptador na interface

tagDestino: tag para indentificar qual é a ligaçao final de cada conexao, seja cidade ou adaptador

chanceFalha: chance de falha por segundo

tempoConcerto: tempo de concerto em caso de falha em segundos

custoConcerto: custo do concerto em segundos

contadorTempoConserto: contabiliza quanto tempo ja esta no concerto

numeroFalha: total de falhas

tagFalha: indica se houve falha

capacidadeMaxima: capacidade maxima da interconexao

recursoTranstortado: quantidade de recuso que esta sendo transportado pela conexao no turno

proximo: representa a proxima interconexao na lista de interconexoes

entradaAdaptador: aponta, caso a entrada seja um adaptador, para o adaptador cuja saída eh esta interconexao

saidaAdaptador: aponta, caso a saida seja um adaptador, para o adaptador cuja entrada eh esta interconexao

entradaInterconexao: aponta, caso a entrada seja interconexao, para a interconexao cuja saida eh esta interconexao

saidaInterconexao: aponta, caso a saida seja interconexao, para a interconexao cuja entrada eh esta interconexao

proximoEntradaAdaptador: ponteiro para a proxima entrada do adaptador relacionado a interconexao

proximoSaidaAdaptador: ponteiro para a proxima saida do adaptador relacionado a interconexao

proximoEntradaCidade: ponteiro para a proxima cidade de destino

entradaGerador: aponta, caso a entrada seja um gerador, para o gerador cuja saida eh esta interconexao

saidaCidade: aponta, caso a saida seja uma cidade, para a cidade cuja entrada eh esta interconexao

5.6.1.8 typedef struct relatorio Relatorio

----------——Relatorio----------—— Cabecalho onde serao armazenadas as respostas paras as perguntas do relatorio inicial

tempoTotalSimulacao: tempo total da simulacao

custoTotalSimulacao: custo total na simulacao

totalGeradores: total de geradores

energiaTotalGerada: energia total gerada

totalCidades: total de cidades

energiaGastaCidades: energia total gasta pelas cidades

tamanhoTotalInterconexoes: tamanho total das interconexoes

numeroFalhaInterconexoes: numero de falhas nas interconexoes

numeroCidadesNegativadas: numero de cidades que ficaram com menos recurso que o necessário

tempoSemRecurso: tempo que ficaram sem recurso

nomeroCidadesNoVermelho: número de cidades que ficaram com menos de 30% dos recursos

tempoCidadesNoVermelho: tempo que ficaram com menos de 30% de recurso

5.6.1.9 typedef enum vazia Vazia

----------——Enumeracoes----------—— Enumeracao para detectar se a lista esta vazia ou nao

5.6.1.10 typedef enum vazio Vazio

Enumeracao para detectar se a lista esta vazia ou nao

5.6.2 Enumeration Type Documentation

5.6.2.1 enum condicaoCidade

Enumeracao para detectar qual é o estado que a cidade esta de acordo com o recurso rece

Enumerator:

VERDE
AMARELO
VERMELHO

5.6.2.2 enum destino

Enumeracao para detectar qual eh o ponto final de ligacao da conexao

Enumerator:

ADAPTADOR
CIDADE

5.6.2.3 enum falha

Enumeracao para detectar uma falha

Enumerator:

FALHA
SEM_FALHA

5.6.2.4 enum vazia

----------——Enumeracoes----------—— Enumeracao para detectar se a lista esta vazia ou nao

Enumerator:

VAZIA
NAO_VAZIA

5.6.2.5 enum vazio

Enumeracao para detectar se a lista esta vazia ou nao

Enumerator:

VAZIO
NAO_VAZIO

5.7 /home/eduardo/TrabalhoMP/app/src/Adaptadores.c File Reference

```
#include "../header/Adaptadores.h"
#include <assert.h>
```

Include dependency graph for Adaptadores.c:

Functions

- Adaptador * criaListaAdaptador ()
- Vazio adaptadorVazio (Adaptador *listaAlvo)
- Adaptador * insereAdaptador (char *registro, Adaptador *listaAlvo)
- void imprimeListaAdaptador (Adaptador *listaAlvo)
- void liberaListaAdaptador (Adaptador *listaAlvo)
- void defineDistribuicao (Adaptador *listaAlvo)
- void mandarRecursoAdaptado (Adaptador *listaAlvo)

5.7.1 Function Documentation

5.7.1.1 Vazio adaptadorVazio (Adaptador * *listaAlvo*)

Funcao: adaptadorVazio (Iterador)

AssertivaSaida: VAZIO || NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazio, adaptadorVazio, Adaptador *listaAlvo

Interfaces implicitas: Vazio - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de adaptadores AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.7.1.2 Adaptador* criaListaAdaptador ( )

Funcao: criaListaAdaptador (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Adaptador

Interfaces explicitas: Adaptador*, criaListaAdaptador

5.7.1.3 void defineDistribuicao (Adaptador * *listaAlvo*)

Funcao: defineDistribuicao

AssertivaEntrada: adaptadorVazio(listaAlvo) == NAO_VAZIO; para cada adaptador da lista: adaptador->saidas[i] != null;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: definicao da distribuicao do adaptador

Interfaces explicitas: void, defineDistribuicao, Adaptador *listaAlvo

Interfaces implicitas: listaAlvo - lista de adaptadores < conexoes do adaptador

< quantidade de recurso transportado

Assertiva estrutural: aux é a lista nao-nula de adaptadores

AE: aux nao chegou ao fim da lista de adaptadores

Assertiva estrutural: somatorio eh a soma das capacidades maximas do adaptador corrente

AE: o adaptador corrente possui saidas

Assertiva estrutural: conexao eh a lista de saidas do adaptador corrente

Comentarios de argumentacao

Enquanto percorre a lista de conexoes que saem do adaptador somam-se a capaxidade maxima de todas as conexoes

AE: a lista de saidas do adaptador nao chegou ao fim

AS: a lista de saidas do adaptador chegou ao fim

Assertiva estrutural: conexao eh a lista de saidas do adaptador corrente

AE: a lista de saidas do adaptador nao chegou ao fim

AE: a saida corrente do adaptador nao possui falha

Assertiva estrutural: recursoTransportado é a quantidade de recurso que cada conexao vai transportar no turno

Comentarios de argumentacao

Se a a capacidade maxima da saida corrente do adaptador for maior ou igual ao recursoTransportado Entao o recursoTransportado da saida corrente do adaptador recebe o recursoTransportado Senao recebe a capacidade-Maxima da saida corrente do adaptador FimSe

AS: a saida corrente do adaptador possui falha

AS: a lista de saidas do adaptador chegou ao fim

AS: a lista de adaptadores chegou ao fim

5.7.1.4 void imprimeListaAdaptador (Adaptador * *listaAlvo*)

Funcao: imprimeListaAdaptador (Iterador)

AssertivaEntrada: adaptadorVazio(listaAlvo) == NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: impressao da lista de adaptadores

Interfaces explicitas: void, imprimeListaAdaptador, Adaptador *listaAlvo

Interfaces implicitas: listaAlvo - lista de adaptadores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Imprime os atributos do adaptador corrente

AS: listaAlvo chegou ao fim

5.7.1.5 Adaptador* insereAdaptador (char * *registro,* Adaptador * *listaAlvo*)

Funcao: insereAdaptador

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir um novo adaptador na lista de adaptadores

Interfaces explicitas: Adaptador*, insereAdaptador, char *registro, Adaptador *listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de adaptadores < Alocacao de um novo adaptador

< Alocacao de um vetor do tamanho do registro

< Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome do novo adaptador possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos do adaptador

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 4

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome do adaptador Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos da adaptador

Comentarios de argumentacao

Os atributos nao lidos do adaptador inserido recebem o valor nulo, e o proximo adaptador da lista que contem o novo adaptador inserido na cabeca recebe a lista de adaptadores atual

5.7.1.6 void liberaListaAdaptador (**Adaptador** * *listaAlvo*)

Funcao: liberaListaAdaptador (Iterador)

AssertivaEntrada: adaptadorVazio(listaAlvo) == NAO_VAZIO;

AssertivaSaida: adaptadorVazio(aux1) == VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Adaptador

Requisitos: liberacao da lista de adaptadores

Interfaces explicitas: void, liberaListaAdaptador, Adaptador *listaAlvo

Interfaces implicitas: listaAlvo - lista de adaptadores Asseriva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Adaptador corrente

AS: listaAlvo chegou ao fim

5.7.1.7 void mandarRecursoAdaptado (Adaptador * *listaAlvo*)

Funcao: mandarRecursoAdaptado

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Adaptador

A estrategia de distribuicao ja foi previamente definida e as interconxoes ligadas diretamente cada celula do adaptador ja esta com o recursoTransportado preenchido

Requisitos: alterar a interconexao ligada a cada celula da lista de geradores com o valor que sera transportado Altera o adaptador ou cidade de destino com o recurso que sera recebido

Interfaces explicitas: void, mandarRecursoAdaptado Interfaces implicitas:

listaAlvo - lista de adaptadores AE : percorrendo listaAlvo

AE: precorre a lista de conexoes que saem de cada adaptador

AE: vai ater a ulcima interconexao antes do destino

Assertiva de argumentacao

Verifica se a proxima Interconexao nao esta falha se nao esta define o recurso que ela ira transportar

AE : verifica se a proxima conexao nao eh falha

AE : se nao for falho define quanto de recurso sera transportado

AE : posterior esta falho, recursoTransportado = 0

se a saida eh uma cidade

ou se a saida é um adaptador

5.8 /home/eduardo/TrabalhoMP/app/src/Cidades.c File Reference

```
#include "../header/Cidades.h"
#include <assert.h>
```

Include dependency graph for Cidades.c:

Functions

- Cidade * criaListaCidade ()
- Vazia cidadeVazia (Cidade *listaAlvo)
- Cidade * insereCidade (char *registro, Cidade *listaAlvo)
- void imprimeListaCidade (Cidade *listaAlvo)
- void liberaListaCidade (Cidade *listaAlvo)
- int recursoGastoTotal (Cidade *listaAlvo)
- int numeroCidades (Cidade *listaAlvo)
- void gerenciaRecursoRecebido (Cidade *listaAlvo)
- int numeroCidadesNegativadas (Cidade *listaAlvo)
- int tempoSemRecursoNecessario (Cidade *listaAlvo)
- int numeroCidadesNoVermelho (Cidade *listaAlvo)
- int tempoCidadesNoVermelho (Cidade *listaAlvo)

5.8.1 Function Documentation

5.8.1.1 Vazia cidadeVazia (Cidade * *listaAlvo*)

Funcao: cidadeVazia (Iterador)

AssertivaSaida: VAZIA || NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazia, cidadeVazia, Cidade *listaAlvo

Interfaces implicitas: Vazia - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de cidades AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.8.1.2 Cidade* criaListaCidade ( )

Funcao: criaListaCidade (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Cidade

Interfaces explicitas: Cidade*, criaListaCidade

5.8.1.3 void gerenciaRecursoRecebido (Cidade * *listaAlvo*)

Funcao: gerenciaRecursoRecebido

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: gerencia o recurso recebido, aterando o valor do recurso gasto

Interfaces explicitas: int, gerenciaRecursoRecebido, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE : percorrer a lista de cidades

AE : se a porcentagem do recurso Necessario eh positiva

AE: se a porcentagem eh negativa mas nao critica

AE: se a porcentagem eh critica

5.8.1.4 void imprimeListaCidade (Cidade * *listaAlvo*)

Funcao: imprimeListaCidade (Iterador)

AssertivaEntrada: cidadeVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: impressao da lista de cidades

Interfaces explicitas: void, imprimeListaCidade, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Imprime os atributos da cidade corrente

AS: listaAlvo chegou ao fim

5.8.1.5 Cidade∗ insereCidade (char ∗ *registro,* Cidade ∗ *listaAlvo*)

Funcao: insereCidade

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir uma nova cidade na lista de cidades

Interfaces explicitas: Cidade∗, insereCidade, char ∗registro, Cidade ∗listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de cidades < Alocacao da nova cidade

< Alocacao de um vetor do tamanho do registro

< Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome da nova cidade possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos da cidade

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 4

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome da cidade Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos da cidade

Comentarios de argumentacao

Os atributos nao lidos da cidade inserida recebem o valor nulo, e a proxima cidade da lista que contem a nova cidade inserido na cabeca recebe a lista de cidades atual

5.8.1.6 void liberaListaCidade (Cidade * *listaAlvo*)

Funcao: liberaListaCidade (Iterador)

AssertivaEntrada: cidadeVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: cidadeVazia(aux1) == VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: liberacao da lista de cidades

Interfaces explicitas: void, liberaListaCidade, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Cidade corrente

AS: listaAlvo chegou ao fim

5.8.1.7 int numeroCidades (Cidade * *listaAlvo*)

Funcao: numeroCidades (Iterador)

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Cidade

Requisitos: conta o numero total de cidades em uma lista de cidades

Interfaces explicitas: int, numeroCidades, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Contagem do numero de total de cidades na listaAlvo

AS: listaAlvo chegou ao fim

5.8.1.8 int numeroCidadesNegativadas (Cidade * *listaAlvo*)

Funcao: numeroCidadesNegativadas

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar quantas cidades tiveram seus recursos abaixo do necessario

Interfaces explicitas: int, numeroCidadesNegativadas, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer a lista de cidade

AE : teve algum turno negativado

5.8.1.9 int numeroCidadesNoVermelho (Cidade * *listaAlvo*)

Funcao: numeroCidadesNoVermelho

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar quantas cidades tiveram seus recursos abaixo de 30% do necessario

Interfaces explicitas: int, numeroCidades, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer a lista de cidade

AE : teve algum turno no vermelho

5.8.1.10 int recursoGastoTotal (Cidade * *listaAlvo*)

Funcao: recursoGastoTotal

AssertivaEntrada: cidadeVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: total >= 0;

Hipotese: cidade - ponteiro para o inicio da lista de cidades

Requisitos: Somar os recursos gatos por todas as cidades

Interfaces explicitas: int, recursoGastoTotal, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: lista nao chegou ao fim

Comentarios de argumentacao

Incrementa o somatorio com o valor do recurso gasto pela celula atual muda a referencia de cidade para a proxima celula

AS: lista chegou ao fim

5.8.1.11 int tempoCidadesNoVermelho (Cidade * *listaAlvo*)

Funcao: tempoSemRecursoNecessario

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar a soma dos tempos que as cidades ficaram com menos de 30% do recurso necessario

Interfaces explicitas: int, tempoSemRecursoNecessario, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer lista de cidades

5.8.1.12 int tempoSemRecursoNecessario (Cidade * *listaAlvo*)

Funcao: tempoSemRecursoNecessario

Hipóteses: listaAlvo - ponteiro para o inicio da lista do tipo Cidade

Requisitos: contabilizar a soma dos tempos que as cidades ficaram sem recurso

Interfaces explicitas: int, tempoSemRecursoNecessario, Cidade *listaAlvo

Interfaces implicitas: listaAlvo - lista de cidades AE: percorrer lista de cidades

5.9 /home/eduardo/TrabalhoMP/app/src/Geradores.c File Reference

```
#include "../header/Geradores.h"
#include <assert.h>
```

Include dependency graph for Geradores.c:

Functions

- Gerador * criaListaGerador ()
- Vazio geradorVazio (Gerador *listaAlvo)
- Gerador * insereGerador (char *registro, Gerador *listaAlvo)
- void imprimeListaGerador (Gerador *listaAlvo)

- void liberaListaGerador (Gerador ∗listaAlvo)
- int recursoProduzidoTotal (Gerador ∗listaAlvo)
- int custoGeradores (Gerador ∗listaAlvo)
- void mandarRecursoProduzido (Gerador ∗listaAlvo)
- int numeroGeradores (Gerador ∗listaAlvo)

5.9.1 Function Documentation

5.9.1.1 Gerador∗ criaListaGerador ( )

Funcao: criaListaGerador (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Gerador

Interfaces explicitas: Gerador∗, criaListaGerador

5.9.1.2 int custoGeradores (Gerador ∗ *listaAlvo*)

Funcao: custoGeradores

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: realiza a soma dos custos dos geradores

Interfaces explicitas: int, custoGeradores, Gerador ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Somando os custos de cada gerador da lista de geradores

5.9.1.3 Vazio geradorVazio (Gerador ∗ *listaAlvo*)

Funcao: geradorVazio (Iterador)

AssertivaSaida: VAZIO || NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazio, geradorVazio, Gerador ∗listaAlvo

Interfaces implicitas: Vazio - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de geradores AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.9.1.4 void imprimeListaGerador (Gerador ∗ *listaAlvo*)

Funcao: imprimeListaGerador

AssertivaEntrada: geradorVazio(listaAlvo) == NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: impressao da lista de geradores

Interfaces explicitas: void, imprimeListaGerador, Gerador ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Imprime os atributos do gerador corrente

AS: listaAlvo chegou ao fim

5.9.1.5 Gerador∗ insereGerador (char ∗ *registro,* Gerador ∗ *listaAlvo*)

Funcao: insereGerador

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir um novo gerador na lista de geradores

Interfaces explicitas: Gerador∗, insereGerador, char ∗registro, Gerador ∗listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de geradores < Alocacao de um novo gerador

< Alocacao de um vetor do tamanho do registro

< Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome do novo adaptador possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos do gerador

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 5

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome do gerador Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos do gerador

Comentarios de argumentacao

Os atributos nao lidos do gerador inserido recebem o valor nulo, e o proximo gerador da lista que contem o novo gerador inserido na cabeca recebe a lista de geradores atual

5.9.1.6 void liberaListaGerador (Gerador * *listaAlvo*)

Funcao: liberaListaGerador (Iterador)

AssertivaEntrada: geradorVazio(listaAlvo) == NAO_VAZIO;

AssertivaSaida: geradorVazio(listaAlvo) == VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: liberacao da lista de geradores

Interfaces explicitas: void, liberaListaGerador, Gerador *listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Gerador corrente

AS: listaAlvo chegou ao fim

5.9.1.7 void mandarRecursoProduzido (Gerador * *listaAlvo*)

Funcao: mandarRecursoProduzido

AssertivaEntrada: geradorVazio(listaAlvo) == NAO_VAZIO;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: alterar a interconexao ligada a cada celula da lista de geradores com o valor que sera transportado Altera o adaptador de destino com o recurso que sera recebido

Interfaces explicitas: void, mandarRecursoProduzido, Gerador *listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores AE: nao chegou ao final da listaAlvo

AE: o elemento possui saidas

AE: interconexao ligada a essa celula nao teve falha

AE: gerador possui uma saida

AE: laço que cobre os casos de uma interconexao apontar para outra antes

AE: uma interconexao aponta para outra;

Assertiva de argumentacao

Verifica se a proxima Interconexao nao esta falha se nao esta define o recurso que ela ira transportar

AE : verifica se a proxima interconexao nao esta falha

AE: conexao nao falha

AE : conexao Falha transporta 0 de recurso

AE: chegou na ultima interconexao depois do gerador;

AE: a interconexao aponta para o adaptador

AE: interconexao ligada a essa celula falhou

AE: tem saida

AE: percorrrer a lista de inteconexao das saidas

AE: a interconexao aponta para o adaptador

AS: chegou ao final da listaAlvo

5.9.1.8 int numeroGeradores (Gerador * *listaAlvo*)

Funcao: numeroGeradores (Iterador)

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: conta o numero total de geradores em uma lista de geradores

Interfaces explicitas: int, numeroGeradores, Gerador *listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim

Comentarios de argumentacao

Contagem do numero de total de geradores na listaAlvo

5.9.1.9 int recursoProduzidoTotal (Gerador * *listaAlvo*)

Funcao: numeroGeradores (Iterador)

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Gerador

Requisitos: realiza o calculo da energia total gerada

Interfaces explicitas: int, recursoProduzidoTotal, Gerador *listaAlvo

Interfaces implicitas: listaAlvo - lista de geradores Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Somando os recursos produzidos para obter a energia total gerada

5.10 /home/eduardo/TrabalhoMP/app/src/Geral.c File Reference

```
#include "../header/Geral.h"
#include <assert.h>
```

Include dependency graph for Geral.c:

Functions

- void conecta (Cidade *cidades, Gerador *geradores, Interconexao *interconexoes, Adaptador *adaptadores)
- void verifica (Cidade *cidades, Gerador *geradores, Interconexao *interconexoes, Adaptador *adaptadores)

5.10.1 Function Documentation

5.10.1.1 void conecta (Cidade * *cidades,* Gerador * *geradores,* Interconexao * *interconexoes,* Adaptador * *adaptadores*)

Funcao: conecta

Hipóteses: cidades - ponteiro para uma lista do tipo Cidade geradores - ponteiro para uma lista do tipo Gerador interconexoes - ponteiro para uma lista do tipo Interconexao adaptadores - ponteiro para uma lista do tipo Adaptador

Requisitos: realizar a conexao entre as listas, para gerar a rede de distribuicao

Interfaces explicitas: void, conecta, Cidade *cidades, Gerador *geradores, Interconexao *interconexoes, Adaptador *adaptadores

Interfaces implicitas: cidades - lista de cidades geradores - lista de geradores interconexoes - lista de interconexoes adaptadores - lista de adaptadores < Ponteiro auxiliar para cidade

< Ponteiro auxiliar para gerador

< Ponteiro auxiliar para interconexao

< Ponteiro auxiliar para interconexao

< Ponteiro auxiliar para adaptador

AE: o auxI1 ainda nao chegou ao fim da lista de interconexoes

AE: o auxA ainda nao chegou ao fim da lista de adaptadores

AE: a posicao inicial da interconexao apontada por auxI1 coincide com a posicao do adaptador apontado por auxA

Comentarios de argumentacao

insere-se o adaptador como entrada da interconexao insere-se a interconexao na lista de saidas do adaptador

AE: a posicao final da interconexao apontada por auxI1 coincide com a posicao do adaptador apontado por auxA

Comentarios de argumentacao

insere-se o adaptador como saida da interconexao insere-se a interconexao na lista de entradas do adaptador

AS: o auxA chegou ao fim da lista de adaptadores

AE: o auxG ainda nao chegou ao fim da lista de geradores

AE: a posicao inicial da interconexao apontada por auxI1 coincide com a posicao do gerador apontado por auxG

Comentarios de argumentacao

insere-se a interconexao como saida do gerador insere-se o gerador como entrada da interconexao

AS: o auxG chegou ao fim da lista de geradores

AE: o auxC ainda nao chegou ao fim da lista de cidades

AE: a posicao final da interconexao apontada por auxI1 coincide com a posicao da cidade apontado por auxC

Comentarios de argumentacao

insere-se a cidade como saida da lista de interconexao insere-se a interconexao na lista de entradas da cidade

AS: o auxG chegou ao fim da lista de geradores

AE: o auxI2 ainda nao chegou ao fim da lista de interconexoes

AE: a posicao inicial da interconexao apontada por auxI1 coincide com a posicao final da interconexao apontada por auxI2

Comentarios de argumentacao

insere-se a interconexao auxI1 como saida da interconexao auxI2 insere-se a interconexao auxI2 como entrada da interconexao auxI1

AE: a posicao final da interconexao apontada por auxI1 coincide com a posicao inicial da interconexao apontada por auxI2

Comentarios de argumentacao

insere-se a interconexao auxI2 como saida da interconexao auxI1 insere-se a interconexao auxI1 como entrada da interconexao auxI2

AS: o auxI2 chegou ao fim da lista de interconexoes

AS: o auxI1 chegou ao fim da lista de interconexoes

5.10.1.2 void verifica (Cidade * *cidades,* Gerador * *geradores,* Interconexao * *interconexoes,* Adaptador * *adaptadores*)

Funcao: verfica

Hipóteses: cidades - ponteiro para uma lista do tipo Cidade geradores - ponteiro para uma lista do tipo Gerador interconexoes - ponteiro para uma lista do tipo Interconexao adaptadores - ponteiro para uma lista do tipo Adaptador

Requisitos: verificar as conexoes e gerar um arquivo com as inconsistencias encontradas

Interfaces explicitas: void, conecta, Cidade *cidades, Gerador *geradores, Interconexao *interconexoes, Adaptador *adaptadores

Interfaces implicitas: cidades - lista de cidades geradores - lista de geradores interconexoes - lista de interconexoes adaptadores - lista de adaptadores < Ponteiro auxiliar para cidade

< Ponteiro auxiliar para gerador

< Ponteiro auxiliar para interconexao

< Ponteiro auxiliar para adaptador

AE: a lista de cidades nao eh vazia

AE: o auxC ainda nao chegou ao fim da lista de cidades

AE: a cidade corrente nao posssui entradas

AS: o auxC chegou ao fim da lista de cidades

AE: a lista de cidades eh vazia

AE: a lista de geradores nao eh vazia

AE: o auxG ainda nao chegou ao fim da lista de geradores

AE: o gerador corrente nao posssui saidas

AS: o auxG chegou ao fim da lista de geradores

AE: a lista de geradores eh vazia

AE: a lista de interconexoes nao eh vazia

AE: o auxI ainda nao chegou ao fim da lista de interconexoes

AE: a interconexao nao possui entradas em geradores, adaptadores e interconexoes

AE: a interconexao nao possui saidas em geradores, adaptadores e interconexoes

AS: o auxI chegou ao fim da lista de interconexoes

AE: a lista de interconexoes eh vazia

AE: a lista de adaptadores nao eh vazia

AE: o auxA ainda nao chegou ao fim da lista de adaptadores

AE: o adaptador corrente nao posssui entradas

AE: o adaptador corrente nao posssui saidas

AS: o auxA chegou ao fim da lista de adaptadores

AE: a lista de adaptadores eh vazia

5.11 /home/eduardo/TrabalhoMP/app/src/Interconexoes.c File Reference

```
#include "../header/Interconexoes.h"
#include <assert.h>
```

Include dependency graph for Interconexoes.c:

Functions

- Interconexao ∗ criaListaInterconexao ()
- Vazia interconexaoVazia (Interconexao ∗listaAlvo)
- Interconexao ∗ insereInterconexao (char ∗registro, Interconexao ∗listaAlvo)
- void imprimeListaInterconexao (Interconexao ∗listaAlvo)
- void liberaListaInterconexao (Interconexao ∗listaAlvo)
- float tamanhoConexao (Interconexao ∗listaAlvo)
- float tamanhoTotalConexao (Interconexao ∗listaAlvo)
- Falha calculaFalha (Interconexao ∗listaAlvo)
- void mandarRecursoTransportado (Interconexao ∗listaAlvo)
- int custoGastoComConserto (Interconexao ∗listaAlvo)
- int numeroTotalFalhas (Interconexao ∗listaAlvo)
- void gerenciaFalhas (Interconexao ∗listaAlvo)

5.11.1 Function Documentation

5.11.1.1 Falha calculaFalha (Interconexao ∗ *listaAlvo*)

Funcao: calculaFalha

AssertivaEntrada: conexao->chanceFalha >= 0 && conexao->chanceFalha <= 1;

AssertivaSaida: FALHA || SEM_FALHA;

Requisitos: calculo da chance de falha

Interfaces explicitas: Falha, calculaFalha, Interconexao ∗conexao

Interfaces implicitas: Falha - tipo de dado, indicando se houve falha ou nao listaAlvo - lista de interconexoes Asseriva estrutural: num eh um numero gerado aleatoriamente

5.11.1.2 Interconexao∗ criaListaInterconexao ( )

Funcao: criaListaInterconexao (Iterador)

AssertivaSaida: NULL;

Requisitos: criacao de uma nova lista do tipo Interconexao

Interfaces explicitas: Interconexao∗, criaListaInterconexao

5.11.1.3 int custoGastoComConserto (Interconexao ∗ *listaAlvo*)

Funcao: custoGastoComConserto

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponterio para o inicio da lista do tipo Interconexao

Requisitos: calcular o quanto foi gasto com concerto das interconexoes

Interfaces explicitas: int, custoGastoComConcerto, Interconexao ∗listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE : listaAlvo nao chegou ao fim

Comentarios de argumentacao

Incrementa o total com o resultado da multiplicacao do numero de falhas com o custo gasto com o conserto de cada falha

AS: listaAlvo chegou ao fim

5.11.1.4 void gerenciaFalhas (Interconexao * *listaAlvo*)

Funcao: gerenciaFalhas

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponterio para o inicio da lista do tipo Interconexao

Requisitos: marcar as celulas que falharam como falhas e contabilizar o concertos das celulas que estao paradas

Interfaces explicitas: int, numeroTotalFalhas, Interconexao *listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes AE: listaAlvo nao chegou ao fim

AE: interconexao corrente nao falhou

AE: chance falha da interconexao corrente eh FALHA

AE: interconexao corrente falhou

AE: tempo de conserto foi atingido pelo contador

AS: listaAlvo chegou ao fim

5.11.1.5 void imprimeListaInterconexao (Interconexao * *listaAlvo*)

Funcao: imprimeListaInterconexao (Iterador)

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: impressao da lista de interconexoes

Interfaces explicitas: void, imprimeListaInterconexao, Interconexao *listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Imprime os atributos da interconexao corrente

AS: listaAlvo chegou ao fim

5.11.1.6 Interconexao* insereInterconexao (char * *registro,* Interconexao * *listaAlvo*)

Funcao: insereInterconexao

AssertivaEntrada: registro != NULL;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao registro - string, nao vazia, contendo uma linha do arquivo txt de entrada

Requisitos: inserir um novo gerador na lista de interconexoes

Interfaces explicitas: Interconexao*, insereInterconexao, char *registro, Interconexao *listaAlvo

Interfaces implicitas: registro - representa uma linha do arquivo de entrada listaAlvo - lista de interconexoes < Alocacao da nova Interconexao

< Variavel auxiliar para percorrer a lista e inserir o elemento no final

< Alocacao de um vetor do tamanho do registro

< Variaveis de auxilio

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Comecando de registro[2], enquanto registro[i] for um caracter irrelevante, soma-se 1 a variavel de auxilio i

AS: a posicao corrente do registro possui um caracter irrelevante

Asseriva estrutural: o nome da nova interconexao possui tamanho i-1

AE: o valor da variavel auxiliar j deve ser menor ou igual ao numero total de atributos lidos da interconexao

AE: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AE: o valor de j eh igual a 0

Comentarios de argumentacao

A (posicao corrente-k) recebe um caracter finalizador

AS: o valor de j eh maior que 0

AE: o valor de j eh maior que 0

Comentarios de argumentacao

A (posicao corrente-k) do vetor numChar recebe um caracter finalizador

Comentarios de argumentacao

De acordo com o valor da variavel auxiliar j, armazena-se o vetor numChar no seu respectivo atributo lido

AS: o valor de j eh maior que 8

AS: a posicao corrente do registro possui um caracter relevante

AE: a posicao corrente do registro possui um caracter relevante

Comentarios de argumentacao

Se o valor da variavel de auxilio j for 0 Entao armazena-se a posicao corrente do registro na (posicao corrente-k) do nome da interconexao Senao armazena-se a posicao corrente do registro na (posicao corrente-k) do vetor numChar FimSe

AS: a posicao corrente do registro possui um caracter irrelevante, ou um caracter finalizador

AS: o valor da variavel auxiliar j ultrapassou o numero total de atributos lidos da interconexao

Comentarios de argumentacao

Os atributos nao lidos da inteconexao inserida recebem o valor nulo, e a proxima interconexao da lista que contem a nova inteconexao inserido na cabeca recebe a lista de interconexoes atual

AE: se a listaAlvo nao for vazia

AE: o aux nao chegou ao final da lista de interconexoes

AE: se a listaAlvo for vazia

5.11.1.7 Vazia interconexaoVazia (Interconexao * *listaAlvo*)

Funcao: interconexaoVazia (Iterador)

AssertivaSaida: VAZIA || NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: checar se a lista está vazia

Interfaces explicitas: Vazia, interconexaoVazia, Interconexao *listaAlvo

Interfaces implicitas: Vazia - tipo de dado, indicando se a lista eh vazia ou nao listaAlvo - lista de interconexoes AE: listaAlvo eh vazia

AE: listaAlvo nao eh vazia

AS: o retorno deve ser uma variavel do tipo Vazia

5.11.1.8 void liberaListaInterconexao (Interconexao * *listaAlvo*)

Funcao: liberaListaInterconexao (Iterador)

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: interconexaoVazia(listaAlvo) == VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: liberacao da lista de interconexoes

Interfaces explicitas: void, liberaListaInterconexao, Interconexao *listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Assertiva estrutural: aux1 é a listaAlvo, porem sendo percorrida

AE: listaAlvo nao chegou ao fim Comentarios de argumentacao

Liberam os atributos alocados dinamicamente do elemento Interconexao corrente

AS: listaAlvo chegou ao fim

5.11.1.9 void mandarRecursoTransportado (Interconexao * *listaAlvo*)

Funcao: mandarRecursoTransportado

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: transporte dos recursos

Interfaces explicitas: void, mandarRecursoTransportado, Interconexao *listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a lista nao-nula de interconexoes

AE: aux nao chegou ao fim da lista de inteconexoes

AE: a interconexao corrente nao possui falha

Comentarios de argumentacao

Enqaunto a lista de interconexoes eh percorrida, dependendo do destino da interconexao, soma-se o recurso ao respectivo destino

AE: o destino da interconexao eh um Adaptador

AE: o destino da interconexao eh uma Cidade

AS: a lista de interconexoes chegou ao fim

5.11.1.10 int numeroTotalFalhas (Interconexao * *listaAlvo*)

Funcao: numeroTotalFalhas

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: total >= 0;

Hipóteses: listaAlvo - ponterio para o inicio da lista do tipo Interconexao

Requisitos: contabilizar a quantidade de falhas

Interfaces explicitas: int, numeroTotalFalhas, Interconexao *listaAlvo

Interfaces implicitas: listaAlvo - lista de interconexoes Asseriva estrutural: aux é a listaAlvo, porem sendo percorrida

AE: listaAlvo ainda nao acabou Comentarios de argumentacao

Incrementa o somatorio com o numero de falhas da celula atual

5.11.1.11 float tamanhoConexao (Interconexao * *listaAlvo*)

Funcao: tamanhoConexao

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: distancia > 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: calculo da distancia entre a posicao final e inicial

Interfaces explicitas: float, tamanhoConexao, Interconexao *listaAlvo

Interfaces implicitas: float - valor do tamanho listaAlvo - lista de interconexoes < Valor no eixo x da posicao inicial

< Valor no eixo y da posicao inicial

< Valor no eixo x da posicao final

< Valor no eixo y da posicao final

Comentarios de argumentacao

Calculando a distancia utilizando ((xb-xa)^2 + (yb-ya)^2)^(1/2)

5.11.1.12 float tamanhoTotalConexao (Interconexao * *listaAlvo*)

Funcao: tamanhoTotalConexao

AssertivaEntrada: interconexaoVazia(listaAlvo) == NAO_VAZIA;

AssertivaSaida: resultado > 0;

Hipóteses: listaAlvo - ponteiro para uma lista do tipo Interconexao

Requisitos: calculo da soma dos tamanhos das interconexoes

Interfaces explicitas: float, tamanhoTotalConexao, Interconexao *listaAlvo

Interfaces implicitas: float - valor da soma dos tamanhos listaAlvo - lista de interconexoes Asseriva estrutural: aux é a lista nao-nula de interconexoes

AE: aux nao chegou ao fim da lista de inteconexoes

Comentarios de argumentacao

Enquanto a lista de interconexoes eh percorrida, o tamanho das conexoes sao somados e armazenados na variavel resultado

AS: a lista de interconexoes chegou ao fim

5.12 /home/eduardo/TrabalhoMP/app/src/Principal.c File Reference

```
#include "../header/Geral.h"
```

Include dependency graph for Principal.c:

Functions

- int main ()

5.12.1 Function Documentation

5.12.1.1 int main ( )

< String auxiliar para obter registros

AE: o nome do arquivo nao eh valido

< Abre arquivo de entrada

AE: o nome do arquivo eh valido

AE: o arquivo nao chegou ao fim

AE: a linha (registro) obtido do arquivo possui conteudo

AE: O registro obtido eh do tipo Cidade

AE: O registro obtido eh do tipo Gerador

AE: O registro obtido eh do tipo Interconexao

AE: O registro obtido eh do tipo Adaptador

AS: a linha (registro) obtido do arquivo nao possui conteudo, ou seja, o arquivo chegou ao fim

AS: o arquivo chegou ao fim

Comentarios de argumentacao

Conectando e verificando as listas

AE: manda recurso ate os adaptadores

AE: mandar o recurso ate as cidades

Comentarios de argumentacao

Imprimindo as listas obtidas a partir do arquivo de entrada

Comentarios de argumentacao

Preenchimento do relatorio

Comentarios de argumentacao

Desalocando as listas obtidas

5.13 /home/eduardo/TrabalhoMP/README.md File Reference

5.14 /home/eduardo/TrabalhoMP/test/Adaptadores_unittest.c File Reference

```
#include <gtest/gtest.h>
#include "../app/header/Adaptadores.h"
```

Include dependency graph for Adaptadores_unittest.c:

Functions

- TEST (criaListaAdaptador, vazia)

  Testando a funcao criaListaAdaptador.
- TEST (adaptadorVazio, vazio)

  Caso vazio.
- TEST (adaptadorVazio, naoVazio)

  Caso naoVazio.
- TEST (insereAdaptador, Vazio)

  Caso vazio.
- TEST (insereAdaptador, naoVazio)

Caso naoVazio.

5.14.1 Function Documentation

5.14.1.1 TEST (criaListaAdaptador , vazia)

Testando a funcao criaListaAdaptador.

5.14.1.2 TEST (adaptadorVazio , vazio)

Caso vazio.

Testando a funcao adaptadorVazio

5.14.1.3 TEST (adaptadorVazio , naoVazio)

Caso naoVazio.

5.14.1.4 TEST (insereAdaptador , Vazio)

Caso vazio.

Testando a funcao insereAdaptador

5.14.1.5 TEST (insereAdaptador , naoVazio)

Caso naoVazio.

Inicio do teste em si

5.15 /home/eduardo/TrabalhoMP/test/Cidades_unittest.c File Reference

```
#include <gtest/gtest.h>
#include "../app/header/Cidades.h"
```

Include dependency graph for Cidades_unittest.c:

Functions

- TEST (criaListaCidade, vazia)
  Testando a funcao criaListaCidade.
- TEST (cidadeVazia, vazia)
  Caso vazia.
- TEST (cidadeVazia, naoVazia)
  Caso naoVazia.
- TEST (insereCidade, Vazia)
  Caso vazia.
- TEST (insereCidade, naoVazia)
  Caso naoVazia.
- TEST (recursoGastoTotal, geral)
  Testando a funcao recursoGastoTotal.
- TEST (numeroCidades, geral)
  Testando a funcao numeroCidades.

5.15.1 Function Documentation

5.15.1.1 TEST (criaListaCidade , vazia)

Testando a funcao criaListaCidade.

5.15.1.2 TEST (cidadeVazia , vazia)

Caso vazia.

Testando a funcao cidadeVazia

5.15.1.3 TEST (cidadeVazia , naoVazia)

Caso naoVazia.

5.15.1.4 TEST (insereCidade , Vazia)

Caso vazia.

Testando a funcao insereCidade

5.15.1.5 TEST (insereCidade , naoVazia)

Caso naoVazia.

Inicio do teste em si

5.15.1.6 TEST (recursoGastoTotal , geral)

Testando a funcao recursoGastoTotal.

5.15.1.7 TEST (numeroCidades , geral)

Testando a funcao numeroCidades.

Inicio do teste em si

5.16 /home/eduardo/TrabalhoMP/test/Geradores_unittest.c File Reference

```
#include <gtest/gtest.h>
#include "../app/header/Geradores.h"
```

Include dependency graph for Geradores_unittest.c:

Functions

- TEST (criaListaGerador, vazia)

  Testando a funcao criaListaGerador.
- TEST (geradorVazio, vazio)

  Caso vazio.
- TEST (geradorVazio, naoVazio)

  Caso naoVazia.

- TEST (insereGerador, Vazio)

  Caso vazio.
- TEST (insereGerador, naoVazio)

  Caso naoVazio.
- TEST (recursoProduzidoTotal, geral)

  Testando a funcao recursoProduzidoTotal.
- TEST (custoGeradores, geral)

  Testando a funcao custoGeradores.
- TEST (numeroGeradores, geral)

  Testando a funcao numeroGeradores.

5.16.1 Function Documentation

5.16.1.1 TEST (criaListaGerador , vazia)

Testando a funcao criaListaGerador.

5.16.1.2 TEST (geradorVazio , vazio)

Caso vazio.

Testando a funcao geradorVazio

5.16.1.3 TEST (geradorVazio , naoVazio)

Caso naoVazia.

5.16.1.4 TEST (insereGerador , Vazio)

Caso vazio.

Testando a funcao insereGerador

5.16.1.5 TEST (insereGerador , naoVazio)

Caso naoVazio.

Inicio do teste em si

5.16.1.6 TEST (recursoProduzidoTotal , geral)

Testando a funcao recursoProduzidoTotal.

Inicio do teste em si

5.16.1.7 TEST (custoGeradores , geral)

Testando a funcao custoGeradores.

Inicio do teste em si

5.16.1.8 TEST (numeroGeradores , geral)

Testando a funcao numeroGeradores.

Inicio do teste em si

5.17 /home/eduardo/TrabalhoMP/test/Interconexoes_unittest.c File Reference

```
#include <gtest/gtest.h>
#include "../app/header/Interconexoes.h"
```

Include dependency graph for Interconexoes_unittest.c:

Functions

- TEST (criaListaInterconexao, vazia)

  Testando a funcao criaListaInterconexao.
- TEST (interconexaoVazia, vazia)

  Caso vazia.
- TEST (interconexaoVazia, naoVazia)

  Caso naoVazia.
- TEST (insereInterconexao, Vazia)

  Caso vazia.
- TEST (insereInterconexao, naoVazia)

  Caso naoVazia.
- TEST (tamanhoConexao, geral)

  Testando a funcao tamanhoConexao.
- TEST (tamanhoTotalConexao, geral)

  Testando a funcao tamanhoTotalConexao.

5.17.1 Function Documentation

5.17.1.1 TEST (criaListaInterconexao , vazia)

Testando a funcao criaListaInterconexao.

5.17.1.2 TEST (interconexaoVazia , vazia)

Caso vazia.

Testando a funcao interconexaoVazia

5.17.1.3 TEST (interconexaoVazia , naoVazia)

Caso naoVazia.

5.17.1.4 TEST (insereInterconexao , Vazia)

Caso vazia.

Testando a funcao insereInterconexao

5.17.1.5 TEST (insereInterconexao , naoVazia)

Caso naoVazia.

Inicio do teste em si

5.17.1.6 TEST (tamanhoConexao , geral)

Testando a funcao tamanhoConexao.

5.17.1.7 TEST (tamanhoTotalConexao , geral)

Testando a funcao tamanhoTotalConexao.

5.18 /home/eduardo/TrabalhoMP/test/main_unittest.c File Reference

```
#include <gtest/gtest.h>
```

Include dependency graph for main_unittest.c:

Functions

- int main (int argc, char **argv)

5.18.1 Function Documentation

5.18.1.1 int main (int *argc*, char ** *argv*)

Index