Jogo das 20 Perguntas

1

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 no Struct Reference

Função de Criação do No padrao da arvore.

Public Attributes

- char pergunta [50]
- int indice
- struct no * Sim
- struct no * Nao

3.1.1 Detailed Description

Função de Criação do No padrao da arvore.

3.1.2 Member Data Documentation

3.1.2.1 indice

```
int no::indice
```

3.1.2.2 Nao

```
struct no* no::Nao
```

3.1.2.3 pergunta

```
char no::pergunta[50]
```

3.1.2.4 Sim

```
struct no* no::Sim
```

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

- funcoes.c

Chapter 4

File Documentation

4.1 funcoes.c File Reference

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

Classes

- struct no

  Função de Criação do No padrao da arvore.

Typedefs

- typedef struct no arvore

  Função de Criação do No padrao da arvore.

Functions

- arvore * preencherNo (char *pergunta, int indice)

  Funcao para criar um no novo e preenchido, baseado nos parametros.
- void removerNo (arvore *no)

  Funcao remover o no atual.
- void salvarArvore (arvore *salvar, FILE *pSalvar)

  Funcao para salvar a arvore em um arquivo .txt.
- arvore * carregarArvore (FILE *pArquivo, arvore **salvar, arvore **pai, int indice, int direcao)

  Funcao para carregar a arvore de um arquivo .txt para a memoria.
- arvore * percorrer (int resp, arvore **raiz)

  Funcao para percorrer a arvore. É checada a existencia previa de nos, criando-os caso nao existam.
- void novoJogo (arvore **raiz, FILE *pSalvar)

  Funcao para executar o jogo. Utilizando das funcoes anteriores, com lacos de repeticao e condicoes de parada relativas ao input do usuario.

4.1.1 Typedef Documentation

4.1.1.1 arvore

```
typedef struct no arvore
```

Função de Criação do No padrao da arvore.

4.1.2 Function Documentation

4.1.2.1 carregarArvore()

```
arvore* carregarArvore (
            FILE * pArquivo,
            arvore ** salvar,
            arvore ** pai,
            int indice,
            int direcao )
```

Funcao para carregar a arvore de um arquivo .txt para a memoria.

4.1.2.2 novoJogo()

```
void novoJogo (
            arvore ** raiz,
            FILE * pSalvar )
```

Funcao para executar o jogo. Utilizando das funcoes anteriores, com lacos de repeticao e condicoes de parada relativas ao input do usuario.

4.1.2.3 percorrer()

```
arvore* percorrer (
            int resp,
            arvore ** raiz )
```

Funcao para percorrer a arvore. É checada a existencia previa de nos, criando-os caso nao existam.

4.1.2.4 preencherNo()

```
arvore* preencherNo (
            char * pergunta,
            int indice )
```

Funcao para criar um no novo e preenchido, baseado nos parametros.

4.1.2.5 removerNo()

```
void removerNo (
            arvore * no )
```

Funcao remover o no atual.

4.1.2.6 salvarArvore()

```
void salvarArvore (
            arvore * salvar,
            FILE * pSalvar )
```

Funcao para salvar a arvore em um arquivo .txt.

4.2 jogo20perguntas.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <headers.h>
```

Functions

- int main ()

  Header para funcionamento completo do programa.

4.2.1 Function Documentation

4.2.1.1 main()

```
int main ( )
```

Header para funcionamento completo do programa.

Main do jogo.

A main realiza a filtragem inicial para a escolha do usuario. Baseada em input, ela decide por meio de switch case qual caminho seguir, executando as funcoes de 'funcoes.c' para execucao do jogo.

4.3 testes.cpp File Reference

```
#include "catch.hpp"
#include "headers.h"
```

Macros

- #define CATCH_CONFIG_MAIN

Functions

- TEST_CASE ("Remover no ja nulo", "Nao gera erros de memoria")

  Teste para verificar nao quebra do programa Ao tentar desalocar um nó ja nulo, o programa segue normalmente.
- TEST_CASE ("No != NULL", "Assegurar alocacao e preenchimento do no")

  Teste parar verificar a correta funcionalidade da funcao preencherNo. O teste realiza a chamada da funcao preencherNo com parametros criados, retornando seu valor em uma variavel do tipo arvore.
- TEST_CASE ("Carregamento"," Assegurar carregamento de arvore em memoria")

  Teste parar verificar a correta funcionalidade da funcao carregarArvore.

4.3.1 Macro Definition Documentation

4.3.1.1 CATCH_CONFIG_MAIN

```
#define CATCH_CONFIG_MAIN
```

4.3.2 Function Documentation

4.3.2.1 TEST_CASE() [1/3]

```
TEST_CASE (
             "Remover no ja nulo" ,
             "Nao gera erros de memoria"  )
```

Teste para verificar nao quebra do programa Ao tentar desalocar um nó ja nulo, o programa segue normalmente.

4.3.2.2 TEST_CASE() [2/3]

```
TEST_CASE ( )
```

Teste parar verificar a correta funcionalidade da funcao preencherNo. O teste realiza a chamada da funcao preencherNo com parametros criados, retornando seu valor em uma variavel do tipo arvore.

4.3.2.3 TEST_CASE() [3/3]

```
TEST_CASE (
             "Carregamento" ,
             " Assegurar carregamento de arvore em memoria"  )
```

Teste parar verificar a correta funcionalidade da funcao carregarArvore.

Index