Trabalho-EDA 2023/2024

Gerado por Doxygen 1.9.5

Capítulo 1

Índice das estruturas de dados

1.1 Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Capítulo 2

Índice dos ficheiros

2.1 Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

Capítulo 3

Documentação da estruturas de dados

3.1 Referência à estrutura Node

`#include <types.h>`

Campos de Dados

- int data
- struct Node * right
- struct Node * down

3.1.1 Documentação dos campos e atributos

3.1.1.1 data

`int Node::data`

Referenciado por calculateMaxSumUtil(), createNode(), printList(), updateValue() e writeToFile().

3.1.1.2 down

`struct Node* Node::down`

Referenciado por addColumn(), addRow(), calculateMaxSumUtil(), createNode(), getDimensions(), printList(), readFile(), removeColumn(), removeRow(), updateValue() e writeToFile().

3.1.1.3 right

`struct Node* Node::right`

Referenciado por addColumn(), addRow(), calculateMaxSumUtil(), createNode(), getDimensions(), printList(), readFile(), removeColumn(), removeRow(), updateValue() e writeToFile().

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- src/types.h

Capítulo 4

Documentação do ficheiro

4.1 Referência ao ficheiro src/logic.c

Implementa as operações lógicas sobre a matriz representada por uma lista ligada bidimensional.

```
#include "logic.h"
#include "utils.h"
#include "types.h"
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
```

Funções

- void updateValue (Node *head, int oldValue, int newValue)

  Atualiza o primeiro nó encontrado com um valor específico na matriz 2D ligada.
- Node * addRow (Node *head, int rowData[ ], int size, const char *filename)

  Adiciona uma nova linha à matriz representada por uma lista ligada 2D.
- void addColumn (Node *head, int columnData[ ], const char *filename)

  Adiciona uma nova coluna ao final de cada linha da matriz.
- Node * removeRow (Node *head, int rowIndex, const char *filename)

  Remove uma linha específica da matriz.
- void removeColumn (Node *head, int columnIndex, const char *filename)

  Remove uma coluna específica da matriz.
- int calculateMaxSum (Node *head, int numRows)

  Calcula a soma máxima de uma seleção de nós em uma matriz bidimensional representada por uma lista ligada.

4.1.1 Descrição detalhada

Implementa as operações lógicas sobre a matriz representada por uma lista ligada bidimensional.

Este arquivo contém as definições das funções responsáveis por manipular a matriz, incluindo atualização de valores, adição e remoção de linhas e colunas, e o cálculo da soma máxima sob certas condições. As operações são aplicadas sobre a estrutura de dados de lista ligada bidimensional, que permite a representação flexível de matrizes. As funções abordam:

- Atualização de um valor específico dentro da matriz (`updateValue`).
- Adição de novas linhas (`addRow`) e colunas (`addColumn`) à matriz com dados especificados pelo usuário.
- Remoção de linhas (`removeRow`) e colunas (`removeColumn`) selecionadas pelo usuário da matriz.
- Cálculo da soma máxima possível selecionando um elemento por linha e coluna sem repetições (`calculateMaxSum`).

Além disso, o arquivo lida com a escrita da matriz atualizada de volta ao arquivo após modificações, garantindo que as alterações sejam persistidas.

4.1.2 Documentação das funções

4.1.2.1 addColumn()

```
void addColumn (
            Node * head,
            int columnData[],
            const char * filename )
```

Adiciona uma nova coluna ao final de cada linha da matriz.

Esta função percorre todas as linhas da matriz representada por uma lista ligada bidimensional, adicionando um novo nó ao final de cada linha com os dados fornecidos em `columnData`. Cada elemento do array `column↩Data` corresponde ao dado a ser inserido em cada nova célula da coluna. Após a adição da nova coluna, a função grava a matriz atualizada de volta ao arquivo especificado por `filename`.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
columnData	Array contendo os dados a serem adicionados na nova coluna.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências createNode(), Node::down, Node::right e writeToFile().

Referenciado por makeSquare() e requestAddColumn().

4.1.2.2 addRow()

```
Node * addRow (
              Node * head,
              int rowData[],
              int size,
              const char * filename )
```

Adiciona uma nova linha à matriz representada por uma lista ligada 2D.

Esta função adiciona uma nova linha ao final da matriz. A nova linha é composta pelos valores fornecidos no array `rowData`. Após adicionar a nova linha, a matriz é escrita de volta ao arquivo especificado por `filename`. Se a matriz estava inicialmente vazia (i.e., `head` é NULL), a nova linha se tornará a primeira linha da matriz.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada 2D. Pode ser NULL se a lista estiver vazia.
rowData	Array contendo os dados para a nova linha.
size	Número de elementos no array `rowData`.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Retorna

Node∗ Retorna o ponteiro para o nó cabeça da lista ligada 2D atualizada.

Referências createNode(), Node::down, Node::right e writeToFile().

Referenciado por makeSquare() e requestAddRow().

4.1.2.3 calculateMaxSum()

```
int calculateMaxSum (
              Node * head,
              int numRows )
```

Calcula a soma máxima de uma seleção de nós em uma matriz bidimensional representada por uma lista ligada.

Esta função inicia o processo de calcular a soma máxima onde nenhum dois nós selecionados estão na mesma linha ou coluna. Utiliza uma abordagem de backtracking para explorar todas as combinações possíveis de seleção de nós, mantendo a restrição de que apenas um nó por linha e por coluna pode ser escolhido. A função `calculateMaxSumUtil` é chamada recursivamente para percorrer a matriz e acumular a soma de valores selecionados, atualizando `maxSum` com a maior soma encontrada durante a exploração. Um array `visited` é utilizado para marcar as colunas já visitadas e evitar seleções repetidas na mesma coluna.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional que representa a matriz.
numRows	Número de linhas na matriz, utilizado para inicializar o array `visited` e controlar a recursão.

Retorna

A soma máxima obtida pela seleção de nós conforme as regras especificadas.

Referências calculateMaxSumUtil() e maxSum.

Referenciado por requestCalculateMaxSum().

4.1.2.4 removeColumn()

```
void removeColumn (
            Node * head,
            int columnIndex,
            const char * filename )
```

Remove uma coluna específica da matriz.

Esta função itera sobre cada linha da matriz, representada por uma lista ligada bidimensional, e remove o nó correspondente à coluna especificada por `columnIndex`. Se a coluna a ser removida é a primeira, o ponteiro da linha é atualizado para o próximo nó à direita. Para colunas no meio ou no final da linha, ajusta-se o ponteiro `right` do nó anterior para pular o nó da coluna removida. Os nós removidos são liberados para evitar vazamentos de memória. Após a conclusão da remoção em todas as linhas, a matriz atualizada é gravada de volta ao arquivo especificado por `filename`. Mensagens de erro são exibidas se a lista estiver vazia ou o índice da coluna for inválido.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
columnIndex	Índice baseado em zero da coluna a ser removida.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências Node::down, Node::right e writeToFile().

Referenciado por requestRemoveColumn().

4.1.2.5 removeRow()

```
Node * removeRow (
            Node * head,
            int rowIndex,
            const char * filename )
```

Remove uma linha específica da matriz.

Dada a posição especificada por `rowIndex`, esta função remove a linha correspondente da matriz representada por uma lista ligada bidimensional. Se a linha a ser removida é a primeira, o ponteiro de cabeça da lista (`head`) é atualizado. Para linhas subsequentes, a conexão `down` do nó anterior é ajustada para pular a linha removida. Após a remoção, os recursos alocados para os nós da linha removida são liberados. Finalmente, a matriz atualizada é escrita de volta ao arquivo especificado por `filename`. Se `rowIndex` for inválido ou a lista estiver vazia, a função exibe uma mensagem de erro e retorna sem modificar a matriz.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
rowIndex	Índice baseado em zero da linha a ser removida.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Retorna

O novo ponteiro para o nó cabeça da lista após a remoção da linha.

Referências Node::down, Node::right e writeToFile().

Referenciado por requestRemoveRow().

4.1.2.6 updateValue()

```
void updateValue (
            Node * head,
            int oldValue,
            int newValue )
```

Atualiza o primeiro nó encontrado com um valor específico na matriz 2D ligada.

Esta função percorre a matriz 2D ligada, linha por linha e coluna por coluna, em busca do primeiro nó que contém o valor especificado em `oldValue`. Quando encontrado, o valor desse nó é atualizado para `newValue`. Após a atualização, a matriz modificada é escrita de volta ao arquivo especificado. Se o valor `oldValue` não for encontrado na matriz, uma mensagem será exibida para informar o usuário.

Parâmetros

head	Ponteiro para o nó cabeça da matriz 2D ligada.
oldValue	O valor a ser procurado na matriz para atualização.
newValue	O novo valor para substituir o `oldValue`.

Nota

Esta função atualiza apenas a primeira ocorrência de `oldValue`. Se houver múltiplas ocorrências de `oldValue` na matriz, apenas a primeira será atualizada para `newValue`.

A função assume que o arquivo para gravação é "data.txt". Se necessário usar outro arquivo, o nome do arquivo precisa ser ajustado dentro da função `writeToFile`.

Referências Node::data, Node::down, Node::right e writeToFile().

Referenciado por requestUpdateValue().

4.2 Referência ao ficheiro src/logic.h

```
#include "types.h"
```

Funções

- void updateValue (Node *head, int oldValue, int newValue)

  Atualiza o primeiro nó encontrado com um valor específico na matriz 2D ligada.
- Node * addRow (Node *head, int rowData[ ], int size, const char *filename)

  Adiciona uma nova linha à matriz representada por uma lista ligada 2D.
- void addColumn (Node *head, int columnData[ ], const char *filename)

  Adiciona uma nova coluna ao final de cada linha da matriz.
- Node * removeRow (Node *head, int rowIndex, const char *filename)

  Remove uma linha específica da matriz.
- void removeColumn (Node *head, int columnIndex, const char *filename)

  Remove uma coluna específica da matriz.
- int calculateMaxSum (Node *head, int numRows)

  Calcula a soma máxima de uma seleção de nós em uma matriz bidimensional representada por uma lista ligada.

4.2.1 Documentação das funções

4.2.1.1 addColumn()

```
void addColumn (
            Node * head,
            int columnData[],
            const char * filename )
```

Adiciona uma nova coluna ao final de cada linha da matriz.

Esta função percorre todas as linhas da matriz representada por uma lista ligada bidimensional, adicionando um novo nó ao final de cada linha com os dados fornecidos em `columnData`. Cada elemento do array `column↩Data` corresponde ao dado a ser inserido em cada nova célula da coluna. Após a adição da nova coluna, a função grava a matriz atualizada de volta ao arquivo especificado por `filename`.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
columnData	Array contendo os dados a serem adicionados na nova coluna.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências createNode(), Node::down, Node::right e writeToFile().

Referenciado por makeSquare() e requestAddColumn().

4.2.1.2 addRow()

```
Node * addRow (
            Node * head,
```

```
            int rowData[],
            int size,
            const char * filename )
```

Adiciona uma nova linha à matriz representada por uma lista ligada 2D.

Esta função adiciona uma nova linha ao final da matriz. A nova linha é composta pelos valores fornecidos no array `rowData`. Após adicionar a nova linha, a matriz é escrita de volta ao arquivo especificado por `filename`. Se a matriz estava inicialmente vazia (i.e., `head` é NULL), a nova linha se tornará a primeira linha da matriz.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada 2D. Pode ser NULL se a lista estiver vazia.
rowData	Array contendo os dados para a nova linha.
size	Número de elementos no array `rowData`.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Retorna

Node∗ Retorna o ponteiro para o nó cabeça da lista ligada 2D atualizada.

Referências createNode(), Node::down, Node::right e writeToFile().

Referenciado por makeSquare() e requestAddRow().

4.2.1.3 calculateMaxSum()

```
int calculateMaxSum (
            Node * head,
            int numRows )
```

Calcula a soma máxima de uma seleção de nós em uma matriz bidimensional representada por uma lista ligada.

Esta função inicia o processo de calcular a soma máxima onde nenhum dois nós selecionados estão na mesma linha ou coluna. Utiliza uma abordagem de backtracking para explorar todas as combinações possíveis de seleção de nós, mantendo a restrição de que apenas um nó por linha e por coluna pode ser escolhido. A função `calculateMaxSumUtil` é chamada recursivamente para percorrer a matriz e acumular a soma de valores selecionados, atualizando `maxSum` com a maior soma encontrada durante a exploração. Um array `visited` é utilizado para marcar as colunas já visitadas e evitar seleções repetidas na mesma coluna.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional que representa a matriz.
numRows	Número de linhas na matriz, utilizado para inicializar o array `visited` e controlar a recursão.

Retorna

A soma máxima obtida pela seleção de nós conforme as regras especificadas.

Referências calculateMaxSumUtil() e maxSum.

Referenciado por requestCalculateMaxSum().

4.2.1.4 removeColumn()

```
void removeColumn (
            Node * head,
            int columnIndex,
            const char * filename )
```

Remove uma coluna específica da matriz.

Esta função itera sobre cada linha da matriz, representada por uma lista ligada bidimensional, e remove o nó correspondente à coluna especificada por `columnIndex`. Se a coluna a ser removida é a primeira, o ponteiro da linha é atualizado para o próximo nó à direita. Para colunas no meio ou no final da linha, ajusta-se o ponteiro `right` do nó anterior para pular o nó da coluna removida. Os nós removidos são liberados para evitar vazamentos de memória. Após a conclusão da remoção em todas as linhas, a matriz atualizada é gravada de volta ao arquivo especificado por `filename`. Mensagens de erro são exibidas se a lista estiver vazia ou o índice da coluna for inválido.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
columnIndex	Índice baseado em zero da coluna a ser removida.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências Node::down, Node::right e writeToFile().

Referenciado por requestRemoveColumn().

4.2.1.5 removeRow()

```
Node * removeRow (
            Node * head,
            int rowIndex,
            const char * filename )
```

Remove uma linha específica da matriz.

Dada a posição especificada por `rowIndex`, esta função remove a linha correspondente da matriz representada por uma lista ligada bidimensional. Se a linha a ser removida é a primeira, o ponteiro de cabeça da lista (`head`) é atualizado. Para linhas subsequentes, a conexão `down` do nó anterior é ajustada para pular a linha removida. Após a remoção, os recursos alocados para os nós da linha removida são liberados. Finalmente, a matriz atualizada é escrita de volta ao arquivo especificado por `filename`. Se `rowIndex` for inválido ou a lista estiver vazia, a função exibe uma mensagem de erro e retorna sem modificar a matriz.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
rowIndex	Índice baseado em zero da linha a ser removida.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Retorna

O novo ponteiro para o nó cabeça da lista após a remoção da linha.

Referências Node::down, Node::right e writeToFile().

Referenciado por requestRemoveRow().

4.2.1.6 updateValue()

```
void updateValue (
            Node * head,
            int oldValue,
            int newValue )
```

Atualiza o primeiro nó encontrado com um valor específico na matriz 2D ligada.

Esta função percorre a matriz 2D ligada, linha por linha e coluna por coluna, em busca do primeiro nó que contém o valor especificado em `oldValue`. Quando encontrado, o valor desse nó é atualizado para `newValue`. Após a atualização, a matriz modificada é escrita de volta ao arquivo especificado. Se o valor `oldValue` não for encontrado na matriz, uma mensagem será exibida para informar o usuário.

Parâmetros

head	Ponteiro para o nó cabeça da matriz 2D ligada.
oldValue	O valor a ser procurado na matriz para atualização.
newValue	O novo valor para substituir o `oldValue`.

Nota

Esta função atualiza apenas a primeira ocorrência de `oldValue`. Se houver múltiplas ocorrências de `oldValue` na matriz, apenas a primeira será atualizada para `newValue`.

A função assume que o arquivo para gravação é "data.txt". Se necessário usar outro arquivo, o nome do arquivo precisa ser ajustado dentro da função `writeToFile`.

Referências Node::data, Node::down, Node::right e writeToFile().

Referenciado por requestUpdateValue().

4.3 logic.h

Ir para a documentação deste ficheiro.

```
1 #ifndef LOGIC_H
2 #define LOGIC_H
3
4 #include "types.h"
5
24 void updateValue(Node* head, int oldValue, int newValue);
25
40 Node* addRow(Node* head, int rowData[], int size, const char* filename);
41
54 void addColumn(Node* head, int columnData[], const char* filename);
```

```
55
71 Node* removeRow(Node* head, int rowIndex, const char* filename);
72
88 void removeColumn(Node* head, int columnIndex, const char* filename);
89
103 int calculateMaxSum(Node* head, int numRows);
104
105 #endif
```

4.4 Referência ao ficheiro src/main.c

Ponto de entrada principal para a aplicação de manipulação de matrizes.

```
#include <stdio.h>
#include "logic.h"
#include "menu.h"
#include "utils.h"
```

Funções

- int main ()

4.4.1 Descrição detalhada

Ponto de entrada principal para a aplicação de manipulação de matrizes.

Este arquivo contém a função `main`, que serve como o ponto de entrada para uma aplicação que manipula uma matriz armazenada em um arquivo de texto ("data.txt"). A matriz é representada internamente como uma lista ligada bidimensional. O programa oferece ao usuário um menu de opções para visualizar a matriz, alterar elementos, adicionar ou remover linhas/colunas, e calcular a soma máxima sob certas condições.

As operações disponíveis são:

- Visualizar a matriz atual.
- Atualizar um valor específico dentro da matriz.
- Adicionar uma nova linha com valores especificados pelo usuário.
- Adicionar uma nova coluna com valores especificados pelo usuário.
- Remover uma linha selecionada pelo usuário.
- Remover uma coluna selecionada pelo usuário.
- Calcular e exibir a soma máxima de uma seleção de valores da matriz.

O loop principal do programa apresenta essas opções em um menu, executa a escolha do usuário e repete até que a opção de sair seja selecionada. Após a execução de cada operação, o usuário é solicitado a pressionar Enter para continuar, permitindo a visualização dos resultados ou mensagens de status.

4.4.2 Documentação das funções

4.4.2.1 main()

```
int main ( )
```

Referências printList(), readFile(), requestAddColumn(), requestAddRow(), requestCalculateMaxSum(), requestRemoveColumn(), requestRemoveRow(), requestUpdateValue(), showMenuAndGetChoice() e waitForUserInput().

4.5 Referência ao ficheiro src/menu.c

Implementa a interface do usuário e as funções de solicitação para manipulação de matrizes.

```
#include <stdio.h>
#include "logic.h"
#include "menu.h"
#include "utils.h"
#include <stdlib.h>
```

Funções

- void requestUpdateValue (Node *head)

  Solicita ao usuário valores para atualizar na matriz e aplica a atualização.
- Node * requestAddRow (Node *head, const char *filename)

  Solicita ao usuário a inserção de uma nova linha na matriz e adiciona a linha.
- void requestAddColumn (Node *head, const char *filename)

  Solicita ao usuário a inserção de uma nova coluna na matriz e a adiciona.
- void requestRemoveRow (Node **head, const char *filename)

  Solicita ao usuário a remoção de uma linha específica da matriz.
- void requestRemoveColumn (Node *head, const char *filename)

  Solicita ao usuário a remoção de uma coluna específica da matriz.
- void requestCalculateMaxSum (Node *head, const char *filename)

  Solicita o cálculo da soma máxima possível em uma matriz quadrada.
- int showMenuAndGetChoice ()

  Exibe o menu de opções para o usuário e obtém a escolha feita.
- void clearScreen ()

  Limpa a tela do console.
- void waitForUserInput ()

  Aguarda até que o usuário pressione Enter.

4.5.1 Descrição detalhada

Implementa a interface do usuário e as funções de solicitação para manipulação de matrizes.

Este arquivo contém a lógica de interface do usuário necessária para interagir com o programa de manipulação de matrizes. Inclui a exibição de um menu com várias opções, como alterar um valor na matriz, adicionar ou remover linhas/colunas e calcular a soma máxima. Cada opção do menu é vinculada a uma função específica que realiza a operação desejada, solicitando entradas adicionais do usuário quando necessário. As funções de solicitação facilitam a coleta de dados de entrada do usuário para essas operações e chamam as funções lógicas apropriadas definidas em outros arquivos do projeto para executar as ações requisitadas.

Além das funções de solicitação, o arquivo define `clearScreen` e `waitForUserInput`, que melhoram a experiência do usuário ao interagir com o menu e visualizar os resultados das operações executadas.

Principais funções incluídas:

- `requestUpdateValue`: Solicita ao usuário para atualizar um valor específico na matriz.
- `requestAddRow`: Solicita ao usuário para adicionar uma nova linha na matriz.
- `requestAddColumn`: Solicita ao usuário para adicionar uma nova coluna na matriz.
- `requestRemoveRow`: Solicita ao usuário para remover uma linha específica da matriz.
- `requestRemoveColumn`: Solicita ao usuário para remover uma coluna específica da matriz.
- `requestCalculateMaxSum`: Solicita ao usuário para calcular e exibir a soma máxima conforme as condições definidas.
- `showMenuAndGetChoice`: Exibe o menu principal e coleta a escolha do usuário.
- `clearScreen`: Limpa o terminal para uma visualização clara do menu e resultados.
- `waitForUserInput`: Pausa a execução do programa até que o usuário pressione Enter, permitindo a visualização dos resultados.

4.5.2 Documentação das funções

4.5.2.1 clearScreen()

```
void clearScreen ( )
```

Limpa a tela do console.

Executa o comando apropriado para limpar a tela do console, dependendo do sistema operacional em uso. Utiliza "cls" para Windows (definido por _WIN32) e "clear" para sistemas Unix-like (como Linux e macOS). Esta função facilita a visualização das informações atualizadas no console, removendo saídas anteriores.

Referenciado por showMenuAndGetChoice().

4.5.2.2 requestAddColumn()

```
void requestAddColumn (
            Node * head,
            const char * filename )
```

Solicita ao usuário a inserção de uma nova coluna na matriz e a adiciona.

Após verificar que a lista não está vazia, a função obtém as dimensões atuais da matriz para determinar quantos valores são necessários para a nova coluna, equivalente ao número de linhas atual. O usuário é então solicitado a fornecer os valores para a nova coluna, que são lidos e armazenados em um array temporário. Utilizando esses valores, a função `addColumn` é chamada para adicionar a nova coluna à matriz. Por fim, a função informa ao usuário que a nova coluna foi adicionada com sucesso.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo onde a matriz atualizada será escrita após a adição da coluna.

Referências addColumn() e getDimensions().

Referenciado por main().

4.5.2.3 requestAddRow()

```
Node * requestAddRow (
            Node * head,
            const char * filename )
```

Solicita ao usuário a inserção de uma nova linha na matriz e adiciona a linha.

Primeiro, verifica se a lista está vazia. Se não estiver, a função obtém as dimensões atuais da matriz para determinar quantos valores o usuário deve inserir para a nova linha. Em seguida, solicita ao usuário que forneça esses valores, um por um, armazenando-os em um array temporário. Após coletar todos os valores, chama a função `addRow` para adicionar efetivamente a nova linha à matriz com os valores fornecidos pelo usuário. A função retorna o ponteiro atualizado para a cabeça da lista ligada bidimensional.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Retorna

Ponteiro para o nó cabeça atualizado da lista ligada bidimensional.

Referências addRow() e getDimensions().

Referenciado por main().

4.5.2.4 requestCalculateMaxSum()

```
void requestCalculateMaxSum (
            Node * head,
            const char * filename )
```

Solicita o cálculo da soma máxima possível em uma matriz quadrada.

Verifica inicialmente se a matriz não está vazia. Em seguida, ajusta a matriz para garantir que seja quadrada, adicionando linhas ou colunas de zeros se necessário. Após o ajuste, verifica novamente se a matriz é quadrada. Se sim, procede com o cálculo da soma máxima usando a função `calculateMaxSum`, que explora todas as combinações possíveis de seleção de valores sem repetir linhas ou colunas. O resultado é exibido ao usuário. Esta função pode ser utilizada para ajustar a matriz e realizar operações relacionadas a arquivos, como salvar a matriz ajustada, se necessário.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo que pode ser usado para operações relacionadas ao arquivo, se necessário.

Referências calculateMaxSum(), getDimensions(), makeSquare() e maxSum.

Referenciado por main().

4.5.2.5 requestRemoveColumn()

```
void requestRemoveColumn (
            Node * head,
            const char * filename )
```

Solicita ao usuário a remoção de uma coluna específica da matriz.

Primeiro, verifica se a matriz não está vazia. Se não estiver, a função obtém as dimensões atuais da matriz para informar ao usuário o número de colunas disponíveis. Em seguida, solicita ao usuário que forneça o índice da coluna que deseja remover. A função verifica a validade do índice fornecido. Se válido, `removeColumn` é chamada para remover a coluna selecionada da matriz. Após a remoção, a função exibe uma mensagem confirmando o sucesso da operação.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências getDimensions() e removeColumn().

Referenciado por main().

4.5.2.6 requestRemoveRow()

```
void requestRemoveRow (
            Node ** head,
            const char * filename )
```

Solicita ao usuário a remoção de uma linha específica da matriz.

Primeiro, verifica se a matriz não está vazia. Se não estiver, obtém as dimensões atuais da matriz para informar ao usuário quantas linhas existem. O usuário é solicitado a inserir o índice da linha que deseja remover. A função verifica se o índice fornecido é válido. Se for, a função `removeRow` é chamada para remover a linha selecionada, e o ponteiro para a cabeça da lista é atualizado caso necessário. Uma mensagem confirma a remoção bem-sucedida da linha.

Parâmetros

head	Referência ao ponteiro para o nó cabeça da lista ligada bidimensional.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências getDimensions() e removeRow().

Referenciado por main().

4.5.2.7 requestUpdateValue()

```
void requestUpdateValue (
            Node * head )
```

Solicita ao usuário valores para atualizar na matriz e aplica a atualização.

Interage com o usuário para obter um valor existente na matriz (`oldValue`) e o valor que deve substituí-lo (`newValue`). Em seguida, chama a função `updateValue` para procurar na matriz o `oldValue` e substituí-lo pelo `newValue`. Após a conclusão da atualização, uma mensagem é exibida para informar ao usuário que a operação foi realizada com sucesso. Se o `oldValue` não for encontrado na matriz, `updateValue` cuidará de informar ao usuário.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional que representa a matriz.

Referências updateValue().

Referenciado por main().

4.5.2.8 showMenuAndGetChoice()

```
int showMenuAndGetChoice ( )
```

Exibe o menu de opções para o usuário e obtém a escolha feita.

Limpa a tela e exibe um menu de opções numéricas para diferentes operações que podem ser realizadas na matriz (como visualizar a tabela de valores, modificar elementos, adicionar/remover linhas/colunas, e calcular a soma máxima). O usuário é solicitado a escolher uma opção, e a entrada é validada para garantir que seja um número correspondente a uma das opções. A função repete a solicitação até que uma entrada válida seja fornecida.

Retorna

A escolha do usuário como um inteiro, correspondendo a uma das opções do menu.

Referências clearInputBuffer() e clearScreen().

Referenciado por main().

4.5.2.9 waitForUserInput()

```
void waitForUserInput ( )
```

Aguarda até que o usuário pressione Enter.

Exibe uma mensagem solicitando que o usuário pressione Enter para continuar. Primeiro, limpa o buffer de entrada para remover qualquer caractere residual que possa ter sido deixado por entradas anteriores. Então, entra em um loop de espera até que o usuário pressione Enter, garantindo que o programa só prossiga após a ação explícita do usuário. Essa função é útil para pausar a execução entre diferentes seções de um menu ou após a execução de uma tarefa, permitindo que o usuário tenha tempo para ler as informações exibidas antes de continuar.

Referenciado por main().

4.6 Referência ao ficheiro src/menu.h

Funções

- void requestUpdateValue (Node *head)
  Solicita ao usuário valores para atualizar na matriz e aplica a atualização.
- Node * requestAddRow (Node *head, const char *filename)
  Solicita ao usuário a inserção de uma nova linha na matriz e adiciona a linha.
- void requestAddColumn (Node *head, const char *filename)
  Solicita ao usuário a inserção de uma nova coluna na matriz e a adiciona.
- void requestRemoveRow (Node **head, const char *filename)
  Solicita ao usuário a remoção de uma linha específica da matriz.
- void requestRemoveColumn (Node *head, const char *filename)
  Solicita ao usuário a remoção de uma coluna específica da matriz.
- void requestCalculateMaxSum (Node *head, const char *filename)
  Solicita o cálculo da soma máxima possível em uma matriz quadrada.
- int showMenuAndGetChoice ()
  Exibe o menu de opções para o usuário e obtém a escolha feita.
- void clearScreen ()
  Limpa a tela do console.
- void waitForUserInput ()
  Aguarda até que o usuário pressione Enter.

4.6.1 Documentação das funções

4.6.1.1 clearScreen()

```
void clearScreen ( )
```

Limpa a tela do console.

Executa o comando apropriado para limpar a tela do console, dependendo do sistema operacional em uso. Utiliza "cls" para Windows (definido por _WIN32) e "clear" para sistemas Unix-like (como Linux e macOS). Esta função facilita a visualização das informações atualizadas no console, removendo saídas anteriores.

Referenciado por showMenuAndGetChoice().

4.6.1.2 requestAddColumn()

```
void requestAddColumn (
            Node * head,
            const char * filename )
```

Solicita ao usuário a inserção de uma nova coluna na matriz e a adiciona.

Após verificar que a lista não está vazia, a função obtém as dimensões atuais da matriz para determinar quantos valores são necessários para a nova coluna, equivalente ao número de linhas atual. O usuário é então solicitado a fornecer os valores para a nova coluna, que são lidos e armazenados em um array temporário. Utilizando esses valores, a função `addColumn` é chamada para adicionar a nova coluna à matriz. Por fim, a função informa ao usuário que a nova coluna foi adicionada com sucesso.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo onde a matriz atualizada será escrita após a adição da coluna.

Referências addColumn() e getDimensions().

Referenciado por main().

4.6.1.3 requestAddRow()

```
Node * requestAddRow (
            Node * head,
            const char * filename )
```

Solicita ao usuário a inserção de uma nova linha na matriz e adiciona a linha.

Primeiro, verifica se a lista está vazia. Se não estiver, a função obtém as dimensões atuais da matriz para determinar quantos valores o usuário deve inserir para a nova linha. Em seguida, solicita ao usuário que forneça esses valores, um por um, armazenando-os em um array temporário. Após coletar todos os valores, chama a função `addRow` para adicionar efetivamente a nova linha à matriz com os valores fornecidos pelo usuário. A função retorna o ponteiro atualizado para a cabeça da lista ligada bidimensional.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Retorna

Ponteiro para o nó cabeça atualizado da lista ligada bidimensional.

Referências addRow() e getDimensions().

Referenciado por main().

4.6.1.4 requestCalculateMaxSum()

```
void requestCalculateMaxSum (
            Node * head,
            const char * filename )
```

Solicita o cálculo da soma máxima possível em uma matriz quadrada.

Verifica inicialmente se a matriz não está vazia. Em seguida, ajusta a matriz para garantir que seja quadrada, adicionando linhas ou colunas de zeros se necessário. Após o ajuste, verifica novamente se a matriz é quadrada. Se sim, procede com o cálculo da soma máxima usando a função `calculateMaxSum`, que explora todas as combinações possíveis de seleção de valores sem repetir linhas ou colunas. O resultado é exibido ao usuário. Esta função pode ser utilizada para ajustar a matriz e realizar operações relacionadas a arquivos, como salvar a matriz ajustada, se necessário.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo que pode ser usado para operações relacionadas ao arquivo, se necessário.

Referências calculateMaxSum(), getDimensions(), makeSquare() e maxSum.

Referenciado por main().

4.6.1.5 requestRemoveColumn()

```
void requestRemoveColumn (
            Node * head,
            const char * filename )
```

Solicita ao usuário a remoção de uma coluna específica da matriz.

Primeiro, verifica se a matriz não está vazia. Se não estiver, a função obtém as dimensões atuais da matriz para informar ao usuário o número de colunas disponíveis. Em seguida, solicita ao usuário que forneça o índice da coluna que deseja remover. A função verifica a validade do índice fornecido. Se válido, `removeColumn` é chamada para remover a coluna selecionada da matriz. Após a remoção, a função exibe uma mensagem confirmando o sucesso da operação.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências getDimensions() e removeColumn().

Referenciado por main().

4.6.1.6 requestRemoveRow()

```
void requestRemoveRow (
            Node ** head,
            const char * filename )
```

Solicita ao usuário a remoção de uma linha específica da matriz.

Primeiro, verifica se a matriz não está vazia. Se não estiver, obtém as dimensões atuais da matriz para informar ao usuário quantas linhas existem. O usuário é solicitado a inserir o índice da linha que deseja remover. A função verifica se o índice fornecido é válido. Se for, a função `removeRow` é chamada para remover a linha selecionada, e o ponteiro para a cabeça da lista é atualizado caso necessário. Uma mensagem confirma a remoção bem-sucedida da linha.

Parâmetros

head	Referência ao ponteiro para o nó cabeça da lista ligada bidimensional.
filename	Nome do arquivo onde a matriz atualizada será escrita.

Referências getDimensions() e removeRow().

Referenciado por main().

4.6.1.7 requestUpdateValue()

```
void requestUpdateValue (
            Node * head )
```

Solicita ao usuário valores para atualizar na matriz e aplica a atualização.

Interage com o usuário para obter um valor existente na matriz (`oldValue`) e o valor que deve substituí-lo (`new↩Value`). Em seguida, chama a função `updateValue` para procurar na matriz o `oldValue` e substituí-lo pelo `newValue`. Após a conclusão da atualização, uma mensagem é exibida para informar ao usuário que a operação foi realizada com sucesso. Se o `oldValue` não for encontrado na matriz, `updateValue` cuidará de informar ao usuário.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional que representa a matriz.

Referências updateValue().

Referenciado por main().

4.6.1.8 showMenuAndGetChoice()

```
int showMenuAndGetChoice ( )
```

Exibe o menu de opções para o usuário e obtém a escolha feita.

Limpa a tela e exibe um menu de opções numéricas para diferentes operações que podem ser realizadas na matriz (como visualizar a tabela de valores, modificar elementos, adicionar/remover linhas/colunas, e calcular a soma máxima). O usuário é solicitado a escolher uma opção, e a entrada é validada para garantir que seja um número correspondente a uma das opções. A função repete a solicitação até que uma entrada válida seja fornecida.

Retorna

A escolha do usuário como um inteiro, correspondendo a uma das opções do menu.

Referências clearInputBuffer() e clearScreen().

Referenciado por main().

4.6.1.9 waitForUserInput()

```
void waitForUserInput ( )
```

Aguarda até que o usuário pressione Enter.

Exibe uma mensagem solicitando que o usuário pressione Enter para continuar. Primeiro, limpa o buffer de entrada para remover qualquer caractere residual que possa ter sido deixado por entradas anteriores. Então, entra em um loop de espera até que o usuário pressione Enter, garantindo que o programa só prossiga após a ação explícita do usuário. Essa função é útil para pausar a execução entre diferentes seções de um menu ou após a execução de uma tarefa, permitindo que o usuário tenha tempo para ler as informações exibidas antes de continuar.

Referenciado por main().

4.7 menu.h

Ir para a documentação deste ficheiro.

```
1 #ifndef MENU_H
2 #define MENU_H
3
14 void requestUpdateValue(Node* head);
15
29 Node* requestAddRow(Node* head, const char* filename);
30
43 void requestAddColumn(Node* head, const char* filename);
44
57 void requestRemoveRow(Node** head, const char* filename);
58
71 void requestRemoveColumn(Node* head, const char* filename);
72
86 void requestCalculateMaxSum(Node* head, const char* filename);
87
99 int showMenuAndGetChoice();
100
108 void clearScreen();
109
120 void waitForUserInput();
121
122 #endif
```

4.8 Referência ao ficheiro src/types.h

Estruturas de Dados

- struct Node

Definições de tipos

- typedef struct Node Node

4.8.1 Documentação dos tipos

4.8.1.1 Node

```
typedef struct Node Node
```

4.9 types.h

Ir para a documentação deste ficheiro.

```
1 #ifndef TYPES_H
2 #define TYPES_H
3
4 typedef struct Node {
5     int data;
6     struct Node *right;
7     struct Node *down;
8 } Node;
9
10 #endif
```

4.10 Referência ao ficheiro src/utils.c

Fornece funções utilitárias para a manipulação de uma matriz representada como uma lista ligada bidimensional.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "types.h"
#include <stdbool.h>
#include "utils.h"
#include <stddef.h>
#include "logic.h"
```

Funções

- Node ∗ createNode (int data)

  Cria um novo nó para ser utilizado na matriz bidimensional representada por uma lista ligada.
- Node ∗ readFile (char ∗filename)

  Lê uma matriz de inteiros de um arquivo e constrói uma representação de lista ligada bidimensional.
- void writeToFile (Node ∗head, const char ∗filename)

  Escreve o conteúdo da matriz representada por uma lista ligada bidimensional em um arquivo.
- void printList (Node ∗head)

  Imprime a matriz representada por uma lista ligada bidimensional.
- void getDimensions (Node ∗head, int ∗numRows, int ∗numCols)

  Calcula as dimensões de uma matriz representada por uma lista ligada bidimensional.
- void makeSquare (Node ∗∗head)

  Ajusta a matriz para ser quadrada, adicionando linhas ou colunas de zeros conforme necessário.
- void calculateMaxSumUtil (Node ∗row, int ∗visited, int currentSum, int ∗maxSum, int numRows)

  Função auxiliar recursiva para calcular a soma máxima de valores selecionados da matriz.
- void clearInputBuffer ()

  Limpa o buffer de entrada do stdin.

Variáveis

- int maxSum = 0

4.10.1 Descrição detalhada

Fornece funções utilitárias para a manipulação de uma matriz representada como uma lista ligada bidimensional.

Este arquivo contém a implementação de várias funções utilitárias que apoiam a manipulação e visualização da matriz, incluindo a criação de novos nós, leitura e escrita de matrizes em arquivos, impressão da matriz no console, obtenção das dimensões da matriz, ajuste da matriz para garantir que seja quadrada e cálculo auxiliar para a soma máxima. Estas funções são essenciais para o funcionamento das funcionalidades oferecidas pelo programa, permitindo uma interação eficiente e eficaz com a estrutura de dados da matriz e com a persistência dos dados em arquivos.

Principais funcionalidades incluídas:

- `createNode`: Aloca e inicializa um novo nó para a lista ligada.
- `readFile`: Lê uma matriz de um arquivo e constrói uma representação de lista ligada bidimensional.
- `writeToFile`: Escreve a matriz atualizada em um arquivo, preservando as alterações feitas pelo usuário.
- `printList`: Exibe a matriz atual no console em um formato de tabela fácil de ler.
- `getDimensions`: Calcula e retorna as dimensões atuais da matriz.
- `makeSquare`: Ajusta a matriz para ser quadrada, adicionando linhas ou colunas de zeros conforme necessário.
- `calculateMaxSumUtil`: Função auxiliar utilizada para calcular a soma máxima de valores selecionados da matriz.
- `clearInputBuffer`: Limpa o buffer de entrada para evitar leituras indesejadas durante a coleta de entrada do usuário.

Estas funções proporcionam uma base sólida para a realização de operações complexas sobre a matriz de maneira simples e direta.

4.10.2 Documentação das funções

4.10.2.1 calculateMaxSumUtil()

```
void calculateMaxSumUtil (
            Node * row,
            int * visited,
            int currentSum,
            int * maxSum,
            int numRows )
```

Função auxiliar recursiva para calcular a soma máxima de valores selecionados da matriz.

Esta função é chamada recursivamente para explorar todas as possíveis combinações de seleção de nós na matriz, sob a condição de que apenas um nó por linha e por coluna seja escolhido. Ela utiliza um vetor de visitados `visited` para marcar as colunas já exploradas em cada caminho recursivo, evitando seleções repetidas e garantindo que a restrição de um nó por coluna seja respeitada.

A cada chamada recursiva, a função avança para a próxima linha (`row->down`) e tenta adicionar o valor de cada nó acessível (não visitado) na linha atual ao `currentSum`. Se a última linha for atingida e o `currentSum` for maior que o `maxSum` atual, `maxSum` é atualizado com o valor de `currentSum`.

Parâmetros

row	Ponteiro para o nó atual que representa a linha sendo explorada na chamada recursiva.
visited	Vetor que indica quais colunas já foram visitadas/consideradas na seleção atual.
currentSum	Soma acumulada dos valores dos nós selecionados até o momento.
maxSum	Ponteiro para o valor da maior soma encontrada durante a execução da função.
numRows	Número total de linhas da matriz, usado para dimensionar o vetor `visited`.

Referências calculateMaxSumUtil(), Node::data, Node::down, maxSum e Node::right.

Referenciado por calculateMaxSum() e calculateMaxSumUtil().

4.10.2.2 clearInputBuffer()

```
void clearInputBuffer ( )
```

Limpa o buffer de entrada do stdin.

Esta função lê e descarta caracteres do buffer de entrada até que um caractere de nova linha ('
') ou o fim do arquivo (EOF) seja encontrado. É útil para remover caracteres indesejados ou não lidos após uma entrada de usuário, garantindo que leituras subsequentes de entrada funcionem como esperado.

Referenciado por showMenuAndGetChoice().

4.10.2.3 createNode()

```
Node * createNode (
            int data )
```

Cria um novo nó para ser utilizado na matriz bidimensional representada por uma lista ligada.

Aloca memória para um novo nó da lista ligada e inicializa seus campos. O campo `data` é definido com o valor passado como argumento, enquanto os ponteiros `right` e `down`, que apontam, respectivamente, para o próximo nó na mesma linha e para o próximo nó na mesma coluna, são inicializados como NULL. Se a alocação de memória falhar, o programa exibe uma mensagem de erro e encerra a execução.

Parâmetros

data	O valor do inteiro a ser armazenado no novo nó.

Retorna

Um ponteiro para o novo nó criado.

Referências Node::data, Node::down e Node::right.

Referenciado por addColumn(), addRow() e readFile().

4.10.2.4 getDimensions()

```
void getDimensions (
            Node * head,
            int * numRows,
            int * numCols )
```

Calcula as dimensões de uma matriz representada por uma lista ligada bidimensional.

Através da travessia da estrutura da lista ligada bidimensional, esta função determina o número total de linhas e colunas na matriz. Os valores calculados são armazenados nas variáveis apontadas por `numRows` e `numCols`, respectivamente. A função inicia no nó cabeça da lista (`head`) e percorre a estrutura tanto na direção horizontal (para contar colunas) quanto na direção vertical (para contar linhas), atualizando os contadores conforme avança.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
numRows	Ponteiro para a variável onde o número total de linhas será armazenado.
numCols	Ponteiro para a variável onde o número total de colunas será armazenado.

Referências Node::down e Node::right.

Referenciado por makeSquare(), requestAddColumn(), requestAddRow(), requestCalculateMaxSum(), requestRemoveColumn() e requestRemoveRow().

4.10.2.5 makeSquare()

```
void makeSquare (
            Node ** head )
```

Ajusta a matriz para ser quadrada, adicionando linhas ou colunas de zeros conforme necessário.

Avalia as dimensões atuais da matriz e determina se é necessário adicionar linhas ou colunas para torná-la quadrada. Linhas ou colunas adicionais são preenchidas com zeros. Esta operação é essencial para garantir que a matriz atenda aos requisitos de determinados algoritmos que assumem uma estrutura quadrada. Os ajustes são feitos in-place na estrutura de lista ligada bidimensional, e a matriz ajustada é opcionalmente escrita de volta ao arquivo especificado.

Parâmetros

head	Referência ao ponteiro para o nó cabeça da lista ligada bidimensional.
filename	Nome do arquivo onde a matriz ajustada pode ser escrita. Este parâmetro é usado apenas se linhas ou colunas forem efetivamente adicionadas à matriz.

Referências addColumn(), addRow() e getDimensions().

Referenciado por requestCalculateMaxSum().

4.10.2.6 printList()

```
void printList (
            Node * head )
```

Imprime a matriz representada por uma lista ligada bidimensional.

Esta função percorre a estrutura de lista ligada bidimensional apontada por `head`, imprimindo cada valor de nó em um formato de tabela. Cada célula da tabela é delimitada por linhas horizontais e verticais, com cada valor alinhado à direita dentro de uma largura fixa. A função assume que os valores dos nós têm até três dígitos. Esta representação visual facilita a compreensão da estrutura da matriz no console.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.

Referências Node::data, Node::down e Node::right.

Referenciado por main().

4.10.2.7 readFile()

```
Node * readFile (
            char * filename )
```

Lê uma matriz de inteiros de um arquivo e constrói uma representação de lista ligada bidimensional.

Abre o arquivo especificado por `filename` e lê sua conteúdo linha por linha, onde os valores inteiros são esperados estar separados por ponto e vírgula (`;`). Cada linha do arquivo corresponde a uma linha da matriz e cada valor é inserido em um novo nó na lista ligada. Os nós são conectados tanto horizontalmente (na mesma linha) quanto verticalmente (na mesma coluna) para formar a estrutura bidimensional. A função retorna um ponteiro para o nó cabeça da lista ligada, que representa o canto superior esquerdo da matriz. Em caso de falha ao abrir o arquivo, o programa exibe uma mensagem de erro e termina.

Parâmetros

filename	Nome do arquivo de onde a matriz será lida.

Retorna

Um ponteiro para o nó cabeça da lista ligada bidimensional que representa a matriz lida.

Referências createNode(), Node::down e Node::right.

Referenciado por main().

4.10.2.8 writeToFile()

```
void writeToFile (
            Node * head,
            const char * filename )
```

Escreve o conteúdo da matriz representada por uma lista ligada bidimensional em um arquivo.

Esta função abre o arquivo especificado por `filename` para escrita e percorre a estrutura de lista ligada bidimensional apontada por `head`, escrevendo os dados de cada nó no arquivo. Os valores dentro de uma mesma linha são separados por ponto e vírgula (`;`), e cada linha da matriz é escrita em uma nova linha no arquivo. Se não for possível abrir o arquivo para escrita, a função exibirá uma mensagem de erro e terminará o programa.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo no qual a matriz será escrita.

Referências Node::data, Node::down e Node::right.

Referenciado por addColumn(), addRow(), removeColumn(), removeRow() e updateValue().

4.10.3 Documentação das variáveis

4.10.3.1 maxSum

```
int maxSum = 0
```

Referenciado por calculateMaxSum(), calculateMaxSumUtil() e requestCalculateMaxSum().

4.11 Referência ao ficheiro src/utils.h

```
#include "types.h"
```

Funções

- Node ∗ createNode (int data)

  Cria um novo nó para ser utilizado na matriz bidimensional representada por uma lista ligada.
- Node ∗ readFile (char ∗filename)

  Lê uma matriz de inteiros de um arquivo e constrói uma representação de lista ligada bidimensional.
- void writeToFile (Node ∗head, const char ∗filename)

  Escreve o conteúdo da matriz representada por uma lista ligada bidimensional em um arquivo.
- void printList (Node ∗head)

  Imprime a matriz representada por uma lista ligada bidimensional.
- void getDimensions (Node ∗head, int ∗numRows, int ∗numCols)

  Calcula as dimensões de uma matriz representada por uma lista ligada bidimensional.
- void makeSquare (Node ∗∗head)

  Ajusta a matriz para ser quadrada, adicionando linhas ou colunas de zeros conforme necessário.
- void calculateMaxSumUtil (Node ∗row, int ∗visited, int currentSum, int ∗maxSum, int numRows)

  Função auxiliar recursiva para calcular a soma máxima de valores selecionados da matriz.
- void clearInputBuffer ()

  Limpa o buffer de entrada do stdin.

4.11.1 Documentação das funções

4.11.1.1 calculateMaxSumUtil()

```
void calculateMaxSumUtil (
            Node * row,
            int * visited,
            int currentSum,
            int * maxSum,
            int numRows )
```

Função auxiliar recursiva para calcular a soma máxima de valores selecionados da matriz.

Esta função é chamada recursivamente para explorar todas as possíveis combinações de seleção de nós na matriz, sob a condição de que apenas um nó por linha e por coluna seja escolhido. Ela utiliza um vetor de visitados `visited` para marcar as colunas já exploradas em cada caminho recursivo, evitando seleções repetidas e garantindo que a restrição de um nó por coluna seja respeitada.

A cada chamada recursiva, a função avança para a próxima linha (`row->down`) e tenta adicionar o valor de cada nó acessível (não visitado) na linha atual ao `currentSum`. Se a última linha for atingida e o `currentSum` for maior que o `maxSum` atual, `maxSum` é atualizado com o valor de `currentSum`.

Parâmetros

row	Ponteiro para o nó atual que representa a linha sendo explorada na chamada recursiva.
visited	Vetor que indica quais colunas já foram visitadas/consideradas na seleção atual.
currentSum	Soma acumulada dos valores dos nós selecionados até o momento.
maxSum	Ponteiro para o valor da maior soma encontrada durante a execução da função.
numRows	Número total de linhas da matriz, usado para dimensionar o vetor `visited`.

Referências calculateMaxSumUtil(), Node::data, Node::down, maxSum e Node::right.

Referenciado por calculateMaxSum() e calculateMaxSumUtil().

4.11.1.2 clearInputBuffer()

```
void clearInputBuffer ( )
```

Limpa o buffer de entrada do stdin.

Esta função lê e descarta caracteres do buffer de entrada até que um caractere de nova linha ('
') ou o fim do arquivo (EOF) seja encontrado. É útil para remover caracteres indesejados ou não lidos após uma entrada de usuário, garantindo que leituras subsequentes de entrada funcionem como esperado.

Referenciado por showMenuAndGetChoice().

4.11.1.3 createNode()

```
Node * createNode (
            int data )
```

Cria um novo nó para ser utilizado na matriz bidimensional representada por uma lista ligada.

Aloca memória para um novo nó da lista ligada e inicializa seus campos. O campo `data` é definido com o valor passado como argumento, enquanto os ponteiros `right` e `down`, que apontam, respectivamente, para o próximo nó na mesma linha e para o próximo nó na mesma coluna, são inicializados como NULL. Se a alocação de memória falhar, o programa exibe uma mensagem de erro e encerra a execução.

Parâmetros

data	O valor do inteiro a ser armazenado no novo nó.

Retorna

Um ponteiro para o novo nó criado.

Referências Node::data, Node::down e Node::right.

Referenciado por addColumn(), addRow() e readFile().

4.11.1.4 getDimensions()

```
void getDimensions (
            Node * head,
            int * numRows,
            int * numCols )
```

Calcula as dimensões de uma matriz representada por uma lista ligada bidimensional.

Através da travessia da estrutura da lista ligada bidimensional, esta função determina o número total de linhas e colunas na matriz. Os valores calculados são armazenados nas variáveis apontadas por `numRows` e `numCols`, respectivamente. A função inicia no nó cabeça da lista (`head`) e percorre a estrutura tanto na direção horizontal (para contar colunas) quanto na direção vertical (para contar linhas), atualizando os contadores conforme avança.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
numRows	Ponteiro para a variável onde o número total de linhas será armazenado.
numCols	Ponteiro para a variável onde o número total de colunas será armazenado.

Referências Node::down e Node::right.

Referenciado por makeSquare(), requestAddColumn(), requestAddRow(), requestCalculateMaxSum(), requestRemoveColumn() e requestRemoveRow().

4.11.1.5 makeSquare()

```
void makeSquare (
            Node ** head )
```

Ajusta a matriz para ser quadrada, adicionando linhas ou colunas de zeros conforme necessário.

Avalia as dimensões atuais da matriz e determina se é necessário adicionar linhas ou colunas para torná-la quadrada. Linhas ou colunas adicionais são preenchidas com zeros. Esta operação é essencial para garantir que a matriz atenda aos requisitos de determinados algoritmos que assumem uma estrutura quadrada. Os ajustes são feitos in-place na estrutura de lista ligada bidimensional, e a matriz ajustada é opcionalmente escrita de volta ao arquivo especificado.

Parâmetros

head	Referência ao ponteiro para o nó cabeça da lista ligada bidimensional.
filename	Nome do arquivo onde a matriz ajustada pode ser escrita. Este parâmetro é usado apenas se linhas ou colunas forem efetivamente adicionadas à matriz.

Referências addColumn(), addRow() e getDimensions().

Referenciado por requestCalculateMaxSum().

4.11.1.6 printList()

```
void printList (
            Node * head )
```

Imprime a matriz representada por uma lista ligada bidimensional.

Esta função percorre a estrutura de lista ligada bidimensional apontada por `head`, imprimindo cada valor de nó em um formato de tabela. Cada célula da tabela é delimitada por linhas horizontais e verticais, com cada valor alinhado à direita dentro de uma largura fixa. A função assume que os valores dos nós têm até três dígitos. Esta representação visual facilita a compreensão da estrutura da matriz no console.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.

Referências Node::data, Node::down e Node::right.

Referenciado por main().

4.11.1.7 readFile()

```
Node * readFile (
            char * filename )
```

Lê uma matriz de inteiros de um arquivo e constrói uma representação de lista ligada bidimensional.

Abre o arquivo especificado por `filename` e lê sua conteúdo linha por linha, onde os valores inteiros são esperados estar separados por ponto e vírgula (`;`). Cada linha do arquivo corresponde a uma linha da matriz e cada valor é inserido em um novo nó na lista ligada. Os nós são conectados tanto horizontalmente (na mesma linha) quanto verticalmente (na mesma coluna) para formar a estrutura bidimensional. A função retorna um ponteiro para o nó cabeça da lista ligada, que representa o canto superior esquerdo da matriz. Em caso de falha ao abrir o arquivo, o programa exibe uma mensagem de erro e termina.

Parâmetros

filename	Nome do arquivo de onde a matriz será lida.

Retorna

Um ponteiro para o nó cabeça da lista ligada bidimensional que representa a matriz lida.

Referências createNode(), Node::down e Node::right.

Referenciado por main().

4.11.1.8 writeToFile()

```
void writeToFile (
            Node * head,
            const char * filename )
```

Escreve o conteúdo da matriz representada por uma lista ligada bidimensional em um arquivo.

Esta função abre o arquivo especificado por `filename` para escrita e percorre a estrutura de lista ligada bidimensional apontada por `head`, escrevendo os dados de cada nó no arquivo. Os valores dentro de uma mesma linha são separados por ponto e vírgula (`;`), e cada linha da matriz é escrita em uma nova linha no arquivo. Se não for possível abrir o arquivo para escrita, a função exibirá uma mensagem de erro e terminará o programa.

Parâmetros

head	Ponteiro para o nó cabeça da lista ligada bidimensional representando a matriz.
filename	Nome do arquivo no qual a matriz será escrita.

Referências Node::data, Node::down e Node::right.

Referenciado por addColumn(), addRow(), removeColumn(), removeRow() e updateValue().

4.12 utils.h

Ir para a documentação deste ficheiro.

```
1 #ifndef UTILS_H
2 #define UTILS_H
3
4 #include "types.h"
5
18 Node* createNode(int data);
19
33 Node* readFile(char* filename);
34
46 void writeToFile(Node* head, const char* filename);
47
58 void printList(Node* head);
59
72 void getDimensions(Node *head, int *numRows, int *numCols);
73
86 void makeSquare(Node **head);
87
106 void calculateMaxSumUtil(Node* row, int* visited, int currentSum, int* maxSum, int numRows);
107
115 void clearInputBuffer();
116
117 #endif
```

Índice