

Sparse Matrices Operations

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	Sparse Struct Reference	5
3.1.1	Member Data Documentation	5
3.1.1.1	Column	5
3.1.1.2	Line	5
3.1.1.3	Value	5
3.1.1.4	nrElements	5
3.1.1.5	nrLines	5
3.1.1.6	nrColumns	5
4	File Documentation	7
4.1	C:/Users/Paul/Desktop/ProiectC/functions.c File Reference	7
4.1.1	Detailed Description	7
4.2	C:/Users/Paul/Desktop/ProiectC/functions.h File Reference	7
4.2.1	Detailed Description	8
4.2.2	Typedef Documentation	8
4.2.2.1	SparseMatrix	8
4.2.3	Function Documentation	8
4.2.3.1	allocate(int ***mat, int m, int n)	8

4.2.3.2	<code>allocate_and_read(int ***matrice1, int ***matrice2, int *m, int *n, int *p, int *q)</code>	8
4.2.3.3	<code>ERROR(SparseMatrix R)</code>	8
4.2.3.4	<code>freeEverything(SparseMatrix rare)</code>	8
4.2.3.5	<code>generate_input(int choice)</code>	8
4.2.3.6	<code>generate_matrix(int **mat, int m, int n)</code>	9
4.2.3.7	<code>generate_number(int x)</code>	9
4.2.3.8	<code>main()</code>	9
4.2.3.9	<code>print_matrix(int **matrice, int m, int n, FILE *g)</code>	9
4.2.3.10	<code>print_sparse_matrix(SparseMatrix sparse, FILE *g)</code>	9
4.2.3.11	<code>resolve(int **NormalMat1, int **NormalMat2, int m, int n, int p, int q, int choice)</code>	9
4.2.3.12	<code>Sparse_to_Normal(SparseMatrix *matR)</code>	9
4.2.3.13	<code>SparseMatrix_Addition(SparseMatrix *M1, SparseMatrix *M2)</code>	10
4.2.3.14	<code>SparseMatrix_Conversion(int **mat, int m, int n)</code>	10
4.2.3.15	<code>SparseMatrix_Product(SparseMatrix *M1, SparseMatrix *M2)</code>	10
4.2.3.16	<code>write_matrix(int **mat, int m, int n, FILE *f)</code>	10
4.3	<code>C:/Users/Paul/Desktop/ProiectC/generare.c</code> File Reference	11
4.3.1	Detailed Description	11
4.4	<code>C:/Users/Paul/Desktop/ProiectC/main.c</code> File Reference	11
4.4.1	Detailed Description	11
4.4.2	Macro Definition Documentation	11
4.4.2.1	<code>number_of_testes</code>	11
	Index	13

Chapter 1

Class Index

1.1 Class List

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

Sparse	5
----------------------------------	---

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

C:/Users/Paul/Desktop/ProiectC/ functions.c	
C library implementation for sparse matrix operations	7
C:/Users/Paul/Desktop/ProiectC/ functions.h	
C library for sparse matrices operations and sparse matrices generation	7
C:/Users/Paul/Desktop/ProiectC/ generare.c	
C library implementation to generate sparse matrices	11
C:/Users/Paul/Desktop/ProiectC/ main.c	
Libraries 7. : A library for operations with sparse matrices	11

Chapter 3

Class Documentation

3.1 Sparse Struct Reference

```
#include <functions.h>
```

Public Attributes

- int * [Line](#)
- int * [Column](#)
- int * [Value](#)

3 pointers used to store the nnull elements and their positions on matrix

- int [nrElements](#)
- int [nrLines](#)
- int [nrColumns](#)

3 variables used to store the number of nnull elements,the number of lines and columns from the normal matrix

3.1.1 Member Data Documentation

3.1.1.1 int * Sparse::Column

3.1.1.2 int* Sparse::Line

3.1.1.3 int * Sparse::Value

3 pointers used to store the nnull elements and their positions on matrix

3.1.1.4 int Sparse::nrElements

3.1.1.5 int Sparse::nrLines

3.1.1.6 int Sparse::nrColumns

3 variables used to store the number of nnull elements,the number of lines and columns from the normal matrix

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

- C:/Users/Paul/Desktop/ProiectC/[functions.h](#)

Chapter 4

File Documentation

4.1 C:/Users/Paul/Desktop/ProiectC/functions.c File Reference

C library implementation for sparse matrix operations.

```
#include <stdlib.h>
#include <stdio.h>
#include "functions.h"
```

4.1.1 Detailed Description

C library implementation for sparse matrix operations.

Implements the summ and the product of two sparse matrices, the conversions between a normal matrix and a sparse one and the printing of the matrices in both formes.

4.2 C:/Users/Paul/Desktop/ProiectC/functions.h File Reference

C library for sparse matrices operations and sparse matrices generation.

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

Classes

- struct [Sparse](#)

Typedefs

- typedef struct [Sparse](#) * [SparseMatrix](#)
The alias used to declare variable of [Sparse](#) type.

Functions

- `int main ()`
- `void freeEverything (SparseMatrix rare)`
- `SparseMatrix SparseMatrix_Conversion (int **mat, int m, int n)`
- `SparseMatrix SparseMatrix_Addition (SparseMatrix *M1, SparseMatrix *M2)`
- `SparseMatrix SparseMatrix_Product (SparseMatrix *M1, SparseMatrix *M2)`
- `void print_matrix (int **matrice, int m, int n, FILE *g)`
- `void print_sparse_matrix (SparseMatrix sparse, FILE *g)`
- `void allocate_and_read (int ***matrice1, int ***matrice2, int *m, int *n, int *p, int *q)`
- `int ** Sparse_to_Normal (SparseMatrix *matR)`
- `int ERROR (SparseMatrix R)`
- `void resolve (int **NormalMat1, int **NormalMat2, int m, int n, int p, int q, int choice)`
- `void generate_input (int choice)`
- `int generate_number (int x)`
- `void generate_matrix (int **mat, int m, int n)`
- `void write_matrix (int **mat, int m, int n, FILE *f)`
- `void allocate (int ***mat, int m, int n)`

4.2.1 Detailed Description

C library for sparse matrices operations and sparse matrices generation.

4.2.2 Typedef Documentation

4.2.2.1 typedef struct Sparse * SparseMatrix

The alias used to declare variable of `Sparse` type.

4.2.3 Function Documentation

4.2.3.1 void allocate (int *** mat, int m, int n)

Function used to allocate a matrix using double pointer transmitted by reference. It is used the `calloc` function to initialize simultaneously elements with null elements.

4.2.3.2 void allocate_and_read (int *** matrice1, int *** matrice2, int * m, int * n, int * p, int * q)

It allocates and reads two matrices. The elements are read from an input file.

4.2.3.3 int ERROR (SparseMatrix R)

If the fields of structure are not filled, results that the addition or the product operation failed so the function returns true or false.

4.2.3.4 void freeEverything (SparseMatrix rare)

Deallocate memory used for structure variable and the structure fields.

4.2.3.5 void generate_input (int choice)

Function that uses the above functions to provide the input, such that the conditions for sum or product of matrices will be satisfied in equal measure, also for the fail case, when the dimensions are entirely random generated.

Parameters

<i>choice</i>	Parameter used to generate matrices that satisfies the conditions for matrices summ or product or it generates completely random matrices used for one of the operations.
---------------	---

4.2.3.6 void generate_matrix (int ** *mat*, int *m*, int *n*)

Function used to create a sparse matrix by generating a small number of nennull elements and their positions in the matrix

4.2.3.7 int generate_number (int *x*)

Function used to generate a random number between 0 and *x* ,using rand() function

4.2.3.8 int main ()

Main function : generate matrices, allocate space for matrices, resolve operations with matrices. Functions calls of generate_input,allocate_and_read and resolve are imported from [functions.h](#) header file

4.2.3.9 void print_matrix (int ** *matrice*, int *m*, int *n*, FILE * *g*)

It prints a matrix in the file specified by **g* file pointer

4.2.3.10 void print_sparse_matrix (SparseMatrix *sparse*, FILE * *g*)

It prints a sparse matrix ih this form: the line index, the column index and the nennull value

4.2.3.11 void resolve (int ** *NormalMat1*, int ** *NormalMat2*, int *m*, int *n*, int *p*, int *q*, int *choice*)

Function where the above operations are executed and the results are printed in an output file

Parameters

<i>**NormalMat1,**NormalMat2</i>	Two double pointers used for the matrices operations
<i>m,n,p,q</i>	The number of lines and columns of the matrices
<i>choice</i>	Parameter which will randomly decide what operation is executed

4.2.3.12 int ** Sparse_to_Normal (SparseMatrix * *matR*)

Create a normal matrix from the compressed form.

Parameters

<i>*matR</i>	A SparseMatrix type parameter
--------------	-------------------------------

4.2.3.13 SparseMatrix SparseMatrix_Addition (SparseMatrix * *M1*, SparseMatrix * *M2*)

Computes the addition of two sparse matrices. It returns the resulted matrix as a structure/

Parameters

<i>*M1,*M2</i>	Two matrices stored in structure used for addition
----------------	--

4.2.3.14 SparseMatrix SparseMatrix_Conversion (int ** *mat*, int *m*, int *n*)

Computes the conversion of a matrix to a form where only the nennull elements are stored with their positions, using a structure . It returns the resulted structure.

Parameters

<i>**mat</i>	Double pointer which stores the elements of matrix
<i>m,n</i>	The number of lines and columns of matrix

4.2.3.15 SparseMatrix SparseMatrix_Product (SparseMatrix * *M1*, SparseMatrix * *M2*)

Computes the product of two sparse matrices stored using the structure. It returns the resulted matrix by a structure.

Parameters

<i>*M1,*M2</i>	Two structures used to store two sparse matrices used for product
----------------	---

4.2.3.16 void write_matrix (int ** *mat*, int *m*, int *n*, FILE * *f*)

Function used to write a matrix in the file specified by the file pointer parameter

Parameters

<i>**mat</i>	Double pointer used to store the elements of matrix
<i>m,n</i>	Dimensions of matrix
<i>*f</i>	File pointer used to write at the position described

4.3 C:/Users/Paul/Desktop/ProiectC/generare.c File Reference

C library implementation to generate sparse matrices.

```
#include "functions.h"
#include <stdio.h>
#include <stdlib.h>
```

4.3.1 Detailed Description

C library implementation to generate sparse matrices.

The program implements functions to generate a sparse matrix, to write a matrix, to allocate a matrix and a final function which includes all of the previous functions and is used in the main program.

4.4 C:/Users/Paul/Desktop/ProiectC/main.c File Reference

Libraries 7. : A library for operations with sparse matrices.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "functions.h"
```

Macros

- `#define number_of_testes 5`

The number of operations we will make on matrices.

4.4.1 Detailed Description

Libraries 7. : A library for operations with sparse matrices.

Created by Ionescu Iulian

4.4.2 Macro Definition Documentation

4.4.2.1 `#define number_of_testes 5`

The number of operations we will make on matrices.

Index

allocate
 functions.h, 8

allocate_and_read
 functions.h, 8

C:/Users/Paul/Desktop/ProiectC/functions.c, 7

C:/Users/Paul/Desktop/ProiectC/functions.h, 7

C:/Users/Paul/Desktop/ProiectC/generare.c, 11

C:/Users/Paul/Desktop/ProiectC/main.c, 11

Column
 Sparse, 5

ERROR
 functions.h, 8

freeEverything
 functions.h, 8

functions.h
 allocate, 8
 allocate_and_read, 8
 ERROR, 8
 freeEverything, 8
 generate_input, 8
 generate_matrix, 9
 generate_number, 9
 main, 9
 print_matrix, 9
 print_sparse_matrix, 9
 resolve, 9
 Sparse_to_Normal, 9
 SparseMatrix, 8
 SparseMatrix_Addition, 10
 SparseMatrix_Conversion, 10
 SparseMatrix_Product, 10
 write_matrix, 10

generate_input
 functions.h, 8

generate_matrix
 functions.h, 9

generate_number
 functions.h, 9

Line
 Sparse, 5

main
 functions.h, 9

main.c
 number_of_testes, 11

nrColumns
 Sparse, 5

nrElements
 Sparse, 5

nrLines
 Sparse, 5

number_of_testes
 main.c, 11

print_matrix
 functions.h, 9

print_sparse_matrix
 functions.h, 9

resolve
 functions.h, 9

Sparse, 5
 Column, 5
 Line, 5
 nrColumns, 5
 nrElements, 5
 nrLines, 5
 Value, 5

Sparse_to_Normal
 functions.h, 9

SparseMatrix
 functions.h, 8

SparseMatrix_Addition
 functions.h, 10

SparseMatrix_Conversion
 functions.h, 10

SparseMatrix_Product
 functions.h, 10

Value
 Sparse, 5

write_matrix
 functions.h, 10