Project 3

刘一郴 12112609

1. Code Component

1.1. head.h

Some include and define to easy following programing.

文本

描述已自动生成

1.2. Matrix.h

1.2.1. struction Matrix

Specially it has a pointer to next Matrix which can be used to form the **Linked List** to record current existed Matrixs.

文本

描述已自动生成

1.2.2. pFirstMatrix

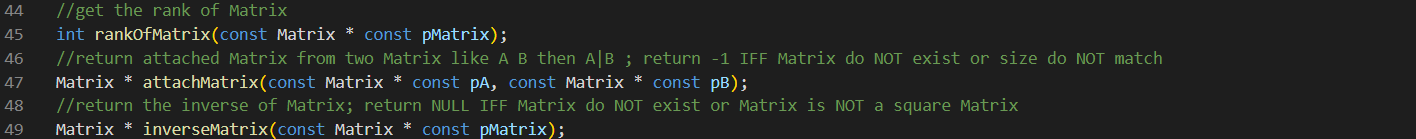
It is used to show the head of **Linked List** proposed above, **initially NULL**.



1.2.3. functions

Claims of Matrix Functions including all requested functions as well as some **extra functions** like **transposeMatrix(), standardMatrix(), gussianEliminationMatrix(), rankOfMatrix(), attachMatrix(), inverseMatrix()**.

文本

描述已自动生成

1.3. Matrix.c

Because the code length is to long, we pick some import function as examples. Moreover, some function is **quite easy and needless to explain**.

1.3.1. gussianEliminationMatrix

return Gussian Elimination of Matrix ; return NULL IFF Matrix do NOT exist or size do NOT match.

Reduce the matrix into triangular form.

文本

描述已自动生成

1.3.2. inverseMatrix

return the inverse of Matrix; return NULL IFF Matrix do NOT exist or Matrix is NOT a square Matrix.

**Using the theorem [A | I] = A \* [I | A^-1]**

文本

描述已自动生成电脑萤幕画面

描述已自动生成

2. requirements

2.1. The programming language can only be C, not C++. Please save your source code into \*.c files, and compile them using a C compiler such as gcc (not g++). Try to use Makefile or CMake to manage your source code.

图形用户界面, 文本, 应用程序

描述已自动生成

2.2. Design a struct for matrices, and the struct should contain the data of a matrix, the number of columns, the number of rows, etc.

图片包含 图形用户界面

描述已自动生成

2.3. Only float elements in a matrix are supported. You do not need to implement some other types.

As above figure

2.4. Implement some functions

All the required functions are implemented as well as some **extra functions** like **transposeMatrix(), standardMatrix(), gussianEliminationMatrix(), rankOfMatrix(), attachMatrix(), inverseMatrix()**.

2.5. The designed functions should be safe and easy to use. Suppose you are designing a library for others to use. You do not need to focus on the optimization in this project, ease of use is more important.

In this project all the existing Matrixs are stored in one **Linked List**, so we can check if the Matrixs given by user is valid by following function **existMatrix()**.

屏幕上有字

描述已自动生成

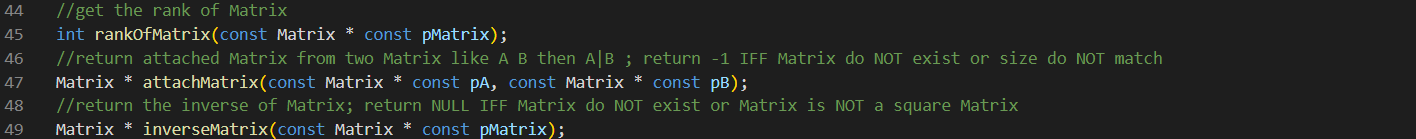
3. highlights

3.1. well written description for functions

All the **parameter type** are well claimed.

All the **description for functions** are well written.

文本

描述已自动生成

3.2. pointer check

For every given pointer of Matrix, function will check if it exists which make functions **rubust**.

**However**, it indeed consume more time each time the function are invoked which most of time are **even duplicated**. So **it would be better for user to avoid the unexisted pointer by themselves**.

3.3. more useful functions

**transposeMatrix(), standardMatrix(), gussianEliminationMatrix(), rankOfMatrix(), attachMatrix(), inverseMatrix()** are useful and necessary tools for practical using to solve Matrix problem. So they are added specially.