project 3: a library for matrix operations in C

12010508华羽霄

整体分为两个文件：main.c和function.c，因为层次较为简单，没有用CMake。

requirements

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

```
struct matrix {
int total_row;
int total_col;
int total_element;
float data[];
};
```

代码分析：

为了方便查看结果，我创建了print方法，以矩阵的形式打印矩阵中的数据。

```
void printMatrix(struct matrix* this_matrix){
    const int row=this_matrix->total_row;
    const int col=this_matrix->total_col;
    const int total=row*col;

    printf("print start-------\n");
    int location=0;
    for(int r=0;r<row;r++){
        for(int c=0;c<col;c++) {
            printf("%d:%f,",location,this_matrix->data[location]);
            location++;
        }
        printf("\n");
    }
    printf("print end---------\n");
}
```

2. create a matrix, createMatrix ().

```
void createMatrix(struct matrix* this_matrix,const int this_total_row,const int this_total_col,float* p_this_test_data,int this_data_l
    this_matrix->total_row=this_total_row;
    this_matrix->total_col=this_total_col;
    const int this_total_element=this_total_col*this_total_row;
    this_matrix->total_element=this_total_element;

    float test_data[this_total_element];

    // error:
    // int this_length=sizeof(this_data)/sizeof(this_data[0]);
    // printf("sizeof:%d\n",sizeof(this_data));
    // printf("sizeof0:%d\n",sizeof(this_data[0]));

    if(this_total_element<=this_data_length){
        // printf("this_total_element<=this_data_length\n");
        for(int i=0;i<this_total_element;i++){
            // error:
            // this_matrix->data[i]=test_data[i];
            this_matrix->data[i]=*p_this_test_data;
            p_this_test_data++;
        }

    }else{
        // printf("this_total_element>this_data_length\n");
        for(int i=0;i<this_data_length;i++){
            this_matrix->data[i]=*p_this_test_data;
            p_this_test_data++;
        }
        for(int i=this_data_length;i<this_total_element;i++){
            this_matrix->data[i]=0;
        }
    }
```

```
    p_this_test_data-=this_total_element;
}
```

代码分析：

使用者并不一定会输入合法的数据，如data长度比实际需求的多或者少，对此我们有不同的create方式：

如果data等于total，正常输入输出：

```
input data:
1.000000,2.000000,3.000000,4.000000,5.000000,6.000000,
this_total_element<=this_data_length
print start-------
0:1.000000,1:2.000000,2:3.000000,
3:4.000000,4:5.000000,5:6.000000,
print end---------
```

如果data大于total，我们舍去多余的部分：

```
input data:
1.000000,2.000000,3.000000,4.000000,5.000000,6.000000,7.000000,
this_total_element<=this_data_length
print start-------
0:1.000000,1:2.000000,2:3.000000,
3:4.000000,4:5.000000,5:6.000000,
print end---------
```

如果data小于total，我们将缺失的部分补零：

```
input data:
1.000000,2.000000,3.000000,4.000000,5.000000,
this_total_element>this_data_length
print start-------
0:1.000000,1:2.000000,2:3.000000,
3:4.000000,4:5.000000,5:0.000000,
print end---------
```

问题分析：

在一开始我想要获取一个数组的长度，我想到了sizeof/sizeof[0]的方法，但实际操作后发现结果完全错误，经过查询了解到，当一个数组是以指针形式存在时，上述提及的方法将会返回指针长度。

```
// error:
    int this_length=sizeof(test_data)/sizeof(test_data[0]);
    printf("sizeof:%d\n",sizeof(test_data));
    printf("sizeof0:%d\n",sizeof(test_data[0]));
```

因此我改为在调用create方法之前，就先计算好data数组的长度，这时data还是array类型的数据而非指针。

```
//correct:
struct matrix test_matrix;
    // test_matrix = (struct matrix*)malloc(sizeof(struct matrix));
```

```
    int test_data_length=sizeof(test_data)/sizeof(test_data[0]);
    createMatrix(&test_matrix,2,3,&test_data[0],test_data_length);
```

3. copy a matrix (copy the data from a matrix to another), copyMatrix().

```
void copyMatrix(struct matrix* copy_matrix,struct matrix* paste_matrix){
    const int shared_row=copy_matrix->total_row;
    const int shared_col=copy_matrix->total_col;
    const int total=shared_row*shared_col;

    float* p_paste_data=copy_matrix->data;

    for(int i=0;i<total;i++){
        paste_matrix->data[i]=*p_paste_data;
        p_paste_data++;
    }
    p_paste_data-=total;
}
```

代码分析：

考虑到用户并不一定会将两个行列大小相同的矩阵进行拷贝操作，我们以被拷贝矩阵作为参考，对复制矩阵进行修剪，被拷贝矩阵超出的部分直接舍去，没有涉及的部分补零。

问题分析：

```
//error:
    float empty[];
```

此处发生编译错误“variable-sized object may not be initialized”，经检查发现，动态长度数组在初始化时，必须对其中元素进行赋值。

```
//correct:
    float empty[]={0};
```

4. add two matrices, addMatrix().

```
struct matrix* addMatrix(struct matrix* matrix1,struct matrix* matrix2){
    const int shared_row=matrix1->total_row > matrix2->total_row ? matrix1->total_row : matrix2->total_row;
    const int shared_col=matrix1->total_col > matrix2->total_col ? matrix1->total_col : matrix2->total_col;
    const int total=shared_row*shared_col;

    struct matrix* expand_matrix1;
    struct matrix* expand_matrix2;
    struct matrix* result_matrix;

    expand_matrix1=expandMatrix(expand_matrix1,shared_row,shared_col);
    expand_matrix2=expandMatrix(expand_matrix2,shared_row,shared_col);

    int location=0;
    for(int r=0;r<shared_row;r++){
        for(int c=0;c<shared_col;c++) {
            result_matrix->data[location]=expand_matrix1->data[location]+expand_matrix2->data[location];
            location++;
        }
    }
    return result_matrix;
}
```

代码分析：

在进行矩阵加法运算之前，我们首先要判断两个矩阵大小是否相同，对此我认为应该将两个矩阵扩展成一样大小，不同则取大，为此我创建了expand方法，将一个矩阵扩展成更大矩阵，多出来的部分补零。

```
struct matrix* expandMatrix(struct matrix* this_matrix,const int expand_row,const int expand_col){
    const int origin_row=this_matrix->total_row;
```

```
    const int origin_col=this_matrix->total_col;
    const int total=expand_row*expand_col;

    float empty[total];
    for(int i=0;i<total;i++) empty[i]=0;
    struct matrix* result_matrix;
    result_matrix->total_row=expand_row;
    result_matrix->total_col=expand_col;
    result_matrix->total_element=total;
    // result_matrix.data=empty;

    createMatrix(result_matrix,expand_row,expand_col,&empty[0],sizeof(empty)/sizeof(empty[0]));
    float* p_data=&this_matrix->data[0];

    int location_expand=0;
    for(int r=0;r<origin_row;r++){
        for(int c=0;c<origin_col;c++) {
            result_matrix->data[location_expand]=*p_data;
            location_expand++;
            p_data++;
        }
        for(int c=origin_col;c<expand_col;c++) {
            result_matrix->data[location_expand]=0;
            location_expand++;
        }
    }
    for(int r=origin_row;r<expand_row;r++){
        for(int c=0;c<expand_col;c++) {
            result_matrix->data[location_expand]=0;
            location_expand++;
        }
    }
    p_data=0;
    location_expand=0;
    return result_matrix;
}
```

5. subtraction of two matrices, subtractMatrix()

```
struct matrix* subtractMatrix(struct matrix* matrix1,struct matrix* matrix2){
    const int shared_row=matrix1->total_row > matrix2->total_row ? matrix1->total_row : matrix2->total_row;
    const int shared_col=matrix1->total_col > matrix2->total_col ? matrix1->total_col : matrix2->total_col;
    const int total=shared_row*shared_col;

    struct matrix* expand_matrix1;
    struct matrix* expand_matrix2;
    struct matrix* result_matrix;

    expand_matrix1=expandMatrix(expand_matrix1,shared_row,shared_col);
    expand_matrix2=expandMatrix(expand_matrix2,shared_row,shared_col);

    int location=0;
    for(int r=0;r<shared_row;r++){
        for(int c=0;c<shared_col;c++) {
            result_matrix->data[location]=expand_matrix1->data[location]-expand_matrix2->data[location];
            location++;
        }
    }
    return result_matrix;
}
```

代码分析：

该方法与add方法类似，唯一不同点在于加号改成减号。

6. add a scalar to a matrix.

代码分析：

这里我认为应该将向量复制扩展成一个矩阵，大到能够在行或者列方向上覆盖原矩阵，然后再进行add操作。

7. subtract a scalar from a matrix.

代码分析：

该方法与add方法类似，唯一不同点在于加号改成减号。

8. multiply a matrix with a scalar.

9. multiply two matrices.

10. find the minimal and maximal values of a matrix.

```
float findmax(struct matrix* this_matrix){
    float result=FLT_MIN;
    int location=0;
    for(int r=0;r<this_matrix->total_row;r++){
        for(int c=0;c<this_matrix->total_col;c++) {
            result=this_matrix->data[location] > result ? this_matrix->data[location] : result;
            location++;
        }
    }
}
```

```
float findmin(struct matrix* this_matrix){
    float result=FLT_MAX;
    int location=0;
    for(int r=0;r<this_matrix->total_row;r++){
        for(int c=0;c<this_matrix->total_col;c++) {
            result=this_matrix->data[location] < result ? this_matrix->data[location] : result;
            location++;
        }
    }
}
```

代码分析：

需要注意的是，寻找最大值的时候，result的初始值需置为FLT_MIN。同理，寻找最小值的时候，result的初始值需置为FLT_MAX。

```
print start-------
0:1.000000,1:2.000000,2:3.000000,
3:4.000000,4:5.000000,5:6.000000,
print end---------
max value:6.000000
min value:1.000000
```

作业总结：

经过这次作业，我对结构体、指针、内存的理解更加深刻，能够灵活运用指针对结构体中的数据进行各种操作。但奈何个人能力和时间精力实在有限，只完成了以上部分，有些遗憾。