实验6 数组的应用

一、【实验目的】  
1、掌握数组的抽象数据类型

2、掌握动态数组的设计方法

3、理解动、静态数组的对比

4、掌握特殊矩阵的压缩存储及运算

二、【实验内容】

1、设矩阵A、矩阵B为n阶对称矩阵，矩阵元素为整数类型，要求:

（1）若A、B采用压缩存储方式，请编写算法实现矩阵乘法运算C=A\*B，要求C用动态二维数组存储。

（2）写一压缩矩阵的元素输出函数，要求按矩阵方式输出元素。

（3）编写一个主程序调用以上两个函数进行测试，输出矩阵A,B,C。

**三、【**实验源代码**】**

#include <stdio.h>

#include <malloc.h>

int \*\*Get2DArray(int row, int col);

void Destroy2DArray(int \*\*a, int row);

void multiplyArray(int \*arrayA, int \*arrayB, int \*\*arrayC, int n);

void printCompressedArray\_to2D(int \*array, int n);

int main(void)

{

    /\*定义三阶对称矩阵\*/

    int n = 3;

    /\*对称矩阵A的压缩存储 \*/

    int ArraycompressedA[] = {1, 2, 3, 4, 5, 6};

    /\*对称矩阵B的压缩存储 \*/

    int ArraycompressedB[] = {7, 8, 9, 10, 11, 12};

    int\*\* C = Get2DArray(n, n);

    /\*矩阵C = 矩阵A \* 矩阵B\*/

    multiplyArray(ArraycompressedA, ArraycompressedB, C, n);

    /\*将压缩矩阵A按矩阵方式输出\*/

    printf("Matrix A:\n");

    printCompressedArray\_to2D(ArraycompressedA, n);

    /\*将压缩矩阵B按矩阵方式输出\*/

    printf("Matrix B:\n");

    printCompressedArray\_to2D(ArraycompressedB, n);

    /\*输出矩阵C\*/

    printf("Matrix C (A \* B):\n");

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n; j++)

        {

            printf("%5d", C[i][j]);

        }

        printf("\n");

    }

    /\*释放内存\*/

    Destroy2DArray(C, n);

}

/\*创建row行col列的二维动态数组，函数返回其首地址\*/

int \*\*Get2DArray(int row, int col)

{

    int \*\*a;

    a = (int \*\*)calloc(row, sizeof(int \*));

    for (int i = 0; i < row; i++)

    {

        a[i] = (int \*)calloc(col, sizeof(int));

    }

    return a;

}

/\*销毁二维数组\*/

void Destroy2DArray(int \*\*a, int row)

{

    for (int i = 0; i < row; i++)

    {

        free(a[i]);

    }

    free(a);

}

void multiplyArray(int \*arrayA, int \*arrayB, int \*\*arrayC, int n)

{

    /\*temp用于临时存储压缩矩阵转为n阶矩阵的值\*/

    int \*\*tempA =Get2DArray(n, n);

    int \*\*tempB =Get2DArray(n, n);

    /\*k为一维数组(存储压缩矩阵)的下标序号\*/

    int k;

    /\*数学中，n阶对称矩阵元素的下标满足条件：1<=i<=n和1<=j<=n\*/

    for (int i = 1, row = 0; i <= n; i++, row++)

    {

        for (int j = 1, col = 0; j <= n; j++, col++)

        {

            if (i >= j)

            {

                k = i \* (i - 1) / 2 + j - 1;

            }

            else

            {

                k = j \* (j - 1) / 2 + i - 1;

            }

            tempA[row][col] = arrayA[k];

            tempB[row][col] = arrayB[k];

        }

    }

    /\*将矩阵A乘矩阵B，结果存储在矩阵C\*/

    for (int row = 0; row < n; row++)

    {

        for (int col = 0; col < n; col++)

        {

            for (int k = 0; k < n; k++)

            {

                /\*矩阵C的第i行第j列的元素是A的第i行与B的第j列对应元素乘积的和\*/

                arrayC[row][col] +=  tempA[row][k] \* tempB[k][col];

            }

        }

    }

    /\*释放内存\*/

    Destroy2DArray(tempA, n);

    Destroy2DArray(tempB, n);

}

/\*将一个压缩矩阵以矩阵方式输出元素\*/

void printCompressedArray\_to2D(int \*array, int n)

{

    /\*k为一维数组(存储压缩矩阵)的下标序号\*/

    int k;

    for (int i = 1; i <= n; i++)

    {

        for (int j = 1; j <= n; j++)

        {

            if (i >= j)

            {

                k = i \* (i - 1) / 2 + j - 1;

            }

            else

            {

                k = j \* (j - 1) / 2 + i - 1;

            }

            printf("%5d", array[k]);

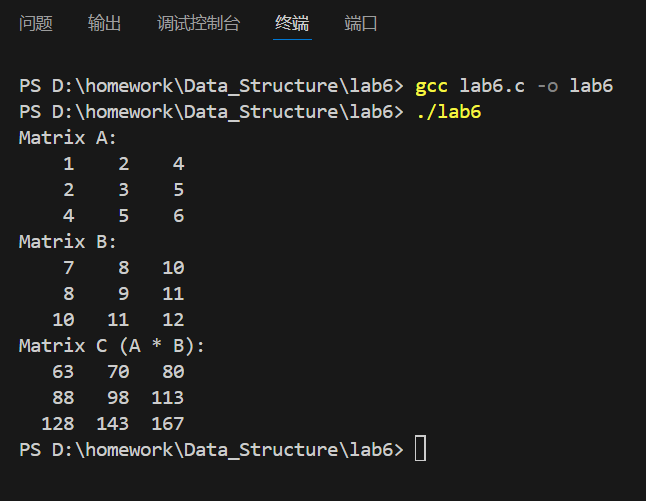
        }

        printf("\n");

    }

}

**四、实验结果**



**五、实验心得**

答：本次实验中，我掌握了数组的抽象数据类型，掌握动态数组的设计方法，理解动、静态数组的对比，掌握特殊矩阵的压缩存储及运算。深刻理解了对称矩阵的压缩存储方法，并掌握了如何在C语言中使用动态数组来存储和操作这些矩阵。通过实现矩阵的乘法运算，我进一步加深了对数组和指针操作的理解。此外，编写和调试代码的过程中，我也遇到了一些挑战，比如如何正确计算矩阵乘法中的索引和如何管理动态内存，但也一一克服。