第四章 分治法求矩阵乘积

```
#include <iostream>
using namespace std;
//STRASSEN矩阵乘法算法
/*
* 矩阵的加法运算
*/
void Add(int** matrixA, int** matrixB, int** matrixResult, int length)
{
  for (int i = 0; i < length; i++) {
    for (int j = 0; j < length; j++) {
      matrixResult[i][j] = matrixA[i][j] + matrixB[i][j];
    }
  }
}
/*
* 矩阵乘法
*/
void Mul(int** matrixA, int** matrixB, int** matrixResult){
  for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 2; ++j) {
      matrixResult[i][j] = 0;
      for (int k = 0; k < 2; ++k) {
        matrixResult[i][j] += matrixA[i][k] * matrixB[k][j];
      }
    }
  }
}
void Strassen(int** matrixA, int** matrixB, int** matrixResult, int length)
{
  int halfLength = length / 2;
  int **a11 = new int*[halfLength];
  int **a12 = new int*[halfLength];
  int **a21 = new int*[halfLength];
  int **a22 = new int*[halfLength];

  int **b11 = new int*[halfLength];
  int **b12 = new int*[halfLength];
int **b21 = new int*[halfLength];
int **b22 = new int*[halfLength];

int **matrixResult11 = new int*[halfLength];
int **matrixResult12 = new int*[halfLength];
int **matrixResult21 = new int*[halfLength];
int **matrixResult22 = new int*[halfLength];

int **temp = new int*[halfLength];
int **temp1 = new int*[halfLength];
if (halfLength == 1){
  Mul(matrixA, matrixB, matrixResult);
}
else{
  //首先将矩阵A，B 分为4块
  for (int i = 0; i < halfLength; i++) {
    a11[i] = new int[halfLength];
    a12[i] = new int[halfLength];
    a21[i] = new int[halfLength];
    a22[i] = new int[halfLength];

    b11[i] = new int[halfLength];
    b12[i] = new int[halfLength];
    b21[i] = new int[halfLength];
    b22[i] = new int[halfLength];

    matrixResult11[i] = new int[halfLength];
    matrixResult12[i] = new int[halfLength];
    matrixResult21[i] = new int[halfLength];
    matrixResult22[i] = new int[halfLength];

    temp[i] = new int[halfLength];
    temp1[i] = new int[halfLength];
    for (int j = 0; j < halfLength; j++) {
      a11[i][j] = matrixA[i][j];
      a12[i][j] = matrixA[i][j + halfLength];
      a21[i][j] = matrixA[i + halfLength][j];
      a22[i][j] = matrixA[i + halfLength][j + halfLength];
      b11[i][j] = matrixB[i][j];
      b12[i][j] = matrixB[i][j + halfLength];
```

```
        b21[i][j] = matrixB[i + halfLength][j];
        b22[i][j] = matrixB[i + halfLength][j + halfLength];
    }
}
Strassen(a11, b11, temp, halfLength);
Strassen(a12, b21, temp1, halfLength);
Add(temp, temp1, matrixResult11, halfLength);
Strassen(a11, b12, temp, halfLength);
Strassen(a12, b22, temp1, halfLength);
Add(temp, temp1, matrixResult12, halfLength);
Strassen(a21, b11, temp, halfLength);
Strassen(a22, b21, temp1, halfLength);
Add(temp, temp1, matrixResult21, halfLength);
Strassen(a21, b12, temp, halfLength);
Strassen(a22, b22, temp1, halfLength);
Add(temp, temp1, matrixResult22, halfLength);
//结果送回matrixResult中
for (int i = 0; i < halfLength; i++) {
    for (int j = 0; j < halfLength; j++) {
        matrixResult[i][j] = matrixResult11[i][j];
        matrixResult[i][j + halfLength] = matrixResult12[i][j];
        matrixResult[i + halfLength][j] = matrixResult21[i][j];
        matrixResult[i + halfLength][j + halfLength] = matrixResult22[i][j];
    }
    delete(a11[i]);
    delete(a12[i]);
    delete(a21[i]);
    delete(a22[i]);

    delete(b11[i]);
    delete(b12[i]);
    delete(b21[i]);
    delete(b22[i]);



    delete(matrixResult11[i]);
    delete(matrixResult12[i]);
    delete(matrixResult21[i]);
    delete(matrixResult22[i]);
```

```
            delete(temp[i]);
            delete(temp1[i]);
        }
        delete(a11);
        delete(a12);
        delete(a21);
        delete(a22);

        delete(b11);
        delete(b12);
        delete(b21);
        delete(b22);

        delete(matrixResult11);
        delete(matrixResult12);
        delete(matrixResult21);
        delete(matrixResult22);

        delete(temp);
        delete(temp1);
    }
}
int main()
{
    int n;
    cout << "输入矩阵行列数(2的幂指数倍)：" << endl;
    cin >> n;
    int **a = new int*[n];
    int **b = new int*[n];
    int **c = new int*[n];
    for (int i = 0; i < n; i++)
    {
        a[i] = new int[n];
        b[i] = new int[n];
        c[i] = new int[n];
    }
    cout << "输入第一个矩阵的数字：" << endl;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            cin >> a[i][j];
        }
    }
    cout << "输入第二个矩阵的数字：" << endl;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            cin >> b[i][j];
        }
    }
    Strassen(a, b, c, n);
    cout << "结果矩阵为：" << endl;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cout << c[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```