**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационной безопасности**

отчет

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Вычисление произведения матриц по строкам

|  |  |  |
| --- | --- | --- |
| Студент гр. 2363 |  | Морозов А.И. |
| Преподаватель |  | Абросимов И.К. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на Лабораторную работу**

|  |  |  |
| --- | --- | --- |
| Студент Морозов А.И. | | |
| Группа 2363 | | |
| Тема работы: Вычисление произведения матриц по строкам | | |
| Исходные данные:  Для выполнения лабораторных работ по оцениванию сложности заданного алгоритма используется одна из структур данных, основанных на одномерных или двухмерных массивах – длинные целые числа, многочлены одной переменной с целыми коэффициентами, матрицы.  Матрицы представляют собой двухмерный массив, над которым определены операции сложения, вычитания, умножения на число, выделения подматрицы, а также преобразования строки в матрицу и матрицы в строку. | | |
| Содержание пояснительной записки:  Титульный лист, задание на лабораторную работу, теоретическое введение, решение поставленной задачи, вывод, приложение с полным исходным кодом разработанной программы. | | |
| Предполагаемый объем пояснительной записки:  Не менее страниц. | | |
| Дата выдачи задания: | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Морозов А.И. |
| Преподаватель |  | Абросимов И.К. |

**Цель работы.**

Экспериментально оценить временную сложность предложенного алгоритма по его реализации на указанном языке программирования.

**Основные теоретические положения.**

Определение: Матрица () – это элемент множества всех матриц (), где:

;

Матрица записывается в виде прямоугольной таблицы, состоящей из строк и столбцов.

Пример:

где

Где означает, что из матрицы был взят элемент, который находится в -ой строке и -ом столбце.

Определение: Подматрица матрицы – матрица, которая образована элементами, стоящими на пересечении выбранных строк и столбцов.

Операции, которые определены над матрицами:

*;*

Сложение матриц:

*;*

Вычитание матриц:

;

Умножение матрицы на число:

;

;

Описание используемой структуры данных – TMatrix:

Поля:

Поле Mat

Поле Mat является приватным и хранит указатель на динамический двумерный массив (матрицу).

Исходный код поля Mat находится в файле matrix.h

Объявление поля:

int\*\* Mat;

Тип поля: int\*\*

Поле size\_m

Поле size\_m является приватным и хранит значение, равное количеству строк или столбцов в квадратной матрице.

Исходный код поля size\_m находится в файле matrix.h

Объявление поля:

int size\_m;

Тип поля: int

Методы:

Метод ShowResult:

Метод ShowResult выводит в консоль матрицу, хранящуюся в поле Mat.

Исходный код метода ShowResult находится в файле matrix.cpp.

Объявление метода:

void ShowResult();

Тип метода: void.

Аргументы метода: нет

Возвращаемое методом значение: нет

Метод selsubmatrix:

Метод selsubmatrix выделяет подматрицу в заданной матрице.

Исходный код метода selsubmatrix находится в файле matrix.cpp.

Объявление метода:

int selsubmatrix(int,int);

Тип метода: int.

Аргументы метода:

* strdel – количество строк в подматрице;
* coldel – количество столбцов в подматрице.

Возвращаемое методом значение:

* 1 – ошибка;
* 0 – нет ошибок.

Метод wrmattofile:

Метод wrmattofile записывает в файл заданную матрицу.

Исходный код метода wrmattofile находится в файле matrix.cpp.

Объявление метода:

void wrmattofile();

Тип метода: void.

Аргументы метода: нет.

Возвращаемое методом значение: нет.

Метод rematfromfile:

Метод rematfromfile чтения матрицы из файла.

Исходный код метода rematfromfile находится в файле matrix.cpp.

Объявление метода:

void rematfromfile();

Тип метода: void.

Аргументы метода: нет.

Возвращаемое методом значение: нет.

Метод deleteMatrix:

Метод deleteMatrix освобождает память, которую занимает матрица.

Исходный код метода deleteMatrix находится в файле matrix.cpp.

Объявление метода:

void deleteMatrix();

Тип метода: void.

Аргументы метода: нет

Возвращаемое методом значение: нет

Перегруженные стандартные операторы:

Оператор operator \* (int):

operator \* (int) – перегруженный стандартный оператор “\*” для корректной работы умножения матрицы на число.

Исходный код оператора operator \* (int) находится в файле matrix.cpp.

Объявление перегрузки оператора:

TMatrix operator \* (int);

Тип оператора: TMatrix.

Возвращаемое значение: структура данных TMatrix.

Оператор operator \* (TMatrix):

operator \* (TMatrix) – перегруженный стандартный оператор “\*” для корректной работы умножения матрицы на матрицу.

Исходный код оператора operator \* (TMatrix) находится в файле matrix.cpp.

Объявление перегрузки оператора:

TMatrix operator \* (TMatrix);

Тип оператора: TMatrix.

Возвращаемое значение: структура данных TMatrix.

Оператор operator = (TMatrix):

operator = (TMatrix) – перегруженный стандартный оператор “=” для корректной работы присваивания структуры данных.

Исходный код оператора operator = (TMatrix) находится в файле matrix.cpp.

Объявление перегрузки оператора:

TMatrix operator = (TMatrix);

Тип оператора: TMatrix.

Возвращаемое значение: структура данных TMatrix.

Оператор operator - (TMatrix):

operator - (TMatrix) – перегруженный стандартный оператор “-” для корректной работы вычитания матриц.

Исходный код оператора operator - (TMatrix) находится в файле matrix.cpp.

Объявление перегрузки оператора:

TMatrix operator - (TMatrix);

Тип оператора: TMatrix.

Возвращаемое значение: структура данных TMatrix.

Тип оператора: TMatrix.

Возвращаемое значение: структура данных TMatrix.

Оператор operator + (TMatrix):

operator + (TMatrix) – перегруженный стандартный оператор “+” для корректной работы сложения матриц.

Исходный код оператора operator + (TMatrix) находится в файле matrix.cpp.

Объявление перегрузки оператора:

TMatrix operator + (TMatrix);

Тип оператора: TMatrix.

Возвращаемое значение: структура данных TMatrix.

**Экспериментальные результаты.**

Вычисление произведения матриц по строкам состоит в использовании, соответственно, формулы:

;

Где записи и означают, что взята i-ая строка и k-ая строка.

Пример нахождения произведения матриц по строкам:

;

;

;

;

;

;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | 1 | 2 | 3 | 4 | 5 |
| (строки) | 64 | 128 | 256 | 512 | 1024 |
| (мc) | 9 | 72 | 563 | 4451 | 36166 |
|  | 8 | 7,819 | 7,906 | 8,125 |  |

Согласно предположению, данная функция относится к классу степенных функций, т.е.

Таким образом,

;

Коэффициент

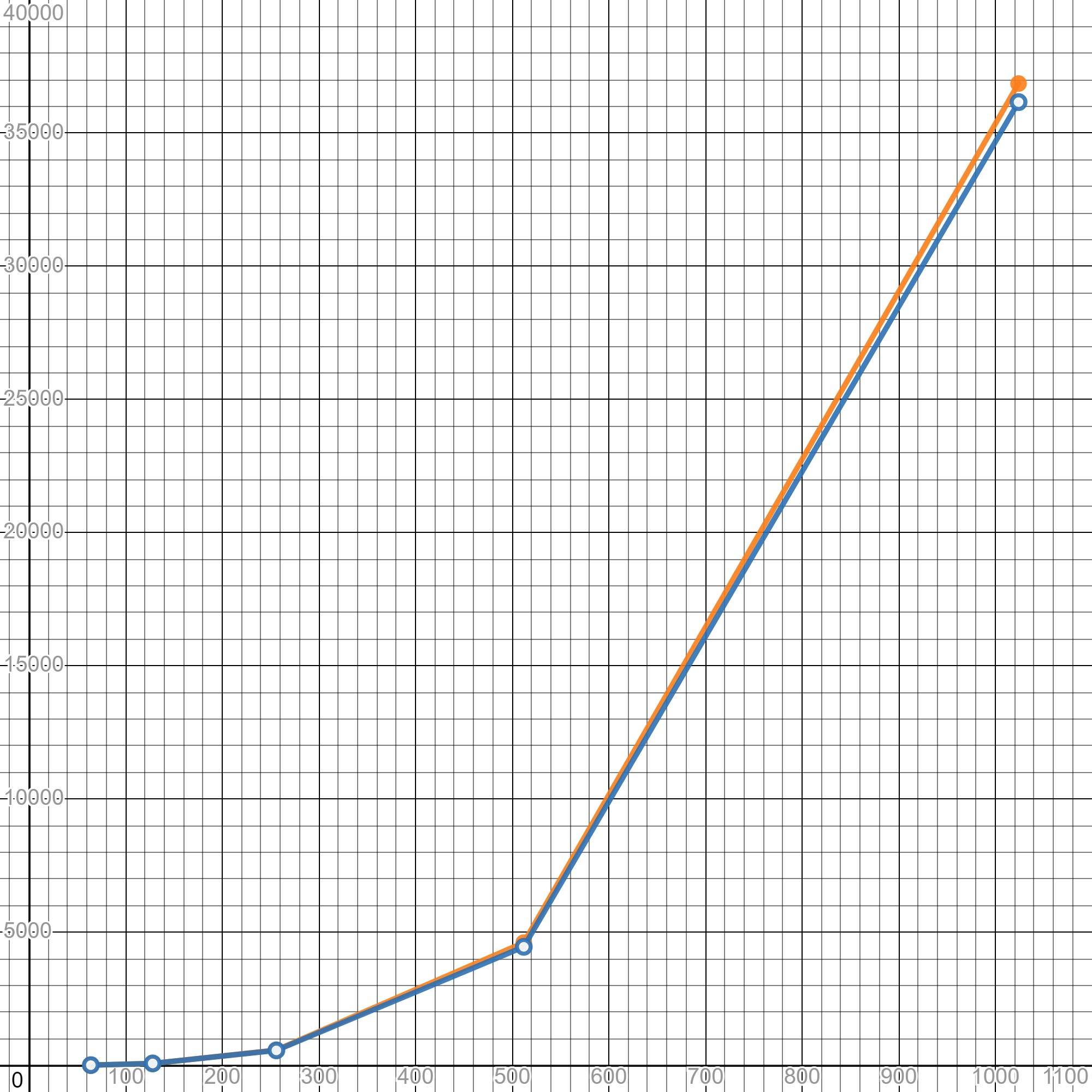


Рисунок 1 – График сложности алгоритма, где оранжевый – теоретический и синий – экспериментальный

**Выводы.**

В результате выполнения лабораторной работы была найдена сложность алгоритма перемножения матриц по строкам т.е. время выполнения алгоритма возрастает кубически пропорционально размеру матрицы n. Это связано с тем, что для каждого элемента результирующей матрицы требуется выполнить n умножений и n суммирований.

ПрИложение

matrix.h:

#ifndef TMATRIX\_H

#define TMATRIX\_H

class TMatrix

{

public:

TMatrix(const TMatrix &);

TMatrix(int, int);

TMatrix operator \*(const int size\_m);

TMatrix operator \*(const TMatrix &);

TMatrix operator =(const TMatrix &);

TMatrix operator -(const TMatrix &);

TMatrix operator +(const TMatrix &);

int selsubmatrix(int,int);

void wrmattofile();

void rematfromfile();

void ShowResult();

void deleteMatrix();

void initMatrix();

void SetZero();

long long\*\* Mat;

int col;

int str;

};

#endif // TMATRIX\_H

main.cpp:

#include "matrix.h"

#include <iostream>

#include <fstream>

#include <cmath>

using namespace std;

int main()

{

int ch;

cout<<"Enter operation:" << endl;

cout<<"1 - Multiplie of matrix(for lab)" << endl;

cout<<"2 - Multiplie of matrix" << endl;

cout<<"3 - Multiplie in value" << endl;

cout<<"4 - Summation of matrix" << endl;

cout<<"5 - Subtraction of matrix" << endl;

cout<<"6 - Select a submatrix in the matrix" << endl;

cout<<"7 - Write the matrix to a file" << endl;

cout<<"8 - Read an array from a file" << endl;

cout<<"0 - Exit"<<endl<<"> ";

cin >> ch;

switch(ch)

{

case 1:

{

for(int i=6; i<=10;i++)

{

unsigned int \_start = 0;

unsigned int \_end = 0;

double time = 0;

TMatrix First\_M(pow(2,i),pow(2,i));

TMatrix Second\_M(pow(2,i),pow(2,i));

for(int j=0;j<10;j++)

{

\_start = clock();

(First\_M \* Second\_M);

\_end = clock();

time += (double) (\_end - \_start)/1000;

}

int n = pow(2,i);

printf("%1d) str = %4d; time = %f;\n",i-5,n,time);

}

break;

}

case 2:

{

int str1,col1,col2;

cout << "Enter size of matrix:\n\n"

"number of rows of the first matrix: ";

cin >> str1;

cout << "number of columns of the first matrix"

" (number of rows of the second): ";

cin >> col1;

cout << "number of columns of the second matrix: ";

cin >> col2;

TMatrix First\_M(str1,col1);

First\_M.initMatrix();

TMatrix Second\_M(col1,col2);

Second\_M.initMatrix();

First\_M.ShowResult();

Second\_M.ShowResult();

(First\_M \* Second\_M).ShowResult();

break;

}

case 3:

{

int str, col, val;

cout << "Enter size of matrix:\n\n"

"number of rows of the matrix: ";

cin >> str;

cout << "number of columns of the matrix: ";

cin >> col;

cout << "Enter value:";

cin >> val;

TMatrix First\_M(str,col);

First\_M.initMatrix();

(First\_M \* val).ShowResult();

break;

}

case 4:

{

int str,col;

cout << "Enter size of matrix:\n\n"

"number of rows of the matrix: ";

cin >> str;

cout << "number of columns of the matrix: ";

cin >> col;

TMatrix First\_M(str,col);

First\_M.initMatrix();

TMatrix Second\_M(str,col);

Second\_M.initMatrix();

(First\_M + Second\_M).ShowResult();

break;

}

case 5:

{

int str,col;

cout << "Enter size of matrix:\n\n"

"number of rows of the matrix: ";

cin >> str;

cout << "number of columns of the matrix: ";

cin >> col;

TMatrix First\_M(str,col);

First\_M.initMatrix();

TMatrix Second\_M(str,col);

Second\_M.initMatrix();

(First\_M - Second\_M).ShowResult();

break;

}

case 6:

{

int str,col,strdel = 1,coldel = 1;

cout << "Enter size of matrix:\n\n"

"number of rows of the matrix: ";

cin >> str;

cout << "number of columns of the matrix: ";

cin >> col;

TMatrix First\_M(str,col);

First\_M.initMatrix();

First\_M.ShowResult();

cout << "number of rows in the submatrix: ";

cin >> strdel;

if(strdel <= 0) cout << "Error";

cout << "number of columns in the submatrix: ";

cin >> coldel;

if(coldel <= 0) cout << "Error";

if(First\_M.selsubmatrix(strdel,coldel) != 0) cout << "Error";

}

case 7:

{

int str,col;

cout << "Enter size of matrix:\n\n"

"number of rows of the matrix: ";

cin >> str;

cout << "number of columns of the matrix: ";

cin >> col;

TMatrix First\_M(str,col);

First\_M.initMatrix();

First\_M.wrmattofile();

}

case 8:

{

int str,col;

ifstream inputFile("test.txt");

inputFile >> str;

inputFile >> col;

inputFile.close();

TMatrix First\_M(str,col);

First\_M.rematfromfile();

First\_M.ShowResult();

}

case 0:

{

return 0;

}

}

}

matrix.cpp:

#include <iostream>

#include <fstream>

#include <stdlib.h>

#include <ctime>

#include "matrix.h"

using namespace std;

TMatrix::TMatrix(int a,int b)

{

srand(time(0));

str = a;

col = b;

Mat = new long long\*[a];

for(int i=0;i<a;i++) Mat[i] = new long long[b];

for(int i=0;i<a;i++)

{

for(int j=0;j<b;j++)

{

Mat[i][j] = 16+rand()%16;

}

}

}

TMatrix::TMatrix(const TMatrix &a)

{

str = a.str;

col = a.col;

Mat = new long long\*[str];

for(int i=0;i<col;i++) Mat[i] = new long long[col];

for(int i=0;i<str;i++)

{

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

{

Mat[i][j] = a.Mat[i][j];

}

}

}

void TMatrix::initMatrix()

{

for(int i = 0; i < str; i++)

{

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

{

cout<<"mtrx["<<i<<"]["<<j<<"] = ";

cin >> Mat[i][j];

}

}

}

void TMatrix::ShowResult()

{

cout<<"\n";

for(int i=0;i<str;i++){

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

cout<<Mat[i][j]<<" ";

cout<<"\n";

}

cout<<"\n";

}

TMatrix TMatrix::operator \* (const int number)

{

TMatrix mtrx1(str,col);

for(int i = 0; i < str; i++)

{

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

{

mtrx1.Mat[i][j] = Mat[i][j] \* number;

}

}

return mtrx1;

}

TMatrix TMatrix::operator \* (const TMatrix &a)

{

TMatrix result(str,a.col);

long long \*row;

for(int i = 0; i < str; i++)

{

row = new long long[a.col];

for(int z=0; z<a.col;z++) row[z] = 0;

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

{

for (int s = 0; s < a.col; s++)

{

row[s] += this->Mat[i][k] \* a.Mat[k][s];

}

}

result.Mat[i] = row;

}

return result;

}

TMatrix TMatrix::operator =(const TMatrix &a)

{

TMatrix mtrx3(str,col);

for(int i = 0; i < str; i++)

{

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

{

mtrx3.Mat[i][j] = a.Mat[i][j];

}

}

return mtrx3;

}

TMatrix TMatrix::operator + (const TMatrix &a)

{

TMatrix mtrx4(str,col);

for(int i = 0; i < str; i++)

{

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

{

mtrx4.Mat[i][j] = (Mat[i][j] + a.Mat[i][j]);

}

}

return mtrx4;

}

TMatrix TMatrix::operator - (const TMatrix &a)

{

TMatrix mtrx5(str,col);

for(int i = 0; i < str; i++)

{

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

{

mtrx5.Mat[i][j] = (Mat[i][j] - a.Mat[i][j]);

}

}

return mtrx5;

}

int TMatrix::selsubmatrix(int strdel,int coldel)

{

if(str < strdel || col < coldel) return 1;

TMatrix mtrx6(strdel,coldel);

int strmas[strdel];

int colmas[coldel];

cout << "write the row numbers in ascending order: ";

for(int i=0;i<strdel;i++)

{

cin >> strmas[i];

if(str < strmas[i])

{

mtrx6.deleteMatrix();

return 1;

}

}

cout << "write the column numbers in ascending order: ";

for(int i=0;i<coldel;i++)

{

cin >> colmas[i];

if(col < colmas[i])

{

mtrx6.deleteMatrix();

return 1;

}

}

for(int i=0;i<strdel;i++)

{

for(int j=0;j<coldel;j++){

mtrx6.Mat[i][j] = Mat[strmas[i]-1][colmas[j]-1];

}

}

mtrx6.ShowResult();

return 0;

}

void TMatrix::wrmattofile()

{

ofstream fout;

fout.open("test.txt");

fout << str << " ";

fout << col << endl;

if (fout.is\_open()) {

for (int i = 0; i < str; i++){

for(int j = 0; j < col; j++){

fout << Mat[i][j] << " ";

}

fout << "\n";

}

fout.close();

cout << "\nThe array has been successfully written to the file." <<endl;

}

else {

cout << "Error opening the file." << endl;

}

}

void TMatrix::rematfromfile()

{

ifstream inputFile("test.txt");

inputFile >> str;

inputFile >> col;

for (int i = 0; i < str; i++){

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

{

inputFile >> Mat[i][j];

}

}

inputFile.close();

}

void TMatrix::deleteMatrix()

{

for(int i = 0; i < str; i++)

{

delete[] Mat[i];

}

delete[] Mat;

}

void TMatrix::SetZero()

{

for(int i = 0; i < str; i++)

{

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

{

Mat[i][j] = 0;

}

}

}