**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по лабораторной работе №8

«**Обработка и печать числовой матрицы**»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Выполнил:** | |  | **Принял:** | |
| ФИО: | \_Цыпышев Т. А.\_\_\_\_\_ |  | ФИО: | \_**Самохвалов А. Э.**\_\_\_ |
| Группа: | \_ИУ5-11Б\_\_\_\_\_\_\_\_\_\_ |  | Должность: | \_Преподаватель\_\_\_\_ |
| Дата: | \_21.11.2022\_\_\_\_\_\_\_\_ |  | Дата: | \_21.11.2022\_\_\_\_\_\_\_\_ |
| Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва, 2022 г.

**Постановка задачи**

**Цели работы:**

* создание двухмерных динамических массивов
* обработка матриц
* использование файлов для хранения матриц (на примере приведенной ниже программы сортировки строк матрицы)
* передача двумерных массивов в функцию через параметры
* форматированный вывод матриц на экран
* доступ к элементам матрицы через указатели и с помощью индексов
* освоение технологии структурного программирования

**Задание:**

Создать квадратную матрицу A размером N\*N (где N вводится с клавиатуры), и заполнить её следующими значениями:

* все элементы главной диагонали равны 1
* элементы, лежащие выше главной диагонали, вычисляются по формуле **A i,j = xi / (j!)i**
* элементы, лежащие ниже главной диагонали, по формуле **A i,j = (-x)i / (j!)i**, где **i,j =1,2,…,N**

Для вычисления значений элементов матрицы использовать рекуррентные соотношения**.**

Реализовать алгоритм заполнения матрицы в виде функции.

В зависимости от размера матрицы и ширины поля вывода элемента матрицы, обеспечить удобное для пользователя отображение матрицы на экране. Оформить вывод матрицы размером N\*M на экран в виде функции с целью использования ее в последующих лабораторных работах для распечатки двумерных массивов.

Матрица должна передаваться в разрабатываемые функции через параметры.

Не изменяя кода функции вывода матрицы, распечатать матрицу в «научном» формате и в формате с фиксированной точкой с точностью 8 знаков после запятой.

Распечатать с помощью разработанной функции, используя вспомогательный массив указателей на строки, матрицу размером **B[10][10],** заданную с помощью оператора описания (нединамическую). Значение элементов матрицы **В** определяется соотношением: **B[i][j]=i\*10+j**.

Объясните, как передаются матрицы A и В в функцию вывода матриц на экран.

Вставьте в программу и объясните результаты выполнения следующих операторов для матрицы В[10][10]:

* cout<<B<<" "<<B[0]<<" "<<B[2]<<endl;
* cout<<B[0][0]<<" "<<\*\*B <<" "<<\*B[0]<<endl;
* cout<<\*(\*(B+1))<<" "<<\*B[1]<<endl;
* cout<<\*(B[0]+1)<<" " <<\*(\*B+1)<<endl;
* cout<<B[0][20]<<" "<<\*(B[0]+20)<<" "<<\*B[2]<<endl;

**Разработка алгоритма**

**Описание входных, выходных и вспомогательных данных:**

Входные данные:

* int N – размер матрицы A
* int X – значение x

Выходные данные:

* double\*\* A – квадратная матрица A размером N\*N
* double B[10][10] – квадратная матрица B размером N\*N

Вспомогательные данные:

* int width – ширина консоли

**Список функций:**

* double fact(int n) – считает факториал N
* void Clear(double \*\*A, int N) – очищает память
* double \*\*CreateDinamicMatrix(int N) – создаёт единичную матрицу
* void FillDinamicMatrix(double \*\*A, int N, int x) – заполняет матрицу A по правилам из условия
* void PrintDinamicMatrix(double \*\*A, int N) – выводит динамичный двумерный массив на экран
* void LoggingDinamicMatrix(double \*\*A, int N) – сохраняет в файл двумерный массив в виде матрицы
* void FillStaticMatrix(double ( &B)[10][10] ) – заполняет матрицу B по правилам из условия
* void PrintStaticMatrix(double B[10][10]) – выводит статический двумерный массив на экран
* void LoggingStaticMatrix(double B[10][10] ) – сохраняет в файл двумерный массив в виде матрицы

**Текст программы**

**Main.cpp**

#include <iostream>

#include <fstream>

#include <cmath>

#include <iomanip>

using namespace std;

double fact(int n) {

if (n < 0) return 0;

if (n == 0) return 1;

else return n \* fact(n - 1);

}

void Clear(double \*\*A, int N) {

for (int i = 1; i < N; i++)

delete[] A[i];

delete[] A;

}

double \*\*CreateDinamicMatrix(int N) {

double \*\*A = new double \*[N];

for (int i = 1; i < N; ++i) {

A[i] = new double[N];

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

A[i][j] = 1;

}

}

return A;

}

void FillDinamicMatrix(double \*\*A, int N, int x) {

for (int i = 1; i < N; ++i) {

for (int j = 1; j < N; ++j) {

if (i == j) { A[i][j] = 1; }

else if (i < j) { A[i][j] = pow(x, i) / pow(fact(j), i); }

else if (i > j) { pow(-x, i) / pow(fact(j), i); }

}

}

}

void PrintDinamicMatrix(double \*\*A, int N) {

int width = 100;

for (int i = 1; i < N; i++) {

for (int j = 1; j < N; j++) {

int K = floor(width / N) - 1;

if (A[i][j] > 0.01)

cout << left << fixed << setw(K) << setprecision(K - 3) << A[i][j] << " ";

else

cout << left << fixed << setw(K) << setprecision(K - 7) << scientific << A[i][j] << " ";

}

cout << "\n";

}

}

void LoggingDinamicMatrix(double \*\*A, int N) {

ofstream log\_file("../log.txt", ios\_base::app);

if (!(log\_file)) { log\_file.open("../log.txt"); }

log\_file << "\n\n\n";

int width = 100;

for (int i = 1; i < N; i++) {

for (int j = 1; j < N; j++) {

int K = floor(width / N) - 1;

if (A[i][j] > 0.01)

log\_file << left << fixed << setw(K) << setprecision(K - 3) << A[i][j] << " ";

else

log\_file << left << fixed << setw(K) << setprecision(K - 7) << scientific << A[i][j] << " ";

}

log\_file << "\n";

}

}

void FillStaticMatrix(double ( &B)[10][10]) {

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

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

B[i][j] = 10 \* i + j;

}

void PrintStaticMatrix(double B[10][10]) {

double \*\*tmp = new double \*[10];

for (int i = 0; i < 10; i++) {

tmp[i] = new double[10];

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

tmp[i][j] = B[i][j];

}

}

PrintDinamicMatrix(tmp, 10);

Clear(tmp, 10);

}

void LoggingStaticMatrix(double B[10][10]) {

double \*\*tmp = new double \*[10];

for (int i = 0; i < 10; i++) {

tmp[i] = new double[10];

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

tmp[i][j] = B[i][j];

}

}

LoggingDinamicMatrix(tmp, 10);

Clear(tmp, 10);

}

int main() {

int N, x;

cout << "Enter the value of N and X: ";

cin >> N >> x;

++N;

cout << "\n----- The matrix A -----\n";

double \*\*A = CreateDinamicMatrix(N);

FillDinamicMatrix(A, N, x);

PrintDinamicMatrix(A, N);

LoggingDinamicMatrix(A, N);

Clear(A, N);

cout << "\n----- The matrix B -----\n";

double B[10][10];

FillStaticMatrix(B);

PrintStaticMatrix(B);

LoggingStaticMatrix(B);

cout << "\n----- Additional calculations -----\n";

cout << "(1) - " << B << " " << B[0] << " " << B[2] << endl;

cout << "(2) - " << B[0][0] << " " << \*\*B << " " << \*B[0] << endl;

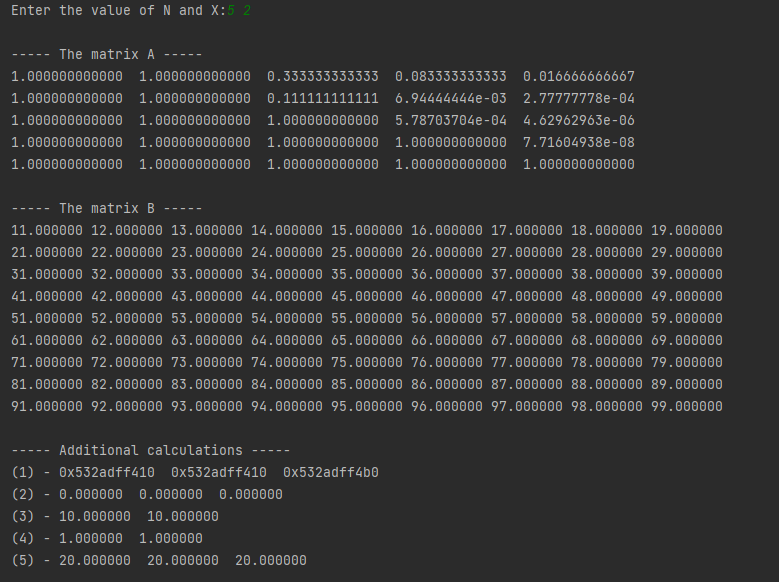
cout << "(3) - " << \*(\*(B + 1)) << " " << \*B[1] << endl;

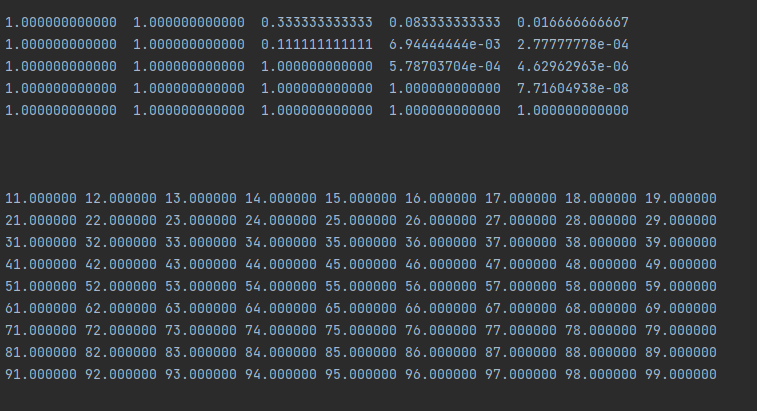
cout << "(4) - " << \*(B[0] + 1) << " " << \*(\*B + 1) << endl;

cout << "(5) - " << B[0][20] << " " << \*(B[0] + 20) << " " << \*B[2] << endl;

}

**Анализ результатов**

****

****

**Вывод**

Я научился:

* Создавать двумерные динамические массивы
* Обрабатывать матрицы
* Использовать текстовые файлы для хранения данных
* Форматировать и выводить матрицу на экран