**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №8

«Обработка и печать числовой матрицы.»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-13Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Аксёнова М.В. |
| Подпись и дата |  | Подпись и дата: |

Москва, 2021 г.

**Постановка задачи**

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 , а 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 алгоритма**

Алгоритм вывода матрицы работает следующим образом:  
программа получается количество столбцов в окне терминала, после чего, зная сколько будет занимать каждое выводимое значение исходя из введённой пользователем точности, находит количество столбцов, которые поместятся в одной “строке”. Количество всех столбцов делится на это число, и становится количество “строк” в итоговом выводе. Далее цикл идёт по каждой “строке”, рассчитывая на каком номере столбца закончится конкретная по формуле count \* (z + 1). После выхода из цикла по “строкам” выводится оставшаяся часть матрицы.

**Описание используемых переменных**

**void PrintMatrix(double \*\*matrix, int n, int m, bool isScientific, int precision) – функция вывода матрицы**

**double \*\*matrix – указатель на первый элемент матрицы**

**int n, int m – размеры матрицы**

**bool isScientific – вывод в научном виде или в фиксированном**

**int precision - точность**

**int z - итератор для цикла по “строкам”**

**int count – количество помещающихся столбцов**

**Текст программы**

**Листинг кода программы:**

**main.cpp**

#include "functions.hpp"

int main()

{

int n, precision;

bool isScientific;

std::cout << "Enter n:\n";

std::cin >> n;

std::cout << "Scientific(1) or fixed(0)?\n";

std::cin >> isScientific;

std::cout << "Enter precision:\n";

std::cin >> precision;

double \*\*matrix = CreateMatrix(n);

PrintMatrix(matrix, n, n, isScientific, precision);

DestructMatrix(matrix, n);

int B[10][10];

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

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

}

}

PrintMatrix(B);

std::cout << B << " " << B[0] << " " << B[2] << std::endl;

std::cout << B[0][0] << " " << \*\*B << " " << \*B[0] << std::endl;

std::cout << \*(\*(B + 1)) << " " << \*B[1] << std::endl;

std::cout << \*(B[0] + 1) << " " << \*(\*B + 1) << std::endl;

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

}

**functions.hpp**

#include <math.h>

#include <iostream>

#include <iomanip>

#include <sys/ioctl.h>

#include <stdio.h>

#include <unistd.h>

int factorial(int);

double \*\*CreateMatrix(int);

void DestructMatrix(double \*\*, int);

void PrintMatrix(double \*\*, int, int, bool, int);

void PrintMatrix(int[10][10]);

**functions.cpp**

#include "functions.hpp"

int factorial(int n)

{

if (n <= 1)

return 1;

else

return (n \* factorial(n - 1));

}

double \*\*CreateMatrix(int N)

{

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

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

{

matrix[i] = new double[N];

}

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

{

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

{

if (i == j)

{

matrix[i][j] = 1;

}

else if (i < j)

{

matrix[i][j] = pow(1, i) / pow(factorial(j), i);

}

else if (i > j)

{

matrix[i][j] = pow(-1, i) / pow(factorial(j), i);

}

}

}

return matrix;

}

void DestructMatrix(double \*\*matrix, int N)

{

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

{

delete[] matrix[i];

}

delete[] matrix;

}

void PrintMatrix(int matrix[10][10])

{

std::cout << std::endl;

for (size\_t i = 0; i < 10; i++)

{

for (size\_t j = 0; j < 10; j++)

{

std::cout << std::setw(4) << matrix[i][j];

}

std::cout << std::endl;

}

std::cout << std::endl;

}

void PrintMatrix(double \*\*matrix, int n, int m, bool isScientific, int precision)

{

struct winsize w;

ioctl(STDOUT\_FILENO, TIOCGWINSZ, &w);

int count = 0;

if (isScientific)

{

int z = 0;

int q = 0;

count = w.ws\_col / (precision + 9);

for (z = 0; z < m / count; z++)

{

for (size\_t i = 0; i < n; i++)

{

for (size\_t j = count \* z; j < count \* (z + 1); j++)

{

std::cout << std::scientific << std::setprecision(precision) << std::setw(precision + 9) << matrix[i][j];

q = j;

}

std::cout << std::endl;

}

std::cout << std::endl;

}

if (q + 1 < m)

{

for (size\_t i = 0; i < n; i++)

{

for (size\_t j = q + 1; j < m; j++)

{

std::cout << std::scientific << std::setprecision(precision) << std::setw(precision + 9) << matrix[i][j];

}

std::cout << std::endl;

}

}

}

else

{

int z = 0;

int q = 0;

count = w.ws\_col / (precision + 4);

for (int z = 0; z < m / count; z++)

{

for (size\_t i = 0; i < n; i++)

{

for (size\_t j = count \* z; j < count \* (z + 1); j++)

{

std::cout << std::fixed << std::setprecision(precision) << std::setw(precision + 4) << matrix[i][j];

q = j;

}

std::cout << std::endl;

}

std::cout << std::endl;

}

if (q + 1 < m)

{

for (size\_t i = 0; i < n; i++)

{

for (size\_t j = q + 1; j < m; j++)

{

std::cout << std::fixed << std::setprecision(precision) << std::setw(precision + 4) << matrix[i][j];

}

std::cout << std::endl;

}

}

}

}

**Анализ результатов**



