Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №1

по курсу «Технология программирования»

**«Исследование перегруженных функций и шаблонов»**

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

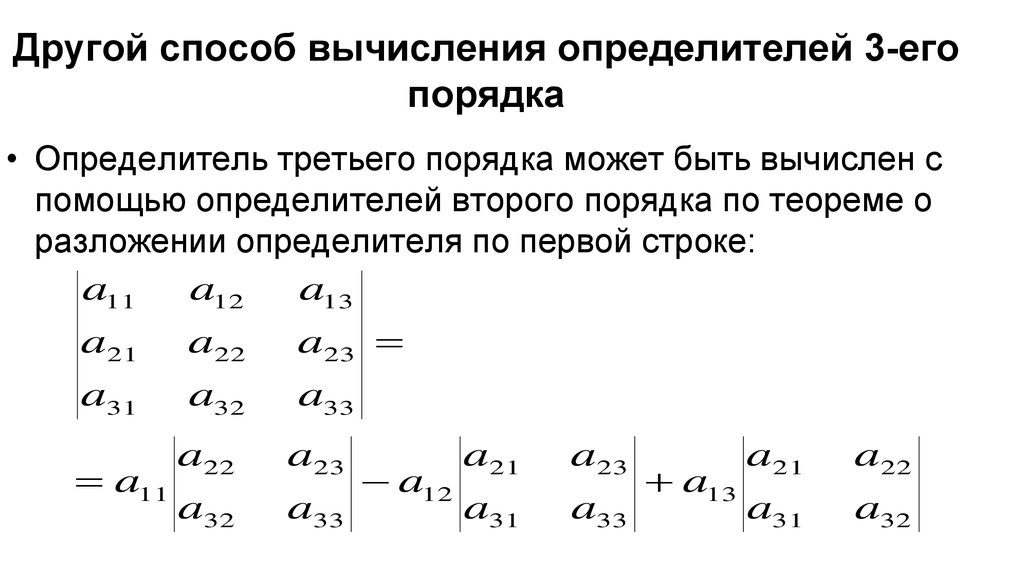
Проверила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Долженкова М.Л/

Киров 2022

**Задание:** Разработать консольное приложение для вычисления обратной матрицы, позволяющее обработать данные типов int и double с использованием перегруженных функций и шаблонов. Обеспечить обработку некорректного ввода данных. Сравнить применимость шаблонных и перегруженных функций

**Математические выкладки:**

Из свойств определителя известно, что определитель матрицы порядка N может быть представлен в виде суммы N определителей N-1 порядка (разложение по строке или столбцу). Предположим мы раскладываем по первому столбцу. При этом определитель равен сумме произведений элементов этого столбца на минор данного элемента матрицы и на -1 в степени суммы индексов элемента. Минор элемента а[i,j] матрицы - это определитель матрицы, полученной вычеркиванием i-той строки и j-того столбца.

  
Таким образом, можно воспользоваться рекурсией и, написав одну простую процедуру, вычислять определители матрицы любого порядка.

**Листинг:**

#include <iostream>

#include <ctime>

#include <cmath>

#include <typeinfo>

using namespace std;

template <typename T> void FreeMem(T\*\* matr, int n);

template <typename T> void PrintMtx(T\*\* matr, int n);

template <typename T> void SetMtx(T\*\* matr, int n);

template <typename T> void TransponMtx(T\*\* matr, T\*\* tMatr, int n);

int Get\_matr(int\*\* matr, int n, int\*\* temp\_matr, int indRow, int indCol);

double Get\_matr(double\*\* matr, int n, double\*\* temp\_matr, int indRow, int indCol);

int Det(int\*\* matr, int n);

double Det(double\*\* matr, int n);

int main()

{

setlocale(0, "");

int n;

int str;

while(true){

system("cls");

cout << "Введите размер матрицы: ";

cin >> n;

if (std::cin.fail()) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else if (n <= 1 || n>10) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else if (std::cin.peek() != '\n') {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else break;

}

while (true){

system("cls");

std::cout << "Введите '1' для целочисленного массива и '0' для дробного: ";

cin >> str;

if (std::cin.fail()) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else if (str != 1 && str != 0) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else if (std::cin.peek() != '\n') {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else break;

}

switch (str)

{

case (1): {

int det;

int\*\* matr = new int\* [n];

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

matr[i] = new int[n];

SetMtx(matr, n);

cout << std::endl;

PrintMtx(matr, n);

det = Det(matr, n);

cout << "Определитель матрицы = " << det << endl;

double\*\* obr\_matr = new double\* [n];

double\*\* tobr\_matr = new double\* [n];

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

obr\_matr[i] = new double[n];

tobr\_matr[i] = new double[n];

}

if (det) {

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

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

int m = n - 1;

int\*\* temp\_matr = new int\* [m];

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

temp\_matr[k] = new int[m];

Get\_matr(matr, n, temp\_matr, i, j);

obr\_matr[i][j] = pow(-1.0, i + j + 2) \* Det(temp\_matr, m) / det;

FreeMem(temp\_matr, m);

}

}

// Транспонирование матрицы

TransponMtx(obr\_matr, tobr\_matr, n);

//Печать обратной матрицы после транспонирования

PrintMtx(tobr\_matr, n);

}

else

cout << "Т.к. определитель матрицы = 0,\nто матрица вырожденная и обратной не имеет!!!" << endl;

FreeMem(tobr\_matr, n);

FreeMem(matr, n);

FreeMem(obr\_matr, n);

break;

}

case (0): {

double det;

double \*\*matr = new double\* [n];

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

matr[i] = new double[n];

SetMtx(matr, n);

cout << std::endl;

PrintMtx(matr, n);

det = Det(matr, n);

std::cout << "Определитель матрицы = " << det << endl;

double\*\* obr\_matr = new double\* [n];

double\*\* tobr\_matr = new double\* [n];

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

obr\_matr[i] = new double[n];

tobr\_matr[i] = new double[n];

}

if (det) {

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

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

int m = n - 1;

double\*\* temp\_matr = new double\* [m];

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

temp\_matr[k] = new double[m];

Get\_matr(matr, n, temp\_matr, i, j);

obr\_matr[i][j] = pow(-1.0, i + j + 2) \* Det(temp\_matr, m) / det;

FreeMem(temp\_matr, m);

}

}

// Транспонирование матрицы

TransponMtx(obr\_matr, tobr\_matr, n);

//Печать обратной матрицы после транспонирования

PrintMtx(tobr\_matr, n);

}

else

cout << "Т.к. определитель матрицы = 0,\nто матрица вырожденная и обратной не имеет!!!" << endl;

FreeMem(tobr\_matr, n);

FreeMem(matr, n);

FreeMem(obr\_matr, n);

break;

}

default:

break;

}

return 0;

}

//Функция транспонирования матрицы

template <typename T> void TransponMtx(T\*\* matr, T\*\* tMatr, int n) {//

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

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

tMatr[j][i] = matr[i][j];

}

//Функция освобождения памяти

template <typename T> void FreeMem(T\*\* matr, int n)

{

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

delete[] matr[i];

delete[] matr;

}

//функция заполнения матрицы

template <typename T> void SetMtx(T\*\* matr, int n)

{

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

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

while (true) {

cout << "Введите matr[" << i <<"," << j << "]= ";

std::cin >> matr[i][j];

if (std::cin.fail()) {

cout << "Первое число получено.\nНекорректный ввод данных,буфер очищен"<<std::endl;

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else break;

}

}

}

//функция печати матрицы

template <typename T> void PrintMtx(T\*\* matr, int n)

{

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

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

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

cout << endl;

}

}

//функция вычеркивания строки и столбца

int Get\_matr(int\*\* matr, int n, int\*\* temp\_matr, int indRow, int indCol)

{

int ki = 0;

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

if (i != indRow) {

for (int j = 0, kj = 0; j < n; j++) {

if (j != indCol) {

temp\_matr[ki][kj] = matr[i][j];

kj++;

}

}

ki++;

}

}

return 0;

}

double Get\_matr(double\*\* matr, int n, double\*\* temp\_matr, int indRow, int indCol)

{

int ki = 0;

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

if (i != indRow) {

for (int j = 0, kj = 0; j < n; j++) {

if (j != indCol) {

temp\_matr[ki][kj] = matr[i][j];

kj++;

}

}

ki++;

}

}

return 0;

}

//функция вычисления определителя матрицы

int Det(int\*\* matr, int n)

{

int temp = 0; //временная переменная для хранения определителя

int k = 1; //степень

if (n == 1)

temp = matr[0][0];

else if (n == 2)

temp = matr[0][0] \* matr[1][1] - matr[1][0] \* matr[0][1];

else {

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

int m = n - 1;

int\*\* temp\_matr = new int\* [m];

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

temp\_matr[j] = new int[m];

Get\_matr(matr, n, temp\_matr, 0, i);

temp = temp + k \* matr[0][i] \* Det(temp\_matr, m);

k = -k;

FreeMem(temp\_matr, m);

}

}

return temp;

}

double Det(double\*\* matr, int n)

{

double temp = 0; //временная переменная для хранения определителя

int k = 1; //степень

if (n == 1)

temp = matr[0][0];

else if (n == 2)

temp = matr[0][0] \* matr[1][1] - matr[1][0] \* matr[0][1];

else {

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

int m = n - 1;

double\*\* temp\_matr = new double\* [m];

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

temp\_matr[j] = new double[m];

Get\_matr(matr, n, temp\_matr, 0, i);

temp = temp + k \* matr[0][i] \* Det(temp\_matr, m);

k = -k;

FreeMem(temp\_matr, m);

}

}

return temp;

}

**Вывод:** в ходе выполнения лабораторной работы было изучено применение перегрузок и шаблона в функциях. Шаблонные функции необходимо применять в случае, если алгоритм одинаковый и не зависит от типа входных и выходных параметров. Например, ввод и вывод значений матрицы. Что касается перегрузки, то её следует применять в тех случаях, когда алгоритмы, выполняемые функциями, отличны и могут отличаться типами входных или выходных параметров, количеством формальных параметров. Стоит отметить, что отличие в одном лишь типе возвращаемого значения недостаточно для создания перегрузки. В лабораторной работе перегрузка использовалась для функции вычеркивания столбца и строки из матрицы, а также для функции вычисления определителя. Это было связано с необходимостью создания временной матрицы и выделением памяти под неё в зависимости от типа.