МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

(ФГБОУ ВО «ВятГУ»)

Факультет автоматики и вычислительной техники

Кафедра ЭВМ

Отчёт

Лабораторная работа № 1 по дисциплине

«Технология программирования»

«Исследование перегруженных функций и шаблонов»

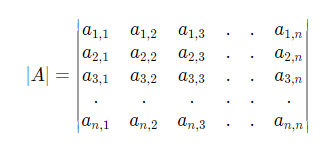
Выполнил студент группы ИВТб-2301-04-00 / Жеребцов К. А.

Проверил преподаватель / Долженкова М. Л.

Киров 2022

1. **Цель:** изучить перегруженные функции и шаблоны.
2. **Задание:** разработать консольное приложение, которое вычисляет определитель матриц вещественных или целых чисел с использованием перегруженных функций и шаблонов. Обеспечить обработку некорректного ввода данных. Сравнить применимость шаблонных и перегруженных функций.
3. **Теория:**

Определитель матрицы равен сумме произведений элементов любой строки или любого столбца и их алгебраических дополнений.



Можно посчитать определитель, например, используя строку i:

Либо же можно посчитать определитель, используя столбец j:

1. **Исходный код:**

#include <iostream>

#include <windows.h>

#include<stdio.h>

#include<conio.h>

using namespace std;

const int n = 10; //размерность массива

int m;

bool f = 1;

// Вывод информации

void print\_help()

{

cout << "What do you want to do: " << endl;

cout << "1 - Calculate the determinant of a matrix of integers" << endl;

cout << "2 - Calculate the determinant of a matrix of real numbers" << endl;

cout << "3 - Exit" << endl;

}

// Перегруженная функция для считывания элементов матрицы DOUBLE

void Get\_Matr(double aa[n][n], int count)

{

int i, j;

for (i = 0; i < count; ++i) {

for (j = 0; j < count; ++j) {

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

cin >> aa[i][j];

while (!std::cin.good() || std::cin.peek() != '\n') {

cout << "Wrong input!" << endl;

cin.clear();

cin.ignore(255, '\n');

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

cin >> aa[i][j];

}

}

}

}

// Перегруженная функция для считывания элементов матрицы INT

void Get\_Matr(int aa[n][n], int count)

{

int i, j;

for (i = 0; i < count; ++i) {

for (j = 0; j < count; ++j) {

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

cin >> aa[i][j];

while (!std::cin.good() || std::cin.peek() != '\n') {

cout << "Wrong input!" << endl;

cin.clear();

cin.ignore(255, '\n');

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

cin >> aa[i][j];

}

}

}

}

//Перегруженная процедура печати матрицы INT

void printMtx(int matrix[n][n], int count)

{

int i, j;

for (i = 0; i < count; i++) {

for (j = 0; j < count; j++) {

cout << "|" << matrix[i][j] << "\t";

}

cout << endl;

}

}

//Перегруженная процедура печати матрицы DOUBLE

void printMtx(double matrix[n][n], int count)

{

int i, j;

for (i = 0; i < count; i++) {

for (j = 0; j < count; j++) {

cout << "|" << matrix[i][j] << "\t";

}

cout << endl;

}

}

//Шаблонная функция вычисления определителя(матрица,размерность)

template<class Type>

Type det(Type matrix[n][n], int count)

{

Type temp\_matrix[n][n]{};

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

int k = 1; //степень

int i, j;

if (count < 1) {

std::cout << "not run";

return 0;

}

else if (count == 1)

temp = matrix[0][0];

else if (count == 2)

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

else

{

for (i = 0; i < count; i++)

{

int ki, kj, di, dj;

di = 0;

for (ki = 0; ki < count - 1; ki++)

{

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < count - 1; kj++)

{

if (kj == 0) dj = 1;

temp\_matrix[ki][kj] = matrix[ki + di][kj + dj];

}

}

temp = temp + k \* matrix[i][0] \* det(temp\_matrix, count - 1);

k = -k;

}

}

return temp;

}

// Выбор действия

void operation()

{

int num = 0;

while (1)

{

print\_help();

cout << "---------------------------------------" << endl;

cin >> num;

while (!std::cin.good() || std::cin.peek() != '\n')

{

cout << "Wrong input!" << endl;

cin.clear();

cin.ignore(255, '\n');

cin >> num;

}

if ((num < 1) || (num > 3))

{

cout << "Invalid code of operation" << endl;

cout << "Enter a number from 1 to 3" << endl;

}

if (num == 1)

{

cout << "Enter the size of the matrix (no more than 10)" << endl;

cin >> m;

while (!std::cin.good() || std::cin.peek() != '\n' || (m > n) || (m < 1))

{

cout << "Wrong input!" << endl;

cin.clear();

cin.ignore(255, '\n');

cin >> m;

}

int mtx[n][n]; //матрица

Get\_Matr(mtx, m);

cout << "---------------------------------------" << endl;

printMtx(mtx, m);

cout << "---------------------------------------" << endl;

cout << "The determinant is " << det(mtx, m) << endl;

cout << "Complete!" << endl;

break;

}

if (num == 2)

{

cout << "Enter the size of the matrix (no more than 10)" << endl;

cin >> m;

while (!std::cin.good() || std::cin.peek() != '\n' || (m > n) || (m < 1))

{

cout << "Wrong input!" << endl;

cin.clear();

cin.ignore(255, '\n');

cin >> m;

}

double mtx[n][n]; //матрица

Get\_Matr(mtx, m);

cout << "---------------------------------------" << endl;

printMtx(mtx, m);

cout << "---------------------------------------" << endl;

cout << "The determinant is " << fixed << det(mtx, m) << endl;

cout << "Complete!" << endl;

break;

}

if (num == 3)

{

f = false;

break;

}

}

}

int main()

{

char c;

while (f) {

operation();

if (f == true) {

do {

cout << "Press <Enter> to continue the program..." << endl;

c = \_getch();

} while (c != 13);

}

system("cls");

}

return 0;

}

1. **Вывод:** было разработано консольное приложение, которое вычисляет определитель матриц вещественных или целых чисел с использованием перегруженных функций и шаблонов. Также была обеспечена обработка некорректного ввода данных. В ходе выполнения лабораторной работы было выявлено, что перегруженные функции лучше использовать в случаях, когда необходимо избежать дублирования имен функции, выполняющих сходные действия, но с различной программной логикой. А шаблонные функции применяются, когда нужно реализовать определенный алгоритм без его привязки к некоторым параметрам (например, типами данных). Другими словами, шаблон функции позволяет определять семейство функций. Это семейство характеризуется общим алгоритмом, который может применяться к данным различных типов.