МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

**«Вятский государственный университет»**

**(ФГБОУ ВПО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Перегрузка функций и шаблонные функции

Отчет по лабораторной работе №1 дисциплины

«Технологии программирования»

Выполнил студент группы ИВТб-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Седов М.Д./

Проверил доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М.Л./

Киров 2019

**Цель работы**: получить базовые навыки работы с шаблонами и перегруженными функциями

**Задание:**

1) Написать программу, выполняющую LU-разложение исходной матрицы с заданным типом элементов.

2) Тип элемент может быть int или double.

3) В работе должны быть применены как перегруженные функции, так и шаблоны

**Описание работы алгоритма:**

Будем использовать следующие обозначения для элементов матриц: L=(lij), U=(uij), i, j = 1…n; причем диагональные элементы матрицы L: lii = 1, i = 1…n.

1. uij=a1j, j = 1…n
2. lj1 = aj1 / u11, j = 1…n (u11 !=0)

Для i = 2…n

1. uij=aij -
2. lij=1/uii \* (aji -

**Исходный текст программы:**

#include <iostream>

#include <conio.h>

#include <string>

#include <Windows.h>

using namespace std;

const int NUMBER\_FIELD = 2;

template <class T> //шаблон для ввода матрицы

T\*\* read\_matrix(int n);

template <class T> //шаблон для вывода матрицы

void write\_matrix(T\*\*, int);

template <class T> // шаблон структуры, которая возвращается в качестве LU-разложения

struct LU {

T\*\* L;

T\*\* U;

};

//перегруженные функции для разложения LU-матрицы

LU <int> LU\_matrix(int\*\* table, int n);

LU <double> LU\_matrix(double\*\* table, int n);

void print\_menu(string\* menu, int n, int pos) {

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(handle, FOREGROUND\_GREEN);

for (int i = 0; i < pos; i++) cout << menu[i] << endl;

SetConsoleTextAttribute(handle, FOREGROUND\_RED);

cout << menu[pos] << endl;

SetConsoleTextAttribute(handle, FOREGROUND\_GREEN);

for (int i = pos + 1; i < n; i++) cout << menu[i] << endl;

}

int main() {

int choice;

int pos = 0;

string menu[NUMBER\_FIELD] = { "1 - int" , "2 - double" };

cout << "Calculation LU-matrix:\n";

cout << "Choose the type (1 or 2)\n";

print\_menu(menu, NUMBER\_FIELD, pos);

char ch;

int code;

do {

ch = getch();

code = static\_cast<int> (ch);

switch (code)

{

case 80:

if (pos < NUMBER\_FIELD - 1) {

system("cls");

++pos;

print\_menu(menu, NUMBER\_FIELD, pos);

}

break;

case 72:

if (pos > 0) {

system("cls");

--pos;

print\_menu(menu, NUMBER\_FIELD, pos);

}

break;

case 13:

choice = ++pos;

break;

default:

break;

}

} while (code != 13);

int n;

while (true) {

cout << "\nEnter the size matrix\n";

cin >> n;

if (!cin && n > 0) {

cout << "Wrong size entry. Repeat input\n";

cin.clear();

while (cin.get() != '\n');

}

else break;

}

cout << endl;

int\*\* table\_i;

double\*\* table\_d;

int\*\* L\_i;

double\*\* L\_d;

int\*\* U\_i;

double\*\* U\_d;

switch (choice)

{

case 1:

table\_i = read\_matrix <int>(n);

LU <int> temp\_i = LU\_matrix(table\_i, n);

L\_i = temp\_i.L;

U\_i = temp\_i.U;

break;

case 2:

table\_d = read\_matrix <double>(n);

LU <double> temp\_d = LU\_matrix(table\_d, n);

L\_d = temp\_d.L;

U\_d = temp\_d.U;

break;

default:

return 0;

}

switch (choice)

{

case 1:

cout << "\nCurrent matrix:\n";

write\_matrix <int>(table\_i, n);

cout << "\nMatrix L: \n";

write\_matrix <int>(L\_i, n);

cout << "\nMatrix U: \n";

write\_matrix <int>(U\_i, n);

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

delete[] table\_i[i];

delete[] L\_i[i];

delete[] U\_i[i];

}

delete[] table\_i;

delete[] L\_i;

delete[] U\_i;

break;

case 2:

cout << "\nCurrent matrix:\n";

write\_matrix <double>(table\_d, n);

cout << "\nMatrix L: \n";

write\_matrix <double>(L\_d, n);

cout << "\nMatrix U: \n";

write\_matrix <double>(U\_d, n);

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

delete[] table\_d[i];

delete[] L\_d[i];

delete[] U\_d[i];

}

delete[] table\_d;

delete[] L\_d;

delete[] U\_d;

break;

default:

return 0;

}

\_getch();

return 0;

}

template <class T>

T\*\* read\_matrix(int n) {

T\*\* table = new T\*[n];

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

table[i] = new T[n];

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

while (true) {

cout << "A[" << i << "] [" << j << "]: ";

cin >> table[i][j];

if (!cin || (table[i][j] == 0 && i == j)) {

cout << "Wrong value of the matrix element. Repeat input\n";

cin.clear();

while (cin.get() != '\n');

}

else break;

}

}

}

return table;

}

template <class T>

void write\_matrix(T\*\* table, int n) {

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

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

cout << table[i][j] << "\t";

cout << endl;

}

cout << endl;

}

LU <int> LU\_matrix(int\*\* table, int n) {

LU <int> temp;

temp.U = new int\*[n];

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

temp.U[i] = new int[n];

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

temp.U[i][j] = table[i][j];

}

temp.L = new int\*[n];

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

temp.L[i] = new int[n];

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

temp.L[i][j] = 0;

}

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

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

temp.L[j][i] = temp.U[j][i] / temp.U[i][i];

for (int k = 1; k < n; k++) {

for (int i = k - 1; i < n; i++)

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

temp.L[j][i] = temp.U[j][i] / temp.U[i][i];

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

for (int j = k - 1; j < n; j++)

temp.U[i][j] -= temp.L[i][k - 1] \* temp.U[k - 1][j];

}

return temp;

}

LU <double> LU\_matrix(double\*\* table, int n) {

LU <double> temp;

temp.U = new double\*[n];

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

temp.U[i] = new double[n];

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

temp.U[i][j] = table[i][j];

}

temp.L = new double\*[n];

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

temp.L[i] = new double[n];

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

temp.L[i][j] = 0;

}

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

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

temp.L[j][i] = temp.U[j][i] / temp.U[i][i];

for (int k = 1; k < n; k++) {

for (int i = k - 1; i < n; i++)

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

temp.L[j][i] = temp.U[j][i] / temp.U[i][i];

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

for (int j = k - 1; j < n; j++)

temp.U[i][j] = temp.U[i][j] - temp.L[i][k - 1] \* temp.U[k - 1][j];

}

return temp;

}

**Вывод**

В ходе данной лабораторной работы были освоены базовые возможности для работы с перегруженными функциями и шаблонными функциями. Перегруженные функции имеют место, например, если необходимо при вызове функции с одинаковым именем при разных типах фактических параметрах выполнять различные действия. Например, для функции, принимающей типа данных int выполнять действия сложения матриц, а при типе данных double использовать действия для сложения двух целых чисел. Шаблонные функции применяются для создания универсальной функции, которая отличается лишь типом входных данных, либо типом возврата значения функции. Например, создается шаблон функции, которая создаёт матрицу заданного типа, и возвращающую её в качестве результата. Действия, выполнимые для создания матрицы будут совпадать для int, double и т.д. В данной лабораторной работе использования перегруженных функций было нецелесообразно, поскольку операция разложения LU-матрицы идентична как для типа int, так и для типа double. Ввод и вывод матриц осуществлялось так же при помощи шаблонных функций.