МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**«Создание консольного приложения Изучение основ С++.**

**Перегрузка функций и шаблонные функции.»**

Отчет по лабораторной работе № 1 по дисциплине

«Технологии программирования»

Выполнил студент группы: ПИб-2301\_\_\_\_\_\_\_\_\_\_\_\_\_ /Морозов Д.А.

Проверил преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М.Л.

**Киров 2023**

**Цели работы:**

* Разработать консольное приложение, позволяющее обработать данные заданных типов с использованием перегруженных функций и шаблонов;
* Обеспечить обработку некорректного ввода данных.
* Сравнить применимость шаблонных и перегруженных функций

**Задание:**

Разработать консольное приложение, в котором пользователь должен выбрать тип данных элементов (int или float) матрицы и ввести размерность, после чего программа должна произвести разность матриц.

**Словесное описание алгоритма:**

Первым делом идет ввод переменной type, которая имеет тип данных int. Ввод реализован при помощи функции input, в которой проверяются введенные пользователем значения. При помощи функции check производится обход, введенной пользователем строки, посимвольно, и в случае, если проверка была пройдена, то есть введенная строка содержит только цифры, функция передает значение преобразованное в тип int в переменную type.

Далее производится проверка введенного значения, после чего вводятся переменные n и m(кол-во строк и столбцов соответственно).

В соответствии с выбранным типом инициализируется динамическая матрица, после чего вызывается перегруженная функция inputMatrix. В её теле происходит заполнение матрицы данными от пользователя.

После чего вызывается шаблонная функция addMatrix, которая выполняет действие разности с матрицами и выводит готовую матрицу в консоль.

После выполнения операций очищается память, выделенная для динамической матрицы.

**Листинг кода:**

//

// main.cpp

// Testing

//

// Created by Danil Morozov on 08.02.2023.

//

#include <iostream>

using namespace std;

bool is\_valid\_input(string input) {

for (char c : input) {

if (!isdigit(c)) {

return false;

}

}

return true;

}

bool checkInteger(string input)

{

bool isNeg=false;

int itr=0;

if(input.size()==0)

return false;

if(input[0]=='-')

{

isNeg=true;

itr=1;

}

for(int i=itr;i<input.size();i++)

{

if(!isdigit(input[i]))

return false;

}

return true;

}

// Function to input the elements of the matrix

void inputMatrix(int\*\* matrix, int numRows, int numCols) {

cout << "Enter the elements of the matrix:\n";

string input;

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

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

cout << "Enter element " << i + 1 << ", " << j + 1 << ": \n";

cin >> input;

if (checkInteger(input)){

cout << "Good\n";

matrix[i][j] = stoi(input);

}else {

cout << "Error: Invalid input. Please enter a valid integer.\n";

j--;

}

// Check for data entry errors

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Error: Invalid input. Please enter a valid integer.\n";

}

}

}

}

// Function to input the elements of the matrix (for float type)

void inputMatrix(float\*\* matrix, int numRows, int numCols) {

cout << "Enter the elements of the matrix:\n";

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

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

cout << "Enter element " << i + 1 << ", " << j + 1 << ": \n";

cin >> matrix[i][j];

// Check for data entry errors

if (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Error: Invalid input. Please enter a valid float.\n";

j--;

}

}

}

}

template<class T>

void addMatrix(T\*\* matrix1, T\*\* matrix2, T\*\* result, int numRows, int numCols) {

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

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

result[i][j] = matrix1[i][j] - matrix2[i][j];

}

}

}

int main() {

int numRows, numCols;

char type;

string input;

// rows

do {

cout << "Enter the number of rows: ";

cin >> input;

while (!is\_valid\_input(input)) {

cout << "Incorrect input. Try again! \n";

cout << "Enter the number of rows: ";

cin.clear();

cin >> input;

}

numRows = stoi(input);

if (numRows < 1) {

cout << "Incorrect input. Try again! \n" << endl;

continue;

}

break;

} while (true);

// cols

do {

cout << "Write number of cols: ";

cin >> input;

while (!is\_valid\_input(input)) {

cout << "Incorrect input. Try again! \n";

cout << "Write number of cols: ";

cin.clear();

cin >> input;

}

numCols = stoi(input);

if (numCols < 1) {

cout << "Incorrect input. Try again! \n" << endl;

continue;

}

break;

} while (true);

// type

do {

cout << "Choose the type ('1' - int, '2' - float): ";

cin >> input;

while (!is\_valid\_input(input)) {

cout << "Incorrect input. Try again! \n";

cout << "Write 1 or 2 \n";

cin.clear();

cin >> input;

}

type = stoi(input);

if (type != 1 && type != 2) {

cout << "Incorrect input. Try again! \n" << endl;

continue;

}

break;

} while (true);

if (type == 1) {

// Input the elements of the first matrix (for integer type)

int\*\* matrix1 = new int\* [numRows];

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

matrix1[i] = new int[numCols];

}

int\*\* matrix2 = new int\* [numRows];

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

matrix2[i] = new int[numCols];

}

int\*\* result = new int\* [numRows];

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

result[i] = new int[numCols];

}

cout << "Enter elements of the first matrix (for integer type):\n";

inputMatrix(matrix1, numRows, numCols);

// Input the elements of the second matrix (for integer type)

cout << "Enter elements of the second matrix (for integer type):\n";

inputMatrix(matrix2, numRows, numCols);

// Add the two matrices (for integer type)

addMatrix(matrix1, matrix2, result, numRows, numCols);

// Print the result (for integer type)

cout << "The difference of the two matrices (for integer type) is:\n";

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

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

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

}

cout << endl;

}

// Free the memory

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

delete[] matrix1[i];

delete[] matrix2[i];

delete[] result[i];

}

delete[] matrix1;

delete[] matrix2;

delete[] result;

} else {

float\*\* matrix3 = new float\* [numRows];

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

matrix3[i] = new float[numCols];

}

float\*\* matrix4 = new float\* [numRows];

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

matrix4[i] = new float[numCols];

}

float\*\* result = new float\* [numRows];

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

result[i] = new float[numCols];

}

// Input the elements of the first matrix (for float type)

cout << "Enter elements of the first matrix (for float type):\n";

inputMatrix(matrix3, numRows, numCols);

// Input the elements of the second matrix (for float type)

cout << "Enter elements of the second matrix (for float type):\n";

inputMatrix(matrix4, numRows, numCols);

// Add the two matrices (for float type)

addMatrix(matrix3, matrix4, result, numRows, numCols);

// Print the result (for float type)

cout << "The difference of the two matrices (for float type) is:\n";

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

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

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

}

cout << endl;

}

// Free the memory

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

delete[] matrix3[i];

delete[] matrix4[i];

delete[] result[i];

}

delete[] matrix3;

delete[] matrix4;

delete[] result;

}

return 0;

}

**Экранные формы:**

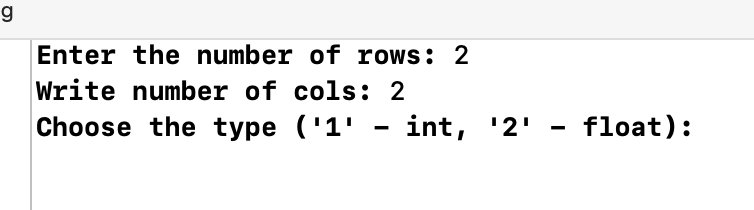
****

Рис. 1 – Ввод используемых данных

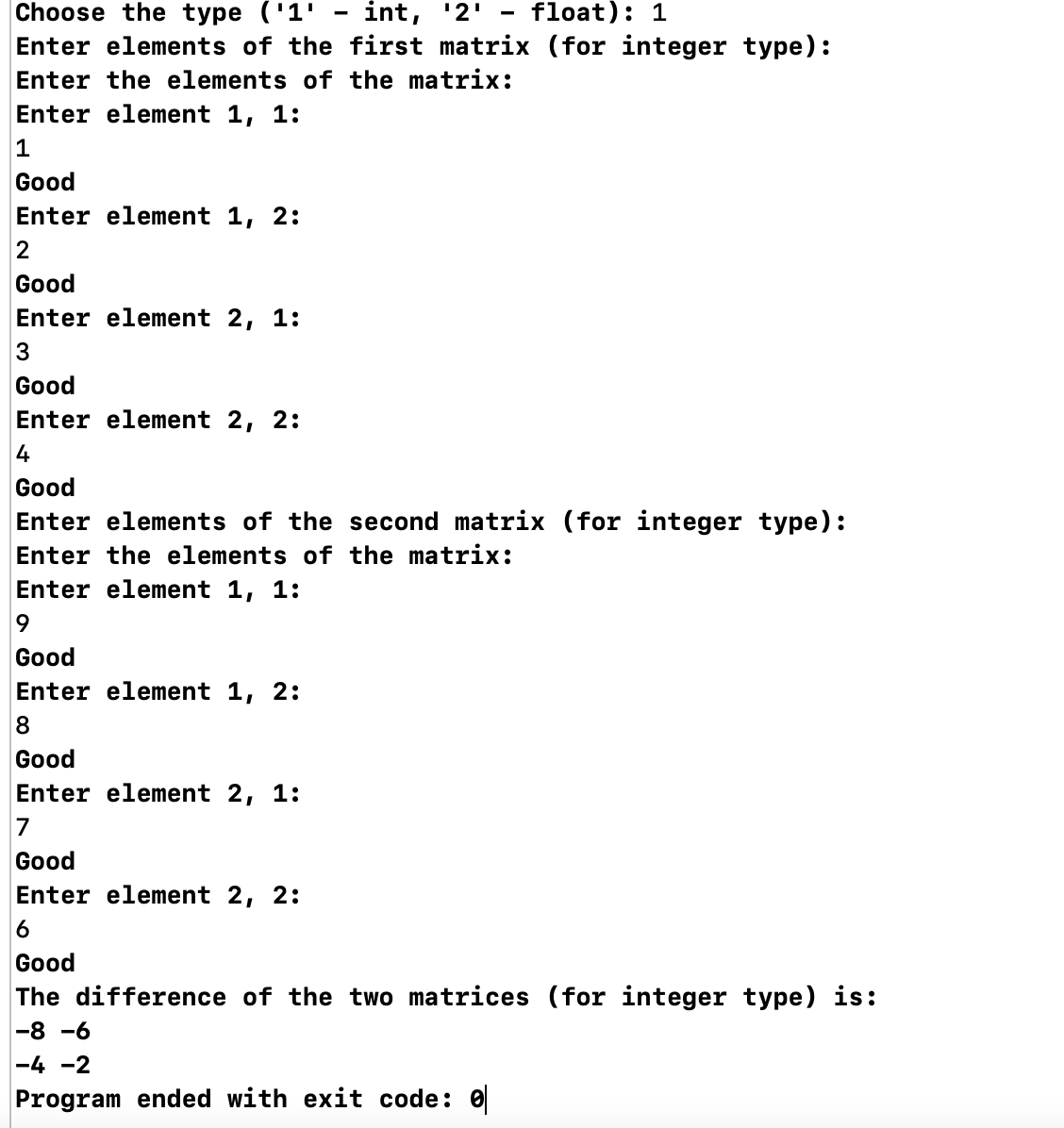
****

Рис. 2 – матрица с типом данных int

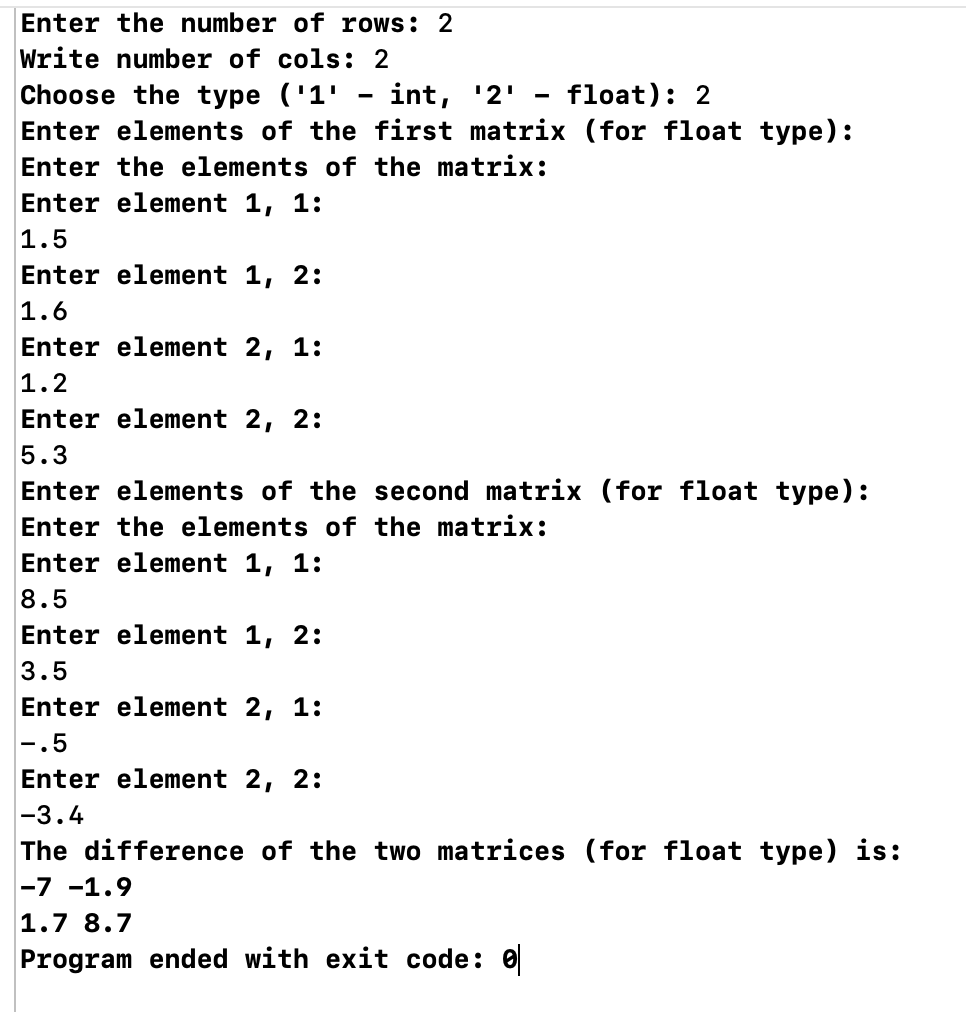
****

Рис. 2 – матрица с типом данных float

**Вывод:**

В ходе данной лабораторной работы было разработано консольное приложение, позволяющее обрабатывать данные заданных типов с использованием перегруженных функций и шаблонов.

Так же была обеспечена обработка некорректного ввода данных.

При сравнении шаблонных и перегруженных функций стоит отметить, что шаблонные функции более удобны, так как они занимают меньше места, по сравнению с перегруженными функциями, которые более просты в понимании. В данной лабораторной работе более предпочтительным вариантом является использование шаблонов, а от перегруженных функций можно отказаться для того, чтобы уменьшить объем кода. Но, стоит отметить, что использование перегруженных функций будет более уместно в случае, если необходимо иметь разные алгоритмы для каждой конкретной функции.