**6 ЛАБОРАТОРНАЯ РАБОТА №6**

**«ИССЛЕДОВАНИЕ ШАБЛОНОВ ФУНКЦИЙ»**

**6.1 Цель работы**

Исследование назначения и способа описания шаблонов функций, применение их при написании объектно-ориентированных программ.

**6.2 Вариант задания – 8**

Разработать программу на языке C++, которая обрабатывает данные разных типов (int, char, и др.). Функция обработки данных должна быть реализована как шаблон. Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы.

Требуется написать функцию-шаблон, переставляющую i-ю и j-ю строки в матрице, проиллюстрировать ее корректную работу на различных по типу наборах данных (не менее трех: int, char и др.).

**6.3 Ход работы**

6.3.1 В программе были создан шаблон класса Matrix полями которого являютя размерность матрицы и непосредственно матрица. Созданы конструкторы и деструктор. Среди конструкторов можно выделить специализацию для типов double и char. Реализованы функции демонстрации матрицы и непосредственно требуемая функция переставляющая строки в матрице. В функции main проведены действия с тремя объектами разных типов этого класса, продемонстрирована работа шаблонов.

6.3.2 Написана программа на С++ согласно вышеописанного алгоритма.

#include <iostream>

#include <windows.h>

#include <ctime>

// Класс - матрица из целых числел

template <class T>

class Matrix {

private:

int numRows;

int numColumns;

T \*\*matrix;

public:

Matrix();

Matrix(int \_numRows, int \_numColumns);

~Matrix();

void showMatrix();

void swapStrings();

};

// Конструктор по умолчанию класса Matrix

template <typename T>

Matrix<T> :: Matrix() {

numRows = 5;

numColumns = 5;

//выделение памяти под массив указателей на массивы

matrix = new T\* [numRows];

// выделение памяти под массивы

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

matrix[i] = new T [numColumns];

}

// заполняем массив

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

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

matrix[i][j] = rand();

}

}

}

// Конструктор по умолчанию класса Matrix

// СПЕЦИАЛИЗАЦИЯ КОНСТРУКТОРА ДЛЯ ТИПА DOUBLE

template <>

Matrix<double> :: Matrix() {

numRows = 5;

numColumns = 5;

//выделение памяти под массив указателей на массивы

matrix = new double\* [numRows];

// выделение памяти под массивы

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

matrix[i] = new double [numColumns];

}

// заполняем массив

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

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

matrix[i][j] = rand() \* 1.0 / rand();

}

}

}

// Конструктор по умолчанию класса Matrix

// СПЕЦИАЛИЗАЦИЯ КОНСТРУКТОРА ДЛЯ ТИПА CHAR

template <>

Matrix<char> :: Matrix() {

numRows = 5;

numColumns = 5;

//выделение памяти под массив указателей на массивы

matrix = new char\* [numRows];

// выделение памяти под массивы

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

matrix[i] = new char [numColumns];

}

// заполняем массив

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

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

matrix[i][j] = rand() % 64 + 192; //рандомные нормальные символы

}

}

}

// Конструтор с параметрами (кол-во строк, столбцов матрицы) класса Matrix

template <class T>

Matrix<T> :: Matrix(int \_numRows, int \_numColumns) {

numRows = \_numRows;

numColumns = \_numColumns;

// выделяем память

matrix = new T\* [numRows];

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

matrix[i] = new T [numColumns];

}

// заполняем массив ручками

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

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

std::cout << "Введите [" << i+1 << "][" << j+1 << "] элемент матрицы: ";

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

}

}

std::cout << std::endl;

}

// Деструктор класса Matrix

template <class T>

Matrix<T> :: ~Matrix() {

// очистить память выделенную матрице

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

delete [] matrix[i];

}

delete [] matrix;

}

// Показать матрицу класса Matrix

template <class T>

void Matrix<T> :: showMatrix() {

std::cout << "Матрица:" << std::endl;

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

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

std::cout.setf(std::ios::left);

std::cout.width(15);

std::cout << matrix[i][j];

}

std::cout << std::endl << std::endl;

}

}

// Функция шаблон меняющая строки в матрице объекта

template <typename T>

void Matrix<T> :: swapStrings() {

if (this->numRows <= 1) {

std::cout << "В матрице мало строк для обмена ими\a" << std::endl;

system("pause");

system("cls");

return;

}

int numStr1, numStr2;

std::cout << "В матрице " << this->numRows << " строк. Какие строки меняем?" << std::endl;

std::cout << "Введите номера меняемых строк: " << std::endl;

std::cout << "-> ";

std::cin >> numStr1;

std::cout << "-> ";

std::cin >> numStr2;

if ((numStr1 > this->numRows) || (numStr2 > this->numRows)){

std::cout << "ОШИБКА: Вы ввели слишком большие номера строк!\a" << std::endl;

system("pause");

system("cls");

return;

}

T temp;

for(int i = 0; i < this->numColumns; i++) {

temp = this->matrix[numStr1 - 1][i];

this->matrix[numStr1 - 1][i] = this->matrix[numStr2 - 1][i];

this->matrix[numStr2 - 1][i] = temp;

}

}

int main(int argc, char const \*argv[])

{

system("color B0");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand((unsigned int)time(NULL));

char ch;

while(1){

std::cout << "Исходная матрица с элементами типа int: " << std::endl;

Matrix<int> \*obj1 = new Matrix<int>();

obj1->showMatrix();

obj1->swapStrings();

std::cout << "измененная матрица с элементами типа int: " << std::endl;

obj1->showMatrix();

std::cout << std::endl;

system("pause");

delete obj1;

system("cls");

std::cout << "Исходная матрица с элементами типа double: " << std::endl;

Matrix<double> \*obj2 = new Matrix<double>();

obj2->showMatrix();

obj2->swapStrings();

std::cout << "измененная матрица с элементами типа double: " << std::endl;

obj2->showMatrix();

std::cout << std::endl;

system("pause");

delete obj2;

system("cls");

std::cout << "Исходная матрица с элементами типа char: " << std::endl;

Matrix<char> \*obj3 = new Matrix<char>();

obj3->showMatrix();

obj3->swapStrings();

std::cout << "измененная матрица с элементами типа char: " << std::endl;

obj3->showMatrix();

std::cout << std::endl;

system("pause");

delete obj3;

system("cls");

std::cout << "Выйти? (y/n) >> ";

std::cin >> ch;

if (ch == 'y') break;

system("cls");

}

return 0;

}

5.3.3 Выполнена отладка программы.

Результаты тестирования отображены на рисунках 6.1– 6.3. На изображениях изображено как генерирутся таблицы с разными типами элементов, а именно с типом integer, double и char. Затем у пользователя требуется ввести 2 номера строк которые будут поменяны местами, далее демонстрируется измененная матрица. В конце спрашивается осуществить ли действия ещё раз или выйти.

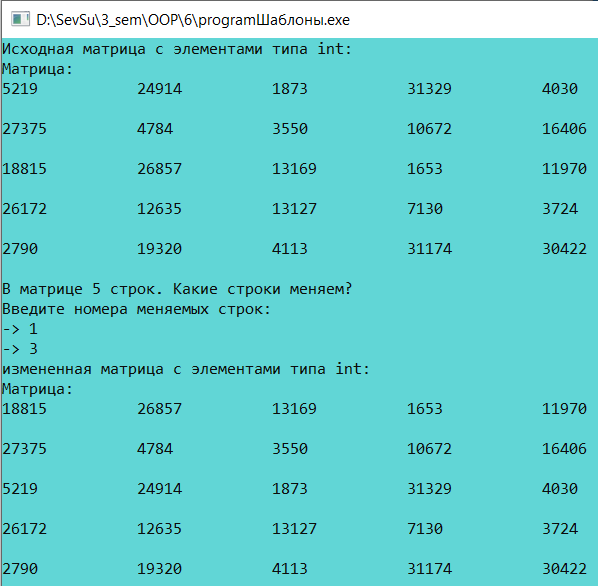


Рисунок 6.1 – Выполнение программы с элементами матрицы типа integer

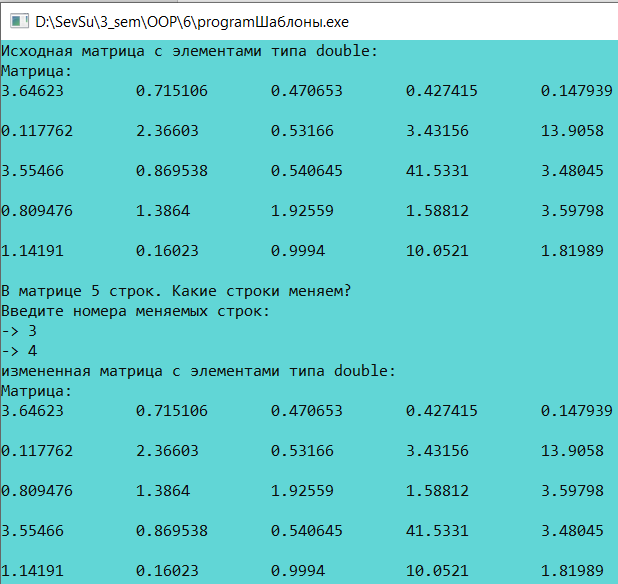


Рисунок 6.2 – Выполнение программы с элементами матрицы типа double

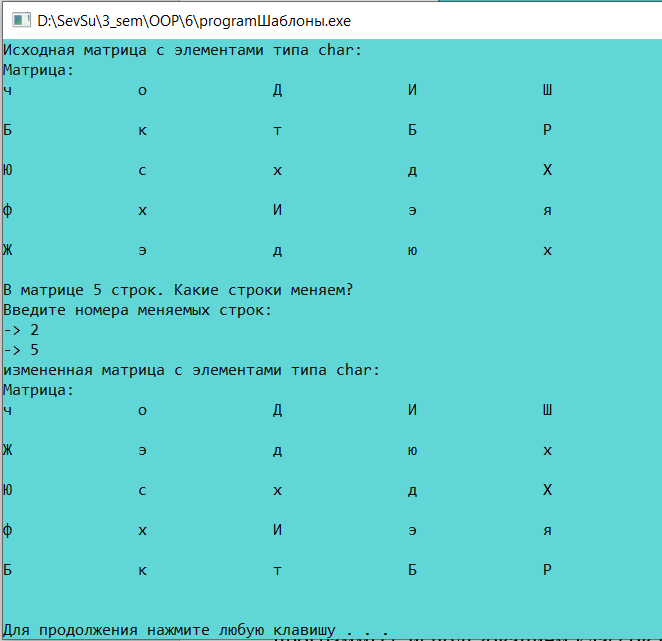


Рисунок 6.3 – Выполнение программы с элементами матрицы типа char

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были получены навыки разработки программ, использующих шаблоны классов и функций. Были закреплены навыки разработки и отладки программ, использующих объекты. Полученные во время разработки навыки помогут разрабатывать более сложные программы с использованием классов и объектов, более эффективные по времени выполнения алгоритмы.