ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение учебной практики обучающегося направления

подготовки/специальности 09.03.01

1. Фамилия, имя, отчество обучающегося: Сыворотнев Владислав Сергеевич

1. Группа: 4141
2. Тема индивидуального задания: изучение методов сортировки структур данных;

совершенствование навыков процедурного программирования на языке *C*/*C*++

при решении задач обработки статических и динамических массивов; разработка

пользовательских интерфейсов.

1. Исходные данные:

- разработать математическую модель описания поставленной задачи и структурировать её с целью достижения требуемого решения;

- используя технологию процедурного программирования реализовать заданный

метод сортировки и применить его для указанных фрагментов числовой матрицы

в соответствии с индивидуальным заданием;

- разработать пользовательский интерфейс, позволяющий одновременно наблюдать

на экране исходную и результирующую матрицу.

1. Содержание отчетной документации:
   1. индивидуальное задание;
   2. отчёт, включающий в себя:
   * титульный лист (отчет по практике);
   * материалы о выполнении индивидуального задания;
   * выводы по результатам практики;
   * список использованных источников.
   1. отзыв руководителя от профильной организации (при прохождении практики в профильной организации).
2. Срок представления отчета на кафедру: «\_30\_»\_\_\_\_\_05\_\_\_\_\_\_\_\_2022 г.

Руководитель практики

доцент, канд. техн. наук, доцент Л.Н. Бариков

должность, уч. степень, звание подпись, дата инициалы, фамилия

Задание принял к исполнению: дата 09.02.2022

Обучающийся

В.С. Сыворотнев

дата подпись инициалы, фамилия

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

Кафедра 44 Вычислительных систем и сетей

(наименование)

ОТЧЁТ ПО ПРАКТИКЕ

ЗАЩИЩЁН С ОЦЕНКОЙ

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, канд. техн. наук, доцент |  |  |  | Л.Н. Бариков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | учебная | |
| тип практики | вычислительная | |
| на тему индивидуального задания | | изучение методов сортировки структур данных; |
| совершенствование навыков процедурного программирования на языке *C*/*C*++ | | | |
| при решении задач обработки статических и динамических массивов. | | | |

|  |  |
| --- | --- |
| выполнен | Сыворотневым Владиславом Сергеевичем |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.01 |  | Информатика и вычислительная техника |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 02 |  | Вычислительные машины, комплексы, |
|  | код |  | наименование направленности |
| системы и сети | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 4141 |  |  |  | В.С. Сыворотнев |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2022

**Индивидуальные задания на программирование**

***Цель работы:*** *изучение методов сортировки структур данных; совершенствование навыков процедурного программирования на языке C/C++ при решении задач обработки статических и динамических массивов. Разработка пользовательского интерфейса.*

***Задание на программирование:*** *используя технологию процедурного программирования реализовать заданный метод сортировки и применить его для указанных фрагментов числовой матрицы в соответствии с индивидуальным заданием. Разработать пользовательский интерфейс, позволяющий:*

*- осуществлять ввод исходных данных;*

*- наблюдать на экране монитора одновременно исходную и отсортированную матрицу.*

***Порядок выполнения работы:***

1. Получить у преподавателя индивидуальное задание: метод сортировки и вид сортируемых фрагментов матрицы. В зависимости от задания стрелка на рисунке указывает направление возрастания или убывания значений элементов фрагментов матрицы после сортировки.

2. Исходная матрица размера 2*n*×2*n* (для вариантов 33-40 размер матрицы (2*n*+1)×(2*n*+1)), не должна содержать элементов с одинаковыми значениями.

3. Разработать математическую модель: описать с помощью формул и рисунков структуру массива и процесс его преобразования. У результирующей матрицы должны быть отсортированы значения элементов заданных фрагментов и обнулены все остальные.

4. Провести структуризацию задачи. С этой целью выделить подзадачи, которые будут реализованы в виде отдельных функций (ввод исходных данных, вывод содержимого массива, обработка массива в соответствии с заданием и т.д.). При этом запрещается совмещать в одной функции решение нескольких подзадач.

5. Обязательное требование – введение собственных типов данных, логически точно разделяющих и группирующих информацию, используемую при решении задачи (тип значения элемента массива, тип массива и т.п.).

6. Построить схему алгоритма функции сортировки.

7. Составить программу на языке C/C++, реализовав при этом функции ввода исходной матрицы, сортировки значений элементов заданных фрагментов, обнуления значений элементов не сортируемых фрагментов и вывода значений элементов матрицы,

8. Входные данные на этапах тестирования и демонстрации работы преподавателю должны задаваться либо с использованием специально подобранных арифметических формул, либо вводиться с клавиатуры по запросу. **Датчики псевдослучайных чисел использовать запрещается**.

9. Выходные данные должны выводиться на экран с пояснениями. Вывод исходной и преобразованной матриц должен быть реализован в виде отдельной функции, вызов которой осуществляется из функции main().

10. Проверить и продемонстрировать преподавателю работу программы на полном наборе тестов. Входные и выходные массивы должны выводиться в одном и том же формате.

11. Использовать стандартные потоковые объекты ввода/вывода cin и cout.

12. Оформить отчет по практике в составе: титульный лист, постановка задачи, схема алгоритма функции сортировки, описание процесса решения задачи, текст программы, контрольные примеры (скриншоты), выводы по результатам практики, список использованных источников.

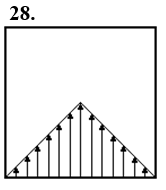
13. Текст программы в отчете не должен представлять из себя скриншот.

14. Скриншоты тестов должны легко читаться. Все их неинформативные части должны быть удалены.

***Вариант № 7/28:***

***Метод сортировки***

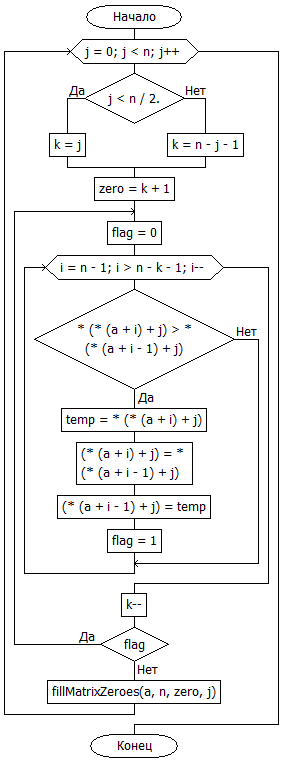
1. *Сортировка по возрастанию методом обмена с флагом перестановки.*



***Математическая модель решения***

Сортировка матрицы происходит методом обмена с флагом перестановки. Суть метода заключается в сравнении соседних элементов на каждой итерации, и если порядок в паре неверный, то элементы меняются местами. В нашем случае сравниваются верхний и нижний элементы матрицы, сравнение происходит по столбцам снизу вверх. Также в алгоритме предусмотрен флаг, который является индикатором того, что матрица еще не отсортирована по умолчанию. Если все элементы матрицы находятся на своих местах, флаг будет изменен, и выполнение алгоритма сортировки будет прекращено.

***Схема алгоритма сортировки матрицы***



***Текст файла main.cpp:***

#include "mainwindow.h"

#include<iostream>

#include <QApplication>

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

{

QApplication a(*argc*, *argv*);

MainWindow w;

w.show();

*return* a.exec();

}

***Текст файла mainwindow.h:***

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **MainWindow**; }

QT\_END\_NAMESPACE

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_pushButton\_Create\_clicked**();

void **on\_AutomaticInputButton\_clicked**();

void **on\_sortButton\_clicked**();

void **on\_clearButton\_clicked**();

void **on\_info\_triggered**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

***Текст файла mainwindow.cpp:***

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QTableWidgetItem>

#include <QMessageBox>

*typedef* int telem; */\*определение* *типа* *значений* *элементов* *массива\*/*

*typedef* telem\* tstr; */\*определение* *типа* *"указатель* *на* *telem"\*/*

*typedef* tstr\* tmatr; */\*определение* *типа* *"указатель* *на* *указатель* *на* *telem"\*/*

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**on\_pushButton\_Create\_clicked**()

{

int spinBoxValue = ui->spinBox->value();

int n = spinBoxValue \* 2;

ui->tableWidget->setRowCount(n);

ui->tableWidget->setColumnCount(n);

ui->tableWidget\_2->setRowCount(n);

ui->tableWidget\_2->setColumnCount(n);

}

void MainWindow::**on\_AutomaticInputButton\_clicked**()

{

*for* (int i = 0; i < ui->tableWidget->rowCount(); i++)

*for* (int j = 0; j < ui->tableWidget->columnCount(); j++)

{

QTableWidgetItem \*itm = *new* QTableWidgetItem(tr("%1").arg(10 \* i + j));

ui->tableWidget\_2->setItem(i, j, *itm*);

}

}

void MainWindow::**on\_sortButton\_clicked**()

{

int j, i, k, cItem, pItem;

int zero;

int temp;

bool flag;

int n = ui->tableWidget\_2->columnCount();

bool error = *false*;

*//* *копирование* *матрицы*

*for* (int i = 0; i < ui->tableWidget\_2->rowCount(); i++)

{

*if* (!error)

{

*for* (int j = 0; j < ui->tableWidget\_2->columnCount(); j++)

{

QTableWidgetItem \*fItm = ui->tableWidget\_2->item(i, j);

*if* (fItm == *nullptr* || !fItm || fItm->text().isEmpty()) {

QMessageBox::critical(*this*, "Ошибка", "Исходная матрица заполнена не полностью!");

error = *true*;

*break*;

}

QTableWidgetItem \*sItm = *new* QTableWidgetItem(tr("%1").arg(fItm->text().toInt()));

ui->tableWidget->setItem(i, j, *sItm*);

}

}

}

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

*/\*начальное* *количество* *не* *сортированных* *элементов* *строки\*/*

*if* (j < n / 2.) {

k = j;

} *else* {

k = n - j - 1;

}

*//запоминаем* *изначальное* *количество* *неотсортированных* *элементов*

zero = k + 1;

*do* {

flag = *false*;

*for* (i = n - 1; i > n - k - 1; i--) {

*if* (error){

*break*;

}

QTableWidgetItem \*itm1 = ui->tableWidget->item(i, j);

QTableWidgetItem \*itm2 = ui->tableWidget->item(i - 1, j);

cItem = itm1->text().toInt();

pItem = itm2->text().toInt();

*if* (cItem > pItem) {

temp = cItem;

cItem = pItem;

pItem = temp;

QTableWidgetItem \*itm3 = *new* QTableWidgetItem(tr("%1").arg(cItem));

QTableWidgetItem \*itm4 = *new* QTableWidgetItem(tr("%1").arg(pItem));

ui->tableWidget->setItem(i, j, *itm3*);

ui->tableWidget->setItem(i - 1, j, *itm4*);

flag = *true*;

}

}

k--;

} *while* (flag);

*for* (i = 0; i < n - zero; i++) {

*if* (error){

ui->tableWidget->clearContents();

*break*;

}

QTableWidgetItem \*itm3 = *new* QTableWidgetItem(tr("%1").arg(0));

ui->tableWidget->setItem(i, j, *itm3*);

}

}

}

void MainWindow::**on\_clearButton\_clicked**()

{

ui->tableWidget\_2->clearContents();

ui->tableWidget->clearContents();

}

void MainWindow::**on\_info\_triggered**()

{

QMessageBox::information(*this*, "Информация",

"Данная программа предназначена для сортировки треугольной области"

" матрицы методом обмена с флагом перестановки. \n"

"\nДля выбора размера матрицы введите в специальном поле размер n."

" (матрица размера 2\*n) \n"

"\nДля ускоренного заполнения элементов матрицы создан режим автоматического"

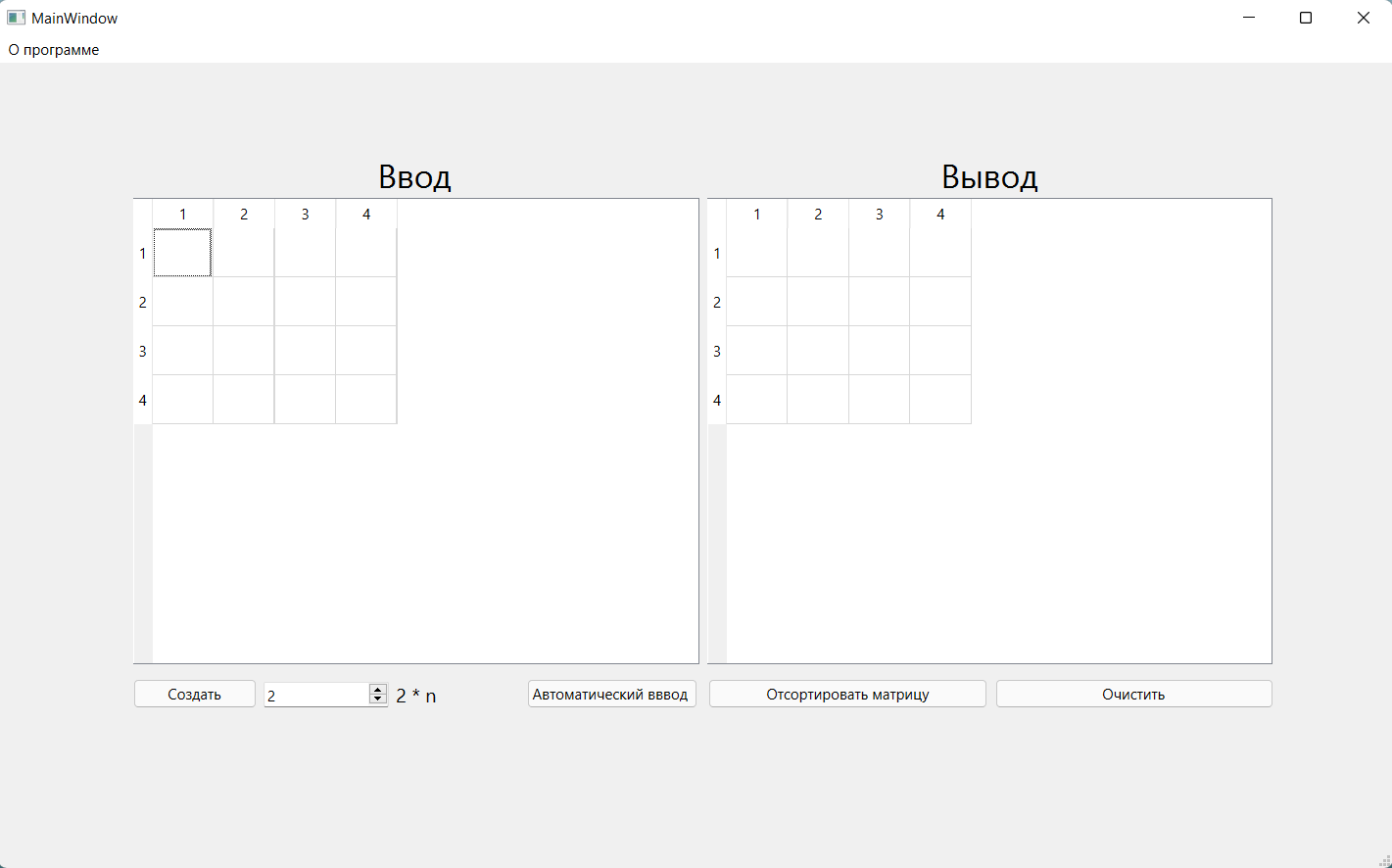
" ввода, достаточно просто нажать на кнопку. \n"

"\nПосле выбора размера матрицы и ввода ее элементов можно нажать на кнопку"

" сортировки, чтобы получить результат.");

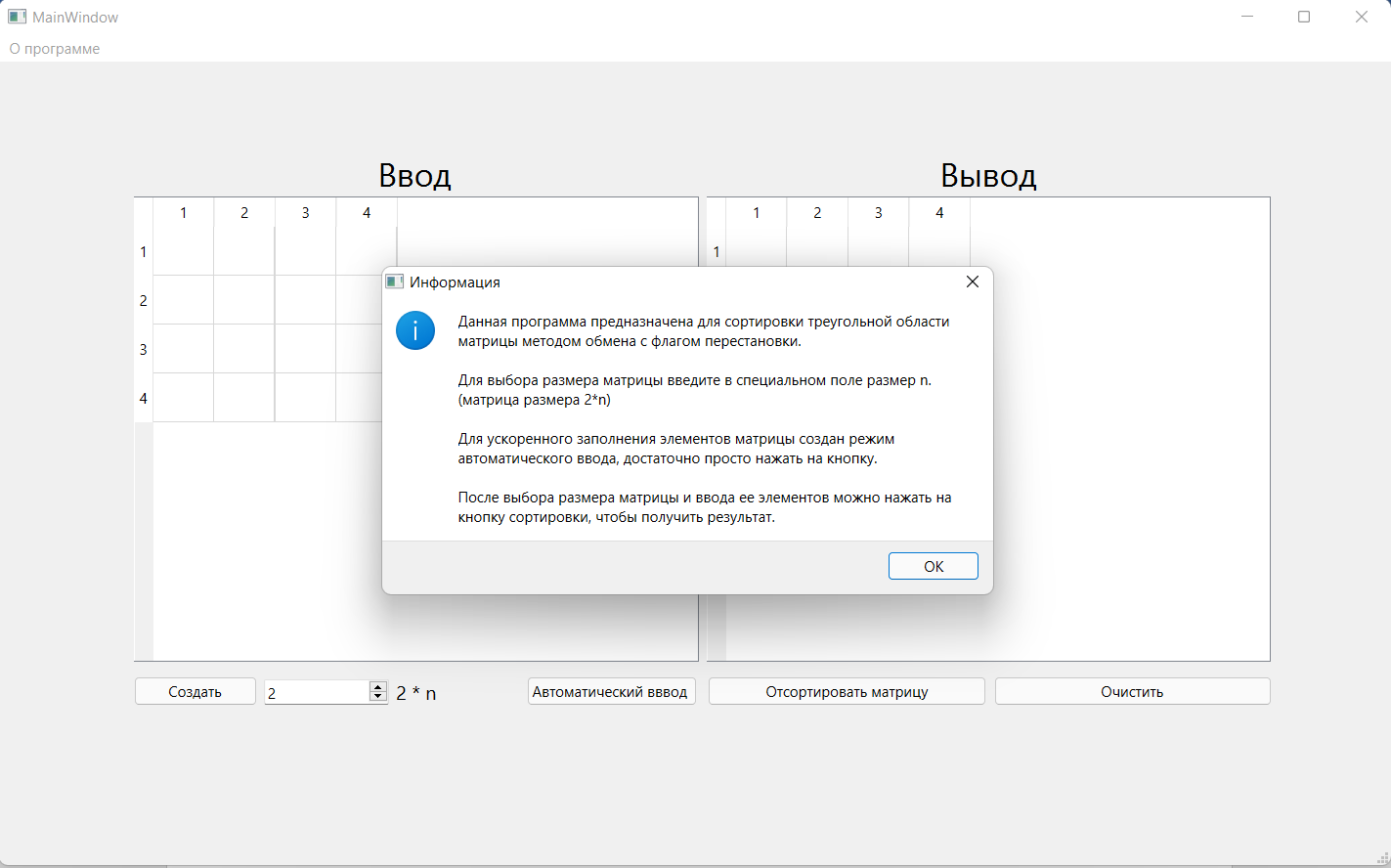
}

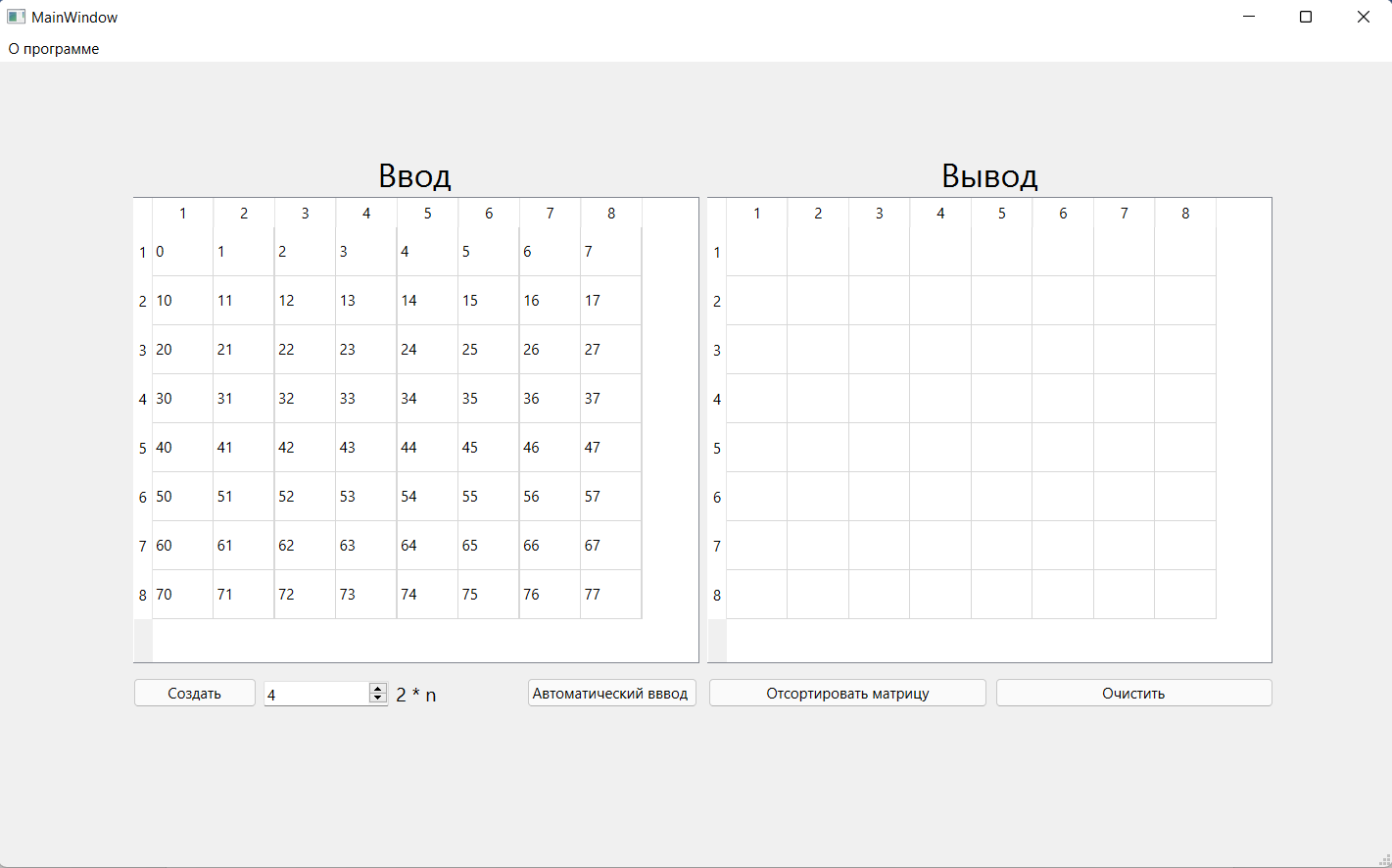
***Контрольные примеры:***

******

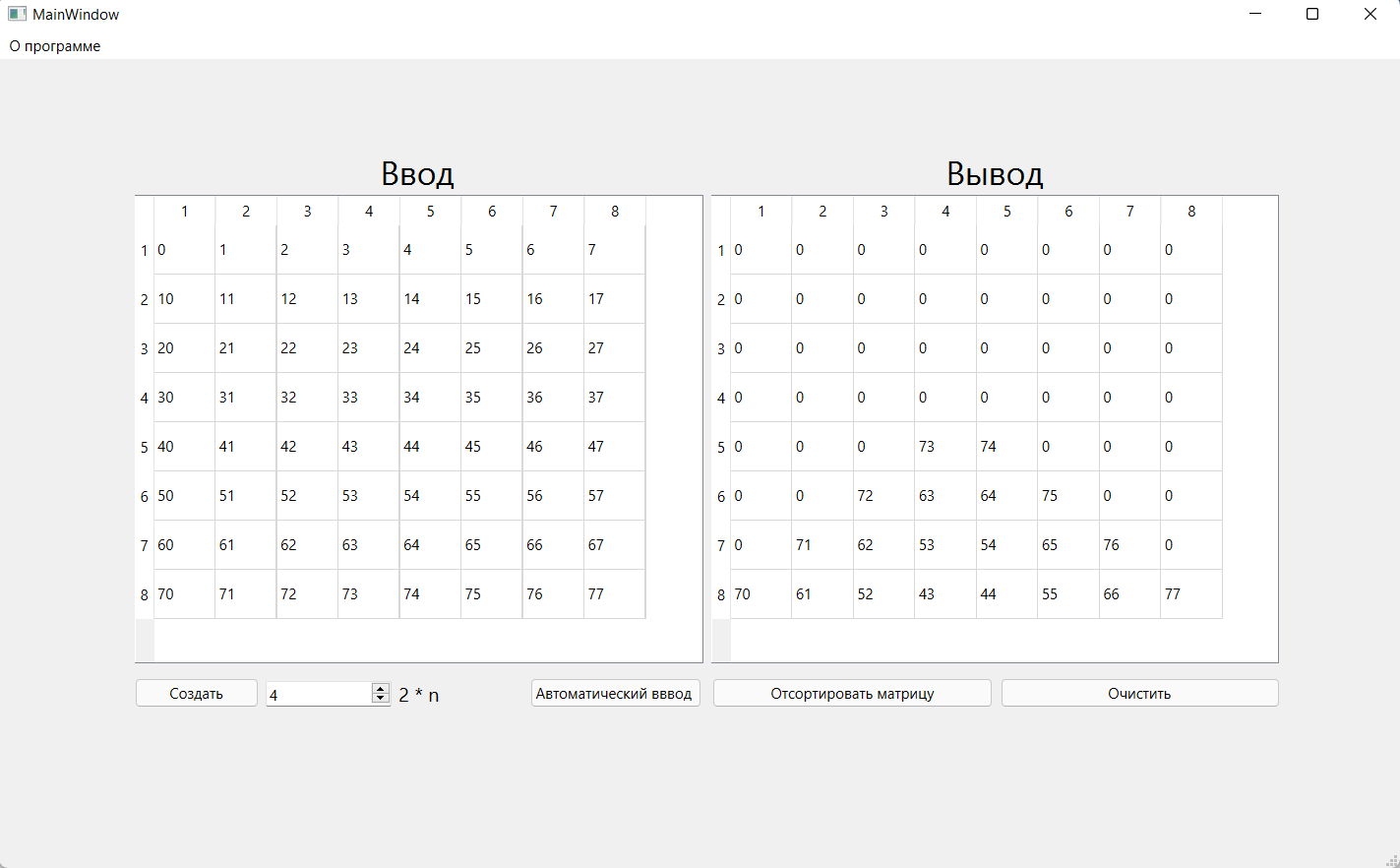
После запуска программы сразу становится доступным ввод значений матрицы в таблицу. По умолчанию матрица размера 4\*4(2\*n).   
С помощью специального поля можно поменять размерность матрицы, задав другое значение n, затем нажать кнопку «Создать» и размерность матрицы изменится.

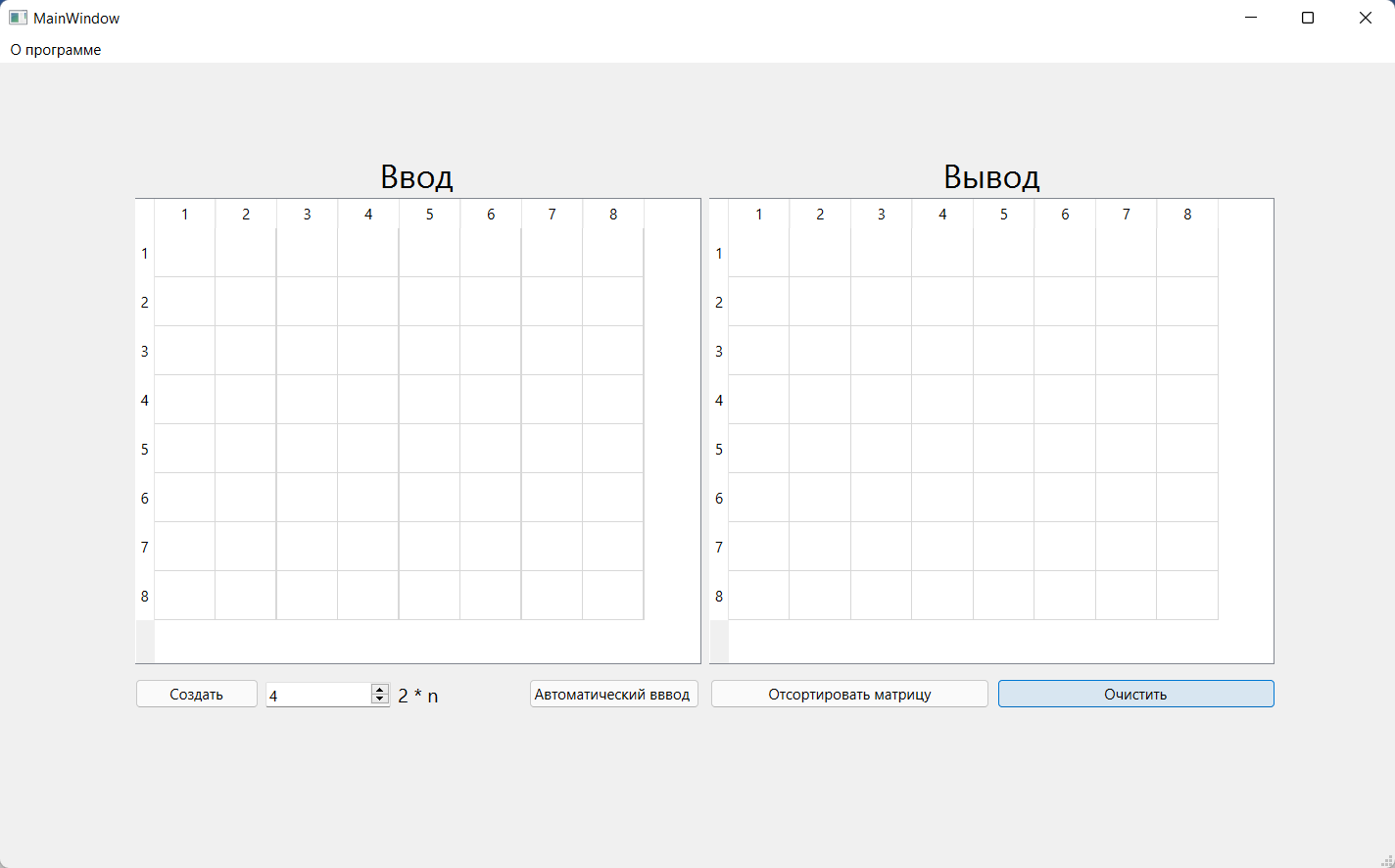
В левом верхнем углу программы можно получить информацию о работе программы.



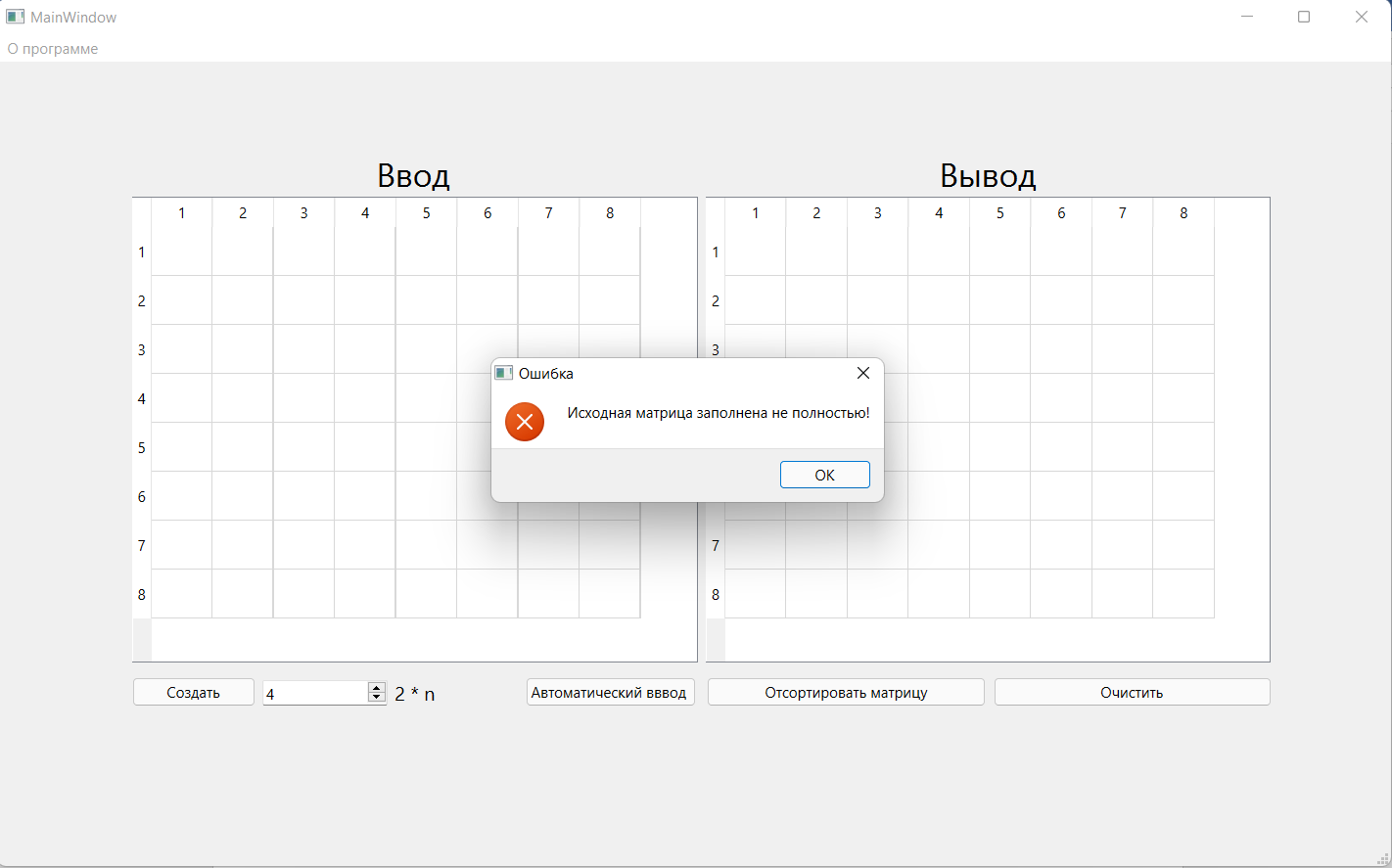
Кнопка «Автоматический ввод» заполняет матрицу автоматически по формуле (10 \* i + j)

После нажатия кнопки «Отсортировать матрицу» введенная матрица будет отсортирована.



Кнопка «Очистить» удаляет все введенные значения 

Если попытаться отсортировать пустую матрицу, программа выведет ошибку.



**Выводы по результатам практики**

В процессе выполнения практической работы были изучены методы сортировки двумерных массивов, усовершенствованы навыки процедурного программирования на языке C/C++, разработан пользовательский интерфейс.

Была разработана математическая модель решения задачи, которая помогла структурировать задачу.

Используя технологию процедурного программирования, была реализована сортировка по возрастанию методом обмена с флагом перестановки. Был разработан пользовательский интерфейс, позволяющий взаимодействовать с программой через оконное приложение.

**Список использованных источников**

1. Документация фрейморка Qt для разработки интерфейса -https://doc.qt.io/qt.html
2. Документация языка c++ - https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170