МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Очереди и стеки

Студент гр. 8381 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными характеристиками и особенностями типов данных стек и очередь, изучить особенности их реализации на языке программирования C++. Разработать программу, использующую иерархические списки и их рекурсивную обработку, высчитывающую значение выражения.

Задание.

«Болты и гайки». Имеется куча перемешанных n болтов и n гаек, отличающихся диаметрами. Нужно быстро найти все соответствующие пары болтов и гаек. Известно, что каждая гайка подходит по диаметру ровно к одному болту, и наоборот. Нет ни двух болтов одинакового диаметра, ни двух гаек одинакового диаметра.

Попробовав навинтить гайку на болт, можно определить, что из них больше (или они соответствуют друг другу), но невозможно (а в алгоритме – нельзя) непосредственно сравнить два болта или две гайки.

Простой алгоритм – $O(n^2)$. Это не быстро. Предложить аналог рандомизированной быстрой сортировки – $O(n \log n)$.

Основные теоретические положения.

Одной из быстрых сортировок, имеющих среднюю сложность $O(n \log n)$, является алгоритм быстрой сортировки QuickSort.

Общая идея алгоритма состоит в следующем:

- Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
- Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».

- Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

Выполнение работы.

Написание работы производилось на базе операционной системы Lubuntu 18.10 в среде разработки QtCreator с использованием фреймворка Qt. Сборка, отладка производились в QtCreator, запуск программы осуществлялся через консоль.

Для реализации программы был разработан графический интерфейс с помощью встроенного в QtCreator UI-редактора. Он представляет из себя два поля ввода для массивов болтов и гаек, кнопку считывания и переноса в поле ввода информации из файла, флаги записи дополнительной информации в вывод и пошагового режима, поле вывода, а также кнопку выгрузки в файл информации, содержащейся в поле вывода (см. приложения А-К). Основные методы для работы графического интерфейса приведены в табл. 1.

Таблица 1 – Методы класса MainWindow и их назначение

Метод	Назначение
`void getInfo(string path)`	Считывает из файла по заданному пути строки массивов болтов и гаек и устанавливает их в поля ввода
`void getOutput()`	Переносит всю информацию из файла `output.txt` в поле вывода
`void on_openFile_clicked()`	Слот, отвечающий за считывание из файла
`void on_saveFile_clicked()`	Слот, отвечающий за запись из поля вывода в файл
`void on_start_clicked()`	Слот, отвечающий за запуск алгоритма нахождения пар, за вывод информации об ошибках и результате

Описание алгоритма нахождения пар

Алгоритм представляет собой сортировку двух массивов таким образом, чтобы массивы были одинаковы, то есть элементы с одинаковым индексом образовывали пару в условиях задачи. Алгоритм основан на базе быстрой сортировки QuickSort. В первом массиве фиксируется опорный элемент, который сравнивается с элементами другого массива. В итоге массив разделяется на элементы, меньшие опорного, большие опорного, а равный опорному помещается в конце массива. Далее проводится аналогичная операция, но опорный элемент берется последним из второго массива, то есть равный опорному элементу из первого, и сравнивается с элементами первого массива. В итоге оба массива поделены на пару найденных равных элементов, на элементы, меньшие этой пары, и большие. Алгоритм рекурсивно запускается для последних двух частей массива.

В алгоритме предусмотрен вывод промежуточных результатов, в том числе пошаговый режим, при котором выводится информация о всех сравнениях и перестановках, совершенных алгоритмом. Полный список функций, реализующих визуализацию алгоритма и некоторые его функции, приведен в табл. 2.

Таблица 2 – Вспомогательные функции для алгоритма и визуализации

Функция	Назначение
`int strCount(string &in)`	Считает количество чисел, разделенных пробелом в строке, а также проверяет корректность строки
`int checkStr(string &bolts, string &nuts)`	Проверяет совпадение строк по количеству чисел в них, а также обрабатывает ошибки на корректность строк, возвращает кол-во чисел
`dArr makeArray (string &in, int l)`	Создает динамический массив чисел из строки
`string getStrFromArray (dArr &input)`	Возвращает строку, состоящую из элементов массива, разделенных пробелом

Окончание таблицы 2

`string getStrCmpAction (dArr &bolts, int ptr1, dArr &nuts, int ptr2)`	Возвращает строку, состоящую из двух массивов, в которых два элемента выделены фигурными скобками.
`string getStrSwapAction (dArr &input, int ptr1, int ptr2)`	Возвращает строку, состоящую из элементов массива, среди которых два элемента выделены фигурными скобками
`int checkPairs (dArr &bolts, dArr &nuts)`	Проверяет, совпадают ли в отсортированных массивах элементы с одинаковыми индексами, то есть проверяет пары
`string getPairs (dArr &bolts, dArr &nuts)`	Возвращает строку, в которой содержатся все пары в формате (b_i, n_i), где b_i и n_i – элементы двух массивов с индексом i

Программа имеет возможность запуска в консольном режиме, для чего при запуске из терминала необходимо указать аргумент `console`. Результат выполнения программы записывается в файл `output.txt` и выводится в терминал.

Оценка сложности алгоритма

В лучшем случае, когда при каждом разбиении оба массива будут разбиваться пополам, глубина рекурсии $N \approx \log_2 n$, где n – длина массивов. Во время работы функция проводит $K_i = 2n + 2K_{i+1}$ сравнений: сначала опорный элемент одного массива с элементами другого, затем наоборот, после чего функция рекурсивно вызывается для двух сегментов массивов. Тогда сложность алгоритма составит $O(n * \log_2 n)$.

В среднем случае, когда разбиения массивов будут неравноценными, глубина рекурсии $N \approx \log_c n$, где c – некоторая константа. В конечном счете, это

приводит к сложности алгоритма $O(n * \log n)$. Так как в функции проводится $2n$ сравнений, а также рекурсивно функция вызывается два раза, то фактически число итераций можно оценить как $4n * \log n$.

В худшем случае, когда разбиения массивов будет таковым, что все элементы либо меньше, либо больше опорного, глубина рекурсии будет $N \approx n$. В таком случае, сложность алгоритма возрастает до $O(n^2)$.

Рост занимаемой памяти напрямую зависит от глубины рекурсии, при этом в функции не создаются копии массивов, соответственно, сложность алгоритма по памяти в среднем случае $O(log\ n)$, в худшем - $O(n)$.

Был проведен ряд тестов программы с фиксацией числа итераций во время работы алгоритма. Входные данные были таковы, чтобы избежать худшего случая. Для оценки теоретического числа итераций N для n элементов в массиве использовалась функция $f(n) = 4n * \log n$. График зависимости числа итераций N от количества элементов n и график функции $f(n)$ изображены на рис. 1.

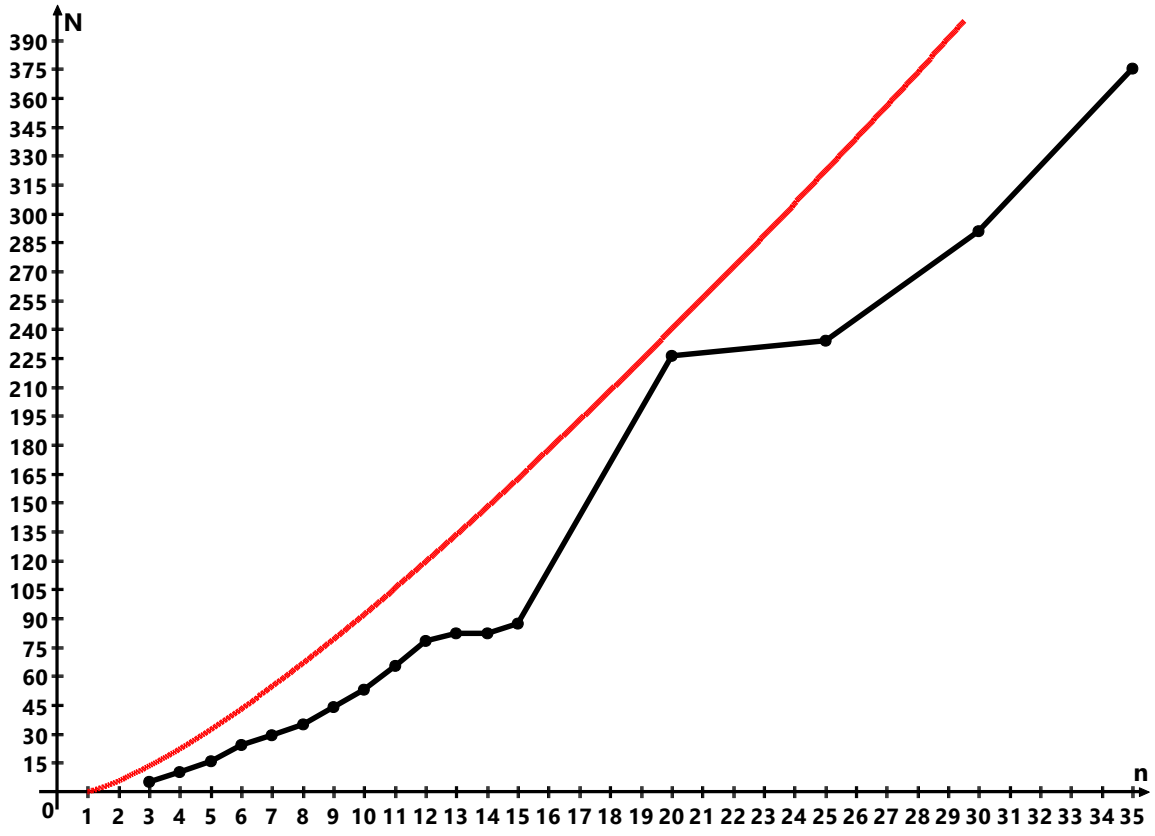

Рисунок 1 – График зависимости числа итераций от длины массивов

Из графиков видно, что практическое число итераций N растет так же, как функция $f(n) = 4n * \log n$. График числа итераций имеет скачкообразную форму, связанную с тем, что алгоритм имеет нестабильную сложность.

Тестирование программы.

Вид программы после выполнения представлен на рис. 2.

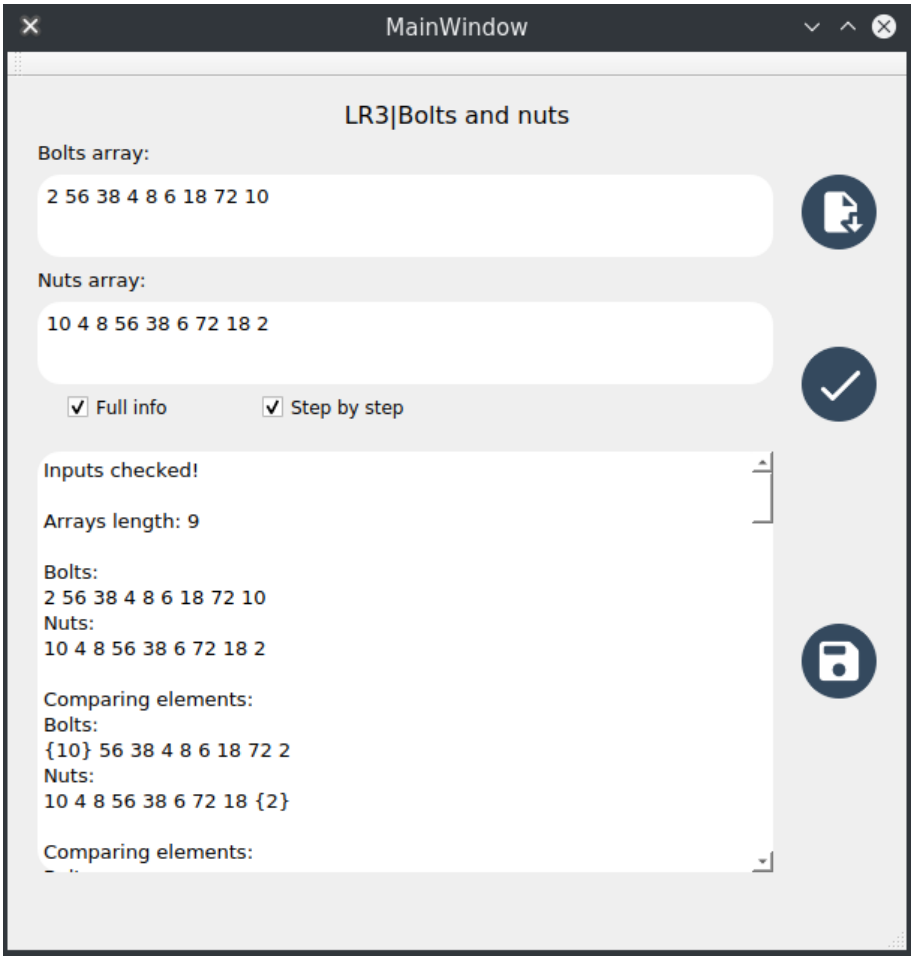

Рисунок 2 – Графический интерфейс программы

Вид диалогового окна пошагового режима представлен на рис. 3.

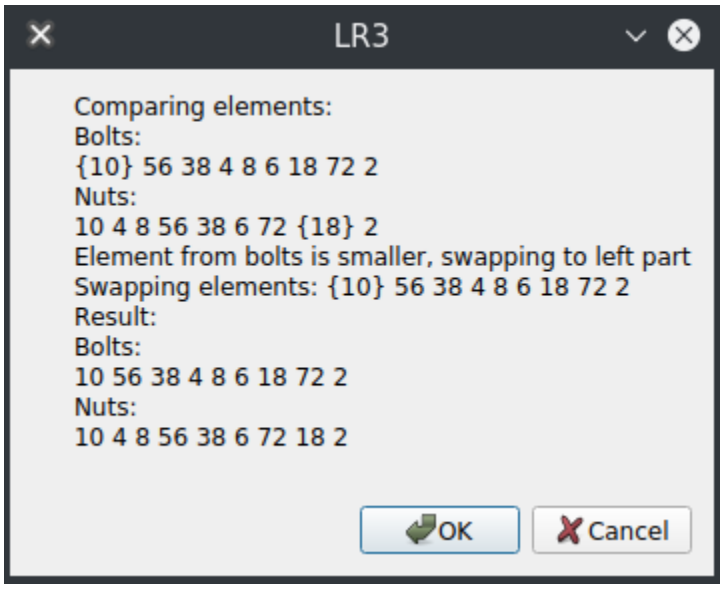

Рисунок 3 – Окно пошагового режима

Было проведено тестирование программы, которое приведено в табл. 3.

Таблица 3 – Тестирование программы

Входные массивы	Результат выполнения (сокращ.)
Bolts: 2 56 38 4 8 6 18 72 10 Nuts: 10 4 8 56 38 6 72 18 2	Arrays length: 9 Bolts: 4 10 8 6 38 56 72 18 2 Nuts: 4 10 8 6 38 56 72 18 2 Pairs: (4, 4), (10, 10), (8, 8), (6, 6), (38, 38), (56, 56), (72, 72), (18, 18), (2, 2) Recursion deep: 6
Bolts: 2 56 38 4 8 98 18 72 10 Nuts: 10 4 8 56 38 6 72 18 2 (ожидаемая ошибка: не все элементы имеют пару)	Arrays length: 9 There are unpairable bolts and nuts
Bolts: 2 56 38 4 8 6 18 72 Nuts: 10 4 8 56 38 6 72 18 2	Error! Number of bolts does not match number of nuts

(ожидаемая ошибка: не совпадает кол-во элементов)	
Bolts: 2 56 38 4 10 8 6 18 72 10 Nuts: 10 4 8 56 38 6 72 10 18 2 (ожидаемая ошибка: есть элементы одного размера)	Error! Found same bolts Sorting was cancelled
Bolts: 2 56 38 4 -8 6 18 72 10 Nuts: 10 4 8 56 38 6 72 18 2 (ожидаемая ошибка: есть недопустимые знаки в вводе)	Error! Bolts input contains not only digits
Bolts: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 Nuts: 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 (ожидается худший случай – большая глубина рекурсии)	Arrays length: 17 Bolts: 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 Nuts: 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 Pairs: (17, 17), (16, 16), (15, 15), (14, 14), (13, 13), (12, 12), (11, 11), (10, 10), (9, 9), (8, 8), (7, 7), (6, 6), (5, 5), (4, 4), (3, 3), (2, 2), (1, 1) Recursion deep: 16

Выводы.

В ходе выполнения лабораторной работы была написана программа, сортирующая два массива по парам, используя сравнения только элементов разных массивов. Был реализован быстрый алгоритм, основанный на сортировке QuickSort, имеющий среднюю сложность $O(n * \log n)$. Были реализованы пошаговый режим выполнения алгоритма, консольный и графический интерфейсы, обработка ошибок.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAIN.C

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    bool console = false;
    for(int i=1;i<argc;i++)
    {
        if(!strcmp("console",argv[i]))
            console = true;
    }
    if (console)
    {
        startConsole();
    }
    else
    {
        QApplication a(argc, argv);
        MainWindow w;
        w.show();
        return a.exec();
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include "console.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    void getInfo(string filePath);
    void getOutput();
    ~MainWindow();

private slots:
    void on_openFile_clicked();

    void on_start_clicked();

    void on_saveFile_clicked();

    void on_checkInfo_toggled(bool checked);

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.C

```
#include "mainwindow.h"
#include "ui_mainwindow.h"


MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->output->setReadOnly(true); //запрет ввода в область вывода
    ui->name->setAlignment(Qt::AlignCenter);  //выравнивание имени по
центру
    ui->checkStep->setEnabled(false);    //выбор отображения графики
недоступен
}


MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::getInfo(string filePath)
{

    ifstream fin(filePath, ios::in);
    string out;
    getline(fin, out);
    ui->inputB->setText(QString::fromStdString(out));
    getline(fin, out);
    ui->inputN->setText(QString::fromStdString(out));
    fin.close();
}


void MainWindow::getOutput()
{
    ifstream fin("output.txt");
    string output;
    string temp;
```

```
    while (!fin.eof())
    {
        getline(fin, temp);
        output += temp;
        output += "\n";
    }
    fin.close();
    ui->output->setPlainText(QString::fromStdString(output));
}

void MainWindow::on_openFile_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Choose input
file (TXT)"), QDir::homePath(), tr("*.txt"));
    if (QString::compare(fileName, QString()) != 0)
    {
        getInfo(qPrintable(fileName));
    }
}

void MainWindow::on_saveFile_clicked()
{
    QString filePath = QFileDialog::getSaveFileName(this, tr("Save to TXT
file"), QDir::homePath(), tr("*.txt"));
    if (QString::compare(filePath, QString()) != 0)
    {
        ofstream fout(qPrintable(filePath));
        fout << qPrintable(ui->output->toPlainText());
        fout.close();
    }
}

void MainWindow::on_checkInfo_toggled(bool checked)
{
    if (checked)
        ui->checkStep->setEnabled(true);
    else
    {
        ui->checkStep->setChecked(false);
        ui->checkStep->setEnabled(false);
    }
}
```

```
void MainWindow::on_start_clicked()
{
    string bolts = qPrintable(ui->inputB->toPlainText());
    string nuts = qPrintable(ui->inputN->toPlainText());
    flags opt;
    opt.info = ui->checkInfo->isChecked();
    opt.step = ui->checkStep->isChecked();
    opt.console = false;
    ofstream fout("output.txt");
    int num = checkStr(bolts, nuts);
    switch (num) {
    case 0:
        fout << "Error! Inputs are empty\n";
        break;
    case ERR_SYMB_B:
        fout << "Error! Bolts input contains not only digits\n";
        break;
    case ERR_SYMB_N:
        fout << "Error! Nuts input contains not only digits\n";
        break;
    case ERR_LENGTH:
        fout << "Error! Number of bolts does not match number of nuts\n";
        break;
    default:
        fout << "Inputs checked!\n\nArrays length: ";
        fout << to_string(num);
        fout << "\n\n";
        dArr boltsArr = makeArray(bolts, num);
        dArr nutsArr = makeArray(nuts, num);
        fout << "Bolts:\n";
        fout << getStrFromArray(boltsArr);
        fout << "Nuts:\n";
        fout << getStrFromArray(nutsArr);
        fout << "\n";
        int n = 0;
        if (!sortBN(boltsArr, nutsArr, opt, fout, n))
        {
            if (!checkPairs(boltsArr, nutsArr))
                fout << "There are unpairable bolts and nuts\n";
            else
            {
                fout << "Sorting is finished\nBolts:\n";
                fout << getStrFromArray(boltsArr);
```

```
                fout << "Nuts:\n";
                fout << getStrFromArray(nutsArr);
                fout << "Pairs of bolts and nuts:\n";
                fout << getPairs(boltsArr, nutsArr);
                fout << "Recursion deep: " << n << endl;
            }
        }
        else
        {
            fout << "Sorting was cancelled\n";
        }
    }
    fout.close();
    getOutput();
}
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ. DARR.H

```
#ifndef DARR_H
#define DARR_H
#include <iostream>
#include <fstream>
#include <sstream>
#include <QMainWindow>
#include <QGraphicsEffect>
#include <QGraphicsView>
#include <QFileDialog>
#include <QStandardPaths>
#include <QtGui>
#include <QLabel>
#include <QColorDialog>
#include <QInputDialog>
#include <QMainWindow>
#include <QPushButton>
#include <QMessageBox>

#define ERR_SYMB_B -1
#define ERR_SYMB_N -2
#define ERR_LENGTH -3

using namespace std;

struct dArr
{
    int length;
    int *base;
};

struct flags
{
    bool step;
    bool info;
    bool console;
};

int strCount(string &input);
int checkStr(string &bolts, string &nuts);
dArr makeArray(string &input, int length);
```

```
string getStrFromArray(dArr &input);
string getPairs(dArr &bolts, dArr &nuts);
string getStrSwapAction(dArr &input, int ptr1, int ptr2);
string getStrCmpAction(dArr &bolts, int ptr1, dArr &nuts, int ptr2);
int checkPairs(dArr &bolts, dArr &nuts);

int sortBN(dArr &bolts, dArr &nuts, flags &opt, ofstream &fout, int &n);

#endif // DARR_H
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД ПРОГРАММЫ. DARR.CPP

```
#include "darr.h"

int strCount(string &input)
{
    unsigned int count = 0;
    unsigned int ptr = 0;
    bool num = false;
    if (input.empty())
        return 0;
    for (ptr = 0; ptr < input.length(); ptr++)
    {
        if (input[ptr] == ' ')
        {
            if (num != false)
            {
                count++;
                num = false;
            }
        }
        else
        {
            if (!isdigit(input[ptr]))
                return -1;
            num = true;
        }
    }
    if (num == true)
        count++;
    return static_cast<int>(count);
}

int checkStr(string &bolts, string &nuts)
{
    int nB = strCount(bolts);
    if (nB == -1)
        return ERR_SYMB_B;
    int nN = strCount(nuts);
    if (nN == -1)
        return ERR_SYMB_N;
    if (nN == nB)
```

```
        return nN;
    return ERR_LENGTH;
}

dArr makeArray(string &input, int length)
{
    dArr output;
    output.base = new int[length];
    output.length = length;
    stringstream temp;
    temp << input;
    for (int i=0; i<length; i++)
    {
        temp >> output.base[i];
    }
    return output;
}

string getStrFromArray(dArr &input)
{
    string output;
    for (int i=0; i<input.length; i++)
    {
        output += to_string(input.base[i]);
        output += " ";
    }
    output += '\n';
    return output;
}

string getStrCmpAction(dArr &bolts, int ptr1, dArr &nuts, int ptr2)
{
    string output;
    output += "Comparing elements:\nBolts:\n";
    for (int i=0; i<bolts.length; i++)
    {
        if (i == ptr1)
            output += "{";
        output += to_string(bolts.base[i]);
        if (i == ptr1)
            output += "}";
        output += " ";
    }
```

```
    output += "\nNuts:\n";
    for (int i=0; i<nuts.length; i++)
    {
        if (i == ptr2)
            output += "{";
        output += to_string(nuts.base[i]);
        if (i == ptr2)
            output += "}";
        output += " ";
    }
    output += '\n';
    return output;
}

string getStrSwapAction(dArr &input, int ptr1, int ptr2)
{
    string output;
    output += "Swapping elements: ";
    for (int i=0; i<input.length; i++)
    {
        if (i == ptr1 || i == ptr2)
            output += "{";
        output += to_string(input.base[i]);
        if (i == ptr1 || i == ptr2)
            output += "}";
        output += " ";
    }
    output += "\n";
    return output;
}

int checkPairs(dArr &bolts, dArr &nuts)
{
    for (int i=0; i<bolts.length; i++)
    {
        if (bolts.base[i] != nuts.base[i])
            return 0;
    }
    return 1;
}

string getPairs(dArr &bolts, dArr &nuts)
{
```

```
    string output;
    for (int i=0; i<bolts.length; i++)
    {
        if (bolts.base[i] == nuts.base[i])
        {
            output += "Pair: (";
            output += to_string(bolts.base[i]);
            output += ", ";
            output += to_string(nuts.base[i]);
            output += ")\n";
        }
    }
    return output;
}

int GUIOut (string output)
{
    QMessageBox out;
    out.setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
    out.setText(QString::fromStdString(output));
    if (out.exec() == QMessageBox::Cancel)
        return 1;
    return 0;
}

int consoleOut (string output)
{
    cout << output << endl;
    cout << "Press any key to continue" << endl;
    getchar();
}

int qsortNB(dArr &bolts, dArr &nuts, int start, int length, flags &opt,
ofstream &fout, int &n)
{
    int ret = 0;
    string output;
    bool equiv = false;
    int bigger = start;
    if (length - start > 1)
    {
        n++;
        int center = nuts.base[length-1];
```

```
        int i = start;
        while (i<length-1)
        {
            if (opt.info)
                output = getStrCmpAction(bolts, i, nuts, length-1);
            if (bolts.base[i] < center)
            {
                if (opt.step)
                {
                    output += "Element from bolts is smaller, swapping to
left part\n";
                    output += getStrSwapAction(bolts, i, bigger);
                }
                swap(bolts.base[i], bolts.base[bigger]);
                if (opt.step)
                {
                    output += "Result:\nBolts:\n";
                    output += getStrFromArray(bolts);
                    output += "Nuts:\n";
                    output += getStrFromArray(nuts);
                }
                bigger++;
            }
            if (bolts.base[i] == center)
            {
                if (equiv == false)
                    equiv = true;
                else
                {
                    fout << "Error! Found same bolts" << endl;
                    return 2;
                }
                if (opt.step)
                {
                    output += "Element from bolts is equal, swapping to
end\n";
                    output += getStrSwapAction(bolts, i, length-1);
                }
                swap(bolts.base[i], bolts.base[length-1]);
                if (opt.step)
                {
                    output += "Result:\nBolts:\n";
                    output += getStrFromArray(bolts);
```

```
                    output += "Nuts:\n";
                    output += getStrFromArray(nuts);
                }
                continue;
            }
            if (opt.info)
                fout << output << endl;
            if (opt.step)
            {
                if (opt.console)
                    consoleOut(output);
                else
                    if (GUIOut(output))
                        return 1;
            }
            i++;
        }
        equiv = false;
        center = bolts.base[length-1];
        bigger = start;
        i = start;
        while (i<length-1)
        {
            if (opt.info)
                output = getStrCmpAction(bolts, length-1, nuts, i);
            if (nuts.base[i] < center)
            {
                if (opt.step)
                {
                    output += "Element from nuts is smaller, swapping to
left part\n";
                    output += getStrSwapAction(nuts, i, bigger);
                }
                swap(nuts.base[i], nuts.base[bigger]);
                if (opt.step)
                {
                    output += "Result:\nBolts:\n";
                    output += getStrFromArray(bolts);
                    output += "Nuts:\n";
                    output += getStrFromArray(nuts);
                }
                bigger++;
            }
```

```
            if (nuts.base[i] == center)
            {
                if (equiv == false)
                    equiv = true;
                else
                {
                    fout << "Error! Found same nuts" << endl;
                    return 2;
                }
                if (opt.step)
                {
                    output += "Element from nuts is equal, swapping to
end\n";
                    output += getStrSwapAction(nuts, i, length-1);
                }
                swap(nuts.base[i], nuts.base[length-1]);
                if (opt.step)
                {
                    output += "Result:\nBolts:\n";
                    output += getStrFromArray(bolts);
                    output += "Nuts:\n";
                    output += getStrFromArray(nuts);
                }
                continue;
            }
            if (opt.info)
                fout << output << endl;
            if (opt.step)
            {
                if (opt.console)
                    consoleOut(output);
                else
                    if (GUIOut(output))
                        return 1;
            }
            i++;
        }
        ret = qsortNB(bolts, nuts, start, bigger, opt, fout, n);
        if (ret)
            return ret;
        ret = qsortNB(bolts, nuts, bigger, length-1, opt, fout, n);
    }
    return ret;
```

```
}

int sortBN(dArr &bolts, dArr &nuts, flags &opt, ofstream &fout, int &n)
{
    return qsortNB(bolts, nuts, 0, bolts.length, opt, fout, n);
}
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД ПРОГРАММЫ. CONSOLE.H

```
#include "darr.h"

int startConsole();
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД ПРОГРАММЫ. CONSOLE.CPP

```
#include "console.h"


int startConsole()
{
    int load;
    string bolts;
    string nuts;
    dArr boltsArr;
    dArr nutsArr;
    flags opt;
    int n;
    opt.console = true;
    ofstream fout("output.txt");
    cout << "Type 1 for load arrays from file, 0 for console input" <<
endl;
    cin >> load;
    if (load)
    {
        string filePath;
        cout << "Type file path" << endl;
        cin >> filePath;
        ifstream fin(filePath, ios::in);
        if (fin)
        {
            getline(fin, bolts);
            getline(fin, nuts);
        }
        else
        {
            cout << "Error! Wrong file name" << endl;
            return 0;
        }
        fin.close();
    }
    else
    {
        cout << "Type bolts array:" << endl;
        cin.ignore(1);
        getline(cin, bolts);
```

```
    cout << "Type nuts array:" << endl;
    getline(cin, nuts);
}
int num = checkStr(bolts, nuts);
switch (num) {
case 0:
    cout << "Error! Inputs are empty\n";
    return 0;
    break;
case ERR_SYMB_B:
    cout << "Error! Bolts input contains not only digits\n";
    return 0;
    break;
case ERR_SYMB_N:
    cout << "Error! Nuts input contains not only digits\n";
    return 0;
    break;
case ERR_LENGTH:
    cout << "Error! Number of bolts does not match number of nuts\n";
    return 0;
    break;
}
boltsArr = makeArray(bolts, num);
nutsArr = makeArray(nuts, num);
cout << "Type 1 for activating full info, 0 for short answer" << endl;
cin >> opt.info;
if (opt.info)
{
    cout << "Type 1 for activating steps, 0 for result only" << endl;
    cin >> opt.step;
}
cin.ignore(1);
if (!sortBN(boltsArr, nutsArr, opt, fout, n))
{
    if (!checkPairs(boltsArr, nutsArr))
        fout << "There are unpairable bolts and nuts\n";
    else
    {
        fout << "Sorting is finished\nBolts:\n";
        fout << getStrFromArray(boltsArr);
        fout << "Nuts:\n";
        fout << getStrFromArray(nutsArr);
        fout << "Pairs of bolts and nuts:\n";
```

```
            fout << getPairs(boltsArr, nutsArr);
        }
    }
    else
    {
        fout << "Sorting was cancelled\n";
    }
    fout.close();
    ifstream fin("output.txt");
    string output;
    string temp;
    while (!fin.eof())
    {
        getline(fin, temp);
        output += temp;
        output += "\n";
    }
    fin.close();
    cout << output;
}
```

ПРИЛОЖЕНИЕ И

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR3.PRO

```
#-------------------------------------------------
#
# Project created by QtCreator 2019-10-17T10:07:33
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = LR3
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain
version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000     # disables all the
APIs deprecated before Qt 6.0.0

CONFIG += c++11

SOURCES += \
        main.cpp \
        mainwindow.cpp \
    darr.cpp \
    console.cpp

HEADERS += \
        mainwindow.h \
    darr.h \
    console.h

FORMS += \
```

```
        mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    images.qrc
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>600</width>
    <height>600</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QTextEdit" name="output">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>250</y>
      <width>491</width>
      <height>281</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
background-color: white;
border-radius: 15px;
</string>
    </property>
   </widget>
   <widget class="QTextEdit" name="inputB">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>65</y>
      <width>491</width>
      <height>55</height>
     </rect>
```

```
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: white;
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
padding: 2px;</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta                name=&quot;qrichtext&quot;
content=&quot;1&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body   style=&quot;   font-family:'Tahoma';
font-size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p   style=&quot;-qt-paragraph-type:empty;   margin-top:0px;   margin-
bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;br
/&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QPushButton" name="openFile">
    <property name="geometry">
     <rect>
      <x>530</x>
      <y>65</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/open.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QLabel" name="name">
    <property name="geometry">
     <rect>
      <x>200</x>
      <y>10</y>
      <width>200</width>
```

```
      <height>30</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 12pt &amp;quot;Tahoma&amp;quot;;</string>
    </property>
    <property name="text">
     <string>LR3|Bolts and nuts</string>
    </property>
   </widget>
   <widget class="QTextEdit" name="inputN">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>150</y>
      <width>491</width>
      <height>55</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: white;
border-radius: 15px;
font: 10pt &amp;quot;Tahoma&amp;quot;;
padding: 2px;</string>
    </property>
   </widget>
   <widget class="QLabel" name="nameN">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>120</y>
      <width>200</width>
      <height>30</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &amp;quot;Tahoma&amp;quot;;</string>
    </property>
    <property name="text">
     <string>Nuts array:</string>
    </property>
   </widget>
   <widget class="QLabel" name="nameB">
```

```
 <property name="geometry">
  <rect>
   <x>20</x>
   <y>35</y>
   <width>200</width>
   <height>30</height>
  </rect>
 </property>
 <property name="styleSheet">
  <string notr="true">font: 10pt &quot;Tahoma&quot;;</string>
 </property>
 <property name="text">
  <string>Bolts array:</string>
 </property>
</widget>
<widget class="QCheckBox" name="checkStep">
 <property name="geometry">
  <rect>
   <x>170</x>
   <y>210</y>
   <width>120</width>
   <height>20</height>
  </rect>
 </property>
 <property name="styleSheet">
  <string notr="true">font: 9pt &quot;Tahoma&quot;;</string>
 </property>
 <property name="text">
  <string>Step by step</string>
 </property>
</widget>
<widget class="QPushButton" name="start">
 <property name="geometry">
  <rect>
   <x>530</x>
   <y>180</y>
   <width>50</width>
   <height>50</height>
  </rect>
 </property>
 <property name="styleSheet">
  <string notr="true">border-image: url(:/ok.png);</string>
 </property>
```

```
  <property name="text">
   <string/>
  </property>
 </widget>
 <widget class="QCheckBox" name="checkInfo">
  <property name="geometry">
   <rect>
    <x>40</x>
    <y>210</y>
    <width>120</width>
    <height>20</height>
   </rect>
  </property>
  <property name="styleSheet">
   <string notr="true">font: 9pt &quot;Tahoma&quot;;</string>
  </property>
  <property name="text">
   <string>Full info</string>
  </property>
 </widget>
 <widget class="QPushButton" name="saveFile">
  <property name="geometry">
   <rect>
    <x>530</x>
    <y>365</y>
    <width>50</width>
    <height>50</height>
   </rect>
  </property>
  <property name="styleSheet">
   <string notr="true">border-image: url(:/savef.png);</string>
  </property>
  <property name="text">
   <string/>
  </property>
 </widget>
</widget>
<widget class="QMenuBar" name="menuBar">
 <property name="geometry">
  <rect>
   <x>0</x>
   <y>0</y>
   <width>600</width>
```

```
      <height>20</height>
     </rect>
    </property>
   </widget>
   <widget class="QToolBar" name="mainToolBar">
    <attribute name="toolBarArea">
     <enum>TopToolBarArea</enum>
    </attribute>
    <attribute name="toolBarBreak">
     <bool>false</bool>
    </attribute>
   </widget>
   <widget class="QStatusBar" name="statusBar"/>
  </widget>
  <layoutdefault spacing="6" margin="11"/>
  <resources/>
  <connections/>
</ui>
```