**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 |  | Гречко В.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.

Быстрая сортировка, рекурсивная реализация.

## Основные теоретические положения.

Одной из быстрых сортировок, имеющих среднюю сложность 𝑂(𝑛log𝑛), является алгоритм быстрой сортировки QuickSort.

Общая идея алгоритма состоит в следующем:

• Выбрать из массива элемент, называемый опорным. тто 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 Elementary OS, в среде QTCreator.

Сначала происходило считывание введенных пользователем данных и проверка на корректность. Для этого используется возвращаемое функцией toInt () булевое значение. При нахождении ошибки пользователю сообщается об этом. Далее происходит заполнение массива значениями и дальнейшая его обработка.

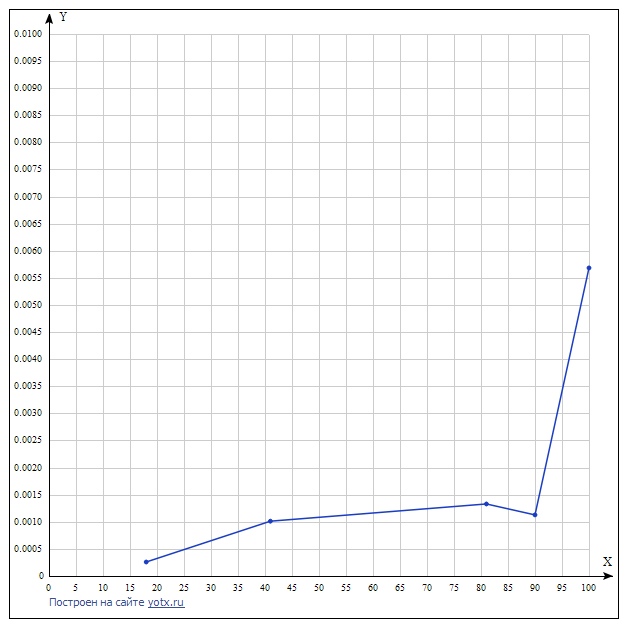
Вызывается функция быстрой сортировки и далее итоговый результат выводится на консоль.

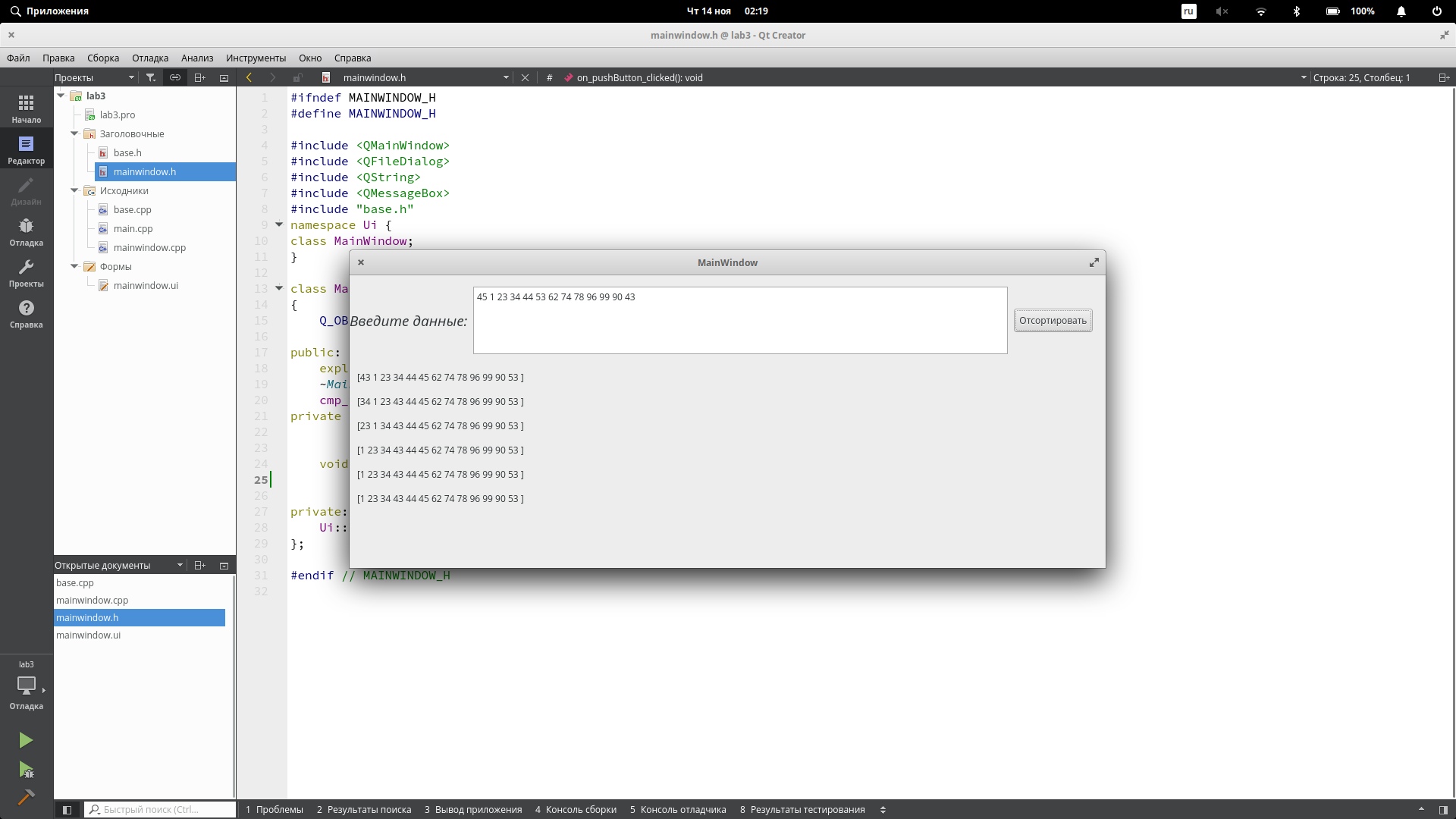
Функция быстрой сортировки quicksort была реализована рекурсивно. Сначала выбирается разрещающий элемент и индексы постепенно начинают двигаться к этому значению. Далее происходит замена найденного элемента разрещающим. А индекс разрещающего используется для дальнейшего запуска программы на каждый из подмассивов.

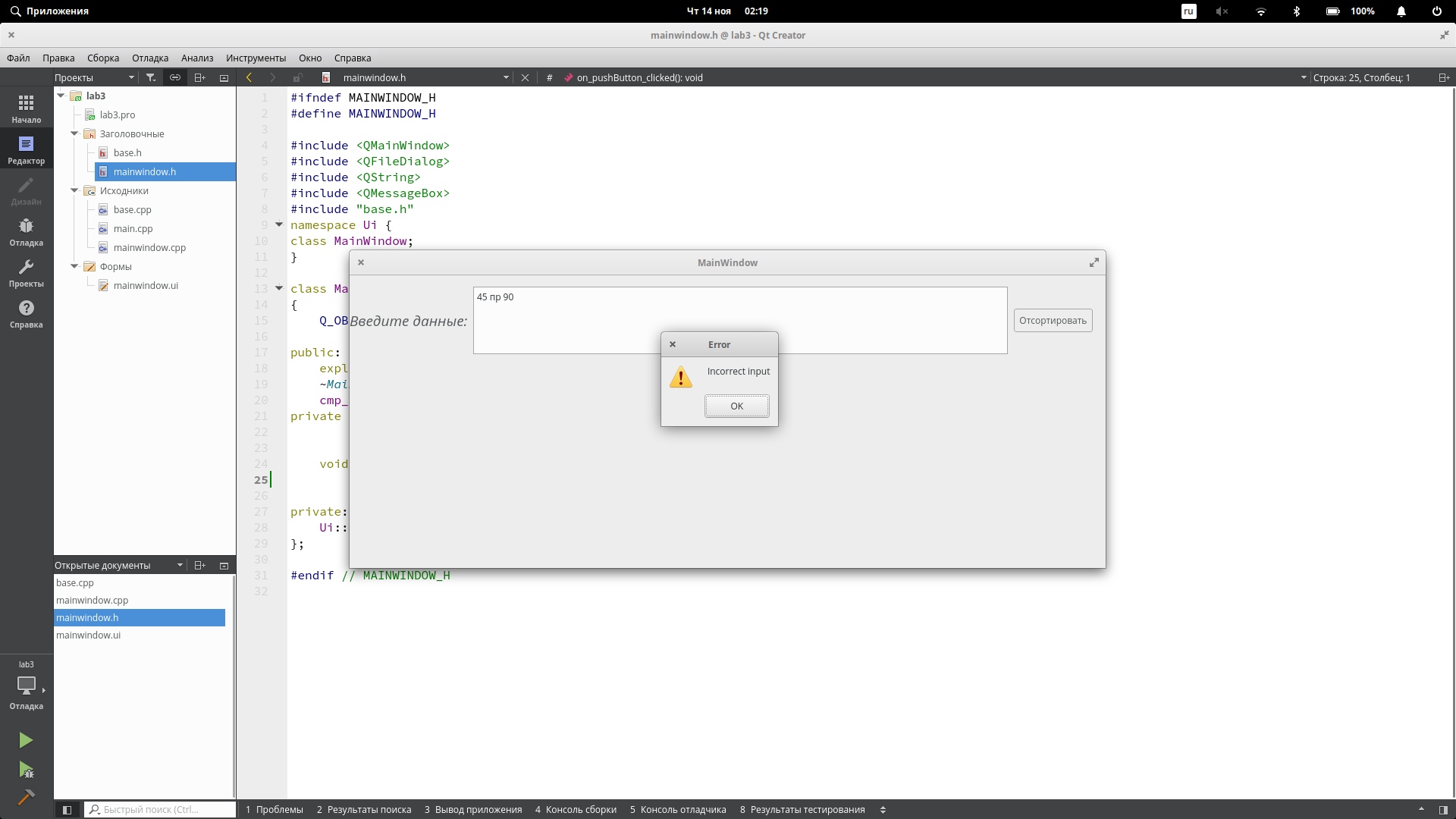
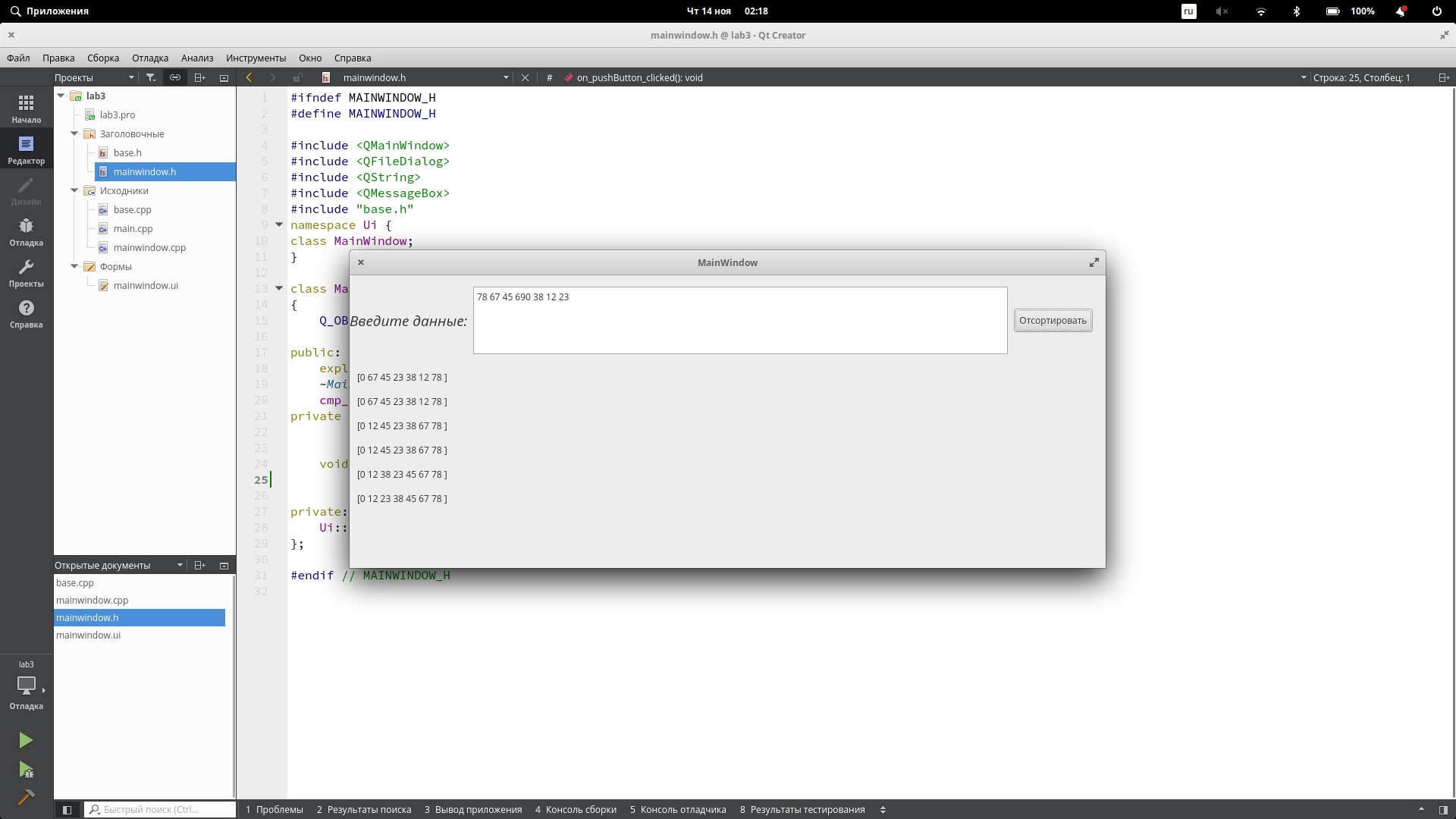
После завершения работы программы результат выводится пользователю. Кроме того, результат представлен в виде последовательной работы алгоритма.

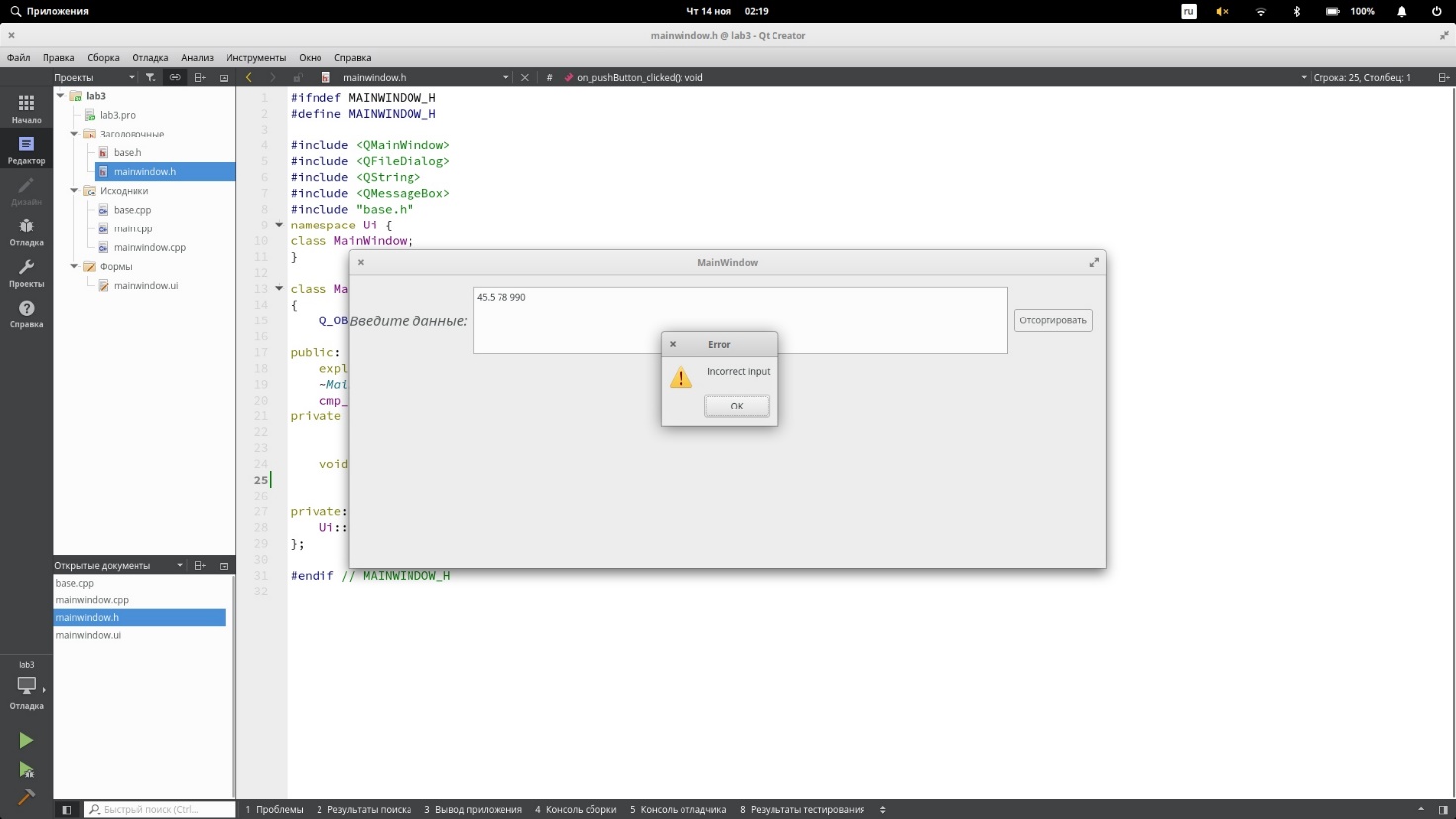
## Оценка эффективности алгоритма.

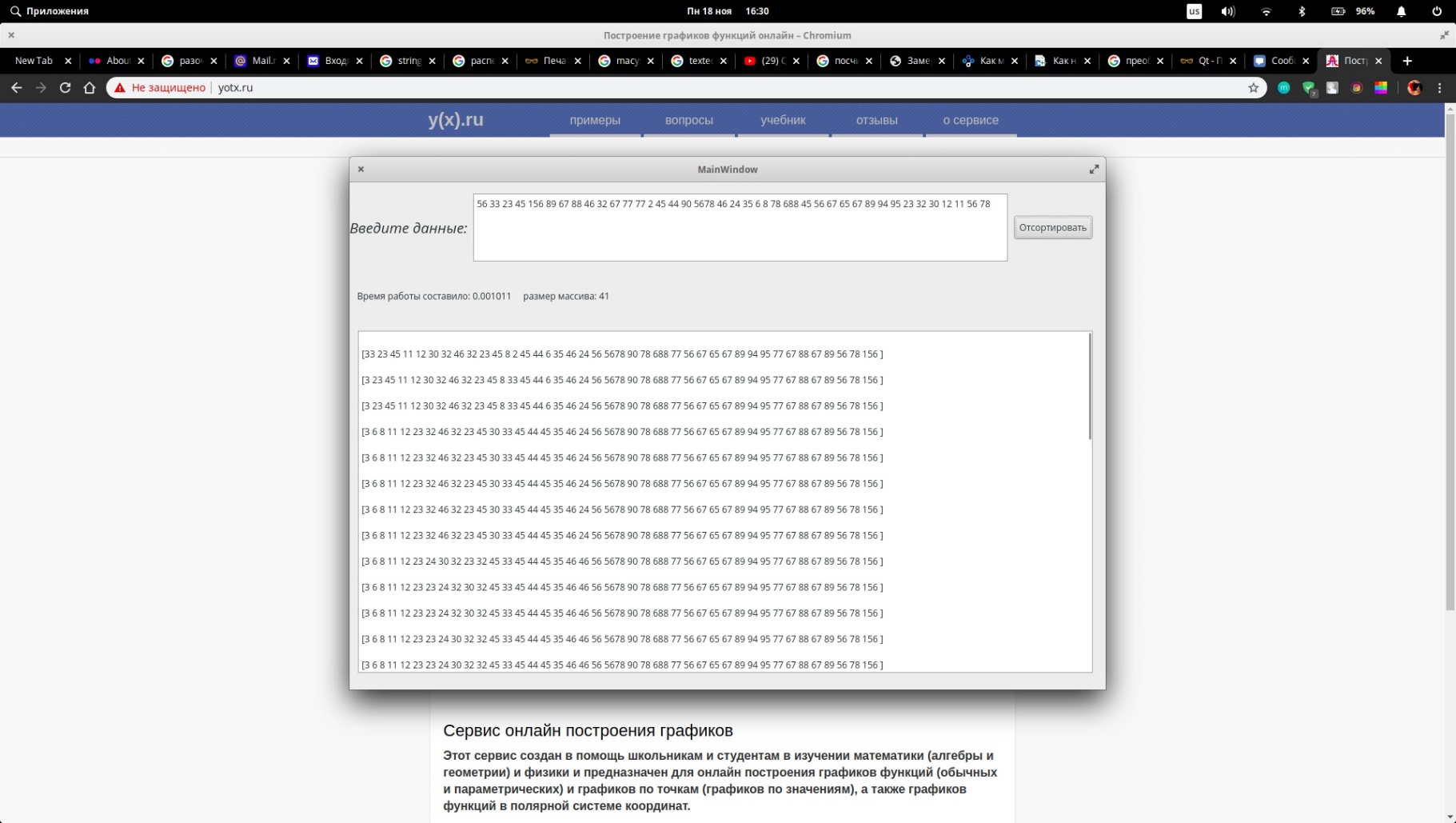
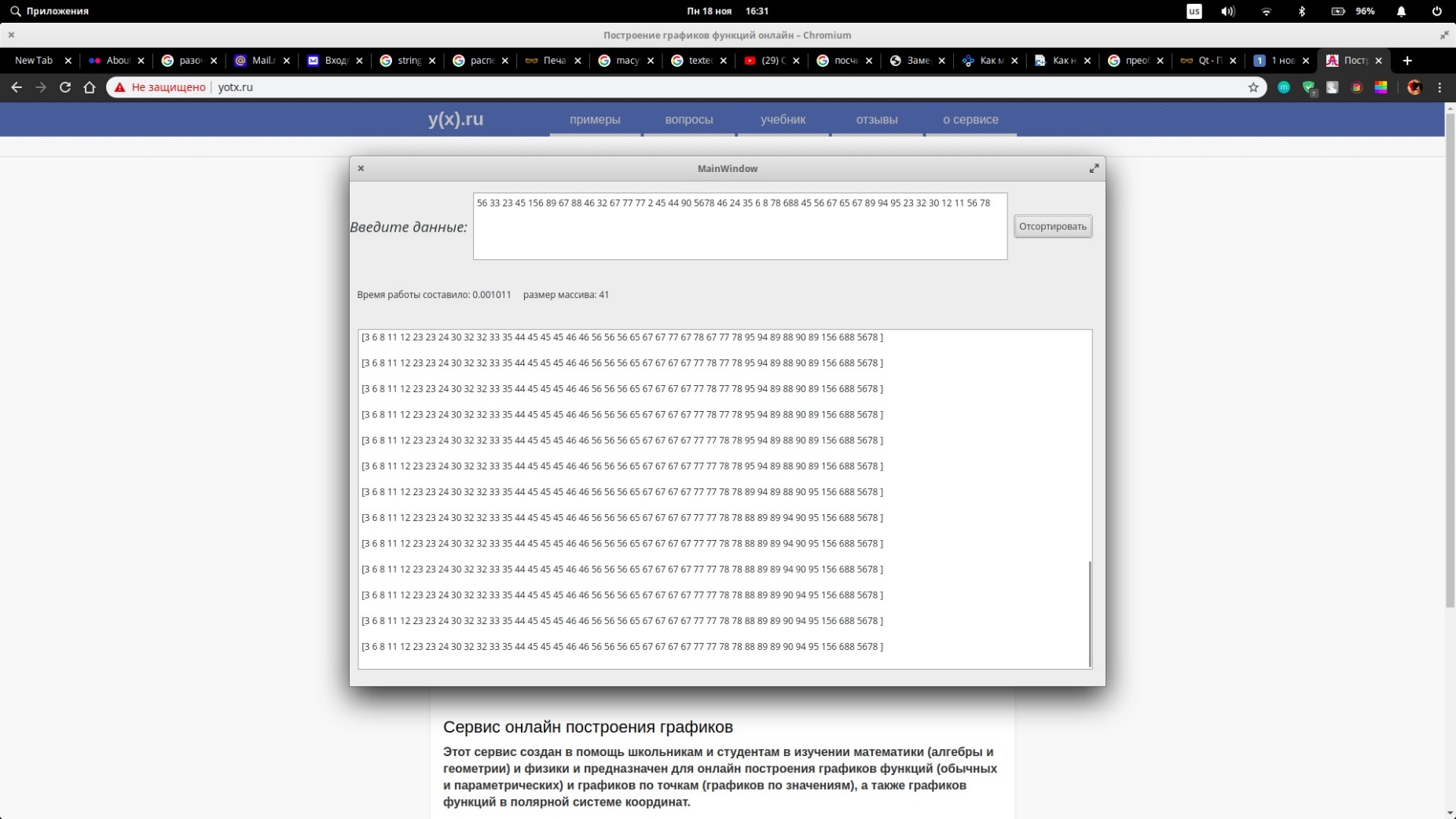
В наиболее сбалансированном варианте при каждой операции разделения массив делится на две одинаковые (плюс-минус один элемент) части, следовательно, максимальная глубина рекурсии, при которой размеры обрабатываемых подмассивов достигнут 1, составит и сложность алгоритма будет равна 𝑂(𝑛×).

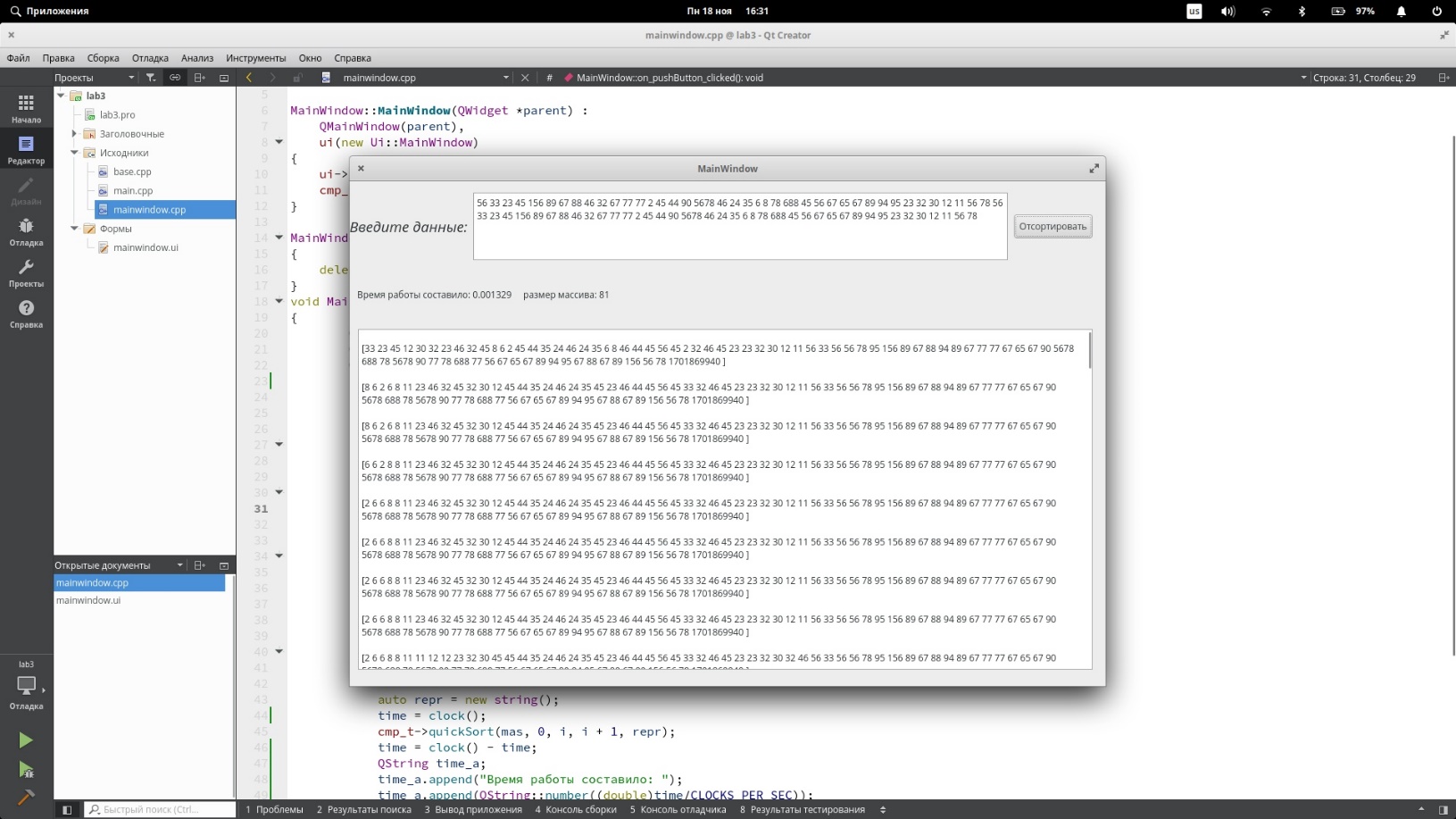
В самом несбалансированном варианте каждое разделение даёт два подмассива размерами 1 и n − 1, то есть при каждом рекурсивном вызове больший массив будет на 1 короче, чем в предыдущий раз. Такое может произойти, если в качестве опорного на каждом этапе будет выбран элемент либо наименьший, либо наибольший из всех обрабатываемых. Общее время работы составит O().

**Тестирование программы.**









## Выводы.

В ходе выполнения лабораторной работы была написана программа, сортирующая массива целочисленных элементов. Был реализован быстрый алгоритм QuickSort, имеющий среднюю сложность 𝑂(𝑛×log(𝑛)).

# Приложение А Исходный код программы

**Файл main.cpp:**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

**Файл base.cpp:**

#include <QFileDialog>

#include "base.h"

#include <string>

void cmp\_type::quickSort(int \*numbers, int left, int right, int size, string\* repr)

{

int pivot;

int l\_hold = left;

int r\_hold = right;

pivot = numbers[left];

while (left < right)

{

while ((numbers[right] >= pivot) && (left < right))

right--;

if (left != right)

{

numbers[left] = numbers[right];

left++;

}

while ((numbers[left] <= pivot) && (left < right))

left++;

if (left != right)

{

numbers[right] = numbers[left];

right--;

}

}

numbers[left] = pivot;

pivot = left;

left = l\_hold;

repr->append("\n[");

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

repr->append(to\_string(numbers[i])).append(" ");

}

repr->append("]\n");

right = r\_hold;

if (left < pivot)

cmp\_type::quickSort(numbers, left, pivot - 1, size, repr);

if (right > pivot)

cmp\_type::quickSort(numbers, pivot + 1, right, size, repr);

}

**Файл base.h:**

#ifndef BASE\_H

#define BASE\_H

#include <string>

using namespace std;

class cmp\_type

{

private:

public:

//cmp\_type();

int type\_m;

void quickSort(int \*numbers, int left, int right, int size);

};

#endif // BASE\_H

**Файл mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QFileDialog>

#include <QString>

#include <QMessageBox>

#include "base.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

cmp\_type \* cmp\_t;

private slots:

void on\_pushButton\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**Файл mainwindow.cpp:**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <string>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

cmp\_t = new(cmp\_type);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

QString data = ui->textEdit->toPlainText();

QStringList abc = data.split(' ');

QStringList result;

int\* mas = new int[100];

int i =0;

bool go = true;

for (auto x:abc){

bool convertOK;

x.toInt(&convertOK);

if(convertOK == false){

go = false;

}

else{

mas[i] = x.toInt();

i++;

}

}

if(go){

QString out\_n;

auto repr = new string();

cmp\_t->quickSort(mas, 0, i, i, repr);

out\_n.append(QString::fromStdString(\*repr));

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

out\_n.append(QString::number(mas[k]));

out\_n.append(" ");

}

ui->label\_3->setText(out\_n);

}

else QMessageBox::warning(this,"Error", "Incorrect input");

}