

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

Преподаватель

Гречко В.Д.

Жангиров Т.Р.

Санкт-Петербург

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, вычисляющую значение выражения.

Задание.

Быстрая сортировка, рекурсивная реализация.

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 меньше 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 Е

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

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

a

r

y

OS, в среде QtCreator.

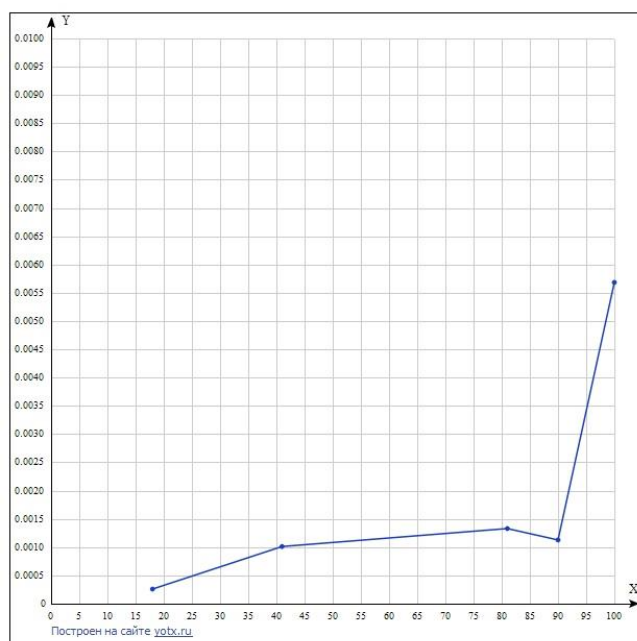
Функция быстрой сортировки quicksort была реализована рекурсивно. Сначала выбирается разрешающий элемент и индексы постепенно начинают двигаться к этому значению. Далее происходит замена найденного элемента разрешающим. А индекс разрешающего используется для дальнейшего запуска программы на каждый из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

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

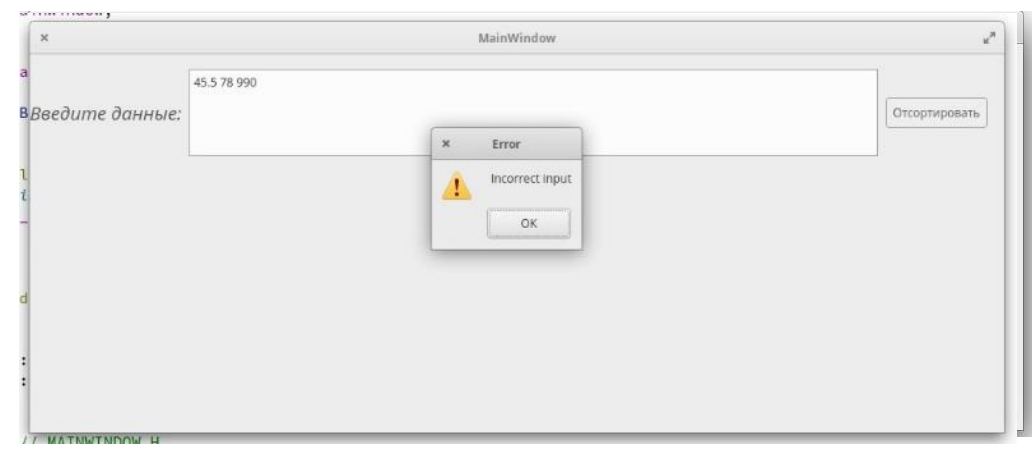
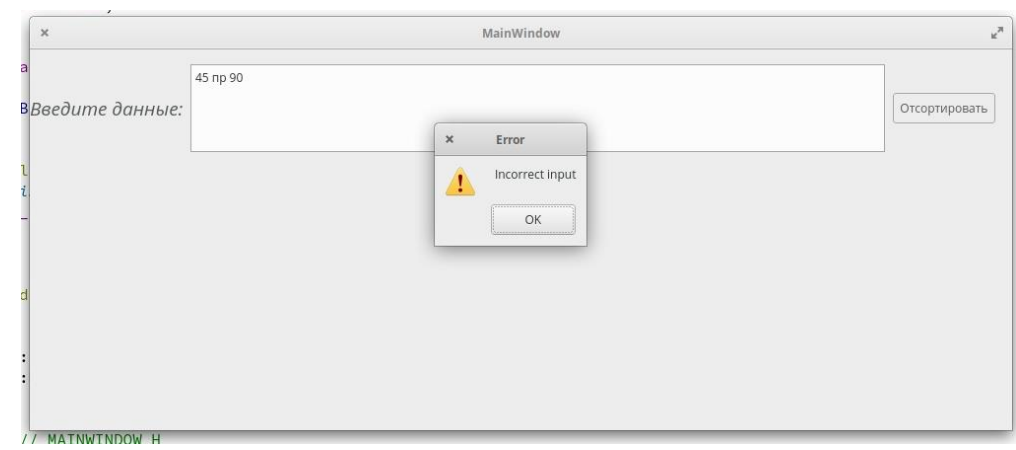
Оценка эффективности алгоритма.

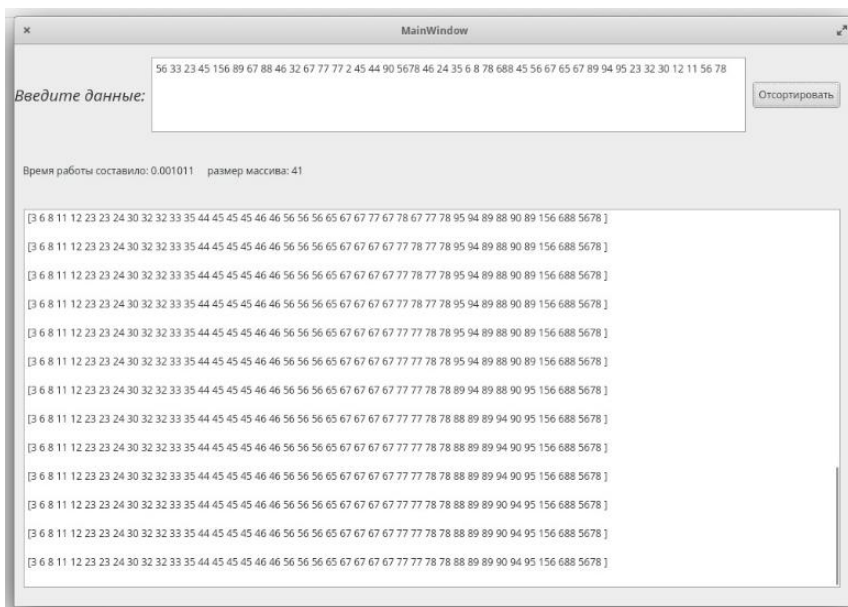
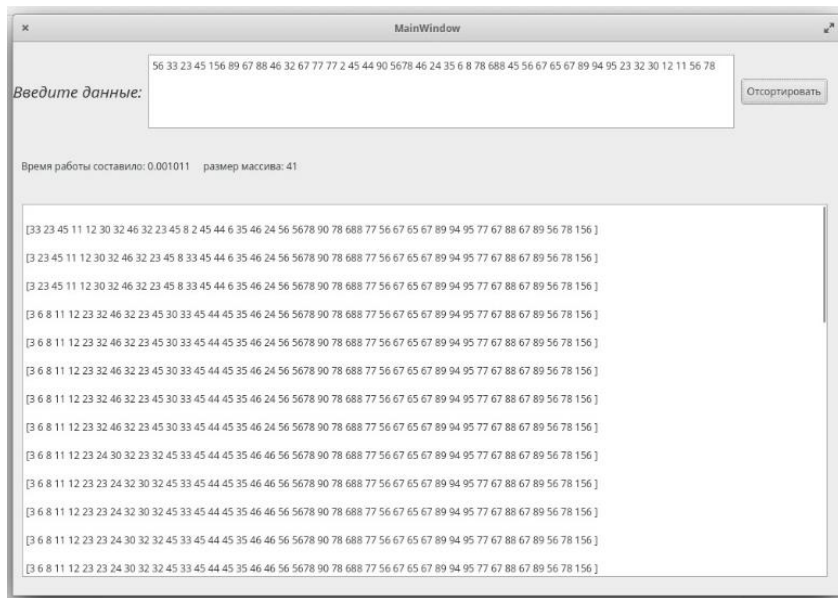
В наиболее сбалансированном варианте при каждой операции деления массив делится на две одинаковые (плюс-минус один элемент) части, следовательно, максимальная глубина рекурсии, при которой размеры

В самом несбалансированном варианте каждое деление 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.



Тестирование программы.





Выводы.

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

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");
}
```