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: Сортировки

Студент гр. 8381 _____ Нгуен Ш. Х.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными понятиями алгоритмов сортировки на языке программирования C++. Разработать программу, реализующую сортировку выбором и сортировку выбором с одновременным выбором максимума и минимума.

Задание

Сортировка выбором; сортировка выбором с одновременным выбором максимума и минимума.

Основные теоретические положения

Сортировка выбором- это простой алгоритм сортировки, основанный на сравнении на месте.

Выбираем наименьший из исходных n элементов, возвращая этот элемент в первое место текущей последовательности. Тогда больше не заботьтесь об этом, посмотрите, что текущая последовательность имеет только (n-1) элементов, и продолжаем со вторым местом. Повторим описанный выше процесс для текущей последовательности, пока в текущей последовательности не останется только один элемент. Первая последовательность имеет n элементов, поэтому идея алгоритма состоит в том, чтобы выполнить n-1 раз, приведя наименьший элемент в текущей последовательности в правильное положение в начале последовательности.

Алгоритм должен как минимум сместить элементы алгоритма сортировки (n! Раз). На массиве из n элементов имеет время выполнения в худшем, среднем и лучшем случае O(n^2).

Алгоритм занимает почти одинаковое время для отсортированных массивов, а также для несортированных массивов.

Шаги алгоритма:

- находим номер минимального значения в текущем списке
- производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)
- теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы

Выполнение работы.

Написание работы производилось на базе операционной системы Linux, 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.

В интерфейсе программы будет 2 опции: сортировка выбором и сортировка выбором с одновременным выбором максимума и минимума. Оба варианта имеют одинаковые результаты, но шаги и количество итераций будут разными.

Функция сортировки выбором selectionSort_1() была реализована рекурсивно. Выбираем наименьший из исходных n элементов, возвращая этот элемент в первую позицию текущей последовательности. Тогда больше не заботьтесь об этом, посмотрите, что текущая последовательность имеет только (n-1) элементов, и продолжаем повторить описанный выше процесс со вторым местом.

Функция сортировки выбором selectionSort_2() была реализована рекурсивно. Выбираем и наименьший и наибольший из исходных n элементов, возвращая наименьший элемент в первое место и наибольший элемент в последное место текущей последовательности. Тогда больше не заботьтесь о первом и последном местах, посмотрите, что текущая последовательность

имеет только (n-2) элементов, и продолжаем повторить описанный выше процесс со вторым местом до (n-1)-ого места.

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

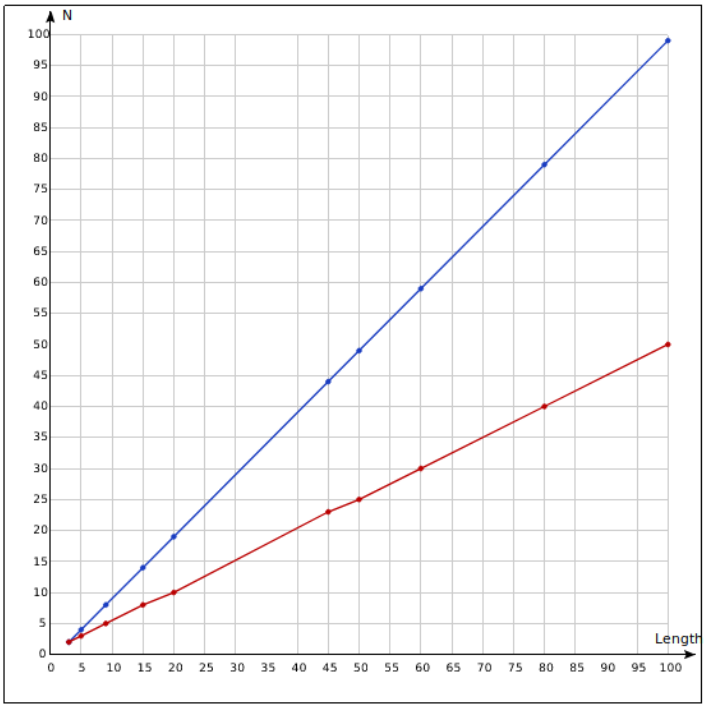
Оценка эффективности алгоритма.

Алгоритм занимает почти одинаковое время для отсортированных массивов, а также для несортированных массивов, поэтому оценка эффективности алгоритма в худшем, среднем и лучшем случае одинаковые.

В обычным алгоритме сортировки выбором есть два циклы : в первым переходить через n1~ n элементов и во втором через n2~ n элементов. Хотя значения n1 n2 уменьшаются в каждом цикле. Поэтому у нас оценка эффективности алгоритма O(n*n) = O(n^2).

А в алгоритме сортировки выбором с одновременным выбором максимума и минимума тоже есть два циклы: в первым переходить через n1~ n элементов и во втором делится на еще две циклы, которые переходят через n/2 элементов, то значит значение n2 ~ 2*(n/2) = n. И у нас же оценка эффективности алгоритма O(n*n) = O(n^2).

График зависимости числа итерраций от длины массивов

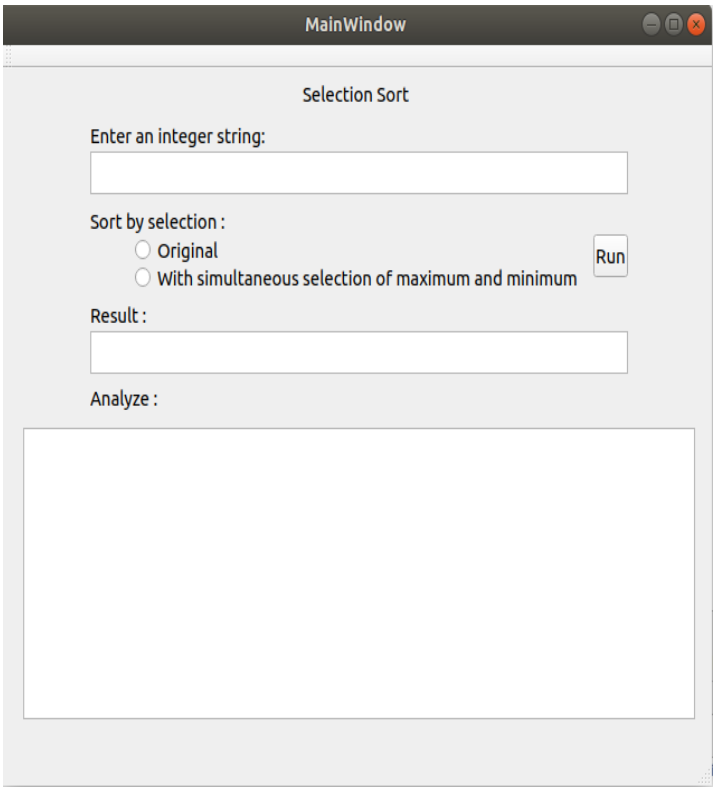


: сортировка выбором.

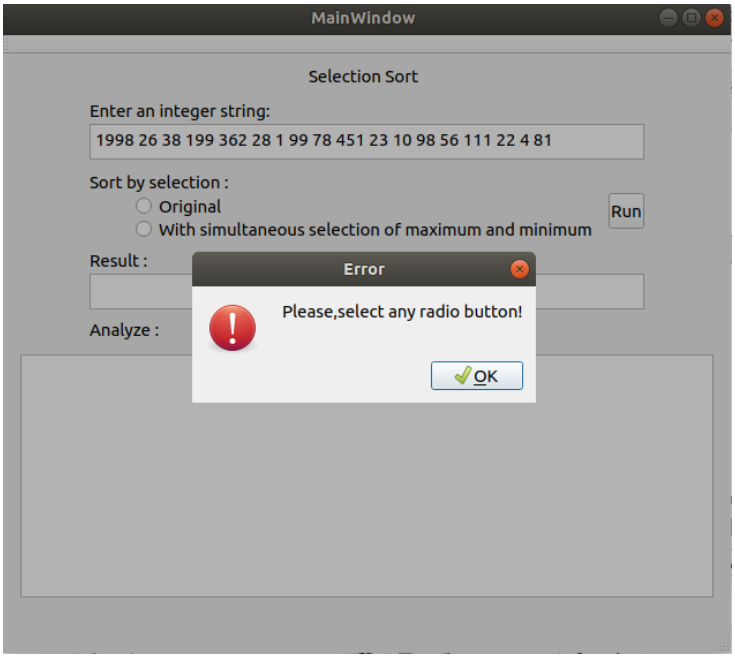
: сортировка выбором с одновременным выбором максимума и минимума.

Тестирование программы.

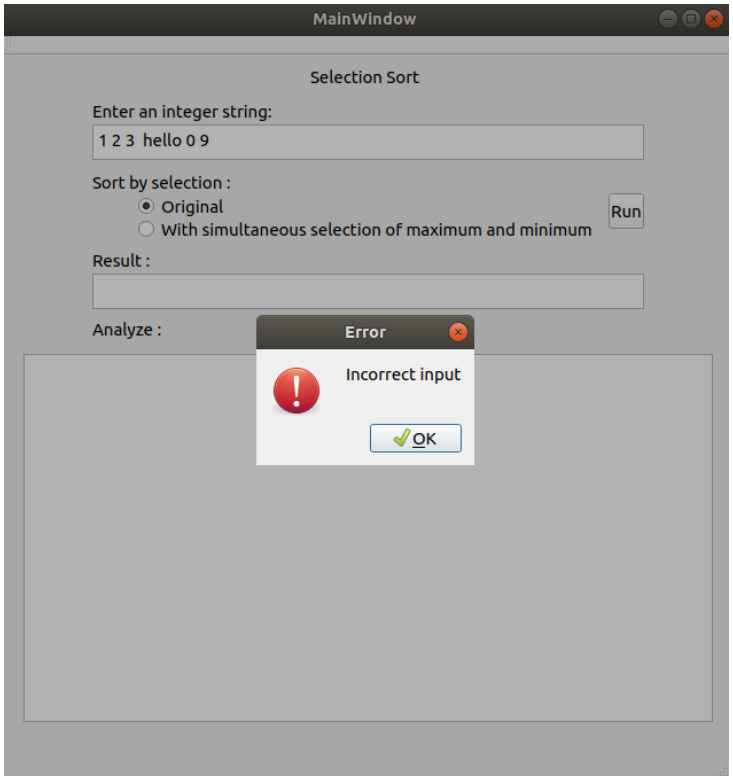
Графический интерфей :



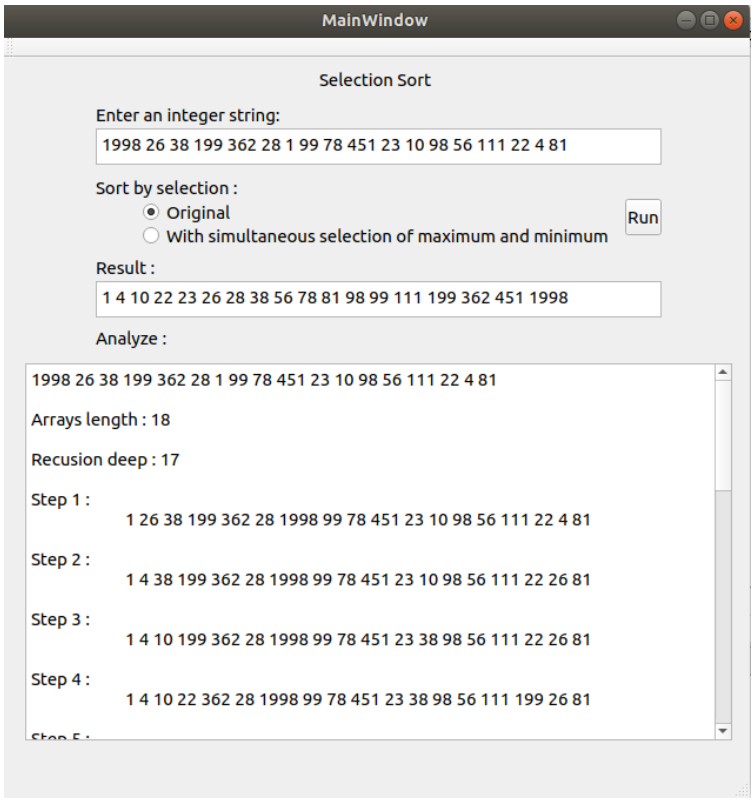
Ошибка не выбрала режим :



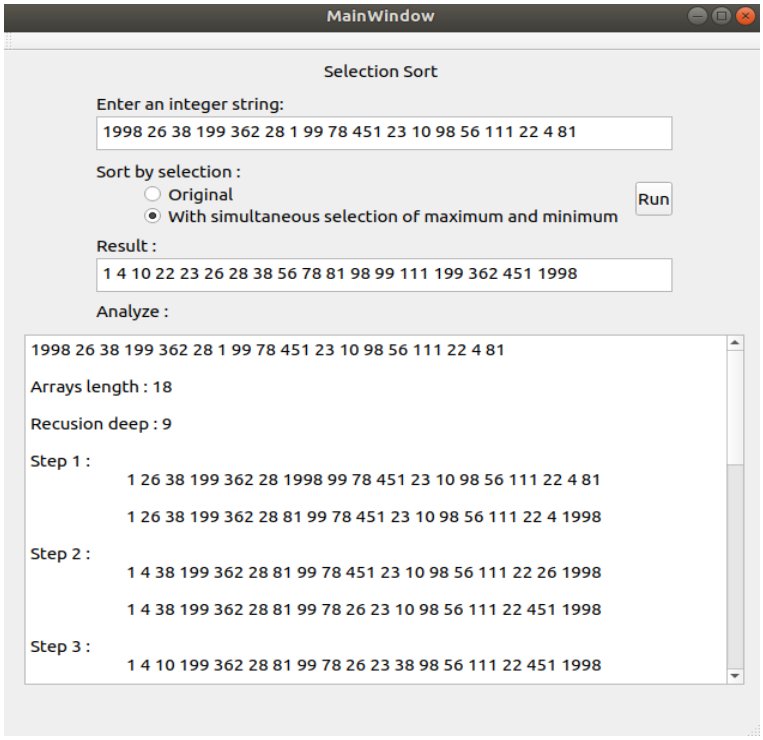
Ошибка ввода данных :



Обычная Сортировка выбором :



Сортировка выбором с одновременным выбором максимума и минимума :



Выводы.

В ходе выполнения лабораторной работы была написана программа, сортирующая массива целочисленных элементов. Был реализован сортировка выбором SelectionSort , имеющий сложность $O(n\text{^}2)$.

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 sort.cpp:

```
#include "sort.h"
void SortSelection::swap(int &a, int &b) {
   int temp;
   temp = a;
   a = b;
   b = temp;
}
void SortSelection::selectionSort_1(int *arr, int size, string *analize, string
*result) {
    int i,j, imin;
    analize->append("\n\n");

    for(i = 0; i < size - 1; i++){
        analize->append("Step ");
        analize->append(to_string(i+1).append(" :\n\t"));
        imin = i;
        for(j = i+1; j < size; j++){
            if(arr[j] < arr[imin]){
                imin = j;
            }
        }
        swap(arr[imin],arr[i]);
        for(int i = 0; i < size; i++){
```

```
            analize->append(to_string(arr[i]).append(" "));
        }
        analize->append("\n\n");
    }

    for(int i = 0; i < size; i++){
        result->append(to_string(arr[i]).append(" "));
    }
}

void SortSelection::selectionSort_2(int *arr, int size, string *analize, string
*result) {
    int i, j, imin, imax;
    analize->append("\n\n");
    for(i = 0; i<int((size+1)/2); i++) {
        analize->append("Step ");
        analize->append(to_string(i+1).append(" :\n\t"));
        imin = i;
        imax = size -1-i;

        for(j = i+1; j<size; j++){
            if(arr[j] < arr[imin])
                imin = j;
        }
        swap(arr[i], arr[imin]);
        for(int i = 0; i < size; i++){
            analize->append(to_string(arr[i]).append(" "));
        }
        analize->append("\n\n");

        for(j = i+1; j<size-i; j++){
            if(arr[j]>arr[imax])
                imax = j;
        }

        swap(arr[size-1-i],arr[imax]);
        analize->append("\t");
        for(int i = 0; i < size; i++){
            analize->append(to_string(arr[i]).append(" "));
        }
        analize->append("\n\n");
    }
```

```
    for(int i = 0; i < size; i++){
        result->append(to_string(arr[i]).append(" "));
    }
}
```

Файл sort.h:

```
#ifndef SORT_H
#define SORT_H
#include <string>
#include <QFileDialog>

using namespace std;
class SortSelection
{
public:
    void selectionSort_1(int *arr, int size, string *analize, string *result);
    void selectionSort_2(int *arr, int size, string *analize, string *result);
    void swap(int &a, int &b);
};

#endif // SORT_H
```

Файл mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QString>
#include <sort.h>
#include <QMessageBox>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
```

```
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    SortSelection* cmp;
    bool flagCase_1 = false;
    bool flagCase_2 = false;

private slots:
    void on_pushButton_clicked();
    void on_radioButton_clicked(bool checked);

    void on_radioButton_2_clicked(bool checked);

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

Файл mainwindow.cpp:

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

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    cmp = new(SortSelection);

}

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

void MainWindow::on_pushButton_clicked()
{
    QString data = ui->textEdit->toPlainText();
    QStringList text_in = data.split(' ');

    int* arr = new int[100];
```

```
    int i = 0;
    bool flagError = false;
    for(auto x:text_in){
        bool flagConvert;
        x.toInt(&flagConvert);
        if(flagConvert == false){
            flagError = true;
        }
        else {
            arr[i] = x.toInt();
            i++;
        }
    }

    if(!flagError){
        QString text_out;
        auto analize = new string();
        auto result = new string();
        analize->append("\n\nArrays length : ");
        analize->append(to_string(i));

        if(flagCase_1){
            analize->append("\n\nRecusion deep : ");
            analize->append(to_string(i-1));
            cmp->selectionSort_1(arr,i,analize,result);
            flagCase_1 = false;
        }
        else if(flagCase_2){
            analize->append("\n\nRecusion deep : ");
            analize->append(to_string(int((i+1)/2)));
            cmp->selectionSort_2(arr,i,analize,result);
            flagCase_2 = false;
        }
        else if (flagCase_1 == false && flagCase_2 == false) {
                    QMessageBox::warning(this,"Error","Please,select  any  radio
button!");
            return;
        }

        data.append(QString::fromStdString(*analize));
        ui->textEdit_2->setText(data);
        ui->textEdit_3->setText(QString::fromStdString(*result));
```

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

void MainWindow::on_radioButton_clicked(bool checked)
{
    flagCase_1 = checked;
}

void MainWindow::on_radioButton_2_clicked(bool checked)
{
    flagCase_2 = checked;
}
```