**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 О(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.

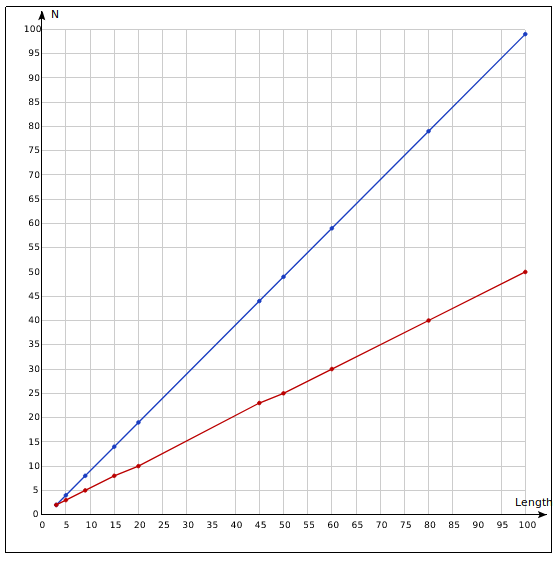
## Оценка эффективности алгоритма.

## 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 О(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 О(n\*n) = O(n^2).

График зависимости числа итерраций от длины массивов



: сортировка выбором.

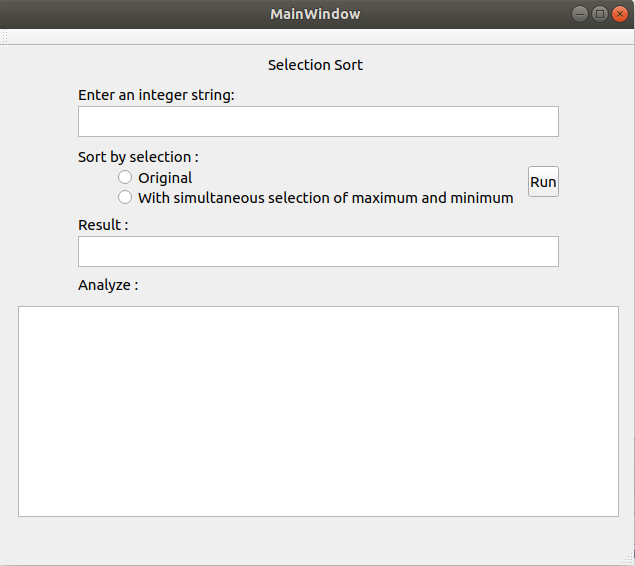
: сортировка выбором с

одновременным выбором

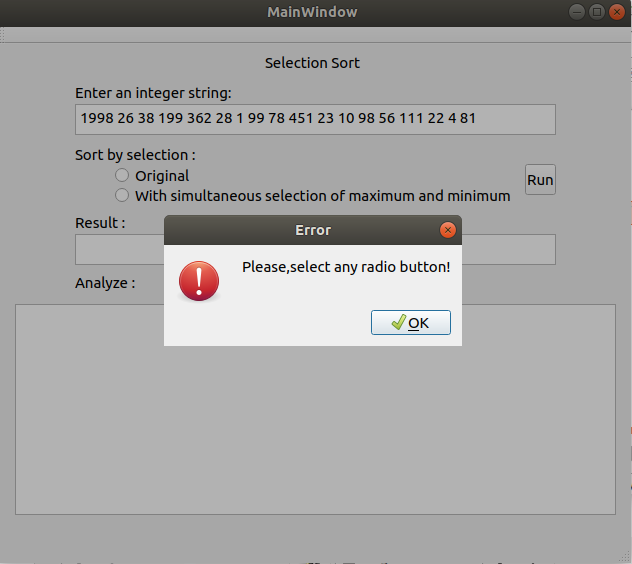
максимума и минимума.

**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^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 console.cpp:**

#include "console.h"

console::console(){

Method = new(method);

}

void console::Console(bool flagCase\_1, bool flagCase\_2,int\* flagError, string \*result, string \*analize,QString data){

if(flagCase\_1 == false && flagCase\_2 == false){

\*flagError = 2;

return;

}

else {

QStringList text\_in = data.split(' ');

int\* arr = new int[100];

int i = 0;

for(auto x:text\_in){

bool flagConvert;

x.toInt(&flagConvert);

if(flagConvert == false){

\*flagError = 1;

return;

}

else {

arr[i] = x.toInt();

i++;

}

}

analize->append("Arrays length : ");

analize->append(to\_string(i));

if(flagCase\_1){

analize->append("\n\nRecusion deep : ");

analize->append(to\_string(i-1));

Method->selectionSort\_1(arr,i,analize,result);

}

else if(flagCase\_2){

analize->append("\n\nRecusion deep : ");

analize->append(to\_string(int((i+1)/2)));

Method->selectionSort\_2(arr,i,analize,result);

}

}

}

**Файл console.h:**

#ifndef CONSOLE\_H

#define CONSOLE\_H

#include <method.h>

#include <QString>

#include <QFileDialog>

class console{

public:

console();

void Console(bool flagCase\_1, bool flagCase\_2,int \*flagError,string \*result, string \*analize, QString data);

method \*Method;

};

#endif // CONSOLE\_H

**Файл method.cpp:**

#include "method.h"

void method::swap(int &a, int &b) {

int temp;

temp = a;

a = b;

b = temp;

}

void method::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 method::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 method.h:**

#ifndef METHOD\_H

#define METHOD\_H

#include <string>

#include <QString>

using namespace std;

class method

{

public:

void swap(int &a, int &b);

void selectionSort\_1(int \*arr, int size, string \*analize, string \*result);

void selectionSort\_2(int \*arr, int size, string \*analize, string \*result);

};

#endif // METHOD\_H

**Файл mainwindow.cpp:**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

process = new(console);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{

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

int flagError = 0;

result = new (string);

analize = new (string);

process->Console(flagCase\_1,flagCase\_2,&flagError,result,analize,data);

if(!flagError){

ui->textEdit\_2->setText(QString::fromStdString(\*result));

ui->textEdit\_3->setText(QString::fromStdString(\*analize));

}

else if (flagError == 2) {

QMessageBox::warning(this,"Error","Please,select any radio button!");

flagCase\_2 = false;

}

else if(flagError == 1){

QMessageBox::warning(this,"Error","Input incorrect");

flagCase\_1 = false;

}

}

void MainWindow::on\_radioButton\_clicked(bool checked)

{

flagCase\_2 = false;

flagCase\_1 = checked;

}

void MainWindow::on\_radioButton\_2\_clicked(bool checked)

{

flagCase\_2 = checked;

flagCase\_1 = false;

}

**Файл mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QString>

#include <QMessageBox>

#include <console.h>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

bool flagCase\_1 = false;

bool flagCase\_2 = false;

console \*process;

string \*result;

string \*analize;

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