

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 III. X.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

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()` булево 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 последнее место текущей последовательности. Тогда больше не заботьтесь о первом и последнем 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 обычном алгоритме сортировки выбором есть два цикла : в первом переходить через $n_1 \sim n$ элементов и во втором через $n_2 \sim n$ элементов. Хотя значения n_1 n_2 уменьшаются в каждом цикле. Поэтому у нас оценка эффективности алгоритма $O(n*n) = O(n^2)$.

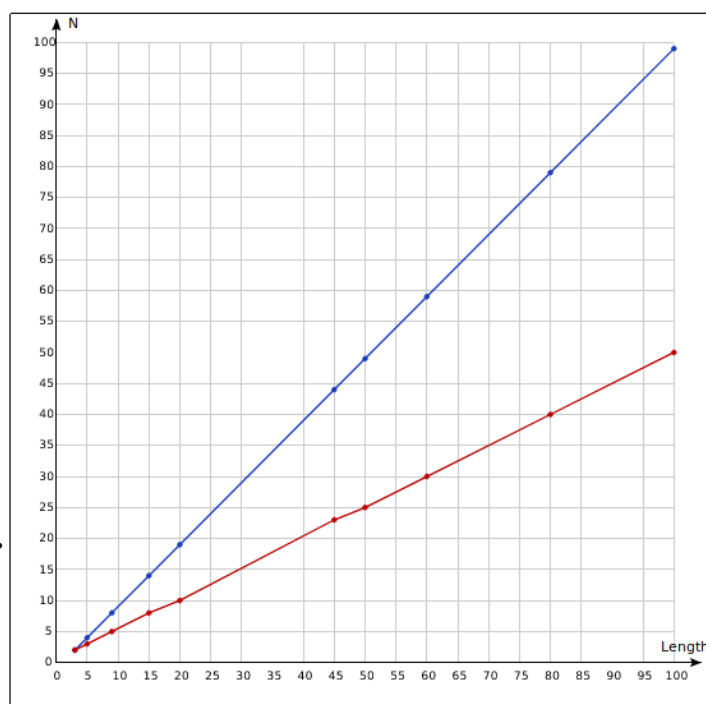
А в алгоритме сортировки выбором с одновременным выбором максимума и минимума тоже есть два цикла: в первом переходить через $n_1 \sim n$ элементов и во втором делится на еще две циклы, которые переходят через $n/2$ элементов, то значит значение $n_2 \sim 2*(n/2) = n$. И у нас же оценка эффективности алгоритма $O(n*n) = O(n^2)$.

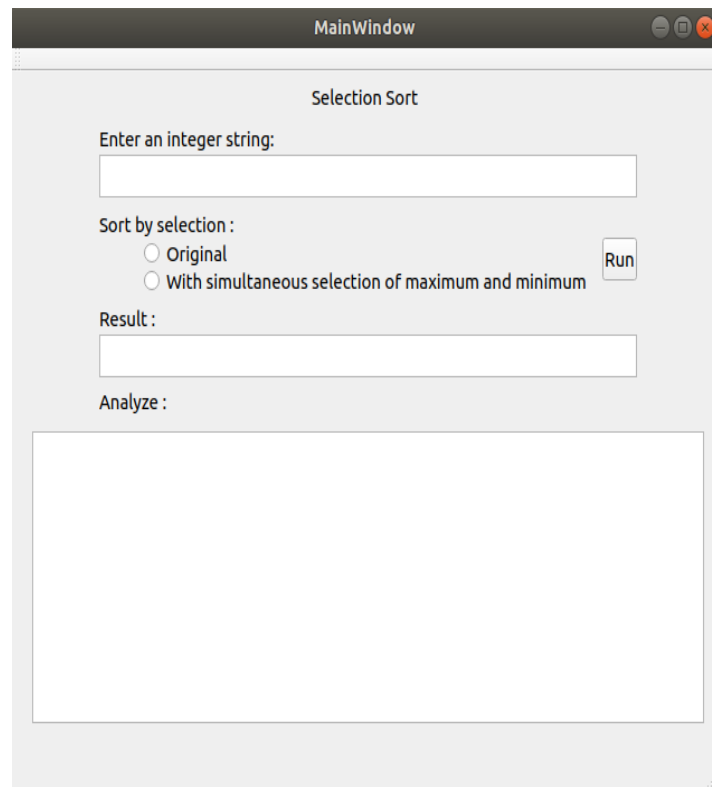
График зависимости числа итераций от длины массивов

— : сортировка выбором.
— : сортировка выбором с
одновременным выбором
максимума и минимума.

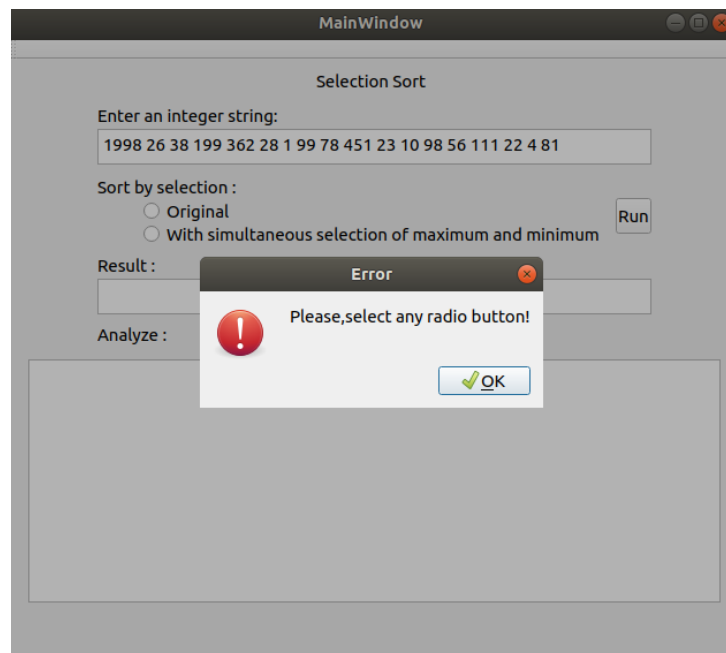
Тестирование программы.

Графический интерфейс :



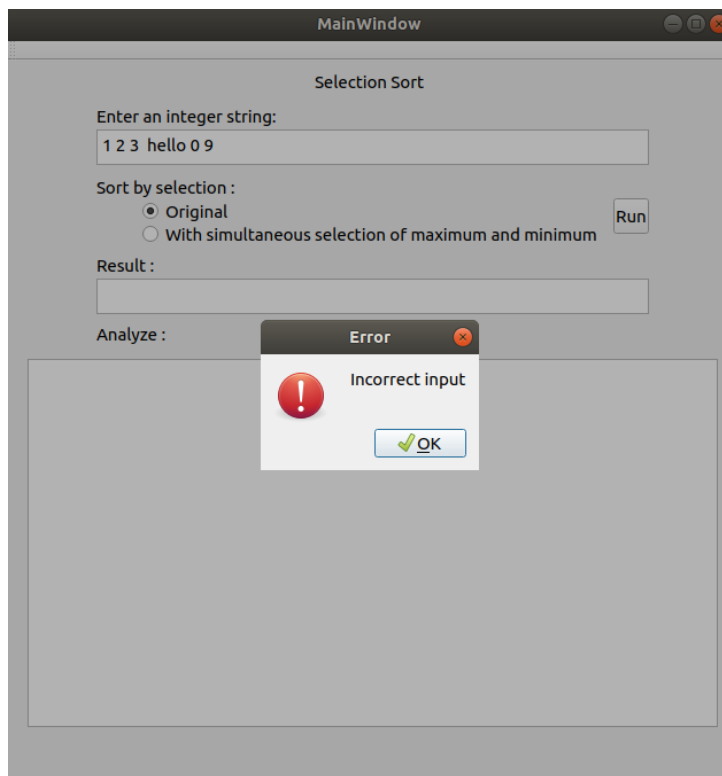


Ошибка не выбрала режим :

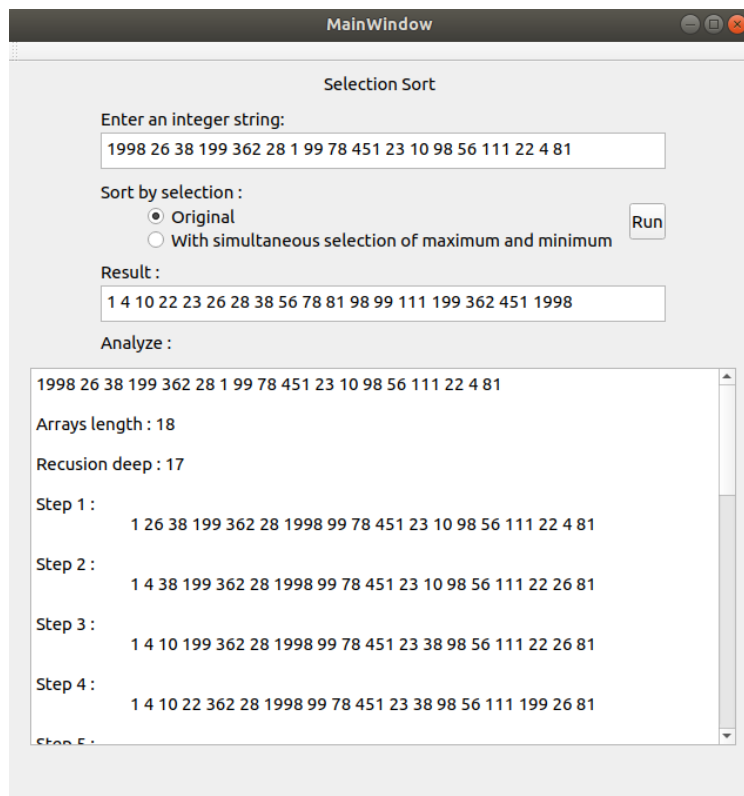


Ошибка

ВВОДА ДАННЫХ :



Обычная Сортировка выбором :



Сортировка выбором с одновременным выбором максимума и минимума :

The screenshot shows a window titled 'MainWindow' with a sub-header 'Selection Sort'. It contains the following elements:

- Enter an integer string:** A text box containing the string '1998 26 38 199 362 28 1 99 78 451 23 10 98 56 111 22 4 81'.
- Sort by selection :** Two radio buttons: 'Original' (unselected) and 'With simultaneous selection of maximum and minimum' (selected). A 'Run' button is to the right.
- Result :** A text box displaying the sorted array: '1 4 10 22 23 26 28 38 56 78 81 98 99 111 199 362 451 1998'.
- Analyze :** A scrollable text area showing the following information:
 - The original array: '1998 26 38 199 362 28 1 99 78 451 23 10 98 56 111 22 4 81'
 - 'Arrays length : 18'
 - 'Recursion deep : 9'
 - Step 1 :**
 - '1 26 38 199 362 28 1998 99 78 451 23 10 98 56 111 22 4 81'
 - '1 26 38 199 362 28 81 99 78 451 23 10 98 56 111 22 4 1998'
 - Step 2 :**
 - '1 4 38 199 362 28 81 99 78 451 23 10 98 56 111 22 26 1998'
 - '1 4 38 199 362 28 81 99 78 26 23 10 98 56 111 22 451 1998'
 - Step 3 :**
 - '1 4 10 199 362 28 81 99 78 26 23 38 98 56 111 22 451 1998'

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 *analyze,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++;
        }
    }

    analyze->append("Arrays length : ");
    analyze->append(to_string(i));
    if(flagCase_1){
        analyze->append("\n\nRecursion deep : ");
        analyze->append(to_string(i-1));
        Method->selectionSort_1(arr,i,analyze,result);
    }
    else if(flagCase_2){
        analyze->append("\n\nRecursion deep : ");
        analyze->append(to_string(int((i+1)/2)));
        Method->selectionSort_2(arr,i,analyze,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
    *analyze, 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 *analyze, string *result) {
    int i,j, imin;
    analyze->append("\n\n");

    for(i = 0; i < size - 1; i++){
        analyze->append("Step ");
        analyze->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++){
            analyze->append(to_string(arr[i]).append(" "));
        }
        analyze->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 *analyze, string *result) {  
    int i, j, imin, imax;  
    analyze->append("\n\n");  
    for(i = 0; i<int((size+1)/2); i++) {  
        analyze->append("Step ");  
        analyze->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++){  
            analyze->append(to_string(arr[i]).append(" "));  
        }  
        analyze->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]);
```

```
analyze->append("\t");
```

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

```
    analyze->append(to_string(arr[i]).append(" "));
```

```
}
```

```
analyze->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 *analyze, string *result);
```

```
    void selectionSort_2(int *arr, int size, string *analyze, string *result);
```

```
};
```

```
#endif // METHOD_H
```

Файлmainwindow.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 *analyze;

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
```