Специальность: 09.02.07 Информационные системы и программирование

Проверил:

\_\_\_\_\_\_\_\_\_\_\_ Антонов М.Б.

Дата: «23» мая 2020 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

(Университет ИТМО)

факультет среднего профессионального образования

Отчет

о Лабораторной работе №18

по теме: Потоки, графики и сортировка Qt

по дисциплине: Разработка программных модулей

Выполнил:

студент группы 2333

\_\_\_\_\_\_\_\_\_\_\_ Агапова Е.А.

Санкт-Петербург 2020

Содержание

[1 Постановка задачи 3](#_Toc38966037)

[1.1 Цель 3](#_Toc38966038)

[1.2 Задачи 3](#_Toc38966039)

[1.3 Задание 3](#_Toc38966040)

[1.4 Общее ограничение на лабораторную работу 3](#_Toc38966041)

[1.5 Требования к отчёту 4](#_Toc38966042)

[2 Диаграммы 5](#_Toc38966043)

[3 Исходный код 6](#_Toc38966044)

[4 ЛИСТИНГ ВЫполнения программы 13](#_Toc38966045)

[5 Вывод 14](#_Toc38966046)

1. Постановка задачи
   1. Цель

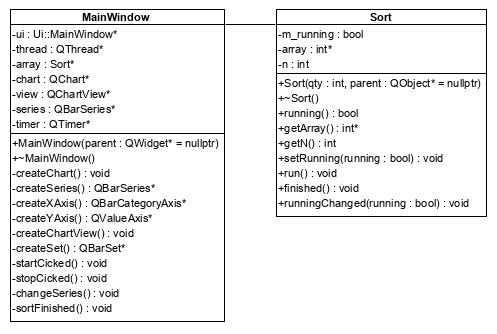
Познакомиться с основами работы с графиками и потоками в Qt.

* 1. Задачи

1. изучить основы работы с QThread;
2. изучить основы работы с QChart;
3. создать приложение для визуализации сортировки.
   1. Задание
4. Эмулируем долгую работу программы над каким-то тяжёлым действием путём сортировки методом пузырька 500 тыс. int значений.
5. Требование: программа должна после запуска генерировать массив в 500
6. элементов.
7. Программа должна на основном экране иметь: столбиковую диаграмму (500 элементов), кнопку запуска и кнопку выхода.
8. Программа должна уметь соотносить 500 тыс. значений из пункта 2 и 500 столбиков диаграммы (к примеру 1 столбик высчитывает среднее значение 1000 элементов и отображает именно их).
9. При нажатии на кнопку запуска программа должна инициализировать сортировку столбиком.
10. Кнопка выхода из программы должна быть доступна всегда (т. е. сортировка должна идти в отдельном потоке).
11. Раз в 3 секунды должно идти обновление текущего состояния сортируемых данных в график на интерфейсе.
    1. Общее ограничение на лабораторную работу

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.
  1. Требования к отчёту
* цели лабораторной работы;
* задачи лабораторной работы;
* выполняемое задание;
* диаграмма(ы) классов;
* диаграммы деятельности (для методов и/или функций больше 10 строк кода (исключая форматирования и пустые строки));
* исходный код программы с комментариями (2 пробела вместо 4, 10pt, моноширинный шрифт);
* листинг выполнения программы;
* исходный код в виде файлов для запуска на произвольном ПК.

1. Диаграммы



*Рисунок 1 – Диаграмма классов*

1. Исходный код

В листингах 2.1-2.5 представлен исходный код программы.

Листинг 2.1 – Код программы

/\*\*

\* @file main.cpp

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Листинг 2.2 – Заголовочный файл mainwindow.h

/\*\*

\* @file mainwindow.h

\* @brief main window whith charts

\* @author Agapova Ekaterina

\*/

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QThread>

#include "sort.h"

#include <QChart>

#include <QChartView>

#include <QBarSeries>

#include <QBarSet>

#include <QBarCategoryAxis>

#include <QValueAxis>

#include <QTimer>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

using namespace QtCharts;

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

/\*\*

\* @brief constructor MainWindow

\* @param parent

\*/

MainWindow(QWidget \*parent = nullptr);

/\*\*

\* @brief destructor MainWindow

\*/

~MainWindow();

private:

Ui::MainWindow \*ui;

QThread \*thread;

Sort\* array;

QChart \*chart;

QChartView \*view;

QBarSeries \*series;

QTimer \*timer;

private:

/\*\*

\* @brief create chart

\*/

void createChart();

/\*\*

\* @brief create series for chart

\* @return

\*/

QBarSeries\* createSeries();

/\*\*

\* @brief create X axis for chart

\* @return

\*/

QBarCategoryAxis\* createXAxis();

/\*\*

\* @brief create Y axis for chart

\* @return

\*/

QValueAxis\* createYAxis();

/\*\*

\* @brief create chart view

\*/

void createChartView();

/\*\*

\* @brief create set for series

\* @return

\*/

QBarSet\* createSet();

private slots:

/\*\*

\* @brief button start is clicked

\*/

void startClicked();

/\*\*

\* @brief button stop is clicked

\*/

void stopClicked();

/\*\*

\* @brief change series

\*/

void changeSeries();

/\*\*

\* @brief sort finished message

\*/

void sortFinished();

};

#endif // MAINWINDOW\_H

Листинг 2.3 – Заголовочный файл mainwindow.cpp

/\*\*

\* @file mainwindow.cpp

\* @brief main window whith charts

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

thread = new QThread();

array = new Sort(500000);

array->moveToThread(thread);

chart = new QChart();

view = new QChartView(ui->widget);

timer = new QTimer();

this->createChart();

this->createChartView();

connect(ui->start, &QPushButton::clicked, this, &MainWindow::startClicked);

connect(ui->stop, &QPushButton::clicked, this, &MainWindow::stopClicked);

connect(thread, &QThread::started, array, &Sort::run);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::startClicked()

{

array->setRunning(true);

thread->start();

timer->start(3000);

connect(timer, &QTimer::timeout, this, &MainWindow::changeSeries);

connect(array, &Sort::finished, thread, &QThread::terminate);

connect(array, &Sort::finished, timer, &QTimer::stop);

connect(array, &Sort::finished, this, &MainWindow::sortFinished);

}

void MainWindow::stopClicked()

{

if(thread->isRunning()){

array->setRunning(false);

thread->exit();

timer->stop();

}

this->close();

}

void MainWindow::createChart()

{

series = new QBarSeries;

this->createSeries();

chart->addSeries(series);

chart->setAnimationOptions(QChart::SeriesAnimations);

}

void MainWindow::createChartView()

{

view->setRenderHint(QPainter::Antialiasing);

chart->legend()->setVisible(false);

view->setChart(chart);

view->resize(ui->widget->size());

}

QBarSeries\* MainWindow::createSeries()

{

series->append(createSet());

return series;

}

QBarSet\* MainWindow::createSet()

{

QBarSet \*set = new QBarSet("data");

int\* arr;

arr = array->getArray();

for (int i=0; i<array->getN(); i++) {

int buf = 0;

while(i%1000 != 0 && i!=0){

buf += arr[i];

i++;

}

buf = buf/1000;

\*set << buf;

}

return set;

}

QBarCategoryAxis\* MainWindow::createXAxis()

{

QBarCategoryAxis \*axisX = new QBarCategoryAxis();

chart->addAxis(axisX, Qt::AlignBottom);

series->attachAxis(axisX);

return axisX;

}

QValueAxis\* MainWindow::createYAxis()

{

QValueAxis \*axisY = new QValueAxis();

chart->addAxis(axisY, Qt::AlignLeft);

series->attachAxis(axisY);

return axisY;

}

void MainWindow::changeSeries()

{

series->clear();

series->append(createSet());

timer->start(3000);

}

void MainWindow::sortFinished()

{

this->changeSeries();

timer->stop();

QMessageBox::information(NULL,QObject::tr("Информация"),

tr("Сортировка завершена"));

}

Листинг 2.4 – Заголовочный файл sort.h

/\*\*

\* @file sort.h

\* @brief bubble sort

\* @author Agapova Ekaterina

\*/

#ifndef SORT\_H

#define SORT\_H

#include <QObject>

class Sort: public QObject

{

Q\_OBJECT

Q\_PROPERTY(bool running READ running WRITE setRunning NOTIFY runningChanged)

private:

bool m\_running;

int \*array;

int n;

public:

/\*\*

\* @brief constructor sort

\* @param qty - quantity of elements

\* @param parent

\*/

explicit Sort(int qty, QObject \*parent = nullptr);

/\*\*

\* @brief destructor sort

\*/

~Sort();

/\*\*

\* @brief running

\* @return m\_running

\*/

bool running() const;

/\*\*

\* @brief get Array

\* @return array

\*/

int \*getArray() const;

/\*\*

\* @brief get N

\* @return n

\*/

int getN() const;

public slots:

/\*\*

\* @brief set m\_running

\* @param running

\*/

void setRunning(bool running);

/\*\*

\* @brief run

\*/

void run();

signals:

/\*\*

\* @brief finished

\*/

void finished();

/\*\*

\* @brief running changed

\* @param running

\*/

void runningChanged(bool running);

};

#endif // SORT\_H

Листинг 2.5 – Заголовочный файл sort.cpp

/\*\*

\* @file sort.cpp

\* @brief bubble sort

\* @author Agapova Ekaterina

\*/

#include "sort.h"

#include <QRandomGenerator>

Sort::Sort(int qty, QObject \*parent) : QObject(parent)

{

n = qty;

array = new int[n];

QRandomGenerator random;

for(int i=0; i<n; i++)

array[i] = random.bounded(1,1000000);

}

Sort::~Sort()

{

delete array;

}

bool Sort::running() const

{

return m\_running;

}

void Sort::setRunning(bool running)

{

if (m\_running == running)

return;

m\_running = running;

emit runningChanged(running);

}

void Sort::run()

{

while (m\_running){

int buf;

for (int i=0; i<n-1; i++){

for (int j=0; j<n;j++){

if (array[j]>array[j+1]){

buf=array[j];

array[j]=array[j+1];

array[j+1]=buf;

}

if(m\_running == false)

return;

}

}

setRunning(false);

}

emit finished();

}

int \*Sort::getArray() const

{

return array;

}

int Sort::getN() const

{

return n;

}

1. ЛИСТИНГ ВЫполнения программы

На рисунке 2 представлен интерфейс работы программы.

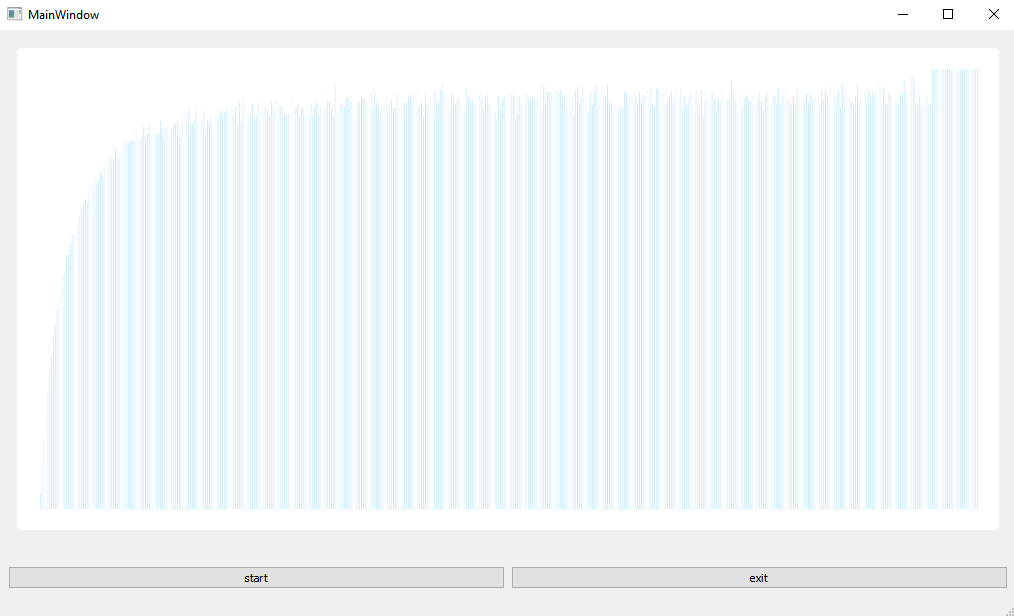


Рисунок 2 – Интерфейс программы

1. Вывод

В ходе выполнения лабораторной работы была написана программа, которая визуализирует процесс сортировки массива в 500 000 элементов с помощью QChart. Для возможности выхода из программы в любой момент времени процесс сортировки реализуется в отдельном потоке.