|  |  |
| --- | --- |
| для прик эмбл | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  **МИРЭА** | |
|  | Институт информационных технологий |
|  | Кафедра инструментального и прикладного программного обеспечения |

|  |  |
| --- | --- |
| **ОТЧЁТ**  **ЛАБОРАТОРНАЯ РАБОТЫ № 2,3,4** | |
| **дисциплина** | |
| **«Разработка программного приложения»**  *(наименование дисциплины)* | |
|  | |
| Выполнил студент группы \_ИКБО-06-16\_\_\_  *(учебная группа)* | *Кондратенко И.А.* |
| Принял доцент кафедры ИТС, доцент, к.т.н.  *должность, звание, ученая степень* | *Торхов Е.А.* |

|  |  |  |
| --- | --- | --- |
| Работы выполнены | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись руководителя)* |

Москва 2018

1. **Цель**

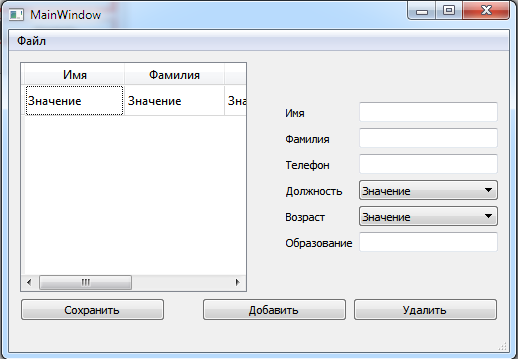
Разработка прикладного приложения с элементами встроенной графики Framework QT.

1. **Задание**

Построить простое приложение, имеющее таблицу с необходимой информацией и способное выполнять несколько базовых функций: изменение информации в ячейке таблицы, добавление/удаление элементов таблицы, загрузка/сохранение файла с данными. Реализовать передачу данных в таблицу с помощью реализации собственной модели.

1. **Описание программы**

В качестве базового шаблона была выбрана данная модель:



1. **Код программы**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONSTATS.H \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef CONSTANTS\_H

#define CONSTANTS\_H

#define COL\_SIZE 4

#define MAX\_COL\_SIZE 7

#endif // CONSTANTS\_H

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CUSTOMTABLEMODEL.H \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef CUSTOMTABLEMODEL\_H

#define CUSTOMTABLEMODEL\_H

#include <QModelIndex>

#include <QVariant>

#include <QAbstractTableModel>

class CustomTableModel : public QAbstractTableModel

{

private:

QList<QList<QString> > list;

int cols;

int rows;

public:

CustomTableModel(QList<QList<QString> > list, int cols, int rows);

QVariant *headerData*(int section, Qt::Orientation orientation, int role) const;

int *rowCount*(const QModelIndex &parent) const;

int *columnCount*(const QModelIndex &parent) const;

QVariant *data*(const QModelIndex &index, int role) const;

};

#endif // CUSTOMTABLEMODEL\_H

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FILE.H \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef FILE\_H

#define FILE\_H

#include <QString>

#include <QFile>

#include <QFileDialog>

#include <QTextStream>

#include <QMessageBox>

class File{

private:

QString file;

QList<QList<QString>>informMatrix;

void set\_info();

void errorDialogWindow(int num);

public:

File(QString file);

File();

QString loadInfo();

QList<QList<QString> > packInfo(QList<QList<QString> > Matrix);

QList<QList<QString> > getDataArray();

bool processInfo(QString raw\_str);

int getArraySize();

};

#endif // FILE\_H

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAINWINDOW.H \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QStandardItemModel>

#include <QMenu>

#include <QMenuBar>

#include <QDebug>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

private:

QList<QList<QString> > Matrix;

short int tableCols;

short int tableRows;

short int currIndex;

public:

explicit MainWindow(QWidget \*parent = 0);

void createDefaultTableView();

void createTableView();

void setDefaultMatrix();

void errorMessage();

~*MainWindow*();

private slots:

void on\_removePushButton\_clicked();

void on\_addPushButton\_clicked();

void on\_savePushButton\_clicked();

void on\_mainTable\_clicked(const QModelIndex &index);

void on\_uploadAction\_triggered();

void on\_SaveAsAction\_triggered();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CUSTOMTABLEMODEL.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "customtablemodel.h"

#include "constants.h"

#include <QDebug>

int CustomTableModel::*rowCount*(const QModelIndex &parent) const{

return this->rows;

}

int CustomTableModel::*columnCount*(const QModelIndex &parent) const{

return this->cols;

}

QVariant CustomTableModel::*data*(const QModelIndex &index, int role) const{

if (role == Qt::DisplayRole) {

QString answer = QString(this->list[index.row()][index.column()]);

return(QVariant(answer));

}

return(QVariant());

}

CustomTableModel::CustomTableModel(QList<QList<QString> > list, int cols, int rows){

this->list = list;

this->cols = cols;

this->rows = rows;

}

QVariant CustomTableModel::*headerData*(int section, Qt::Orientation orientation, int role) const{

if(role != Qt::DisplayRole) {

return(QVariant());

}

if(orientation == Qt::Horizontal) {

switch(section) {

case 0:

return(QVariant(QString::fromUtf8("Имя")));

case 1:

return(QVariant(QString::fromUtf8("Фамилия")));

case 2:

return(QVariant(QString::fromUtf8("Зарплата")));

case 3:

return(QVariant(QString::fromUtf8("Должность")));

case 4:

return(QVariant(QString::fromUtf8("Возраст")));

case 5:

return(QVariant(QString::fromUtf8("Образование")));

default:

return(QVariant());

}

}

return(QVariant());

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FILE.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "file.h"

#include "constants.h"

#include <QDebug>

void File::errorDialogWindow(int num){

switch(num){

case 1:{

QMessageBox \*msg = new QMessageBox;

msg->setText("Внимание");

msg->setInformativeText("В файле присутсвует запрещенная комбинация символов!!!");

msg->setStandardButtons(QMessageBox::Cancel);

msg->*exec*();

delete msg;

break;

}

case 2: {

QMessageBox \*msg = new QMessageBox;

msg->setText("Внимание");

msg->setInformativeText("Неверный формат!!!!");

msg->setStandardButtons(QMessageBox::Cancel);

msg->*exec*();

delete msg;

break;

}

default:{

qDebug() << "Неверное значение";

}

}

}

File::File(QString file){

this->file = file;

}

File::File(){}

QString File::loadInfo(){

QFileDialog \*f\_dial = new QFileDialog;

QFile \*fl = new QFile(f\_dial->getOpenFileName(0,"","","\*.csv"));

QString r\_str;

r\_str.clear();

QTextStream out(fl);

if(fl->*open*(QIODevice::ReadOnly | QIODevice::Text)){

while(!out.atEnd()){

r\_str = r\_str + out.readLine().append("\n");

}

fl->*close*();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Работа со спец. символами \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int size = r\_str.size();

for(int k = 0; k < size; k++){

if((k == 0) && (r\_str[k] == "\"")) r\_str.replace(k,1,"");

else if((r\_str[k] == "\"") && ((r\_str[k+1] == "\n") || (r\_str[k-1] == "\n"))) r\_str.replace(k,1,"");

if(r\_str[k] == "?") return "NULL";

// Обработка символа сплошного печатания

if(((r\_str[k] == ",") && (r\_str[k+1] == "\"") && (r\_str[k+2] == "\""))){

r\_str.replace(k+1,2,"?");

k++;

}

else if(((r\_str[k] == "\"") && (r\_str[k+1] == "\"") && (r\_str[k+2] == ","))) r\_str.replace(k,2,"?");

if(((k == 0) && (r\_str[k] == "\"") && (r\_str[k+1] == "\""))) r\_str.replace(k,2,"?"); // Крайне левое положение

else if((r\_str[k] == "\"") && (r\_str[k+1] == "\"") && (r\_str[k+3] == "\n")) r\_str.replace(k,2,"?"); // Крайне правое положение

//

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delete f\_dial;

delete fl;

return r\_str;

}

QList<QList<QString> >File::packInfo(QList<QList<QString> > Matrix){

QList<QList<QString> >rows;

QList<QString> cell;

QString str;

QString end, begin;

bool isVisit;

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

for(int j = 0; j < MAX\_COL\_SIZE - 1; j++){

int k = 0;

isVisit = false;

Matrix[i][j].append("\n");

while(Matrix[i][j].at(k) != "\n"){

qDebug() << Matrix[i][j].at(k).unicode();

if(!((Matrix[i][j].at(k).unicode() >= 1040 && Matrix[i][j].at(k).unicode() <= 1103)

|| (Matrix[i][j].at(k).unicode() >= 48 && Matrix[i][j].at(k).unicode() <= 57))

&& isVisit == false){

qDebug() << "here";

str = str + Matrix[i][j].at(k);

begin = "\"";

end = "\",";

isVisit = true;

}

else if(Matrix[i][j].at(k) == "\"") str = str + "\"\"";

else str = str + Matrix[i][j].at(k);

k++;

}

qDebug() << str;

str.prepend(begin);

begin.clear();

if(end.isEmpty()){

end = ",";

str.append(end);

}

else str.append(end);

end.clear();

cell.append(str);

qDebug() << cell;

str.clear();

Matrix[i][j].replace(Matrix[i][j].size() - 1,1,"");

}

rows.append(cell);

cell.clear();

}

return rows;

}

bool File::processInfo(QString raw\_str){

if(raw\_str == "NULL"){

this->errorDialogWindow(1);

return false;

}

//qDebug() << raw\_str;

QList<QString>txt\_row;

QString word;

for(int k = 0; k < raw\_str.size(); k++){

if(raw\_str[k] != "\n"){

switch (raw\_str[k].unicode()) {

case 34:{

while(raw\_str[k] == "\""){

k++;

}

break;

}

case 63:{

k++;

while(raw\_str[k] != "?"){

if(raw\_str[k] == "\""){

int counter = 0;

while(raw\_str[k] == "\""){

counter++;

k++;

if(counter == 4){

word = word + "\"";

counter = 0;

}

}

}

else{

word = word + raw\_str[k];

k++;

}

}

break;

}

case 44:{

//qDebug() << word;

txt\_row.append(word);

word.clear();

break;

}

default:{

word = word + raw\_str[k];

break;

}

}

}else{

//qDebug() << word;

txt\_row.append(word);

word.clear();

if(txt\_row.size() < MAX\_COL\_SIZE){

int buff = txt\_row.size() + 1;

for(int i = 0; i < MAX\_COL\_SIZE - buff; i++){

txt\_row.append("Значение");

}

}

if(txt\_row.size() >= MAX\_COL\_SIZE){

this->errorDialogWindow(2);

return false;

}

this->informMatrix.append(txt\_row);

txt\_row.clear();

}

}

return true;

}

QList<QList<QString>> File::getDataArray(){

return this->informMatrix;

}

int File::getArraySize(){

return this->informMatrix.size();

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAINWINDOW.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "file.h"

#include "constants.h"

#include "customtablemodel.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->posComboBox->addItem("Значение");

ui->posComboBox->addItem("Стажер");

ui->posComboBox->addItem("Программист");

ui->posComboBox->addItem("Ведущий программист");

ui->ageComboBox->addItem("Значение");

for(int i = 1; i < 100; i++){

ui->ageComboBox->addItem(QString::number(i));

}

this->tableCols = COL\_SIZE;

this->currIndex = 0;

this->createDefaultTableView();

}

void MainWindow::createTableView(){

CustomTableModel \*model = new CustomTableModel(this->Matrix, COL\_SIZE, this->Matrix.size());

ui->mainTable->*setModel*(model);

}

void MainWindow::setDefaultMatrix(){

this->tableRows = 1;

QList<QString>buff;

for(int j = 0; j < MAX\_COL\_SIZE; j++){

buff.append("Значение");

}

this->Matrix.append(buff);

buff.clear();

}

void MainWindow::errorMessage(){

QMessageBox \*msg = new QMessageBox;

msg->setText("Внимание");

msg->setInformativeText("В таблице нет записей!");

msg->setStandardButtons(QMessageBox::Cancel);

msg->*exec*();

}

void MainWindow::createDefaultTableView(){

this->setDefaultMatrix();

CustomTableModel \*model = new CustomTableModel(this->Matrix, COL\_SIZE, this->Matrix.size());

ui->mainTable->*setModel*(model);

}

void MainWindow::on\_removePushButton\_clicked(){

if(this->tableRows <= 0) this->errorMessage();

else{

int currRow = ui->mainTable->currentIndex().row();

this->Matrix.removeAt(currRow);

this->tableRows--;

this->createTableView();

}

}

void MainWindow::on\_addPushButton\_clicked(){

QList<QString>row;

row.append(ui->nameLineEdit->text());

row.append(ui->lastnameLineEdit->text());

row.append(ui->mfnumbeLineEdit->text());

row.append(ui->posComboBox->currentText());

row.append(ui->ageComboBox->currentText());

row.append(ui->graduateLineEdit->text());

this->tableRows++;

this->Matrix.append(row);

this->createTableView();

}

void MainWindow::on\_savePushButton\_clicked(){

this->Matrix[this->currIndex][0] = ui->nameLineEdit->text();

this->Matrix[this->currIndex][1] = ui->lastnameLineEdit->text();

this->Matrix[this->currIndex][2] = ui->mfnumbeLineEdit->text();

this->Matrix[this->currIndex][3] = ui->posComboBox->currentText();

this->Matrix[this->currIndex][4] = ui->ageComboBox->currentText();

this->Matrix[this->currIndex][5] = ui->graduateLineEdit->text();

this->createTableView();

}

void MainWindow::on\_mainTable\_clicked(const QModelIndex &index){

ui->nameLineEdit->clear();

ui->lastnameLineEdit->clear();

ui->mfnumbeLineEdit->clear();

ui->graduateLineEdit->clear();

this->currIndex = index.row();

QList<QString>row;

for(int i = 0; i < MAX\_COL\_SIZE - 1; i++){

row.append(this->Matrix[index.row()][i]);

}

ui->nameLineEdit->insert(row[0]);

ui->lastnameLineEdit->insert(row[1]);

ui->mfnumbeLineEdit->insert(row[2]);

ui->graduateLineEdit->insert(row[5]);

int buff = 1;

while(buff != ui->posComboBox->count()){

ui->posComboBox->setCurrentIndex(buff);

if(ui->posComboBox->currentText() == row[3]) break;

else if(ui->posComboBox->count() - buff == 1) ui->posComboBox->setCurrentIndex(0);

buff++;

}

buff = 1;

while(buff != ui->ageComboBox->count()){

ui->ageComboBox->setCurrentIndex(buff);

if(ui->ageComboBox->currentText() == row[4]) break;

else if(ui->ageComboBox->count() - buff == 1) ui->ageComboBox->setCurrentIndex(0);

buff++;

}

}

void MainWindow::on\_uploadAction\_triggered(){

File \*fl = new File("person.csv");

bool trigg = fl->processInfo(fl->loadInfo());

if(trigg == false) this->createDefaultTableView();

else{

this->Matrix = fl->getDataArray();

this->tableRows = fl->getArraySize();

this->createTableView();

}

}

void MainWindow::on\_SaveAsAction\_triggered(){

QFileDialog \*fd = new QFileDialog;

fd->setFileMode(QFileDialog::Directory);

fd->*exec*();

QFile saveFile(fd->directory().absolutePath()+"/output.csv");

delete fd;

if(saveFile.*open*(QIODevice::WriteOnly)){

QList<QList<QString> >info;

File \*fl = new File();

info = fl->packInfo(this->Matrix);

QTextStream in(&saveFile);

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

for(int j = 0; j < MAX\_COL\_SIZE - 1; j++){

in << info.at(i).at(j);

}

in << "\r\n";

}

saveFile.*close*();

}

}

MainWindow::~*MainWindow*(){

delete ui;

}

**Скриншоты и примеры работы**

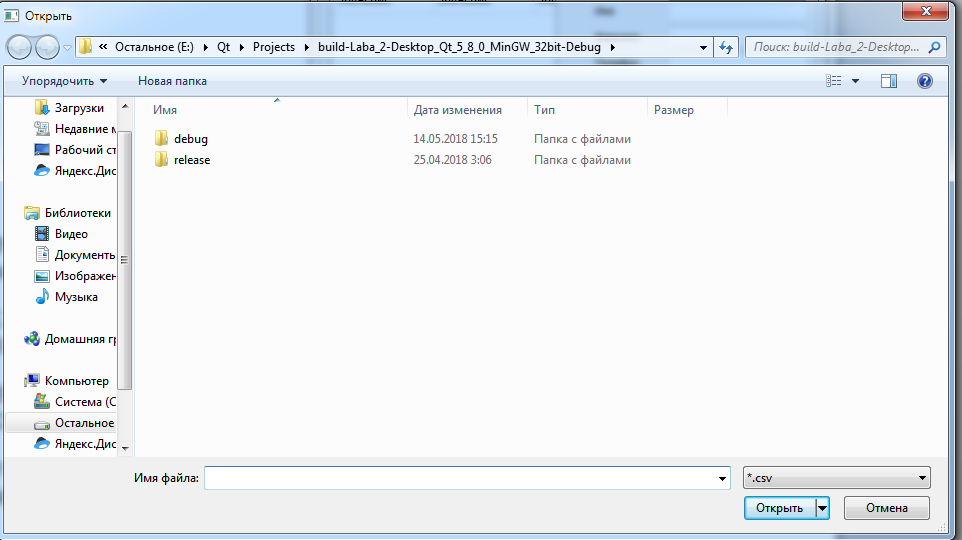
****

Рис. 1 Загрузка файла

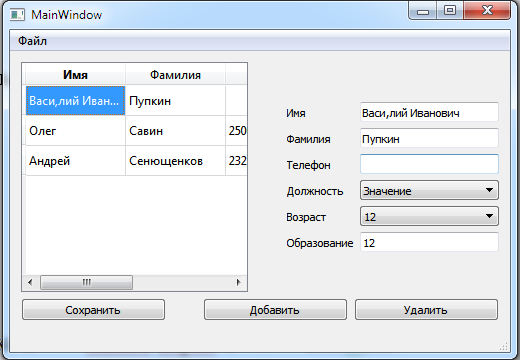


Рис. 2 Отображение элементов таблицы справа

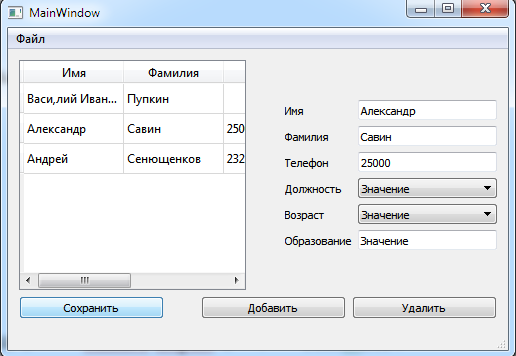


Рис. 3 Изменение данных в таблице

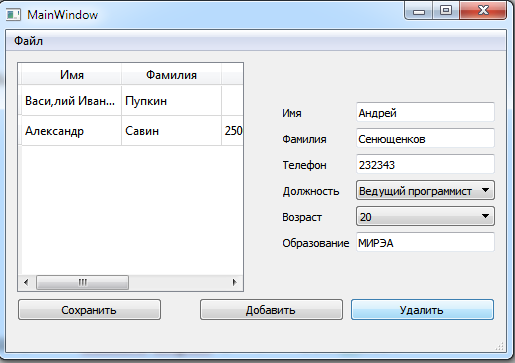


Рис. 4 Удаление элементов таблицы

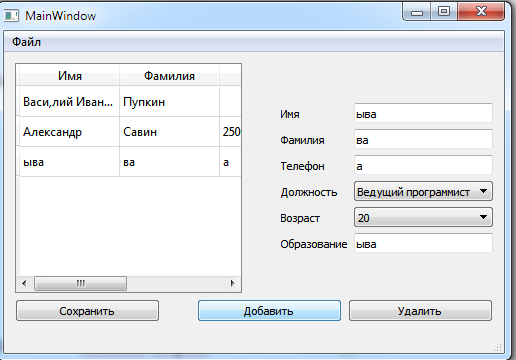


Рис. 5 Добавление элементов в таблицу

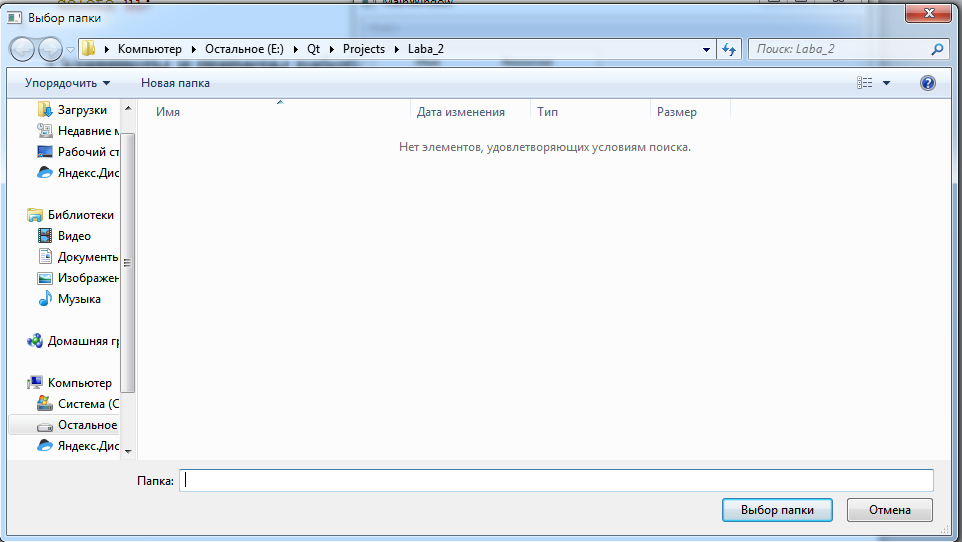


Рис. 6 Сохранение файла с данными

1. **Используемая литература**

*Герберт Шилдт.* Теория и практика С++ = Shildt's Expert C++. — СПб.: BHV — Санкт- Петербург, 1996.

Qt 5.3 Профессиональное программирование на C++. – СПб.:БХВ-Петербург, 2015. - 928 с.

Дейт К. Дж*.* Введение в системы баз данных — 8-е изд. — М.: Вильямс, 2005. — 1328 с.

КузнецовС. Д. Основы баз данных. — 2-е изд. — М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. — 484 с