Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

о лабораторной работе

Тема: «Автоматизированное рабочее место специалиста по вычислению доходов и расходов семьи»

Выполнили

Студенты группы РИС-23-1б

Шароглазов Е.А.

Мокрушин Н.Д.

Проверила доц. кафедры ИТАС

Полякова О.А.

Пермь 2024

**Содержание.**

Введение **3**

Разработка интуитивно понятного интерфейса**4**

Разработка программы, для вычисления и анализа семейного бюджета **6**

Заключение**8**

Приложение А**9**

Приложение Б**16**

Список используемых источников**17**

**Введение.**

Вести учет семейного бюджета сложно для неподготовленного человека, а для того чтобы это делать вручную, нужно знать хотя бы базовые знания работы с таблицами, было бы славно, если бы это сделала за тебя программа.

Отсюда появляется необходимость в создании программы, которая сама будет из вводимых данных получать общую статистику бюджета семьи. Для создания такой программы будем использовать фреймворк QT на языке C++.

Цель лабораторной работы – разработка приложения для упрощения вычисления бюджета семьи.

Задачи:

1. Разработка интуитивно понятного интерфейса

2.Разработка программы, для вычисления и анализа семейного бюджета

**1. Разработка интуитивно понятного интерфейса.**

Для разработки интерфейса был выбран фреймворк QT, в нем есть все самое необходимое для грамотно построенного интерфейса программы, а сама программа только дает пользователю вводить уже полученные или потраченные данные, программа не просчитывает налоги, т.к. предполагается, что большая часть налогов уже либо были включены в цену за определенные товары или услуги, или были высчитаны за пользователя(ЗП работодателем или выигрыш в лотерее государством),

Для ввода данных была использована таблица QTableWidget, так как таблица – одна из лучших визуализаций информации, в таблицу данные вводятся благодаря трем LineEdit’ам: «Примечание», «Сумма», «Дата», а выбрать доход или расход можно с помощью ComboBox. “Примечание” используется для подробностей, какой именно это доход или расход, например, если это доход то в примечании можно указать «Заработная плата отца», а если расход то «Ремонт авто», и в зависимости от положения ComboBox’а в VerticalHeader таблицы QTableWidget будет записан как: Доход/Расход + примечание в скобках, для “Суммы” же, достаточно написать сумму, которую пользователь получил или потратил, то есть, если это доход, то пользователь указывает сумму, и это число помещается в таблицу как положительное число, иначе же в таблицу помещается число отрицательное от того что было введено. Для даты дохода/расхода указывается дата в формате «дд.мм.гггг», иначе программа не позволит дальше вводить данные дальше.

Все три параметра в таблице добавляются после ввода по нажатию кнопки «Добавить источник», и в таблице самой первой строчкой появляются данные пользователя, для удаления строчки необходимо нажать на соответствующую строку и нажать на кнопку «Удалить строку».

Для более подробного анализа в интерфейсе программы присутствует Label для вывода общего бюджета семьи, из суммы в колонке «Сумма», и 2 диаграммы «Доход» и «Расход», в них из данных в таблице из колонки «Сумма» поступают данные в соответствии с тем доход это или расход, и тем самым можно не всматриваясь в таблицу найти самый большой расход или самый меньший доход.

**2. Разработка программы,** **для вычисления и анализа семейного бюджета.**

Для создания программы используем язык С++, так как это одно из условий для написания лабораторной работы.

Для начала необходимо добавлять данные в QTableWidget, у класса QLineEdit есть поле Text, это мы используем для создания переменных типа QString при нажатии кнопки PushButton, а уже на их основе будут созданы переменные типа QTableWidgetItem, т.к. только этот тип можно добавить в ячейки таблицы. Для того чтобы изменять VerticalHeader таблицы в соответствии с тем что написано в ComboBox, создаются конструкции if else, у класса QComboBox есть метод CurrentText(), который возвращает текст виджета, который на данный момент выбрал пользователь, это мы и используем чтобы узнать какой текст писать в VerticalHeader таблицы. После того как были написаны конструкции if else, и мы знаем какой выбор сделал пользователь при переключении ComboBox’а, нам необходимо добавлять уже заготовленные переменные типа QTableWidgetItem в соответствующие колонки строки. После добавления элементов в таблицу, программа передает в глобально созданные контейнеры данные, разные переменные передаются в разные векторы, тем самым заготавливая почву для других механик программы – «сохранения в файл» и «загрузки из файла», и в самом конце после того как данные были перемещенны в таблицу, очищаем LineEdit’ы чтобы этим не занимался пользователь.

После того как данные были перемещены в таблицу, необходимо обновить диаграммы и Label для анализа бюджета, для этого реализован метод updateChartsAndLabel(),который обновляет в соответствии с данными из таблицы диаграммы и Label.

Для удаления была реализован метод get\_clicked\_row\_index(), для получения индекса строчки по клику пользователя, и этот индекс мы используем для удаления строчки, в конце удаляются соответствующие индексы из контейнеров, полученные методом описанным выше, для соответствия данных, и вызов метода для обновления диаграмм и лейбла.

Для сохранения данных в файл, используем данные из заготовленных заранее контейнеров, переписываем данные сначала из первого контейнера в файл, из второго и т.д.. после этого предлагаем пользователю выбрать путь для сохранения файла.

Для выгрузки из файла предлагаем пользователю выбрать файл который программа будет считывать, и программа построчно сначала заполнит VerticalHeader’ы таблицы, а дальше по колонкам заполнит данные, в конце вызовется метод для обновления данных в лейбле и диаграммах

Также реализован метод saveToPdf() для экспорта данных в файл формата PDF, передавая в файл изображения таблицы и диаграмм, и позволяя пользователю выбрать путь для экспорта.

Для создания нового файла реализован метод confirmCreateFile(), который сначала предупреждает пользователя о потери введённых данных, и предлагает либо продолжить либо отменить действие, если пользователь соглашается продолжить, то программа, очищает все данные и контейнеры.

Вкладка About предлагает пользователю узнать о создателях программы и причине её создания

**Заключение**

В ходе выполнения лабораторной работы был получен опыт работы с фреймворком QT и возросло умение работы с языком С++.

Данная программа может быть использована для повседневного вычисления доходов и расходов семьи, но не может быть использована для более сложных вычислений с налогами.

В дальнейшем программа может быть улучшена, добавлено вычисление с учетом налогов, редактирование уже существующих данных без удаления строк, и улучшение дизайна самого приложения для эстетичного вида.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

**Код файла qtclass.h**

#ifndef QTCLASS\_H

#define QTCLASS\_H

#include<QtCharts>

#include<QChartView>

#include<QPieSeries>

#include<QPieSlice>

#include <QMainWindow>

#include <QPdfWriter>

#include <QPainter>

QT\_BEGIN\_NAMESPACE

*namespace* **Ui** { *class* **QtClass**; }

QT\_END\_NAMESPACE

*class* **QtClass** : *public* QMainWindow

{

Q\_OBJECT

*public*:

**QtClass**(QWidget \*parent = *nullptr*);

~***QtClass***();

*private* slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_3\_clicked**();

void **saveToPdf**();

int **get\_clicked\_row\_index**();

void **on\_actionSave\_to\_triggered**();

void **showAboutWindow**();

void **confirmCreateFile**();

void **on\_actionLoad\_from\_triggered**();

void **updateChartsAndLabel**();

void **sortTable**(int column);

void **on\_pushButton\_2\_clicked**();

void **maximised**(QTableWidget table);

*private*:

Ui::QtClass \*ui;

QPieSeries \*series;

QChart \*chart;

QChartView \*chartview;

QPieSeries \*series2;

QChart \*chart2;

QChartView \*chartview2;

};

#endif *//* *QTCLASS\_H*

**Код файла main.cpp**

#include "qtclass.h"

#include <QApplication>

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

{

QApplication a(*argc*, *argv*);

QtClass w;

w.show();

*return* a.exec();

}

**Код файла qtclass.cpp**

#include "qtclass.h"

#include "ui\_qtclass.h"

#include <QPdfWriter>

#include <QPainter>

#include <QPixmap>

QtClass::**QtClass**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(*new* Ui::QtClass)

{

*//* *Первая* *диаграмма* *(Доходы)*

ui->setupUi(*this*);

series = *new* QPieSeries();

chart = *new* QChart();

chart->addSeries(*series*);

chart->setTitle("Доходы");

chartview = *new* QChartView(*chart*);

chartview->setMinimumSize(200, 200);

*//* *Вторая* *диаграмма* *(Расходы)*

series2 = *new* QPieSeries();

chart2 = *new* QChart();

chart2->addSeries(*series2*);

chart2->setTitle("Расходы");

chartview2 = *new* QChartView(*chart2*);

chartview2->setMinimumSize(200, 200);

*//* *Для* *даты* *формата* *дд.мм.гггг*

QRegularExpression rx("\\d{2}\\.\\d{2}\\.\\d{4}"); *//* *Регулярное* *выражение* *для* *формата* *"дд.мм.гг"*

QValidator \*validator = *new* QRegularExpressionValidator(rx, *this*);

ui->lineEdit\_4->setValidator(validator);

QHBoxLayout \*layout = *static\_cast*<QHBoxLayout\*>(ui->horizontalFrame->*layout*());

layout->addWidget(*chartview*);

layout->addWidget(*chartview2*);

*//* *Получаем* *указатель* *на* *действие* *"Export* *File* *to* *PDF"*

QAction \*exportAction = ui->menuExport\_File->actions().at(0); *//* *Предполагаем,* *что* *это* *первое* *действие* *в* *меню*

*//* *Подключаем* *сигнал* *triggered()* *к* *слоту* *exToPdf()*

connect(exportAction, &QAction::triggered, *this*, &QtClass::saveToPdf);

connect(ui->menuAbout, &QMenu::triggered, *this*, &QtClass::showAboutWindow);

connect(ui->actionCreate\_File, &QAction::triggered, *this*, &QtClass::confirmCreateFile);

connect(ui->tableWidget->horizontalHeader(), &QHeaderView::sectionClicked, *this*, &QtClass::sortTable);

}

QtClass::~***QtClass***()

{

*delete* ui;

}

QVector<int> vec;

QVector<QString> vec1;

QVector<QString> vec2;

QVector<QString> vec3;

int QtClass::**get\_clicked\_row\_index**() {

*return* ui->tableWidget->currentRow();

}

int **sum\_vec**(QVector<int> vec)

{

int sum = 0;

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

sum+=vec[i];

}

*return* sum;

}

void QtClass::**updateChartsAndLabel**() {

ui->label->setText("Семейные сбережения: " + QString::number(sum\_vec(vec)));

series->clear();

series2->clear();

*for* (int i = 0; i < vec1.size(); ++i) {

bool ok;

int value = vec1[i].toInt(*&ok*);

*if* (!ok) {

*continue*;

}

*if* (vec2[i].startsWith("Доход")) {

series->append(vec2[i], value);

} *else* *if* (vec2[i].startsWith("Расход")) {

series2->append(vec2[i], value);

}

}

chart->update();

chart2->update();

}

void QtClass::**on\_pushButton\_clicked**()

{

QString str = ui->lineEdit\_2->text();

QString strneg = "-" + str;

QString str1 = ui->lineEdit\_4->text();

QString str2 = ui->lineEdit\_7->text();

QTableWidgetItem \*tbl = *new* QTableWidgetItem(str);

QTableWidgetItem \*tblneg = *new* QTableWidgetItem(strneg);

QTableWidgetItem \*tbl1 = *new* QTableWidgetItem(str1);

ui->tableWidget->insertRow(0);

*if* (ui->comboBox->currentText() == "Доход") {

ui->tableWidget->setVerticalHeaderLabels(QStringList() << "Доход (" + str2 + ")");

ui->tableWidget->setItem(0,0,*tbl*);

ui->tableWidget->setItem(0,1,*tbl1*);

vec.push\_front(str.toInt());

vec1.push\_front(str);

vec2.push\_front("Доход (" + str2 + ")");

vec3.push\_front(str1);

tbl->*setData*(Qt::*UserRole*, str.toInt()); *//* *Для* *сортировки* *по* *сумме*

tbl1->*setData*(Qt::*UserRole*, QDate::fromString(str1, "dd.MM.yyyy").toJulianDay());

}

*else* {

ui->tableWidget->setVerticalHeaderLabels(QStringList() << "Расход (" + str2 + ")");

ui->tableWidget->setItem(0,0,*tblneg*);

ui->tableWidget->setItem(0,1,*tbl1*);

vec.push\_front(-str.toInt());

vec1.push\_front(strneg);

vec2.push\_front("Расход (" + str2 + ")");

vec3.push\_front(str1);

tblneg->*setData*(Qt::*UserRole*, -str.toInt()); *//* *Для* *сортировки* *по* *сумме*

tbl1->*setData*(Qt::*UserRole*, QDate::fromString(str1, "dd.MM.yyyy").toJulianDay());

}

updateChartsAndLabel();

ui->lineEdit\_2->clear();

ui->lineEdit\_4->clear();

ui->lineEdit\_7->clear();

}

void QtClass::**on\_pushButton\_3\_clicked**()

{

int clickedRow = get\_clicked\_row\_index();

*if* (clickedRow != -1) {

vec.remove(clickedRow);

vec1.remove(clickedRow);

vec2.remove(clickedRow);

vec3.remove(clickedRow);

ui->label->setText("Семейные сбережения: " + QString::number(sum\_vec(vec)));

updateChartsAndLabel();

ui->tableWidget->removeRow(clickedRow);

}

}

void QtClass::**saveToPdf**()

{

QString fileName = QFileDialog::getSaveFileName(*this*, "Сохранить как PDF", QDir::homePath(), "PDF (\*.pdf)");

*if* (!fileName.isEmpty()) {

QPdfWriter writer(fileName);

writer.*setPageSize*(QPagedPaintDevice::*A4*);

QPainter painter(*&writer*);

QPixmap tablePixmap = ui->tableWidget->grab();

tablePixmap = tablePixmap.scaledToWidth(writer.width() \* 0.8, Qt::*SmoothTransformation*);

painter.drawPixmap(10, 10, tablePixmap);

QPixmap chartPixmap1 = chartview->grab();

chartPixmap1 = chartPixmap1.scaledToWidth(writer.width() \* 0.4, Qt::*SmoothTransformation*);

QPixmap chartPixmap2 = chartview2->grab();

chartPixmap2 = chartPixmap2.scaledToWidth(writer.width() \* 0.4, Qt::*SmoothTransformation*);

painter.drawPixmap(10, tablePixmap.height() + 20, chartPixmap1);

painter.drawPixmap(chartPixmap1.width() + 20, tablePixmap.height() + 20, chartPixmap2);

painter.end();

}

}

void QtClass::**on\_actionSave\_to\_triggered**()

{

QString fileName = QFileDialog::getSaveFileName(*this*, tr("Сохранить файл"), tr("Имя файла"), tr("Файлы данных (\*.dat)"));

*if* (!fileName.isEmpty()) {

QFile file(fileName);

*if* (file.*open*(QIODevice::*WriteOnly* | QIODevice::*Text*)) {

QTextStream writeStream(*&file*);

*//* *Save* *vec2* *data*

*for* (*const* QString& label : vec2) {

writeStream << label << "\n";

}

*//* *Save* *vec1* *data*

*for* (*const* QString& value : vec1) {

writeStream << value << "\n";

}

*//* *Save* *vec3* *data*

*for* (*const* QString& value : vec3) {

writeStream << value << "\n";

}

file.*close*();

}

}

}

void QtClass::**on\_actionLoad\_from\_triggered**()

{

QString fileName = QFileDialog::getOpenFileName(*this*, tr("Открыть файл"), tr("Имя файла"), tr("Файлы данных (\*.dat)"));

*if* (!fileName.isEmpty()) {

QFile file(fileName);

*if* (file.*open*(QIODevice::*ReadOnly* | QIODevice::*Text*)) {

vec2.clear();

vec1.clear();

vec3.clear();

QTextStream in(*&file*);

QString line;

*while* (!(line = in.readLine()).isNull()) {

*if* (!line.isEmpty()) { *//* *Check* *for* *empty* *lines*

*if* (line.startsWith("Доход") || line.startsWith("Расход")) {

vec2.push\_back(line);

} *else* {

vec1.push\_back(line);

*if* (!(line = in.readLine()).isNull()) {

vec3.push\_back(line);

}

}

}

}

vec.clear();

*for* (int i = 0; i < vec1.size(); ++i) {

bool ok;

int value = vec1[i].toInt(*&ok*);

*if* (!ok) {

*continue*;

}

vec.push\_back(value);

}

file.*close*();

*//* *Clear* *table* *before* *loading* *new* *data*

ui->tableWidget->clear();

ui->tableWidget->setHorizontalHeaderLabels(QStringList() << "Сумма" << "Дата");

ui->tableWidget->setRowCount(vec1.size());

QStringList headerLabels;

*for*(*const* QString& label : vec2) {

headerLabels.append(label);

}

ui->tableWidget->setVerticalHeaderLabels(headerLabels);

*for* (int i = 0; i < vec1.size(); ++i) {

QTableWidgetItem \*item = *new* QTableWidgetItem(vec1[i]);

QTableWidgetItem \*item2 = *new* QTableWidgetItem(vec3.value(i));

ui->tableWidget->setItem(i, 0, *item*);

ui->tableWidget->setItem(i, 1, *item2*);

}

updateChartsAndLabel();

ui->label->setText("Семейные сбережения: " + QString::number(sum\_vec(vec)));

}

}

}

void QtClass::**showAboutWindow**() {

QMessageBox aboutWindow(*this*);

aboutWindow.setWindowTitle("About");

QString aboutText = "<p style=\"font-size: 14px; line-height: 1.5em;\">Приложение разработали студенты ПНИПУ группы РИС-23-1б <br>Шароглазов Егор и Никита Мокрушин в рамках выполнения<br> творческой работы доцента кафедры ИТАС Поляковой Ольги Андреевны.</p>";

aboutWindow.setText(aboutText);

aboutWindow.*exec*();

}

void QtClass::**confirmCreateFile**() {

QMessageBox msgBox(*this*);

msgBox.setWindowTitle("Подтверждение");

msgBox.setText("У вас остались несохраненные изменения. Вы уверены? ");

msgBox.setStandardButtons(QMessageBox::*Yes* | QMessageBox::*Cancel*);

msgBox.setDefaultButton(QMessageBox::*Cancel*);

int result = msgBox.*exec*();

*if* (result == QMessageBox::*Yes*) {

ui->lineEdit\_2->clear(); *//* *Очистка* *поля* *"Сумма"*

ui->lineEdit\_4->clear(); *//* *Очистка* *поля* *"Период"*

ui->lineEdit\_7->clear(); *//* *Очистка* *поля* *"Примечание"*

ui->tableWidget->clearContents();

ui->tableWidget->setRowCount(0);

series->clear();

series2->clear();

chart->update();

chart2->update();

vec.clear();

vec1.clear();

vec2.clear();

vec3.clear();

updateChartsAndLabel();

}

}

void QtClass::**sortTable**(int column) {

*static* Qt::SortOrder order = Qt::*AscendingOrder*; *//* *Запоминаем* *порядок* *сортировки*

ui->tableWidget->sortItems(column, order);

order = (order == Qt::*AscendingOrder*) ? Qt::*DescendingOrder* : Qt::*AscendingOrder*; *//* *Меняем* *порядок* *для* *следующего* *нажатия*

}

void QtClass::**maximised**(QTableWidget table)

{

int max = 0;

QTableWidgetItem \*item = *new* QTableWidgetItem(max);

*for*(int i = 0; i < ui->tableWidget->rowCount(); i++)

{

*if*(table.item(i,0) < item)

{

}

}

}

**ПРИЛОЖЕНИЕ Б**

**Скриншоты результатов работы программы**

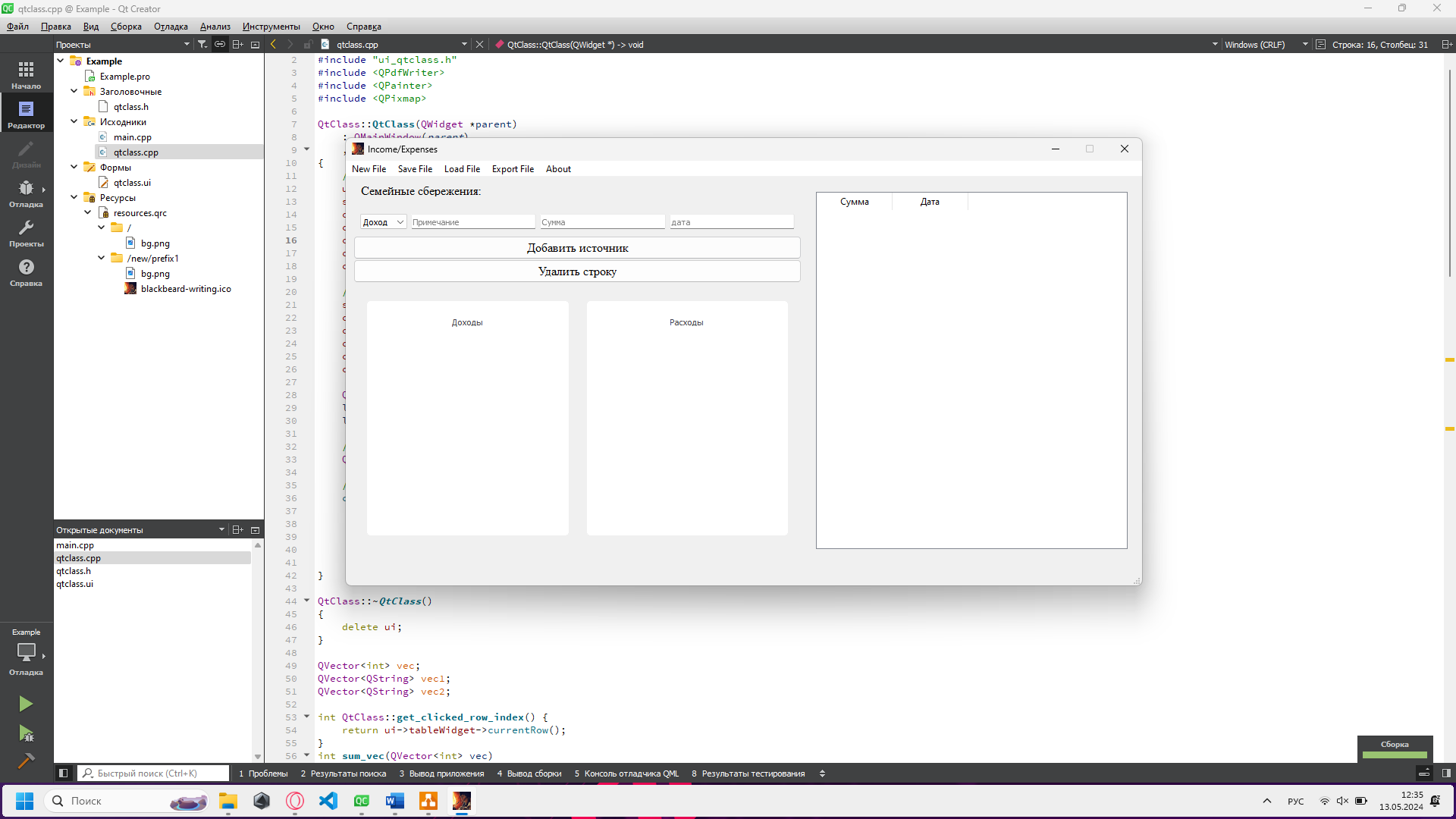


Рис.1 - Стандартный интерфейс программы

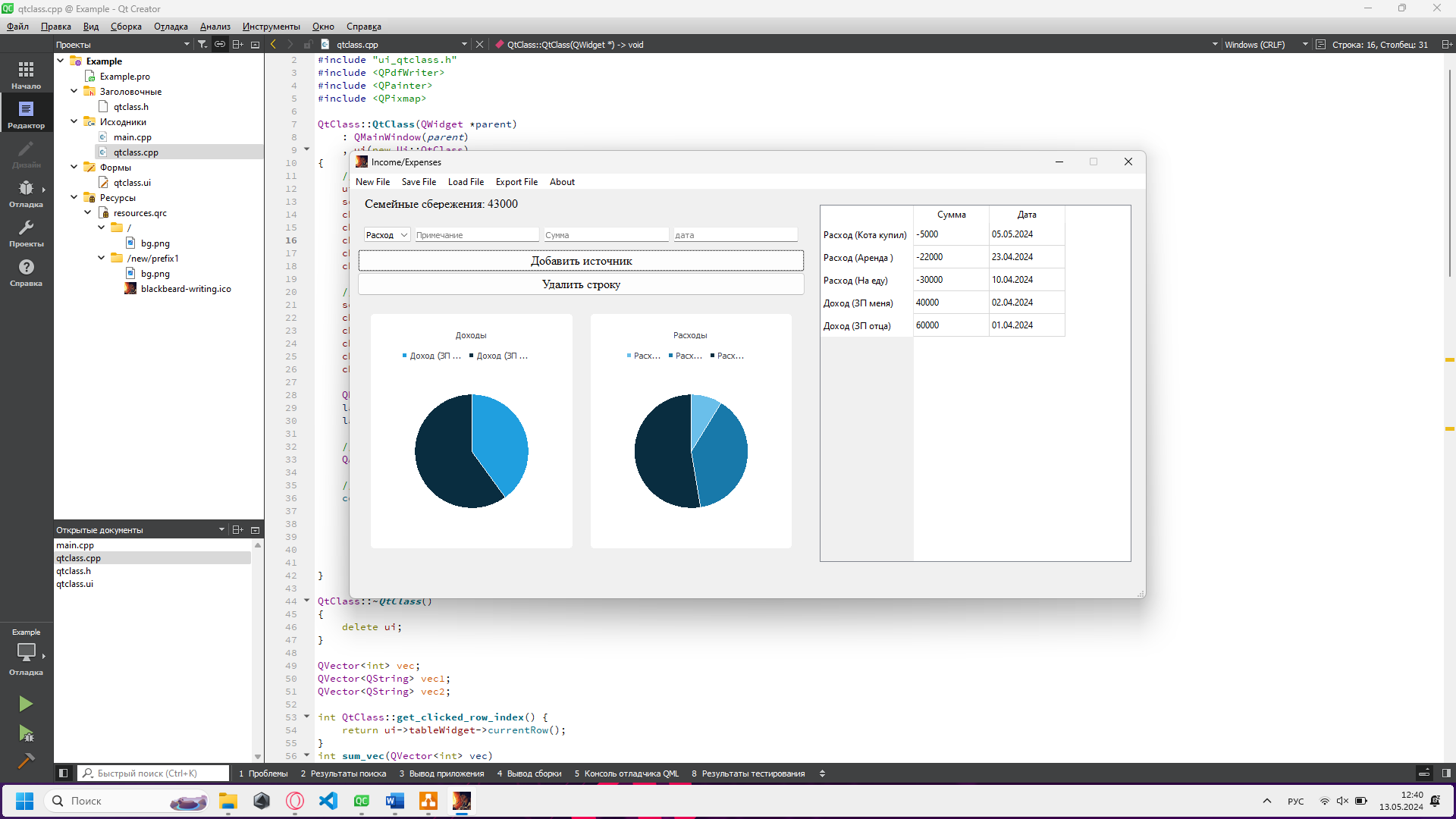


Рис.2 - Программа заполненная данными

**Список использованных источников**

# 1. QT Assistant (Документация): https://doc.qt.io/qt-6/qtassistant-index.html

# 2. Создание приложения Qt на С++: http://doc.crossplatform.ru/qtcreator/2.0.1/creator-writing-program.html

# 3. Иванова, Г.С. Создание пользовательских интерфейсов в программах на С++ с использованием библиотеки Qt [Электронный ресурс]: Учебное пособие по дисциплинам «Объектно-ориентированное программирование», «Системное программное обеспечение» Моск. гос. техн. ун-т: МГТУ им. Баумана, 2011