|  |  |
| --- | --- |
| 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**  **ПО ЛАБОРАТОРНЫМ РАБОТАМ** | |
| **дисциплина** | |
| **«Разработка программного приложения»** | |
|  | |
| Выполнил студент группы \_ИКБО-06-16\_\_\_ | *Васюкова Е.А.* |
| Принял доцент кафедры ИТС, доцент, к.т.н. | *Торхов Е.А.* |

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

Москва 2018

**Лабораторная работа №1**

1. **Цель**

Изучение постраничной навигации, корректность ее отображения в пограничных ситуациях.

1. **Задание**

Отобразить постраничную навигацию, где общее количество страниц, номер текущей страницы и ширину навигации задает пользователь.

1. **Описание программы**

Для реализации программы была выбрана следующая схема:

1 … k-2 k-1 [k] k+1 k+2 … N,

где k – номер выбранной страницы, а N – количество страницы (т.е. номер последней из них).

Алгоритм работы:

1. Если количество страниц <= 5, то в массив записываются все страницы, и возвращается количество страниц, как ограничение для вывода массива.
2. Если номер выбранной страницы <= 4, то в массив записываются все 5 первых значений, 0 и номер последней страницы.
3. Если номер выбранной страницы >= N-3, то в массив записываются 1, 0, N-4, N-3, N-2, N-1, N, и возвращается количество записанных элементов.
4. Если номер выбранной страницы > 3 и < N-2, то значения в массив записываются, как показано на схеме.

Организация вывода: вывод производится посредством массива чисел. 0 – зарезервированное число для троеточия.

1. **Код программы**

#include <iostream>

using std :: cout;

using std :: cin;

using std :: endl;

int f(int \*array, int a, int b)

{

if (a <= 5)

{

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

{

array[i] = i+1;

}

return a;

}

if (b <=4)

{

array[0] = 1;

array[1] = 2;

array[2] = 3;

array[3] = 4;

array[4] = 5;

array[5] = 0;

array[6] = a;

return 7;

}

if (b>=a-3)

{

array[0] = 1;

array[1] = 0;

array[2] = a-4;

array[3] = a-3;

array[4] = a-2;

array[5] = a-1;

array[6] = a;

return 7;

}

if ((b>3)&&(b<a-2))

{

array[0] = 1;

array[1] = 0;

array[2] = b-2;

array[3] = b-1;

array[4] = b;

array[5] = b+1;

array[6] = b+2;

array[7] = 0;

array[8] = a;

return 9;

}

}

int main ()

{

cout << "Enter the total number of pages ";

int a, b;

cin >> a;

cout << "Enter the current page number ";

cin >> b;

int \*array = new int[9];

int n = f(array,a,b);

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

{

if (array[i] == 0)

{

cout << "... ";

}

else

{

if (array[i] == b)

{

cout << "[" << array[i] << "]" << " ";

}

else

{

cout << array[i] << " ";

}

}

}

return 0;

}

1. **Скриншоты и примеры работы**

Результат работы программы представлен на рисунках 1-4.

Рис. 1-4. Возможные виды запросов.

1. **Заключение**

В результате данной лабораторной работы произошло знакомство с постраничной навигацией, были выявлены и обработаны пограничные случаи.

**Лабораторная работа №2**

1. **Цель**

Изучение формата CSV, процесса создания окна и таблицы в графическом приложении.

1. **Задание**

Реализовать графическую программу, которая создаёт окно и таблицу в нём. Реализовать получение данных из CSV файла для таблицы.

1. **Описание программы**

Алгоритм работы:

1. Пока не будут заполнены все строки и столбцы, если обнаружен конец файла, то записать слово в таблицу и закончить чтение из файла.
2. Если встречена запятая записать слово в таблицу и перейти к следующему столбцу.
3. Если встречена кавычка, то записать всё в слово вплоть до следующей кавычки. Если встречено две кавычки подряд, то записать одну из них в слово. Если встречен перенос строки, то записать это сообщение в таблицу и перейти к следующему столбцу.
4. Если не встречена ни запятая, ни кавычка, то просто записать символ в слово.
5. **Код программы**

Файл widget.h:

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

#include <QTableWidget>

namespace Ui {

class Widget;}

class Widget : public QWidget

{

Q\_OBJECT

QTableWidget \*tableWidget = new QTableWidget(1, 4, this);

public:

explicit Widget(QWidget \*parent = 0);

~*Widget*();

private:

Ui::Widget \*ui;

};

#endif // WIDGET\_H

Файл main.cpp:

#include "widget.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

Widget w;

w.show();

return a.exec();

}

Файл widget.cpp:

#include "widget.h"

#include "ui\_widget.h"

#include <QTableWidget>

#include <QFile>

#include <QTextStream>

Widget::Widget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Widget)

{

ui->setupUi(this);

this->resize(450, 300);

QStringList headers;

headers << "ID" << "ФИО" << "Зарплата" << "Отдел";

tableWidget->setHorizontalHeaderLabels(headers);

tableWidget->resize(450,300);

QFile file("C:\\Users\\AcErA\\Desktop\\table\_lr2\\database.txt");

if (file.*open*(QIODevice::ReadOnly))

{

QTextStream fin (&file);

QChar s;

fin >> s;

int end = QString(s).toInt();

tableWidget->setRowCount(end);

int row = 0;

fin >> s;

fin >> s;

fin >> s;

int counter = 0;

QString word = "";

while (row < end)

{

while (counter < 4)

{

if (s == '\x0')

{

QTableWidgetItem \*newItem1 = new QTableWidgetItem();

newItem1->setText(word);

tableWidget->setItem(row, counter, newItem1);

break;

}

if ((QString(s) == ',')||(QString(s) == '\r'))

{

if (QString(s) == '\r')

{

fin >> s;

}

QTableWidgetItem \*newItem1 = new QTableWidgetItem();

newItem1->setText(word);

tableWidget->setItem(row, counter, newItem1);

word = "";

counter++;

fin>>s;

continue;

}

if (QString(s) == '"')

{

fin >> s;

if (QString(s) == '"')

{

word.push\_back(QString(s));

}

else

{

if (QString(s) == '\r')

{

fin >> s;

if (QString (s) == '\n')

{

QTableWidgetItem \*newItem1 = new QTableWidgetItem();

newItem1->setText("ПЕРЕНОС СТРОКИ");

tableWidget->setItem(row, counter, newItem1);

fin >> s;

fin >> s;

fin >> s;

counter++;

continue;

}

}

while (QString(s)!='"')

{

word.push\_back(QString(s));

fin >> s;

}

fin >> s;

if (QString(s) == '"')

{

word.push\_back(QString(s));

fin >> s;

}

}

}

if ((QString(s) != ',')&&(QString(s) != '"'))

{

word.push\_back(QString(s));

fin >> s;

}

}

counter = 0;

row ++;

}

file.*close*();

}

}

Widget::~*Widget*()

{

delete ui;

}

Файл database.txt:

3

137,Иванов\_И.И.,,Развития

115,Петров\_П.П.,65000,"

"

253,"Ульин"", Ф.Н.",75000,Производства

1. **Скриншоты и примеры работы**

Результат работы программы представлен на рисунке 1.

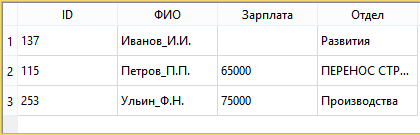


Рис. 1. Окно приложения.

1. **Заключение**

В результате данной лабораторной работы был изучен формат CSV, процесс создания окна и таблицы в графическом приложении. Была реализована графическая программа, создающая окно и таблицу в нём, а также получение данных из CSV файла для таблицы.

**Лабораторная работа №3-4**

1. **Цель**

Изучение основ построения интерфейса и модели данных.

1. **Задание**

Добавить форму к интерфейсу. При клике на запись отображать все данные из записи. Добавить кнопку «Сохранить» на форму, при сохранении – модифицировать данные в таблице. Добавить кнопку «Удалить» на форму, при удалении – удалять строку из таблицы. Добавить возможность загрузки и сохранения файла с данными из меню. В одном из столбцов добавить возможность выбора из всех данных, что есть в данном столбце + возможность написать свой вариант. Реализовать передачу данных в таблицу с помощью реализации собственной модели на основе класса QAbstractTableModel.

1. **Описание программы**

С помощью класса QAbstractTableModel в Qt можно легко представить данные в нужном формате, переопределив некоторые функции, а именно rowCount, columnCount, data, headerData.

1. **Код программы**

Файл widget.h:

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

#include <QAbstractTableModel>

#include <QVariant>

#include <QVector>

#include <QDebug>

namespace Ui {

class Widget;

}

struct MyData

{

QString surname;

QString phone;

QString group;

QString institute;

};

class myTableModel : public QAbstractTableModel

{

QVector <MyData> vec;

public:

myTableModel (QVector <MyData> vect)

{

vec = vect;

}

private:

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

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

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

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

};

class Widget : public QWidget

{

Q\_OBJECT

QVector <MyData> vec;

void show\_table(QVector <MyData> vec);

void add();

void del();

public:

explicit Widget(QWidget \*parent = 0);

~*Widget*();

private slots:

void on\_upload\_clicked(bool checked);

void on\_save\_clicked(bool checked);

void on\_delete\_2\_clicked(bool checked);

void on\_add\_clicked(bool checked);

private:

Ui::Widget \*ui;

};

#endif // WIDGET\_H

Файл main.cpp:

#include "widget.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

Widget w;

w.show();

return a.exec();

}

Файл widget.cpp:

#include "widget.h"

#include "ui\_widget.h"

#include <QAbstractTableModel>

#include <QFile>

#include <QTextStream>

#include <QVariant>

#include <QFileDialog>

using std :: iterator;

Widget::Widget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Widget)

{

ui->setupUi(this);

}

int myTableModel::*rowCount*(const QModelIndex &parent) const

{

return vec.size();

}

int myTableModel::*columnCount*(const QModelIndex &parent) const

{

return 4;

}

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

{

if (role == Qt::DisplayRole)

{

if (index.column() == 0)

{

QString answer = vec[index.row()].surname;

return QVariant(answer);

}

if (index.column() == 1)

{

QString answer = vec[index.row()].phone;

return QVariant(answer);

}

if (index.column() == 2)

{

QString answer = vec[index.row()].group;

return QVariant(answer);

}

if (index.column() == 3)

{

QString answer = vec[index.row()].institute;

return QVariant(answer);

}

}

return QVariant();

}

QVariant myTableModel::*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("Фамилия"));

case 1:

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

case 2:

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

case 3:

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

default:

return QVariant();

}

}

return QVariant();

}

void Widget::show\_table(QVector <MyData> vec)

{

myTableModel \*myModel = new myTableModel(vec);

ui->tableView->*setModel*(myModel);

}

Widget::~*Widget*()

{

delete ui;

}

void Widget::on\_upload\_clicked(bool checked)

{

QString fileName = QFileDialog::getOpenFileName(this,tr("Загрузить файл"),

"",tr("Файлы (\*.txt)"));

QFile file(fileName);

if (file.*open*(QIODevice::ReadOnly))

{

QTextStream fin (&file);

while (!fin.atEnd())

{

MyData new\_el;

fin >> new\_el.surname >> new\_el.phone >> new\_el.group >>

new\_el.institute;

vec.push\_back(new\_el);

}

}

file.*close*();

show\_table(vec);

}

void Widget::on\_save\_clicked(bool checked)

{

QFile file ("C:\\Users\\AcErA\\Desktop\\model\_lr3\\result.txt");

QTextStream fout (&file);

QVector <MyData> :: iterator it;

if (file.*open*(QIODevice::WriteOnly))

{

for (it = vec.begin(); it < vec.end(); it++)

{

fout << (\*it).surname << " " << (\*it).phone << " " << (\*it).group << " " <<(\*it).institute << " ";

}

}

file.*close*();

}

void Widget::on\_delete\_2\_clicked(bool checked)

{

if (ui->tableView->currentIndex().row() >= 0)

{

vec.removeAt(ui->tableView->currentIndex().row());

}

show\_table(vec);

}

void Widget::on\_add\_clicked(bool checked)

{

MyData new\_el;

new\_el.surname = ui->lineEdit->text();

new\_el.phone = ui->lineEdit\_2->text();

new\_el.group = ui->lineEdit\_3->text();

new\_el.institute = ui->comboBox->currentText();

vec.push\_back(new\_el);

show\_table(vec);

}

1. **Скриншоты и примеры работы**

Результат работы программы представлен на рисунке 1.

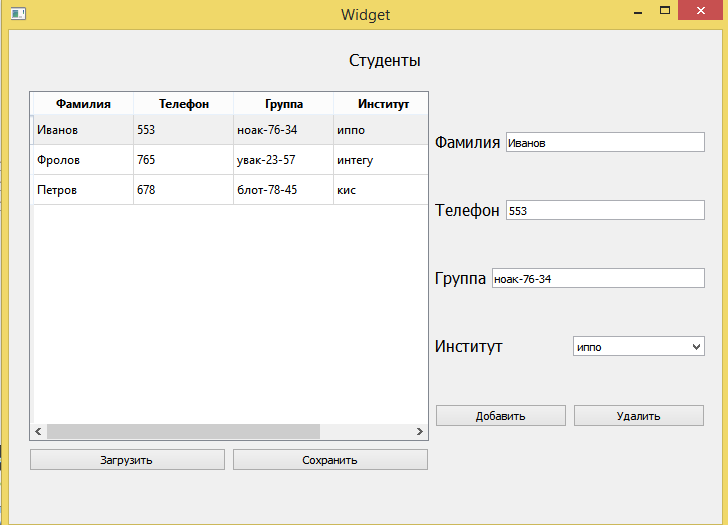


Рис. 1. Окно программы.

1. **Заключение**

В результате данной лабораторной работы были изучены основы построения интерфейса и модели данных. Была реализована собственная модель представления данных.

**Список используемой литературы**

1. Герберт Шилдт. Теория и практика С++ = Shildt's Expert C++. — СПб.: BHV — Санкт- Петербург, 1996.
2. Шлее М. Qt 5.3 Профессиональное программирование на C++. – СПб.:БХВ-Петербург, 2015. - 928 с.: ил. – (В подлиннике)
3. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. СПб.: Бином, 2006. – 560 с.
4. Страуструп Б. Язык программирования С++: Пер. с англ. – 3-е спец. изд. – М.: Бином, 2003. – 1104 с.