|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 10 |

**Название:**

Qt. Создание контейнеров

**Дисциплина:** Объектно-ориентированное программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22Б |  |  | Баканов Роман |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание:**

Моделировать очередь, в качестве элементов которой могут использоваться целые числа и слова. Операции: добавление элемента, удаление элемента, печать элементов. Создать класс-потомок, который содержит процедуру сортировки элементов (числа сортировать по возрастанию, слова по алфавиту). Тестировать полученную модель.

Разработать собственную иерархию классов, готовые контейнеры Qt не использовать. Пользовательский интерфейс для работы с моделью реализовать на Qt. В отчете представить диаграмму классов и обосновать выбранную структуру представления данных.

**Текст программы:**

(graph.h)

#ifndef GRAPH\_H

#define GRAPH\_H

#include <QDialog>

#include "ui\_graph.h"

class FormDialog :public QDialog, public Ui::GraphQueue

{

Q\_OBJECT

public:

FormDialog(QWidget\* parent = 0);

private slots:

void adder();

void remover();

void sorter();

void allremover();

};

#endif

(queue.h)

#include <iostream>

#include <QTextEdit>

#include <string>

using namespace std;

class Queue

{

protected:

string content[20];

int length;

bool existence;

public:

Queue()

{

length = 0;

existence = false;

}

void add(QString arg);

void rem();

void remall();

void print(QTextEdit &textEdit);

};

class AdvQueue :public Queue

{

public:

void sort();

};

(graph.cpp)

#include "graph.h"

#include "queue.h"

#include <QDialog>

using namespace std;

AdvQueue que;

FormDialog::FormDialog(QWidget\* parent):QDialog(parent)

{

setupUi(this);

connect(pushButton, SIGNAL(clicked()),this,SLOT(adder()));

connect(pushButton\_2, SIGNAL(clicked()),this,SLOT(remover()));

connect(pushButton\_3, SIGNAL(clicked()),this,SLOT(allremover()));

connect(pushButton\_4, SIGNAL(clicked()),this,SLOT(sorter()));

};

void FormDialog::adder()

{

if (lineEdit -> text() != "")

{

que.add(lineEdit->text());

lineEdit->clear();

que.print(\**textEdit*);

lineEdit->setFocus();

}

};

void FormDialog::remover()

{

que.rem();

que.print(*\*textEdit*);

lineEdit->setFocus();

};

void FormDialog::sorter()

{

que.sort();

que.print(*\*textEdit*);

lineEdit->setFocus();

};

void FormDialog::allremover()

{

que.remall();

textEdit->clear();

lineEdit->setFocus();

};

(queue.cpp)

#include <iostream>

#include "queue.h"

using namespace std;

void Queue::add(QString arg)

{

length++;

content[length-1] = arg.toStdString();

existence = true;

}

void Queue::rem()

{

if (existence)

{

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

content[i] = content[i + 1];

content[length - 1] = "";

length--;

if (length == 0)

existence = false;

}

}

void Queue::remall()

{

if (existence)

{

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

content[i] = "";

existence = false;

}

}

void Queue::print(QTextEdit &textEdit)

{

textEdit.clear();

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

textEdit.append(QString::fromStdString(content[i]));

}

void AdvQueue::sort()

{

if ((existence) && (length > 1))

{

int j = 0;

string buf;

bool stop = false;

while (!stop)

{

stop = true;

for (int i = 0; i < length - 1 - j; i++)

if (content[i] > content[i + 1])

{

buf = content[i];

content[i] = content[i + 1];

content[i + 1] = buf;

stop = false;

}

j++;

}

stop = false;

j = 0;

while ((!stop) && (j < length))

{

if ((int)content[j][0] > 96)

stop = true;

j++;

}

for (int k = 0; k < j - 2; k++)

for (int i = 0; i < j - 2 -k; i++)

if (stoi(content[i]) > stoi(content[i+1]))

{

buf = content[i];

content[i] = content[i + 1];

content[i + 1] = buf;

}

}

}

(start.cpp)

#include <QApplication>

#include "graph.h"

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

{

QApplication app(*argc*, argv);

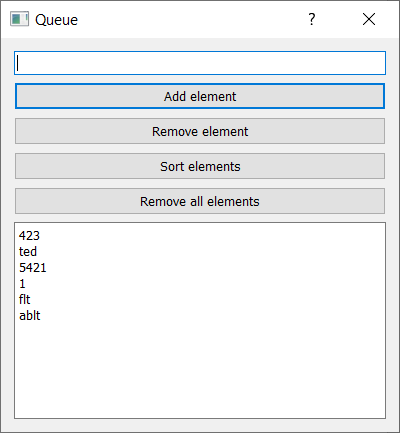
FormDialog\* dialog = new FormDialog();

dialog->show();

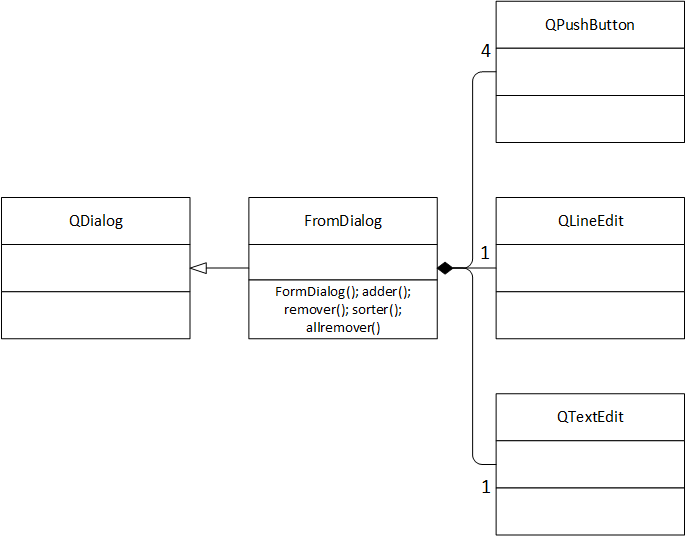
return app.exec();

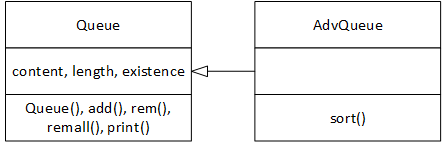
}

**Скриншоты:**

****

**Диаграмма классов:**

****

****

**Выводы:**

Получены навыки работы с контейнерами (в данном примере – с очередью).