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**

О Т Ч Е Т

по лабораторной работе № 1 0

Название: Qt. Создание контейнеров

Дисциплина: Объектно-ориентированное программирование

Студент	ИУ6-22Б		С.В. Астахов
	(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 символы. Операции: добавление элемента, удаление элемента, печать элементов очереди. Создать класс-потомок, который содержит процедуру сортировки элементов очереди (числа по возрастанию, символы 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.

Исходный код

(файл front.cpp)

```
#include <QApplication>
#include "back.h"
//#include "que.h"

int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   FormDialog *dialog = new FormDialog();
   dialog->show(); // отображаем окно
   return app.exec(); // запускаем цикл обработки сообщений
}
```

(файл back.h)

```
#ifndef BACK_H_
#define BACK_H_
#include <QDialog>
#include <QLineEdit>
#include <QSignalMapper>
#include <QTextEdit>
#include <QString>
//#include "que.h"
/// Класс, реализующий редактор
class FormDialog: public QDialog
{
   Q_OBJECT
public:
   FormDialog( QWidget * parent = 0);
   virtual ~FormDialog(){};
```

```
protected:
   QLineEdit *lineEdit1;
   QTextEdit *field1;
   bool lower, isOut;


private slots:
   void pusher();
   void poper();
   void sorter();
   void outer();
};
#endif
```

(файл back.cpp)

```
#include <QPushButton>
#include <QVBoxLayout>
#include <QTextEdit>
#include <QLineEdit>
#include <iostream>
#include <QString>
#include "back.h"
#include "que.h"
using namespace std;

//void FormDialog::newQs(bool& outId);
//void FormDialog::swapper(bool& caseId, bool& outId);
CSmartQ qobj;

FormDialog::FormDialog(QWidget * parent){
        QVBoxLayout *mainLayout = new QVBoxLayout();
        lineEdit1 = new QLineEdit();
        QPushButton *button1 = new QPushButton("Push");
        QPushButton *button2 = new QPushButton("Pop");
        QPushButton *button3 = new QPushButton("Sort");
        QPushButton *button4 = new QPushButton("Out");
        field1 = new QTextEdit();
        field1->setReadOnly(true);
        //QString str1;
        bool lower = true, isOut = false;
```

```
        connect(button1, SIGNAL(clicked()), this, SLOT(pusher()));
        connect(button2, SIGNAL(clicked()), this, SLOT(poper()));
        connect(button3, SIGNAL(clicked()), this, SLOT(sorter()));
        connect(button4, SIGNAL(clicked()), this, SLOT(outer()));
        //connect(lineEdit1, SIGNAL(textEdited(QString)), this, SLOT(newQs()));
        mainLayout->addWidget(lineEdit1);
        mainLayout->addWidget(button1);
        mainLayout->addWidget(button2);
        mainLayout->addWidget(button3);
        mainLayout->addWidget(button4);
        mainLayout->addWidget(field1);
        setLayout(mainLayout);
};

// void FormDialog::newQs(){
//      //isOut = false;
//      field1->setText("");
//      field1->append("input: " + lineEdit1->text());
// };

void FormDialog::pusher(){
        QString str = lineEdit1->text();
   QByteArray a= str.toUtf8(); // to....
   char* d=  a.data();
        qobj.add(*d);

        //QString str2(QChar(*d));
        field1->append("added " + QString::fromLocal8Bit(d,1));
        //field1->append(str2);
};

void FormDialog::poper(){
        if(qobj.getlng()>0){
                qobj.rm();
        field1->append("last elem removed");
        }
        else{
                field1->append("queue is empty");
        }
};

void FormDialog::sorter(){
        qobj.sort();
```

```
        field1->append("sorted");
};

void FormDialog::outer(){
        qobj.reset();
        field1->setText("");
        field1->append("Queue:");
        for(int i=0; i < qobj.getlng(); i++){
                char *ch;
                *ch = qobj.gett();
                field1->append(QString::fromLocal8Bit(ch,1));
                //field1->append("^");
        }
};
```

(файл que.h)

```
#include <iostream>
#include <vector>
using namespace std;

class CQueue {
protected:
        queue<char> queuef;
        int lng;
        int p;
public:
        void add(char arg);
        void rm();
        //void setlng();
        int getlng();
        void reset();
        char gett();
};

class CSmartQ : public CQueue {
public:
        void sort();
};
```

(файл que.cpp)

```
#include <iostream>
```

```
#include <queue>
#include "que.h"
using namespace std;



void CQueue::add(char arg) {
	queuef.push_back(arg);
}
void CQueue::rm() {
	queuef.pop_back();
}
// void CQueue::setlng() {
//	lng = 0;
//	for (vector<char>::iterator i = queuef.begin(); i != queuef.end(); ++i) {
//		lng++;
//	}
// }
int CQueue::getlng() {
	lng = 0;
	for (vector<char>::iterator i = queuef.begin(); i != queuef.end(); ++i) {
		lng++;
	}
	return lng;
}
void CQueue::reset() {
	p = -1;
}
char CQueue::gett() {
	p++;
	return queuef[p];
}



void CSmartQ::sort() {
	bool sorted = false;
	while (!(sorted)) {
		sorted = true;
		for (queue<char>::iterator i = queuef.begin(); i != (queuef.end()-1); ++i) {
			if ((*i) > * (i + 1)) {
				char buf;
				buf = *i;
				*i = *(i + 1);
```

```
                    *(i + 1) = buf;
                    sorted = false;
                }
            }
        }
    }
```

Скриншоты

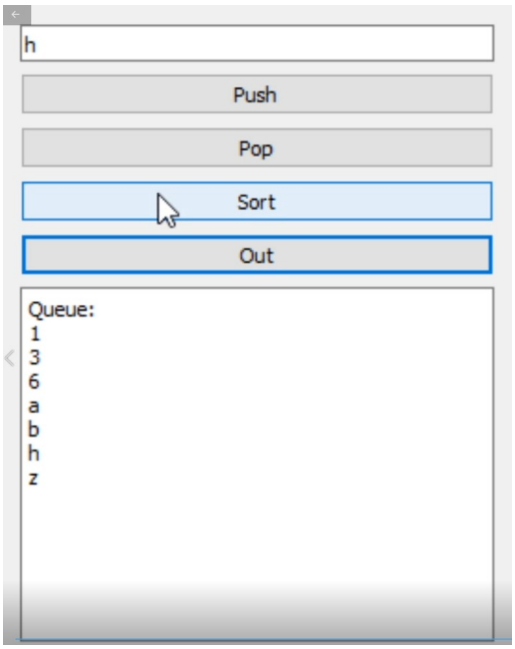
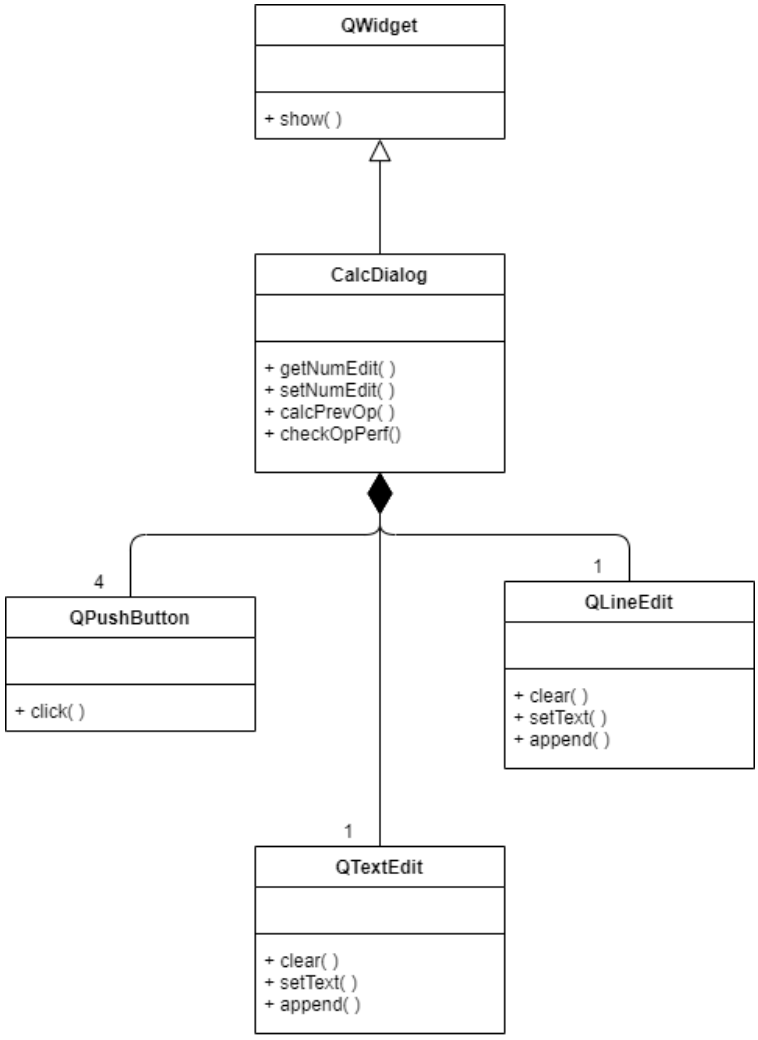
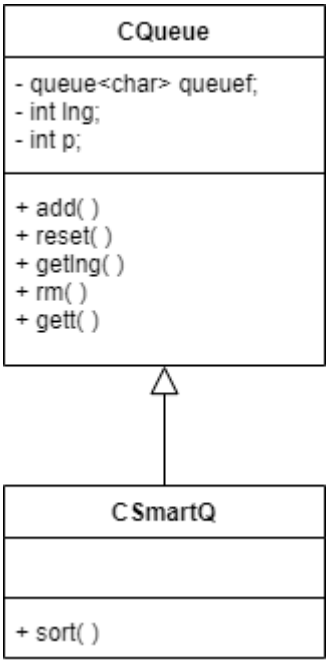


Диаграмма классов

Визуальные классы



Невизуальные классы



Вывод

- С помощью Qt можно осуществлять работу с различными контейнерами (например, очередью), независимо от того созданы они с помощью структуры C++, готового контейнера C++ или готового контейнера Qt.
- Был выбран контейнер queue, так именно он и является стандартным контейнером очереди, где метод push( ) добавляет элемент в конец, а метод pop( ) удаляет элемент из начала.