БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Международный институт дистанционного образования

Кафедра «Информационные системы и технологии»

**КУРСОВОЙ ПРОЕКТ**

по учебной дисциплине

«СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ»

Исполнитель: студент 3 курса, группы 41703120 Реут В.Л.

Руководитель: Бумай А. Ю.

Минск 2023

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](file:///D:\BNTU\BNTU\OOP\Course_Project_OOP.docx#_Toc125733098)

ЗАДАНИЕ 4

**Описание разработанных классов 5**

Диаграмма классов 5

**Описание программы 6**

Испытание программы 12

[Выводы](file:///D:\BNTU\BNTU\OOP\Course_Project_OOP.docx#_Toc125733112) 14

ЛИТЕРАТУРА 15

**Листинг программы 16**

# **ВВЕДЕНИЕ**

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++

С++ является компилируемым языком, а это значит, что компилятор транслирует исходный код на С++ в исполняемый файл, который содержит набор машинных инструкций. Но разные платформы имеют свои особенности, поэтому скомпилированные программы нельзя просто перенести с одной платформы на другую и там уже запустить. Однако на уровне исходного кода программы на С++ по большей степени обладают переносимостью, если не используются какие-то специфичные для текущей ос функции. А наличие компиляторов, библиотек и инструментов разработки почти под все распространенные платформы позволяет компилировать один и тот же исходный код на С++ в приложения под эти платформы.

В 1979-80 годах Бьерн Страуструп разработал расширение к языку Си - "Си с классами". В 1983 язык был переименован в С++.

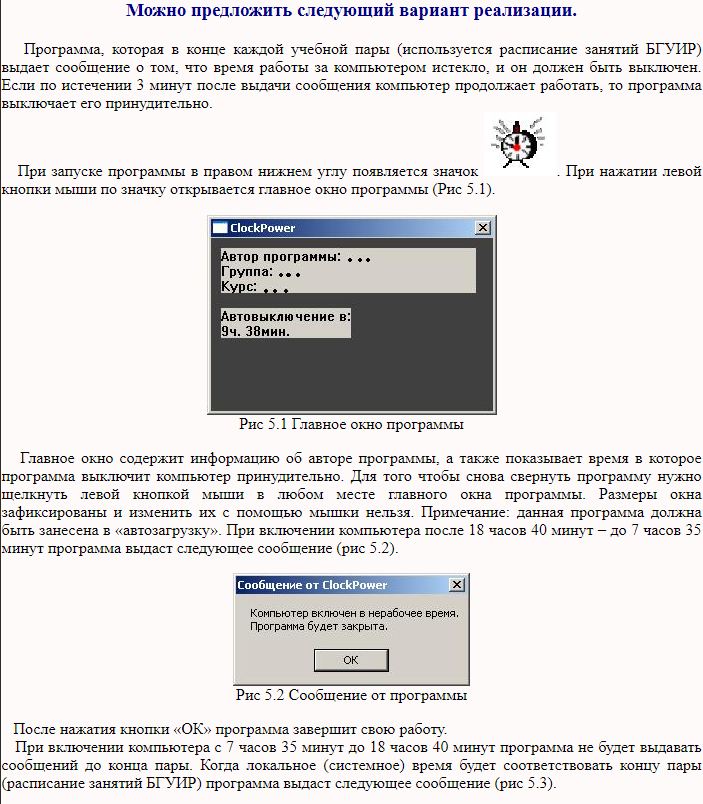
В 1985 году была выпущена первая коммерческая версия языка С++, а также первое издание книги "Языка программирования C++", которая представляла первое описание этого языка при отсутствии официального стандарта.

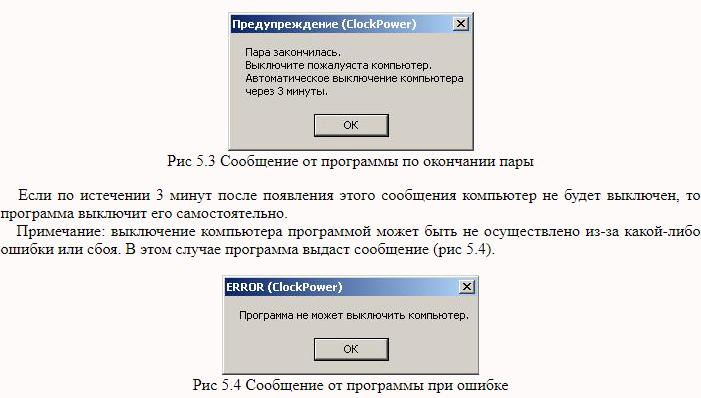
В 1989 была выпущена новая версия языка C++ 2.0, которая включала ряд новых возможностей. После этого язык развивался относительно медленно вплоть до 2011 года. Но при этом в 1998 году была предпринята первая попытка по стандартизации языка организацией ISO (International Organiztion for Standartization). Первый стандарт получил название ISO/IEC 14882:1998 или сокращенно С++98. В дальнейшем в 2003 была издана новая версия стандарта C++03.

**Задание.**

**5.Процессы и потоки (Processes and Threads).**

Разработать программу, демонстрирующую механизмы подготовки, создания и управления процессами. Программа-оболочка должна уметь создавать процесс, ожидать завершения процесса в отдельном потоке, а также завершать запущенные дочерние процессы.

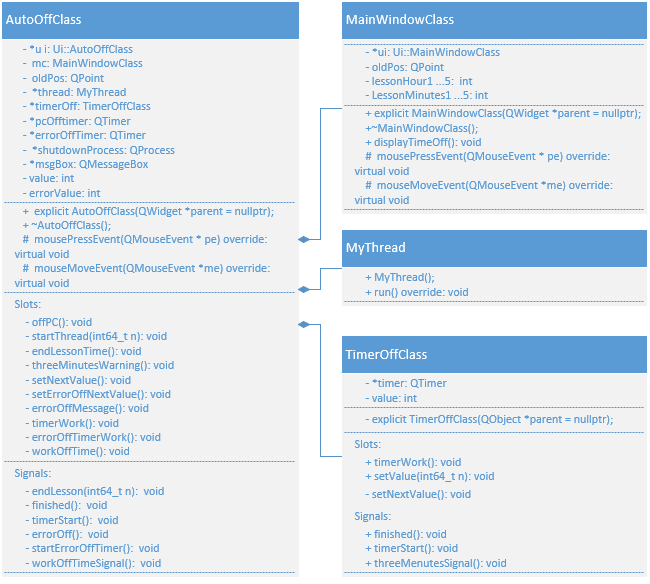




**Описание разработанных классов.**

Программа была создана с помощью фреймворка Qt в QtCreator. Всего реализовано 4 класса. Главный класс-агрегатор **AutoOffClass** (наследован от **QWidget**, в нем содержаться объекты остальных классов) и 3 вспомогательных класса **MainWindowClass** (также наследован от **QWidget**, класс основного окна программы), **MyThread** (наследован от **QThread**, отвечает за многопоточность), **TimerOffClass** (наследован от **QObject**, класс таймер работающий в параллельном потоке).

**Диаграмма классов.**

****

**Описание программы**

**Конструктор класса AutoOffClass**

Основная логическая нагрузка приходится на конструктор класса **AutoOffClass** (Рис.2) . Посредством конструктора задается стартовая заставка приложения (Рис.1):



*Рис.1 С стартовая заставка приложения*

Запускается вторичный поток, а так же прописаны функции connect для соединения сигналов со слотами:

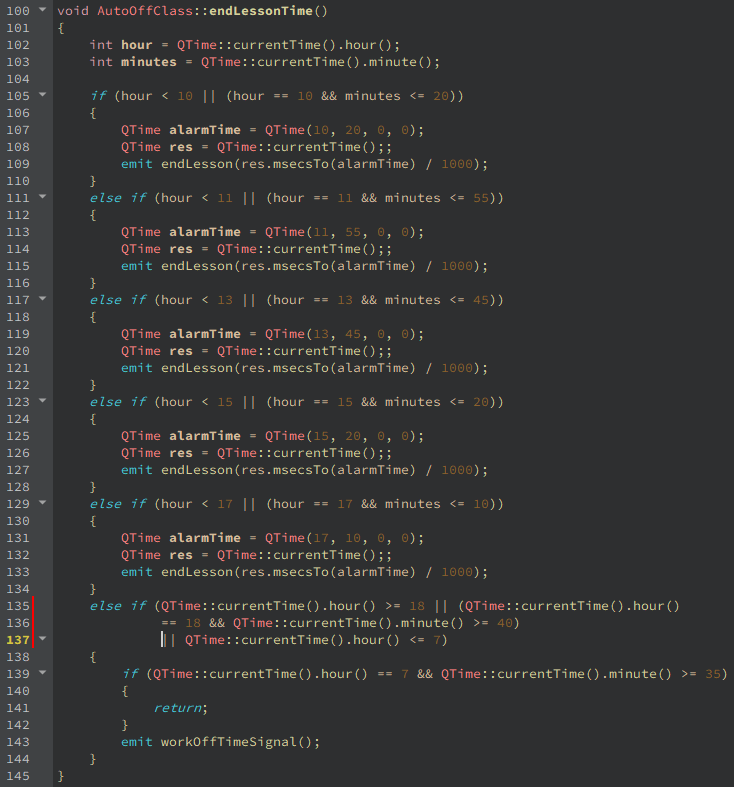


*Рис.2 Конструктор класса* ***AutoOffClass****.*

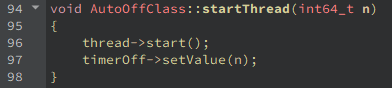
Функция **setAttribute** (строки 9,10) задают прозрачность фона для основного виджета и элемента **QLabel**, в строке 11 по средством функции **setPixmap** мы задаем стартовое изображение которое находится в ресурсах программы. Строки с 13 по 19 это создание **layout** и кастомной кнопки завершения программы, сигнал которой связан функцией **connect** со слотом закрытия программы (строка 15).

В строке 21 инициализируется переменная типа **QMessageBox** для выведения на экран информации об ошибках и предупреждения. Далее инициализируется переменная типа **QProcess** (строка 22), которая отвечает за запуск процесса выключения компьютера в заданное время. Следом 2 переменных 2 таймера. В строках 27 и 28 инициализируются объекты наших классов thread (**MyThread**) и **timerOff** (**TimerOffClass**) и в строке 29 переменная timerOff передается в поток.

На строке 42 у нас находится функция **endLesson** (Рис.3). Она фиксирует текущее время при запуске компьютера ( при условии что компьютер запущен в рабочее время, иначе программа выводит предупреждение и закрывается) и посылает сигнал **endLesson**, по средством библиотечной функции connect (строка 32), слоту **startThread** (Рис.4).



*Рис.3 Функция* ***endLesson****.*



*Рис.4 Слот* ***startThread****.*

В данном случае логика следующая: из функции **endLesson** передается одноименный сигнал который передает слоту **startThread** количество секунд для ближайшей перемены (как получает эти данный сигнал можно увидеть например в строках 107-109 функции **endLesson**). Слот в свою очередь запускает параллельный поток с обратным таймером (класс **TimerOffClass**, файл cpp, Рис.5) который берет за время отсчета то самое количество секунд (передается через **setValue** строка 97, Рис.4).

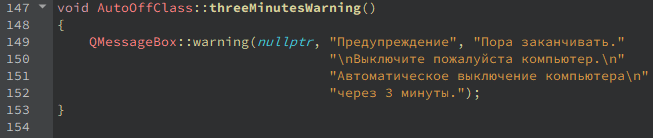


*Рис.5 Файл cpp класса* ***TimerOffClass****.*

Далее работа программы продолжается в **TimerOffClass**. Функция **setValue** по средствам библиотечной функции connect (строка 8, Рис.5) посылает сигнал **timerStart** слоту **timerWork**, тем самым запускает таймер (строка 14, Рис.5). Таймер каждую секунду посылает сигнал **timeout** который через connect (строка 9, Рис.5) передается слоту **setNextValue**.

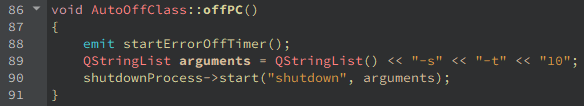
Если заглянуть в функцию **setNextValue** (строки 25-35, Рис.5) то мы увидим что переданное из класса **AutoOffClass** значение **value** каждую секунду уменьшается на единицу. И когда значение достигает нуля, это говорит нам что наступила перемена. Далее мы останавливаем таймер и посылаем 2 сигнала в класс **AutoOffClass** (строки 31,32, Рис.5).

Возвращаемся в конструктор класса **AutoOffClass**. Сигнал **finished** сообщает классу **MyThred**, по средством функции **connect**, что необходимо закрыть поток, так как он нам больше не нужен (строка 31, Рис.2). А сигнал **threeMinutesSignal** класса **TimerOffClass** через connect запускает слоты **timerWork** (строка 33, Рис.2) и **threeMinutesWarning** (строка 35, Рис.2) класса **AutoOffClass**. **threeMinutesWarning** показывает сообщение о том что через три минуты компьютер будет выключен (Рис.6).



*Рис.6 Слот* ***threeMinutesWarning****.*

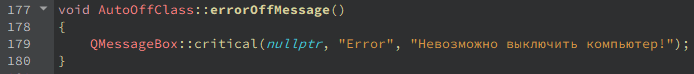
А слот **timerWork** запускает абсолютно аналогичный по логике реализации трехминутный таймер класса **AutoOffClass** который мы наблюдали в классе TimerOffClass. Сигнал таймаут каждую секунду вызывает слот **setNextValue** класса **AutoOffClass** (строка 34, Рис.2), в слоте значение **value** уменьшается на единицу и при значении ноль посылает сигнал **finished** который в свою очередь запускает (строка 36, Рис.2) слот **offPC** (Рис.7).



*Рис.7 Слот* ***offPC****.*

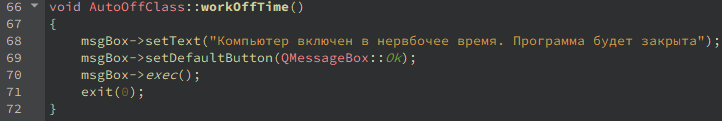
Слот **offPC** используя список аргументов (строка 89, Рис.7) запускает процесс выключения компьютера (строка 90, Рис.7).

Далее следует сказать об обработке ошибки выключения компьютера. Из слота **offPC** посылается сигнал **startErrorOffTimer** который запускает 20-секундный таймер (строка 38, Рис.2) с аналогичной логикой работы выше описанных таймеров, сигнал таймаут которого каждую секунду запускает слот **setErrorOffNextValue** (строка 39, Рис.2). При достижении нуля слот **setErrorOffNextValue** отправляет сигнал **errorOff** который запускает (строка 40, Рис.2) слот **errorOffMessage** (Рис.8). Этот слот сообщает о том что компьютер выключить не удалось.



*Рис.8 Слот* ***errorOffMessage****.*

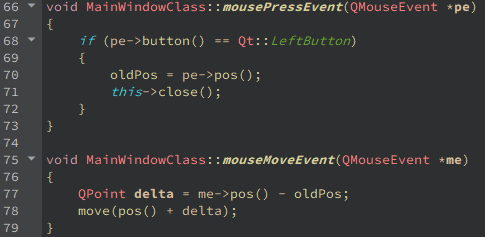
Обработка события включения компьютера в нерабочее время реализована в функции **endLesson** (строки 135-144). Отправляется сигнал **workOffTimeSignal** который запускает (строка 37, Рис.2) слот **workOffTime** (Рис.9)



*Рис.9 Слот* ***workOffTime****.*

Слот выкидывает соответствующее сообщение и завершает программу.

Класс **MainWindowClass** отвечает лишь за выведение лишь актуальной информации когда выключится компьютер. За это отвечает функция **displayTimeOff** (Рис.11). Она предельно проста, поэтому в описании не нуждается. Так же здесь содержаться вспомогательные библиотечные функции **mousePressEvent** и **mouseMoveEvent** (Рис.10). Они отвечают за перемещение виджетов программы на рабочем столе. Абсолютно аналогичные функции содержаться и в классе **AutoOffClass**.



*Рис.10 Функции* ***mousePressEvent*** *и* ***mouseMoveEvent****.*



*Рис.11 Функция* ***displayTimeOff****.*

**Испытание программы**

При включении компьютера автозапуск запускает стартовое окно программы (Рис.1). При нажатии в любой точке стартового окна открывается основное окно программы (Рис.12).

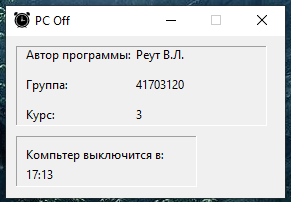


Рис.12 Основное окно программы.

Размер основного окна фиксирован согласно требованиям, а так же закрывается при щелчке левой кнопкой мышки в любом месте.

Согласно расписанию пар, поступает сообщении от том что компьютер будет выключен через 3 минуты (Рис.13).

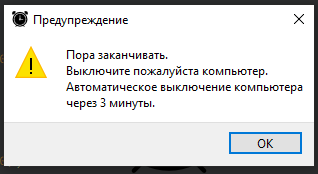


Рис.13 Предупреждение о выключении.

По истечении 3-х минут приходит системное предупреждение о выключении компьютера (Рис.14).

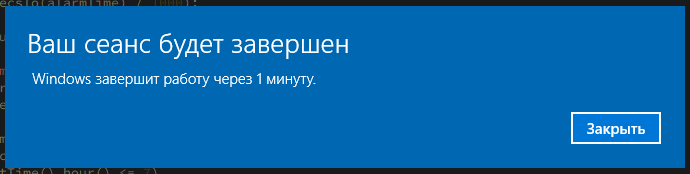


Рис.14 Системное предупреждение о выключении компьютера.

Если что-то мешает завершению работы то мы получаем сообщение о том что не удалось выключить компьютер (Рис.15)

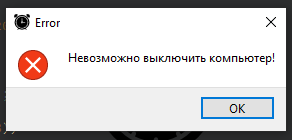


Рис.15 Сообщение о том что не удалось выключить компьютер.

**Выводы**

В процессе создания приложения была разобрана работа потоков и процессов, а так же их отличия. Так же было оценено удобство работы с данными технологиями при использовании современных фреймворков на примере фреймворка Qt.

**ЛИТЕРАТУРА**

1. Qt 5.10 Профессиональное программирование на С++. Макс Шлее.
2. Документация Qt.

**Листинг программы**

**autooffclass.h:**

#ifndef AUTOOFFCLASS\_H

#define **AUTOOFFCLASS\_H**

#include <QWidget>

#include <QMainWindow>

#include "mainwindowclass.h"

#include "mythread.h"

#include "timeroffclass.h"

#include <QProcess>

#include <QPainter>

#include <QTime>

#include <QTimer>

#include <QMouseEvent>

#include <QMessageBox>

#include <QPixmap>

#include <QPushButton>

#include <QBoxLayout>

*namespace* **Ui** {

*class* **AutoOffClass**;

}

*class* **AutoOffClass** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* **AutoOffClass**(QWidget \***parent** = *nullptr*);

~**AutoOffClass**();

*protected*:

*virtual* void ***mousePressEvent***(QMouseEvent \* **pe**) *override*;

*virtual* void ***mouseMoveEvent***(QMouseEvent \***me**) *override*;

signals:

void **endLesson**(int64\_t **n**);

void **finished**();

void **timerStart**();

void **errorOff**();

void **startErrorOffTimer**();

void **workOffTimeSignal**();

*private* slots:

void **offPC**();

void **startThread**(int64\_t **n**);

void **endLessonTime**();

void **threeMinutesWarning**();

void **setNextValue**();

void **setErrorOffNextValue**();

void **errorOffMessage**();

void **timerWork**();

void **errorOffTimerWork**();

void **workOffTime**();

*private*:

Ui::AutoOffClass \***ui**;

MainWindowClass **mc**;

QPoint **oldPos**;

MyThread \***thread**;

TimerOffClass \***timerOff**;

QTimer \***pcOfftimer**;

QTimer \***errorOffTimer**;

QProcess \***shutdownProcess**;

QMessageBox \***msgBox**;

int **value**;

int **errorValue**;

};

#endif *//* *AUTOOFFCLASS\_H*

**autooffclass.cpp:**

#include "autooffclass.h"

#include "ui\_autooffclass.h"

AutoOffClass::**AutoOffClass**(QWidget \***parent**) :

QWidget(*parent*, Qt::*FramelessWindowHint* | Qt::*Window*),

ui(*new* Ui::AutoOffClass)

{

ui->setupUi(*this*);

setAttribute(Qt::*WA\_TranslucentBackground*);

ui->label->setAttribute(Qt::*WA\_TranslucentBackground*);

ui->label->setPixmap(QPixmap(":/pic/alarm-clock-128.png"));

QPushButton\* **pcmdQuit** = *new* QPushButton("X");

pcmdQuit->setFixedSize(16, 16);

connect(pcmdQuit, SIGNAL(clicked()), qApp, SLOT(quit()));

QVBoxLayout\* **pvbx** = *new* QVBoxLayout;

pvbx->addWidget(*pcmdQuit*);

pvbx->addStretch(1);

ui->label->setLayout(*pvbx*);

msgBox = *new* QMessageBox(*this*);

shutdownProcess = *new* QProcess(*this*);

pcOfftimer = *new* QTimer(*this*);

errorOffTimer = *new* QTimer(*this*);

value = 170;

errorValue = 20;

thread = *new* MyThread();

timerOff = *new* TimerOffClass(*this*);

timerOff->moveToThread(*thread*);

connect(timerOff, SIGNAL(finished()), thread, SLOT(quit()));

connect(*this*, SIGNAL(endLesson(int64\_t)), *this*, SLOT(startThread(int64\_t)));

connect(timerOff, SIGNAL(threeMenutesSignal()), *this*, SLOT(timerWork()));

connect(pcOfftimer, SIGNAL(timeout()), *this*, SLOT(setNextValue()));

connect(timerOff, SIGNAL(threeMenutesSignal()), *this*, SLOT(threeMinutesWarning()));

connect(*this*, SIGNAL(finished()), *this*, SLOT(offPC()));

connect(*this*, SIGNAL(workOffTimeSignal()), *this*, SLOT(workOffTime()));

connect(*this*, SIGNAL(startErrorOffTimer()), *this*, SLOT(errorOffTimerWork()));

connect(errorOffTimer, SIGNAL(timeout()), *this*, SLOT(setErrorOffNextValue()));

connect(*this*, SIGNAL(errorOff()), *this*, SLOT(errorOffMessage()));

endLessonTime();

}

AutoOffClass::~**AutoOffClass**()

{

*delete* ui;

}

void AutoOffClass::***mousePressEvent***(QMouseEvent \***pe**)

{

*if* (pe->button() == Qt::*LeftButton*)

{

oldPos = pe->pos();

mc.show();

}

}

void AutoOffClass::***mouseMoveEvent***(QMouseEvent \***me**)

{

QPoint **delta** = me->pos() - oldPos;

move(pos() + delta);

}

void AutoOffClass::**workOffTime**()

{

msgBox->setText("Компьютер включен в нервбочее время. Программа будет закрыта");

msgBox->setDefaultButton(QMessageBox::*Ok*);

msgBox->*exec*();

exit(0);

}

void AutoOffClass::**timerWork**()

{

pcOfftimer->start(1000);

qDebug() << "pcOfftimer";

}

void AutoOffClass::**errorOffTimerWork**()

{

errorOffTimer->start(1000);

qDebug() << "errorOfftimer";

}

void AutoOffClass::**offPC**()

{

emit startErrorOffTimer();

QStringList **arguments** = QStringList() << "-s" << "-t" << "10";

shutdownProcess->start("shutdown", arguments);

}

void AutoOffClass::**startThread**(int64\_t **n**)

{

thread->start();

timerOff->setValue(n);

}

void AutoOffClass::**endLessonTime**()

{

int **hour** = QTime::currentTime().hour();

int **minutes** = QTime::currentTime().minute();

*if* (hour < 10 || (hour == 10 && minutes <= 20))

{

QTime **alarmTime** = QTime(10, 20, 0, 0);

QTime **res** = QTime::currentTime();;

emit endLesson(res.msecsTo(alarmTime) / 1000);

}

*else* *if* (hour < 11 || (hour == 11 && minutes <= 55))

{

QTime **alarmTime** = QTime(11, 55, 0, 0);

QTime **res** = QTime::currentTime();;

emit endLesson(res.msecsTo(alarmTime) / 1000);

}

*else* *if* (hour < 13 || (hour == 13 && minutes <= 45))

{

QTime **alarmTime** = QTime(13, 45, 0, 0);

QTime **res** = QTime::currentTime();;

emit endLesson(res.msecsTo(alarmTime) / 1000);

}

*else* *if* (hour < 15 || (hour == 15 && minutes <= 20))

{

QTime **alarmTime** = QTime(15, 20, 0, 0);

QTime **res** = QTime::currentTime();;

emit endLesson(res.msecsTo(alarmTime) / 1000);

}

*else* *if* (hour < 17 || (hour == 17 && minutes <= 10))

{

QTime **alarmTime** = QTime(17, 10, 0, 0);

QTime **res** = QTime::currentTime();;

emit endLesson(res.msecsTo(alarmTime) / 1000);

}

*else* *if* (QTime::currentTime().hour() >= 18 || (QTime::currentTime().hour()

== 18 && QTime::currentTime().minute() >= 40)

|| QTime::currentTime().hour() <= 7)

{

*if* (QTime::currentTime().hour() == 7 && QTime::currentTime().minute() >= 35)

{

*return*;

}

emit workOffTimeSignal();

}

}

void AutoOffClass::**threeMinutesWarning**()

{

QMessageBox::warning(*nullptr*, "Предупреждение", "Пора заканчивать."

"\nВыключите пожалуйста компьютер.\n"

"Автоматическое выключение компьютера\n"

"через 3 минуты.");

}

void AutoOffClass::**setNextValue**()

{

--value;

*if* (value == 0)

{

pcOfftimer->stop();

emit finished();

}

qDebug() << value;

}

void AutoOffClass::**setErrorOffNextValue**()

{

--errorValue;

*if* (errorValue == 0)

{

errorOffTimer->stop();

emit errorOff();

}

}

void AutoOffClass::**errorOffMessage**()

{

QMessageBox::critical(*nullptr*, "Error", "Невозможно выключить компьютер!");

}

**mainwindowclass.h:**

#ifndef MAINWINDOWCLASS\_H

#define **MAINWINDOWCLASS\_H**

#include <QWidget>

#include <QString>

#include <QMouseEvent>

*namespace* **Ui** {

*class* **MainWindowClass**;

}

*class* **MainWindowClass** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* **MainWindowClass**(QWidget \***parent** = *nullptr*);

~**MainWindowClass**();

void **displayTimeOff**();

*protected*:

*virtual* void ***mousePressEvent***(QMouseEvent \* **pe**) *override*;

*virtual* void ***mouseMoveEvent***(QMouseEvent \***me**) *override*;

*private*:

Ui::MainWindowClass \***ui**;

QPoint **oldPos**;

int **lessonHour1**, **LessonMinutes1**,

**lessonHour2**, **LessonMinutes2**,

**lessonHour3**, **LessonMinutes3**,

**lessonHour4**, **LessonMinutes4**,

**lessonHour5**, **LessonMinutes5**;

};

#endif *//* *MAINWINDOWCLASS\_H*

**mainwindowclass.cpp:**

#include "mainwindowclass.h"

#include "ui\_mainwindowclass.h"

#include <QTime>

#include <QTimer>

MainWindowClass::**MainWindowClass**(QWidget \***parent**) :

QWidget(*parent*),

ui(*new* Ui::MainWindowClass)

{

ui->setupUi(*this*);

setWindowTitle(tr("PC Off"));

setFixedHeight(162);

setFixedWidth(279);

displayTimeOff();

}

MainWindowClass::~**MainWindowClass**()

{

*delete* ui;

}

void MainWindowClass::**displayTimeOff**()

{

lessonHour1 = 10, LessonMinutes1 = 20;

lessonHour2 =11, LessonMinutes2 = 55,

lessonHour3 = 13, LessonMinutes3 = 45,

lessonHour4 = 15, LessonMinutes4 = 20,

lessonHour5 = 17, LessonMinutes5 = 10;

int **hour** = QTime::currentTime().hour();

int **minutes** = QTime::currentTime().minute();

QString **lesson**;

*if* (hour < 10 || (hour == 10 && minutes <= 20))

{

lesson = QString::number(lessonHour1) + ":" + (QString::number(LessonMinutes1 + 3));

ui->l\_time->setText(lesson);

}

*else* *if* (hour < 11 || (hour == 11 && minutes <= 55))

{

lesson = QString::number(lessonHour2) + ":" + (QString::number(LessonMinutes2 + 3));

ui->l\_time->setText(lesson);

}

*else* *if* (hour < 13 || (hour == 13 && minutes <= 45))

{

lesson = QString::number(lessonHour3) + ":" + (QString::number(LessonMinutes3 + 3));

ui->l\_time->setText(lesson);

}

*else* *if* (hour < 15 || (hour == 15 && minutes <= 20))

{

lesson = QString::number(lessonHour4) + ":" + (QString::number(LessonMinutes4 + 3));

ui->l\_time->setText(lesson);

}

*else* *if* (hour < 17 || (hour == 17 && minutes <= 10))

{

lesson = QString::number(lessonHour5) + ":" + (QString::number(LessonMinutes5 + 3));

ui->l\_time->setText(lesson);

}

*else*

{

ui->l\_time->setText("До завтра!");

}

}

void MainWindowClass::***mousePressEvent***(QMouseEvent \***pe**)

{

*if* (pe->button() == Qt::*LeftButton*)

{

oldPos = pe->pos();

*this*->close();

}

}

void MainWindowClass::***mouseMoveEvent***(QMouseEvent \***me**)

{

QPoint **delta** = me->pos() - oldPos;

move(pos() + delta);

}

**mythread.h:**

#ifndef MYTHREAD\_H

#define **MYTHREAD\_H**

#include <QThread>

#include <QObject>

*class* **MyThread** : *public* QThread

{

Q\_OBJECT

*public*:

**MyThread**();

void ***run***() *override*;

};

#endif *//* *MYTHREAD\_H*

**mythread.cpp:**

#include "mythread.h"

#include "qdebug.h"

MyThread::**MyThread**()

{

qDebug() << "MyThread";

}

void MyThread::***run***()

{

exec();

}

**timeroffclass.h:**

#ifndef TIMEROFFCLASS\_H

#define **TIMEROFFCLASS\_H**

#include <QObject>

#include <QTimer>

#include <QTime>

*class* **TimerOffClass** : *public* QObject

{

Q\_OBJECT

*public*:

*explicit* **TimerOffClass**(QObject \***parent** = *nullptr*);

signals:

void **finished**();

void **timerStart**();

void **threeMenutesSignal**();

*public* slots:

void **timerWork**();

void **setValue**(int64\_t **n**);

*private* slots:

void **setNextValue**();

*private*:

QTimer \***timer**;

int **value**;

};

#endif *//* *TIMEROFFCLASS\_H*

**timeroffclass.cpp:**

#include "timeroffclass.h"

#include <QDebug>

TimerOffClass::**TimerOffClass**(QObject \***parent**)

: QObject{*parent*}

{

timer = *new* QTimer(*this*);

connect(*this*, SIGNAL(timerStart()), *this*, SLOT(timerWork()));

connect(timer, SIGNAL(timeout()), *this*, SLOT(setNextValue()));

}

void TimerOffClass::**timerWork**()

{

timer->start(1000);

qDebug() << "work";

}

void TimerOffClass::**setValue**(int64\_t **n**)

{

value = n;

qDebug() << n;

emit timerStart();

}

void TimerOffClass::**setNextValue**()

{

--value;

*if* (value == 0)

{

timer->stop();

emit threeMenutesSignal();

emit finished();

}

qDebug() << value;

}

**main.cpp:**

#include "autooffclass.h"

#include <QApplication>

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

{

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

AutoOffClass **w**;

w.show();

*return* a.exec();

}