Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ "ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра «Компьютерная безопасность»

ОТЧЕТ

К ЛАБОРАТОРНОЙ РАБОТЕ №10

По дисциплине

«Языки программирования»

Работу выполнил

Студент группы СКБ 193 _____ **С.О.Ташкинов**

подпись, дата

Работу проверил _____ **С.А. Булгаков**

Москва 2020

Постановка задачи

Разработать консольное приложение с использованием библиотеки Qt. При запуске формируется вектор, содержащий имена файлов текущего каталога и их размеры. Из этого вектора формируется множество имен файлов таким образом, чтобы их размеры составляли неубывающую последовательность. Данное множество выводится на консоль по следующим правилам: ограничителями являются фигурные скобки; элементы разделяются запятой.

При каждом событии таймера из множества удаляется элемент, среднеквадратическое отклонение размера которого от 3/4 среднего размера файлов, оставшихся в множестве, минимально, после чего обновленное множество выводится на консоль.

Всякий раз после вывода множества на новой строке печатается приглашение ко вводу команды (символ '>').

Организовать работу приложения таким образом, чтобы по таймеру выводилась информация о текущем состоянии множества, а при поступлении данных на стандартный ввод обрабатывалась команда пользователя. Предусмотреть следующие команды:

1) stop – остановка таймера;
2) start – запуск таймера;
3) restart – перезапуск приложения;
4) timeout <целое> – задает новый интервал для таймера;
5) exit – завершение приложения.

1 Общий алгоритм решения поставленной задачи

1.1 Классы

Для решения поставленной задачи был разработан пользовательский класс Klass, являющийся наследником класса QObject, в методах которого производятся необходимые расчеты, а также производится обработка пользовательских команд, поступающих на стандартный ввод.

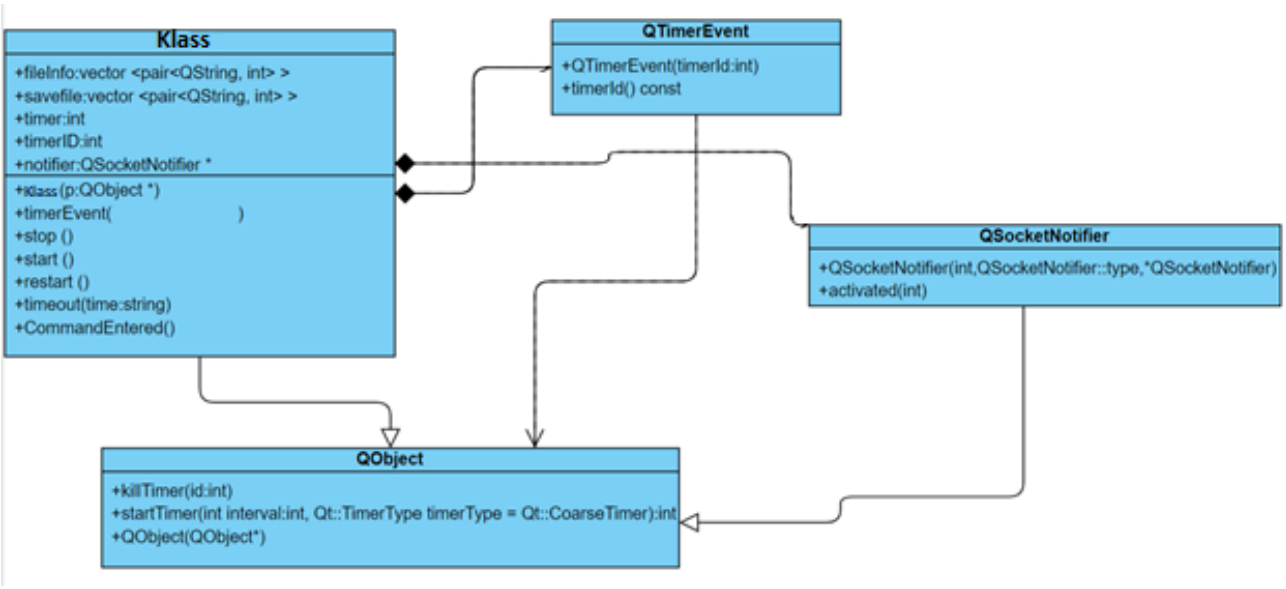

Рисунок 1 – UML 2.0 диаграмма классов

1.2 Поля классов

1.2.1 int Klass:: timer

Интервал тиков таймера.

1.2.2 int Klass:: timerID

Id таймера. Необходим для того, чтобы была возможность остановить таймер.

*1.2.3 QSocketNotifier*Klass:: notifier*

Необходим для считывания команд из стандартного ввода.

1.2.4 std::vector<std::pair<std::string, int>> Klass:: fileInfo

Вектор имен и размеров файлов.

1.2.5 std::vector<std::pair<std::string, int>> Klass:: **savefile**

Запасной вектор имен и файлов

1.3 Методы классов

1.3.1 Метод Klass::Klass()

Является конструктором класса Klass, в котором происходит запуск таймера и формирование вектора названий файлов, находящихся в каталоге, из которого происходит запуск программы**.**

1.3.2 Метод Klass::stop()

Останавливает работу таймера.

1.3.3 Метод Klass::timeout(int t)
Устанавливает интервал тиков таймера.

1.3.4 Метод Klass::start()
Запускает таймер

1.3.5 Метод Klass::restart()
Перезапускает таймер с обновленным вектором имен и размеров файлов

1.3.6 Метод Klass:: timerEvent(QTimerEvent e)*
Является переопределенным методом класса-родителя, с помощью которого осуществляется механизм работы таймера. Этот метод производит создание множества имен файлов из вектора имен и размеров, а также производит вычисление элемента, который должен быть удален с каждым тиком таймера, и удаляет его, после чего выводит оставшиеся элементы.

1.3.7 Метод Klass:: CommandEntered()
Является слотом класса, связанным с сигналом поля класса QSocketNotifier* notifier. С помощью данного метода происходит считывание команд пользователя, переданных на стандартный ввод. В зависимости от команды пользователя вызывается соответствующий метод.

1.4 Расположение функций, классов и их методов по файлам
1.4.1 main.cpp
Основной цикл выполнения программы

1.4.2 Kk.h
Определение класса Klass

1.4.3 rel.cpp
Определение методов класса Klass

1.5 Компиляция программы
Программа была скомпилирована с помощью make-файла, сгенерированного с помощью qmake.

2 Тестирование

2.1 Результат работы программы

{Main.cpp, prov.pro moc_Kk.cpp, Kk.h, Kh.h~, Makefile, mian.o, moc_Kk.o, prov}
>
{Main.cpp, prov.pro moc_Kk.cpp, Kk.h, Kh.h~, Makefile, mian.o, moc_Kk.o}
>
{Main.cpp, prov.pro moc_Kk.cpp, Kk.h, Kh.h~, Makefile, mian.o}
>
{Main.cpp, prov.pro moc_Kk.cpp, Kk.h, Kh.h~, Makefile}
>
{Main.cpp, prov.pro moc_Kk.cpp, Kk.h, Kh.h~}
>
{Main.cpp, prov.pro moc_Kk.cpp, Kk.h}
>
{Main.cpp, prov.pro moc_Kk.cpp}
>
{Main.cpp, prov.pro}
>
{Main.cpp}
>
{}
>

Приложение А

main.cpp

```
#include <QCoreApplication>
#include "Kk.h"

int main(int argc, char *argv[])
{
 QCoreApplication a(argc, argv);
 Klass t;
 return a.exec();
}
```

Kk.h

```
#ifndef KK_H
#define KK_H
#include <QObject>
#include <QTimerEvent>
#include <QFile>

#include <QFileInfo>
#include <QDir>
#include <QDebug>
#include <QString>
#include <QSocketNotifier>

#include <vector>
#include <iostream>
#include <string>
#include <set>
#include <cmath>
#include <cstdlib>
#include <iterator>
using namespace std;


 class Klass :public QObject
 {
 Q_OBJECT

 public:
 vector <pair<QString, int> > fileInfo;
 vector <pair<QString, int> > savefile;
 int timer;
 int timerID;
 QSocketNotifier *notifier;

 void timerEvent(QTimerEvent* e);
```

```
 Klass();
 void stop ();
 void start ();
 void restart ();
 void timeout(int t);
 public slots:

 void CommandEntered();
 };

#endif // KK_H
```

rel.cpp

```
#include "Kk.h"
vector <pair<QString, int> > gfile;

struct cmp{
bool operator()(const QString a, const QString b)
{
int size1=0;
int size2=0;
for (size_t i = 0; i <gfile.size(); ++i)
{
    if(a==gfile[i].first){size1=gfile[i].second;}
    if(b==gfile[i].first){size2=gfile[i].second;}
}
if(size1==0||size2==0){exit(1);}
return size1<size2;
}
};


Klass:: Klass() :timer(5000)
{
    QString directory = QDir::currentPath();
    QDir dir(directory);

    if(!dir.exists())
    {
    qWarning("The directory does not exist");
    }

    dir.setFilter(QDir::Files );


    dir.setSorting(QDir::Size | QDir::Reversed);


    QFileInfoList list = dir.entryInfoList();
    vector <pair<QString, int> > file;

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

    QFileInfo fileInfo = list.at(i);
```

```
    QString name = fileInfo.fileName();
    unsigned int sz = fileInfo.size();
    file.push_back(make_pair(name, sz));
    }

this->fileInfo=file;
this->savefile=file ;
gfile=file;
notifier = new QSocketNotifier( fileno(stdin), QSocketNotifier::Read,
this );
connect(notifier,SIGNAL(activated(int)),this,SLOT(CommandEntered()));
this->timerID=startTimer(timer);
}

void Klass::timerEvent(QTimerEvent *e)
{
    if ( this->timerID==e->timerId()){

    vector <pair<QString, int> > vec;
    vec=this->fileInfo;

    set <QString,cmp> st;
    for(size_t i = 0; i < vec.size(); ++i)
     {
        st.insert(vec[i].first);
     }
   if(st.empty())
    {
      cout<<"{}"<<endl<<">"<<endl;    }
   else
   {
    set<QString,cmp>::iterator it;


    for (it = st.begin(); it != st.end(); ++it)
    {
        if(it==st.begin()){cout << "{"<<"";}
        cout <<(*it).toStdString() ;
        if(it!=(--st.end())){ cout<<", ";};
        if(it==(--st.end())){cout<<"} ";};
    }

    cout<<endl;
    cout<<">"<<endl;

     int t=0;
     int avg=0;
     int sum=0;

    for (size_t i = 0; i < vec.size(); ++i)
     {
      sum+=vec[i].second;
     }
    avg=sum/vec.size();
    int min = avg;
    for (size_t i = 0; i < vec.size(); ++i)
     {
```

```
      if(sqrt((vec[i].second-(avg*3/4)*(vec[i].second-
(avg*3/4)))<=min))
      {
        min=vec[i].second;
        t=i;
      }
    }

    vec.erase(vec.begin() + t);
    this->fileInfo=vec;
   }
   }
else {this->timerID=e->timerId();}
}

void Klass::stop()
{
killTimer(this->timerID);
}
void Klass::start()
{
 this->timerID=startTimer(this->timer);
}
void Klass::restart()
{
    killTimer(this->timerID);
    this->fileInfo=this->savefile;
    this->timerID=startTimer(this->timer);
}
void Klass::timeout(int t)
{
this->timer=t;
this->stop();
this->timerID=startTimer(t);
}
void Klass::CommandEntered()
{
    string cmd;
    getline(std::cin,cmd);
    if(cmd=="stop"){this->stop();}
    else if(cmd=="start"){this->start();}
    else if(cmd=="restart"){this->restart();}
    else if(cmd=="exit"){exit(0);}
    else
    {
    if((0==cmd.find("timeout<") )&& ((cmd.size() - 1) =cmd.find('>')))
    {
    cmd = cmd.substr(8,cmd.size() - 9);
    this->timeout(atoi(cmd.c_str()));
    }
    else{cout<<"Wrong command"<<endl;}
    }
}
```