Государственное бюджетное образовательное учреждение высшего образования
Московской области
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
Колледж космического машиностроения и технологий

ОТЧЕТ

по производственной практике ПП.01.01 по модулю ПМ.01

«Разработка программных модулей программного обеспечения

для компьютерных систем»

по специальности 09.02.03 «Программирование в компьютерных системах»

Выполнил студент гр. П1-16
Партанский И.И.

_____ (подпись)

Принял преподаватель
Гусятинер Л. Б.

_____(подпись)

_____(оценка)

Королев, 2019

Оглавление

Введение 3
1. Общие сведения об организации 4
1.1. Структура организации 4
1.2. Структура отдела 4
1.3. Основные функции отдела 5
1.4. Нормативные документы, которые регламентируют профессиональную деятельность 5
1.5. Информационные технологий предприятия 6
1.6. Программное обеспечение предприятия 6
1.7. Задачи, подлежащие автоматизации 6
2. Содержание выполняемых видов работ 7
2.1. Разработка спецификаций отдельных компонент 7
2.2. Разработка кода программного продукта на основе готовых спецификаций на уровне модуля 8
2.3. Отладка программного модуля с использованием специализированных программных средств 10
2.5. Оптимизация программного кода модуля 12
2.6. Разработка компонентов проектной и технической документации с использованием графических языков спецификаций 12
3. Выводы 14
4. Дневник практики 15
5. Список использованной литературы 16
6. Приложения 17

Введение

На 3 курсе обучения в ККМТ, студентом группы П1-16 Партанским Ильёй была пройдена производственная практика по модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» на предприятии АО «ЭКА». Студент получил задание разработать программный модуль (ПМ) и инструкцию оператору для визуализации исходных данных, текущих и выходных результатов функционирования ПМ, разрабатываемых по индивидуальным заданиям № 1-3, а также изучить структуру предприятия, на котором он проходит практику.

1. Общие сведения об организации

1.1. Структура организации

Организационная структура организации представлена на Рис. 6.1 в Приложении 1.

Информационная структура организации представлена на Рис. 6.2 в Приложении 2.

1.2. Структура отдела

Во главе стоит руководитель отдела, в его подчинении находятся заместитель руководителя и инженер-программист, техники-программисты выполняют поручения инженера-программиста.

Организационная структура отдела представлена на Рис. 1.2.1.

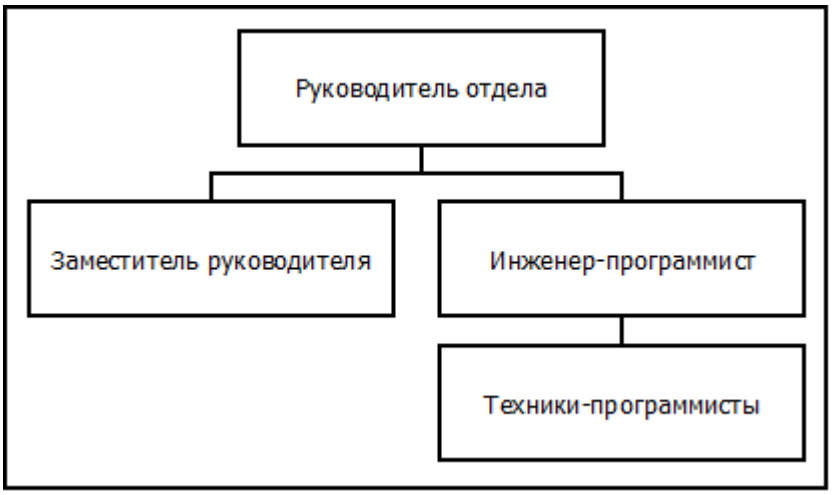
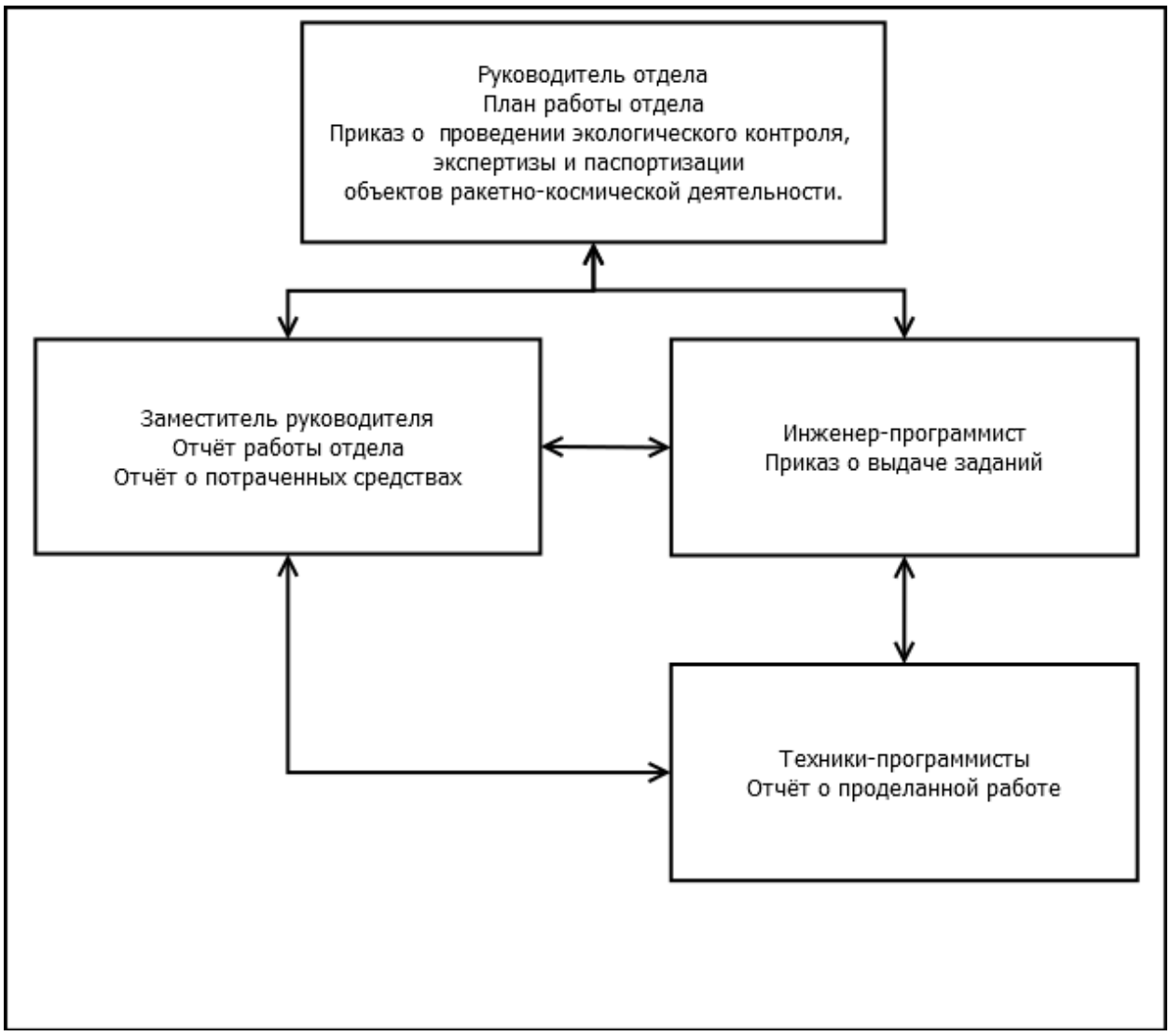


Рис. 1.2.1. Организационная структура отдела.

Информационная структура отдела представлена на Рис. 1.2.2.



(Рис. 1.2.2. Информационная структура отдела)

1.3. Основные функции отдела

- обеспечение создания и развития информационно-телекоммуникационных систем;
- обеспечение обслуживания и технической поддержки информационно-телекоммуникационных систем.

1.4. Нормативные документы, которые регламентируют профессиональную деятельность

- Политика информационной безопасности организации;
- Положение по обеспечению информационной безопасности организации;
- Положение о порядке доступа к информационным ресурсам;

- Положение о коммерческой или служебной тайне;
- Порядок обращения с информацией, подлежащей защите;
- Порядок защиты от несанкционированного доступа к информации и незаконного вмешательства в процесс функционирования информационной системы;
- Положение об отделе информационной безопасности;
- Классификация и перечень основных видов угроз информационной безопасности предприятия;
- Требования и рекомендации по обеспечению информационной безопасности предприятия;
- Положение об отделе технической защиты информации.

1.5. Информационные технологий предприятия

- GitHub
- Skype 8.34.0.78

1.6. Программное обеспечение предприятия

- Qt Creator 4.8.2
- Windows 8.1
- MS Office 2013
- SQLite Studio 3.2.1
- Notepad++ 7.7

1.7. Задачи, подлежащие автоматизации

- Создание систем управления рисками предприятий, органов исполнительной власти федерального и муниципального уровней.
- Разработка систем управления проектами создания сложных технических систем.
- Оценка рисков проектов создания сложных технических систем.

2. Содержание выполняемых видов работ

2.1. Разработка спецификаций отдельных компонент

Общее задание было разделено на 5 модулей:

1. ПМ1. Программный модуль (ПМ) на основе выданного описания алгоритма моделирования процессов синхронизации радиоданных и получения сообщения одноканальным приемником от нескольких источников.

Входные данные модуля: база данных, текстовый файл из ПМ3.

Передача данных в ПМ2, ПМ3: журнал записей, минимальное время получения сообщения приёмника.

2. ПМ2. Программный модуль (ПМ)на основе выданного описания алгоритма оценки вероятностно-временных характеристик получения сообщения многоканальным приемником (несколькими однотипными получателями) от нескольких источников.

Входные данные модуля: база данных, текстовый файл из ПМ1 и ПМ3.

Передача данных в ПМ4: вероятности от конкретного источника и конкретного времени.

3. ПМ3. Программный модуль (ПМ) на основе выданного описания алгоритма поиска минимального времени доведения сообщения по иерархической сети ретрансляторов.

Входные данные модуля: база данных, текстовый файл из ПМ1.

Передача данных в ПМ5, ПМ1, ПМ2: журнал записей, минимальное время, текстовый файл.

4. ПМ4. Программный модуль (ПМ)на основе выданного описания алгоритма выбора рационального состава радиоданных одноканального (многоканального) приёмника для группировки однотипных получателей для группировки однотипных получателей для заданных состава и характеристик источников.

Входные данные модуля: база данных, текстовый файл из ПМ2.

5. ПМ5. Программный модуль (ПМ) для визуализации исходных данных, текущих и выходных результатов функционирования ПМ, разрабатываемых по индивидуальным заданиям № 1-3.

Входные данные модуля: база данных, журнал записей из ПМ3.

2.2. Разработка кода программного продукта на основе готовых спецификаций на уровне модуля

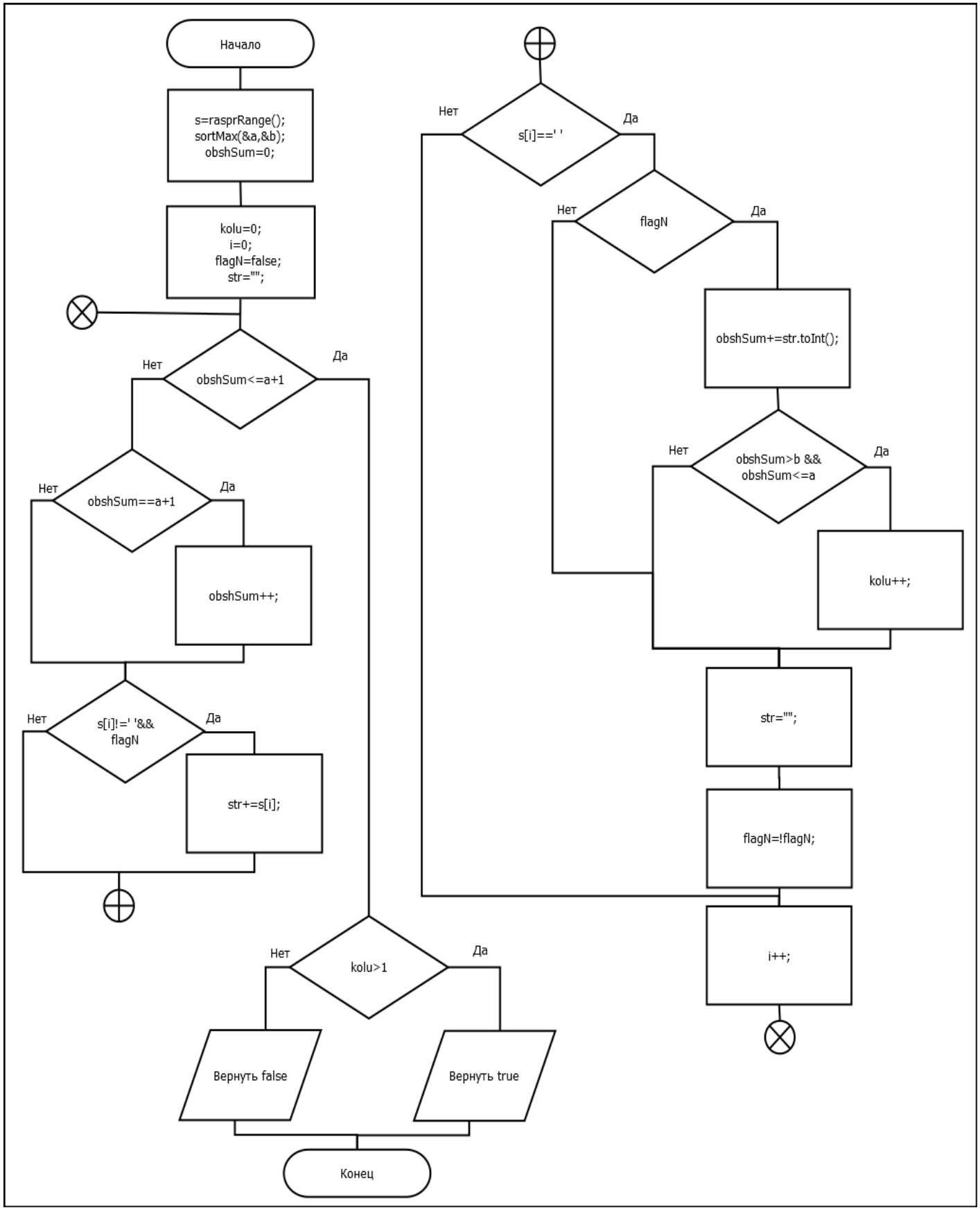
Модуль отображения всех объектов приведён в Листинге 1. Данный модуль располагает объекты, считанные из базы данных в зависимости от его ранга, а также устанавливает нужное изображение (Вид объекта).

Листинг 1. Модуль отображения всех объектов.

```
boolTextOut::dalSvaz(int a, int b)//функция определяющая
находится ли введённые РТР в дальности рангов более чем на один
{
QString s=rasprRange();
    sortMax(&a,&b);
    int obshSum=0;
    int kolu=0;
    int i=0;
    bool flagN=false;
    QString str="";
    while ((obshSum)<=a+1){
        if ((obshSum)==a+1){
            obshSum++;
        }
        if((s[i]!=' ')&&(flagN))
{
str+=s[i];
        }

if (s[i]==' '){
            if (flagN){
obshSum+=str.toInt();
if ((obshSum>b) &&(obshSum<=a)){
                    kolu++;
                }
                str="";
            }
            flagN=!flagN;
        }
        i++;
    }
    if (kolu>1){
        return true;}
    else {
```

```
returnfalse;
    }
}
```



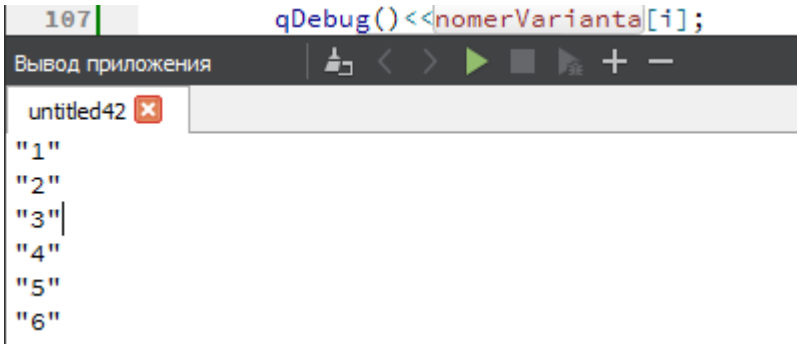
(Рис. 2.2.1. Блок-схема листинга 1)

2.3. Отладка программного модуля с использованием специализированных программных средств

Отладка проводилась с помощью 2 функций qDebug() и console.log()

qDebug- это функция класса QDebug, позволяющая выводить сообщения и данные в консоль.

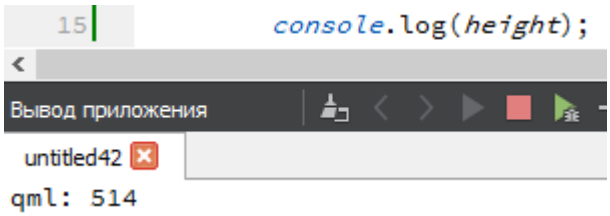
С его помощью я проводил отладку функцииperehodPoiskкласса TextOut (Листинг 6.1 Приложение 3), проводилась проверка правильности вводимых номеров текущих объектов. Пример приведён на Рис.2.3.1.



(Рис.2.3.1. Пример отладки qDebug)

console.log() – этофункция позволяющая выводить сообщения и данные в консоль.

С его помощью я проводил отладку слота onHeightChanged файла qml (Листинг 6.2 Приложение 4), проводилась проверка изменения высоты главного экрана.Пример приведён на Рис.2.3.2.



(Рис.2.3.2. Пример отладки console.log())

2.4. Тестирование программного модуля

Тестирование программы с использованием входных данных из базы данных (Рис.2.4.1) и текстового документа(Рис.2.4.2).

	ID	DeviceTyp	K	Condition	Number	Ak	Bk	StartTime	RepeatDur	MaxRepea	FMessageSent	BrokenTime
1	0	1	NULL	10;1|30;2|90;1		NULL	NULL	NULL	NULL	NULL	NULL	100
2	1	0	1	10;1|30;2|90;1	1	10	0	10	10	15	0;0;0;0|0;0;0;0|0.108;0;0;0|0.121;0;0;0|0.681;0.18;0.05;0.1|0.742;0.24;0.06;0.1|0.937;0.61;0.28;0.3	50
3	2	0	2	10;1|30;2|90;1	1	20	180	20	10	15	0;0;0;0|0;0;0;0|0.108;0;0;0|0.121;0;0;0|0.681;0.18;0.05;0.1|0.742;0.24;0.06;0.1|0.937;0.61;0.28;0.3	150
4	3	0	3	10;1|30;2|90;1	1	12	45	40	10	15	0;0;0;0|0;0;0;0|0.108;0;0;0|0.121;0;0;0|0.681;0.18;0.05;0.1|0.742;0.24;0.06;0.1|0.937;0.61;0.28;0.3	170
5	4	0	4	10;1|30;2|90;1	1	8	135	25	10	15	0;0;0;0|0;0;0;0|0.108;0;0;0|0.121;0;0;0|0.681;0.18;0.05;0.1|0.742;0.24;0.06;0.1|0.937;0.61;0.28;0.3	150
6	5	2	5	10;1|30;2|90;1	1	15	270	25	10	15	0;0;0;0|0;0;0;0|0.108;0;0;0|0.121;0;0;0|0.681;0.18;0.05;0.1|0.742;0.24;0.06;0.1|0.937;0.61;0.28;0.3	150

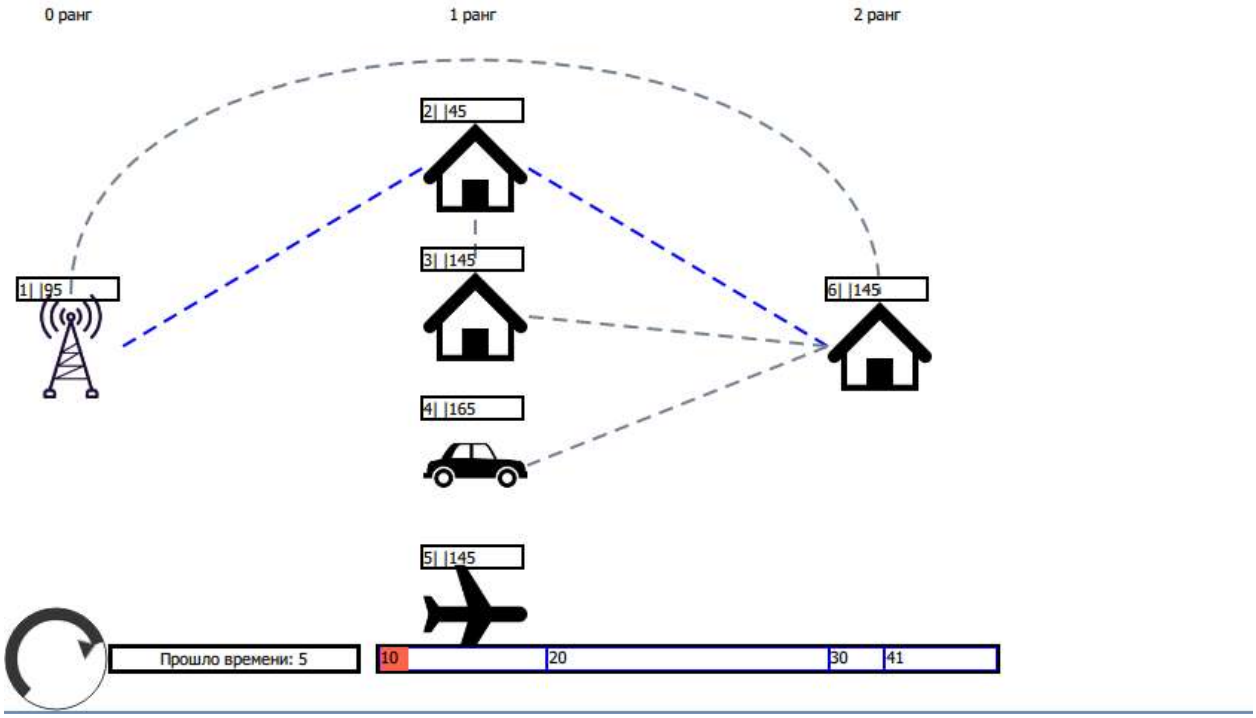
(Рис. 2.4.1. Входные данные из базы данных)

input.txt

```
1 НС 0 1 6
2 НУС 0 1 11
3 НС 0 1 12
4 УС 0 1 17
5 ПСО 0 1 33
```

(Рис. 2.4.2. Входные данные из текстового файла)

Выходные данные программы является отрисовка визуализации радиопередачи от источника к ретранслятору и к приёмнику(Рис. 2.4.3.).



(Рис. 2.4.3. Выходные данные)

2.5. Оптимизация программного кода модуля

Была оптимизирована функция определенияцветашкалы, она приложена в Листинге 2.5.1.

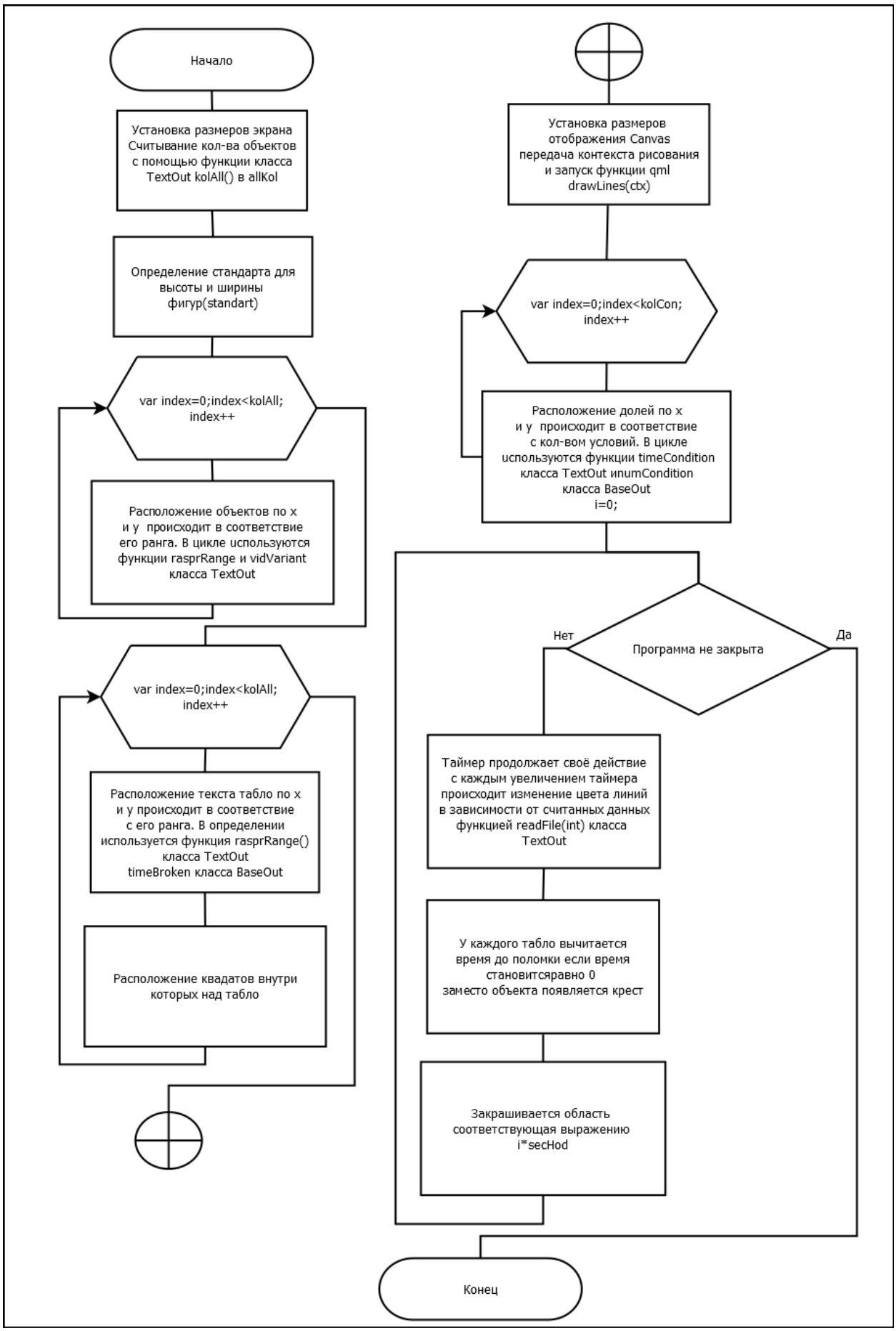
Листинг2.5.1. Функцияопределенияцветашкалы до оптимизации.

```
var a=[10,20,30];
if (parseInt(qwer.numCondition(index))==a[0]){
return "tomato";
                    }
else if (parseInt(qwer.numCondition(index))==a[1]){
return "orangered";
                    }
else if (parseInt(qwer.numCondition(index))==a[2]){
return "red";
                    }
else {
return "darkred";
}
```

Листинг2.5.2. Функцияопределенияцветашкалы после оптимизации.

```
if (parseInt(qwer.numCondition(index))==10){
return "tomato";
                    }
else if (parseInt(qwer.numCondition(index))==20){
return "orangered";
                    }
else if (parseInt(qwer.numCondition(index))==30){
return "red";
                    }
else {
return "darkred";
                    }
```

2.6. Разработка компонентов проектной и технической документации с использованием графических языков спецификаций



(Рис. 2.6.1Общая блок-схема модуля)

3. Выводы

Полученные навыки:

- Работа со структурами в QtCreator
- Работа с базами данных в QtCreator
- Компиляция программы в QtCreator
- Создание баз данных в SQLiteStudio

Полученные умения:

- Разработка программ в Qt Creator
- Работа с файловой системой в QtCreator
- Работа с файлами qml
- Разработка программ на qml

4. Дневник практики

Таблица 1. Дневник практики.

Дата	Содержание работ	Отметка о выполнении
27.04-30.04	Разработка логики программы.	
01.05	Подключение считывания из БД.	
02.05	Подключение считывания из текстового файла.	
03.05-17.05	Разработка функций отображения объектов и табло над ними.	
18.05-01.06	Разработка функций передачи данных между cpp и qml файлами.	
02.06-14.06	Разработка функций отображения шкалы времени и таймера	
15.06-28.06	Разработка функции смены цвета линий связи и их отрисовка	

5. Список использованной литературы

1. Документация по классу QTextStream. http://doc.crossplatform.ru/qt/4.7.x/qtextstream.html
2. Документация по библиотеке QSqlQuery. http://doc.crossplatform.ru/qt/4.7.x/qsqlquery.html
3. Работа с SQLite. https://zametkinapolyah.ru/zametki-o-mysql/tema-12-sql-zaprosy-select-v-bazax-dannyx-sqlite.html
4. Книга. Макс Шлее: “Qt 5.10 Профессиональное программирование на C++”.

6. Приложения

Приложение 1

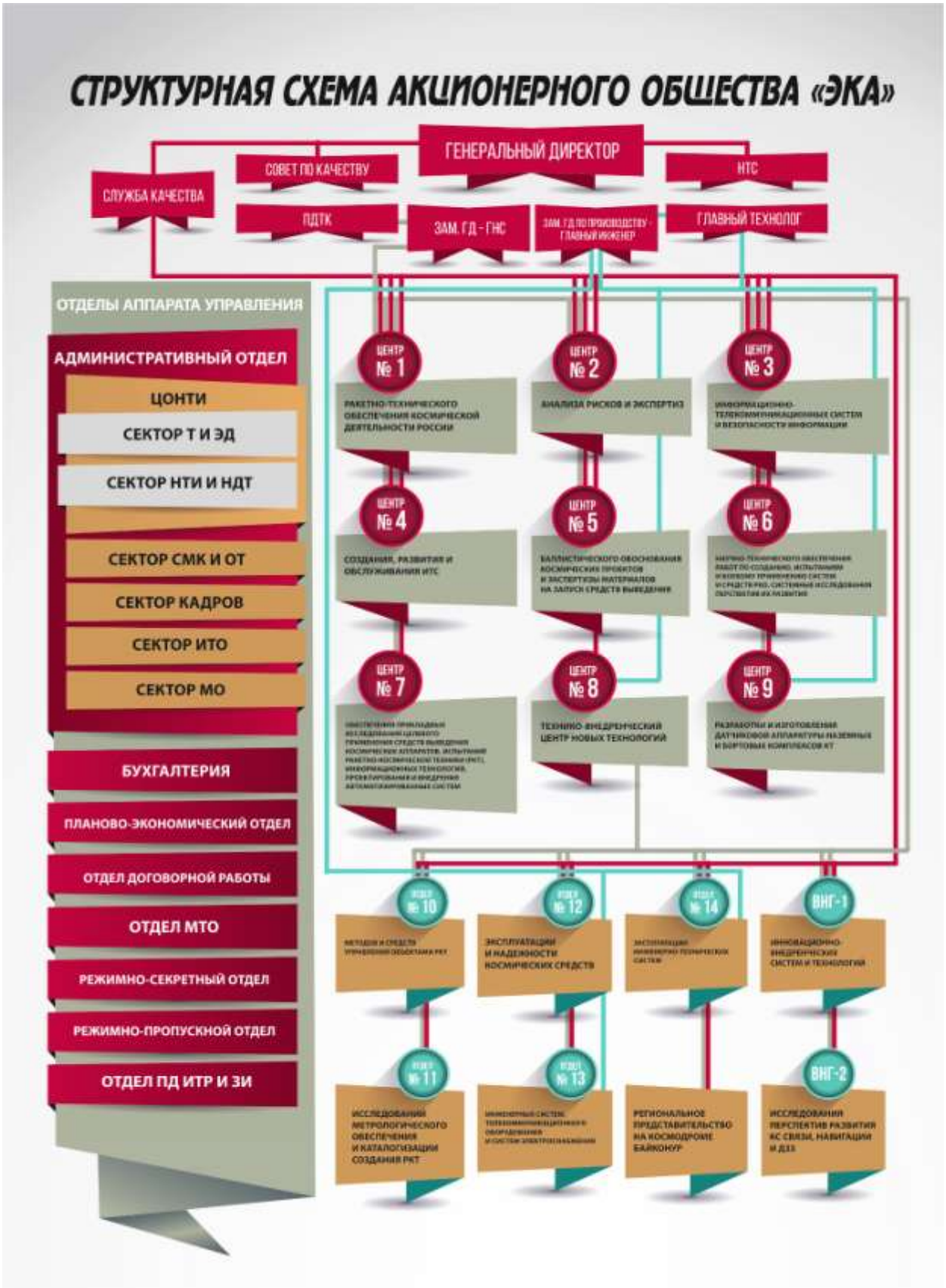
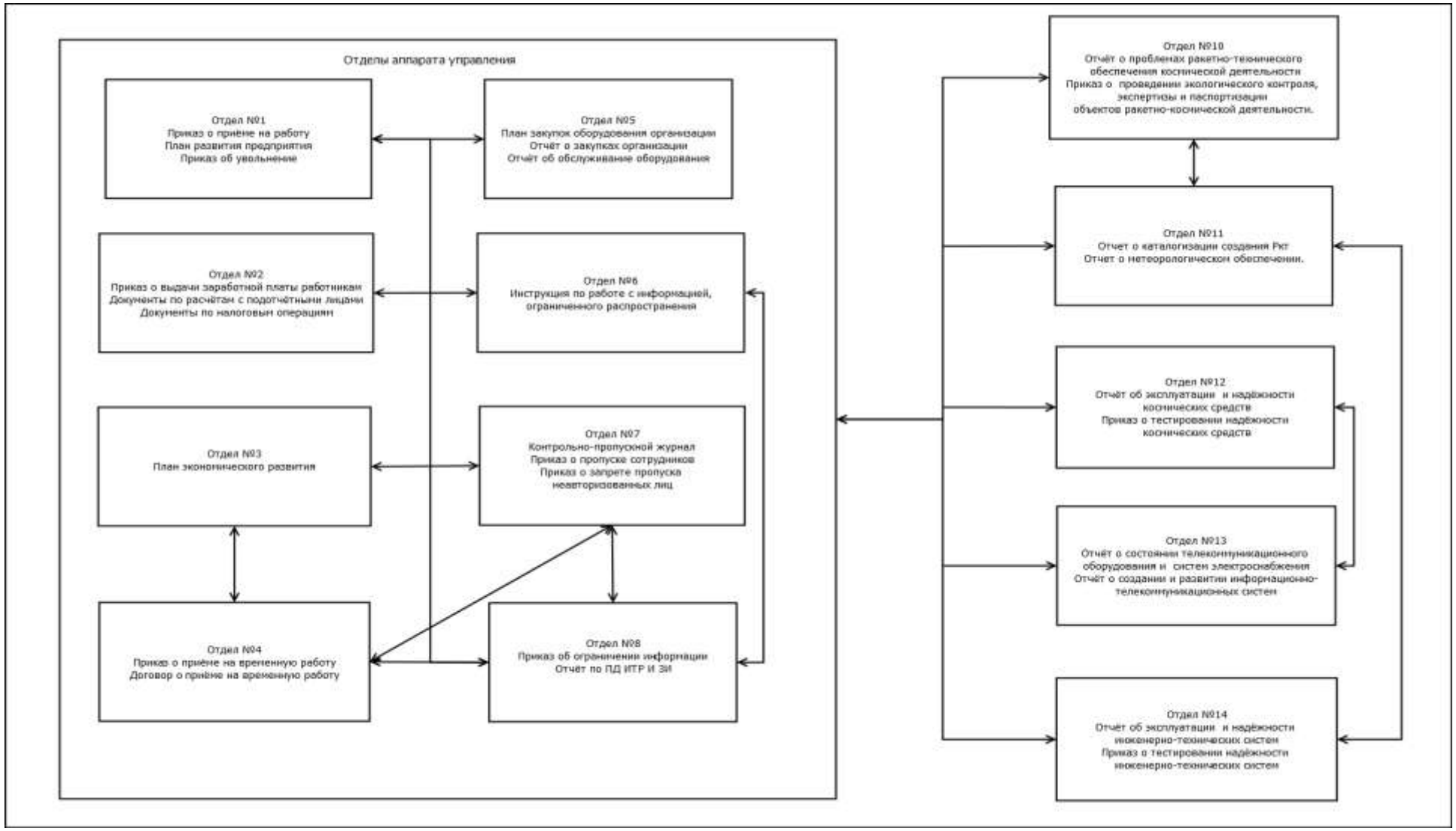


Рис. 6.1. Организационная структура организации.

Приложение 2



(Рис.6.2 Информационная структура организации)

Приложение 3

Листинг 6.1 файлtextout.cpp.

```
#include "textout.h"



TextOut::TextOut(QObject *parent) : QObject(parent)
{
    QString er;
    maxRange(&maxR);
    er=rasprRange();
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("Data.db");
    if (db.open()){
    QSqlQuery query;
query.exec("SELECT * FROM RTR");//Осуществляемзапрос
    while (query.next()){
        QString radio =
query.value(2).toString(),str="",num=query.value(0).toString();//распределяем
радиоданныеиномеробъекта
nowKanalsVariants.append(query.value(3).toString());//устанавливаемкудасмотри
тобъектсейчас
nowKanalsTimes.append("");//добавляем место под время каналов
for (int i = 0; i < nowKanalsVariants[nowKanalsVariants.size()-1].length();
++i) {
            if(nowKanalsVariants[nowKanalsVariants.size()-1][i]==','){
nowKanalsTimes[nowKanalsVariants.size()-
1]+="0,";//устанавливаемкаждомуканалуеготекущеевремя
}
        }
nowKanalsTimes[nowKanalsVariants.size()-1]+="0";//последний канал который
находится за последней зхапятой
radioData.append(radio);//записываем радиоданные
nomerVarianta.append(num);//записываем номер текущего объекта который мы
считали из базы
for (int i = 0; i <radio.length(); ++i) {//формируем линии связи для
отрисовки в qml по радиоданным
if (radio[i]!=','){
                str+=radio[i];
            } else if(radio[i]==','){
                svaz.append(num+'|'+str);
                str="";
            }
        }
        svaz.append(num+'|'+str);

    }
query.exec("SELECT SearchInterval FROM Devices");//запрашиваеминтервалы
    while (query.next()){
        QString strTmp=query.value(0).toString();
        if (strTmp!=""){
intervalSvyazi.append(strTmp.toInt());//Еслизначениевбазене Null
устанавливаемточтовбазе
        }
        else{intervalSvyazi.append(-1);}
    }

    for (int i = 0; i < svaz.size(); ++i) {
boolSvaz.append("slategray");//устанавливаем для каждой линии связи цвет по
умолчанию
```

```
}

    }else{
        qDebug()<<"Data.db is not open";
    }
}
int TextOut::getSomeProperty()const
{
    return kol;
}

int TextOut::getMaxRProperty() const
{
    return maxR;
}

void TextOut::changeBool(bool *t)//сменабулевыхзначений
{
    *t=!(*t);
}
void TextOut::setSomeProperty(const int &i)
{
    kol = i;
}

void TextOut::setMaxRProperty(const int &i)
{
    maxR = i;
}

void TextOut::changeString(QString *s, QString
er)//функцияменяющаяцветввекторе
{
if (er=="НС"){
        *s="yellow";
    } else if (er=="НУС") {
        *s="red";
    } else if (er=="УС") {
        *s="green";
    } else if(er=="С"){
        *s="blue";
    } else {
        *s="slategray";
    }
}

void TextOut::sortMax(int *a, int *b)//функцияменяющаяпеременныеместамиесли
a<b
{
    if (*a<*b){
        int tmp=*b;
        *b=*a;
        *a=tmp;
    }
}

void TextOut::perehodPoisk(int a, int
nowTime)//осуществлениепереходапоискаупередатчиков
{
for (int i = 0; i <nomerVarianta.size(); ++i) {//делаем проход по всем
доступным вариантам
qDebug()<<nomerVarianta[i];
```

```
if ((nomerVarianta[i].toInt()-1)==a &&intervalSvyazi[i]!=-1){//если мы
считали нужный нам вариант продолжаем действия
intschetSinh=0;//счётчик кол-ва элементов для вектора nowSinh
QVector<int>nowSinh;//вектор в который поместим куда смотрит канал в данный
момент
nowSinh.append(0);//добавляем сразу элемент туда
for (int j = 0; j <nowKanalsVariants[i].length(); ++j) {//цикл в котором
помещаем данные из nowKanalsVariants в nowSinh
if (nowKanalsVariants[i][j]!=','){
                    nowSinh[schetSinh]*=10;
                    QString strTmp=""+nowKanalsVariants[i][j];
                    nowSinh[schetSinh]+=strTmp.toInt();
                } else if (nowKanalsVariants[i][j]==',') {
                    nowSinh.append(0);
                    schetSinh++;
                }

            }
            int schetKolTimer=0;//счётчиккол-ваэлементовдлявектора nowSinh
QVector<int>nowTimer;//вектор в который поместим куда смотрит канал в данный
момент
nowTimer.append(0);//добавляем сразу элемент туда
for (int j = 0; j <nowKanalsTimes[i].length(); ++j) {//цикл в котором
помещаем данные из nowKanalsVariants в nowSinh
if (nowKanalsTimes[i][j]!=','){
                    nowTimer[schetKolTimer]*=10;
                    QString strTmp=""+nowKanalsTimes[i][j];
                    nowTimer[schetKolTimer]+=strTmp.toInt();
                } else if (nowKanalsTimes[i][j]==',') {
                    nowTimer.append(0);
                    schetKolTimer++;
                }

            }

for (int y = 0; y < nowSinh.size(); ++y) {//проходимповсемканаламкоторыеесть
if ((nowTimer[y]+intervalSvyazi[i])<=nowTime){//проверяем стоит ли сейчас
переместить поиск на другой вариант
nowSinh[y]++;//если это произошло прибавляем в текущем канале смещение куда
стоит смотреть
intqw=0;//счётчик кол-ва радиоданных
for (int j = 0; j <radioData[i].length(); ++j) {//проходимся по радиоданным
текущего варианта
if(radioData[i][j]==','){
qw++;
                        }
                    }
qw++;//прибавляем к кол-ву 1 из-за того что запятых всегда на 1 меньше чем
радиоданных
if (qw<nowSinh[y]){//если мы зашли за большее кол-во радиоданных чем есть
nowSinh[y]=1;//возвращаемв 1
                        }

                nowTimer[y]+=intervalSvyazi[i];
                }
                QString str=nomerVarianta[i]+'|';
                int tmp=1;
                    for (int j = 0; j < radioData[i].length(); ++j) {
                        if ((tmp==nowSinh[y])&&(radioData[i][j]!=',')){
                            str+=radioData[i][j];
                        }
                        if(radioData[i][j]==','){
                            tmp++;
```

```
                        }

                    }
                    for (int j = 0; j <svaz.size() ; ++j) {
                        if (str==svaz[j]){

                            tmp=j;
                        }
                    }
                    if((nowTimer[y]+intervalSvyazi[i])>nowTime){

if((boolSvaz[tmp]=="slategray")||(boolSvaz[tmp]=="red")){
                            changeString(&boolSvaz[tmp],"C");
                        }else
if((boolSvaz[tmp]=="green")||(boolSvaz[tmp]=="yellow")){
                            nowTimer[y]++;
                        }
                    }

                }
            nowKanalsTimes[i]="";
            nowKanalsVariants[i]="";
            for (int k = 0; k < nowTimer.size(); ++k) {
                if ((nowTimer.size()-1)!=k){
                nowKanalsTimes[i]+=QString::number(nowTimer[k])+',';
                nowKanalsVariants[i]+=QString::number(nowSinh[k])+',';}
                else {
                    nowKanalsTimes[i]+=QString::number(nowTimer[k]);
                    nowKanalsVariants[i]+=QString::number(nowSinh[k]);
}
            }
            }

    }

}

boolTextOut::dalSvaz(int a, int b)//функция определяющая находится ли
введённые РТР в дальности рангов более чем на один
{
    QString s=rasprRange();
    sortMax(&a,&b);
    int obshSum=0;
    int kolu=0;
    int i=0;
    bool flagN=false;
    QString str="";
    while ((obshSum)<=a+1){
        if ((obshSum)==a+1){
            obshSum++;
        }
        if((s[i]!=' ')&&(flagN))
        {
            str+=s[i];
        }

        if (s[i]==' '){
            if (flagN){
                obshSum+=str.toInt();
                if ((obshSum>b) &&(obshSum<=a)){
                    kolu++;
                }
                str="";
```

```
            }
            flagN=!flagN;
        }
        i++;
    }
    if (kolu>1){
        return true;}
    else {
        return false;
    }
}

QString TextOut::sendColor(int a, int b)//получениеизвекторанужногоцвета
{
    QString str1=QString::number(a+1)+'|'+QString::number(b+1);
    for (int i=0;i<svaz.size();i++){
        if (str1==svaz[i]){
            return boolSvaz[i];
        }
    }
    return "-1";
}



void TextOut::getColor(int a, int b,QString s)//измененииецветанужнойсвязи
{
    QString str1=QString::number(a+1)+'|'+QString::number(b+1);
    for (int i=0;i<svaz.size();i++){
        if (str1==svaz[i]){
            changeString(&boolSvaz[i],s);
        }
    }
}



QString TextOut::readFile(int t)//считываниенужнойстрокифайла
{
    int i=0;
    QFile file(":/txt/input.txt");
        if ((file.exists())&&(file.open(QIODevice::ReadOnly)))
        {
            while(!file.atEnd()){
                    QString s=file.readLine();
                if (t==i){
                    return s;
                }
                i++;
            }
        }
        else{
            qDebug()<<"input.txt is not open";
return "-1";
        }

}



boolTextOut::isSvaz(int a, int b)//проверка имеется ли связь между
запрошенными объектами
{
```

```
    QString str1=QString::number(a+1)+'|'+QString::number(b+1);
    for (int i=0;i<svaz.size();i++){
        if (str1==svaz[i]){
            return true;
        }
    }
    return false;
}



void TextOut::maxRange(int *t)//функцияопределениямаксимальногоранга
{
    int maxi=0;
        QSqlDatabase db;
        db = QSqlDatabase::addDatabase("QSQLITE");
        db.setDatabaseName("Data.db");
        if (db.open()){
        //Осуществляемзапрос
        QSqlQuery query;
        query.exec("SELECT Rank FROM Devices");
        while (query.next()){
            QString str=query.value(0).toString();
            if (str.toInt()>maxi){
                maxi=str.toInt();
            }
        }
        db.close();
        *t=maxi+1;
        } else{
            qDebug()<<"Data.db is not open";
}
}

QStringTextOut::rasprRange()//Функция просматривающая как распределяются
ранги
{
    int ar[maxR];
        for (int i = 0; i < maxR; ++i) {
            ar[i]=0;
        }
        /**/
        QSqlDatabase db;
        db = QSqlDatabase::addDatabase("QSQLITE");
        db.setDatabaseName("Data.db");
        if (db.open()){
        //Осуществляемзапрос
        QSqlQuery query;
        query.exec("SELECT Rank FROM Devices");
        while (query.next()){
            QString str=query.value(0).toString();
            ar[str.toInt()]++;
        }
        QString itog="";
        for (int i = 0; i < maxR; ++i) {
            itog+=QString::number(i)+' '+QString::number(ar[i])+' ';
        }
        itog+='.';
        return itog;
        }else {
            qDebug()<<"Data.db is not open";
            return "error";
        }
```

```
}

int TextOut::maxKolRange()
{
    int maxt=0;
        int ar[maxR];
        for (int i = 0; i < maxR; ++i) {
            ar[i]=0;
        }
        /**/
        QSqlDatabase db;
        db = QSqlDatabase::addDatabase("QSQLITE");
        db.setDatabaseName("Data.db");
        if (db.open()){
        //Осуществляемзапрос
        QSqlQuery query;
        query.exec("SELECT Rank FROM Devices");
        while (query.next()){
            QString str=query.value(0).toString();
            ar[str.toInt()]++;
        }
            for (int i = 0; i < maxR; ++i) {
                if (ar[i]>maxt){
                    maxt=ar[i];
                }
            }
        }else{
            qDebug()<<"Data.db is not open";
}
returnmaxt;
}

intTextOut::kolAll()//количество всех PTP приёмников и источников
{
    int sumAll=0;
        int ar[maxR];
        for (int i = 0; i < maxR; ++i) {
            ar[i]=0;
        }
        /**/
        QSqlDatabase db;
        db = QSqlDatabase::addDatabase("QSQLITE");
        db.setDatabaseName("Data.db");
        if (db.open()){
        //Осуществляемзапрос
        QSqlQuery query;
        query.exec("SELECT Rank FROM Devices");
        while (query.next()){
            QString str=query.value(0).toString();
            ar[str.toInt()]++;
        }
            for (int i = 0; i < maxR; ++i) {
                sumAll+=ar[i];
            }
        db.close();
        }else{
            qDebug()<<"Data.db is not open";
        }
            return sumAll;
}

QString TextOut::vidVariant(int a)
{
```

```
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("Data.db");
    if (db.open()){
    //Осуществляемзапрос
    QSqlQuery query;
    int i=0;
    query.exec("SELECT ImageType FROM Devices");
    while (query.next()){
    QString vid = query.value(0).toString();
    if(i==a){
        db.close();
        return vid+".png";
    }
    i++;
    }
    db.close();
    } else{
        qDebug()<<"Data.db is not open";
    }

    return "1.png";
}

int TextOut::timeCondition(int a)
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("Data.db");
    if (db.open()){
    //Осуществляемзапрос
    QSqlQuery query;
    query.exec("SELECT * FROM conditions");
    int i=0,tmp=0;
    while (query.next()){
    QString numCon = query.value(0).toString();
    QString timeCon = query.value(1).toString();
    if(i==a){
        db.close();
        return timeCon.toInt();
    }
    i++;
    tmp=timeCon.toInt();
    }
    db.close();
    return tmp+20;
    }else{
    qDebug()<<"Data.db is not open";
    return -1;
}
}

int TextOut::kolCon()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("Data.db");
    if (db.open()){
    //Осуществляемзапрос
    QSqlQuery query;
    query.exec("SELECT * FROM conditions");
    int i=0;
    while (query.next()){
```

```
    QString numCon = query.value(0).toString();
    QString timeCon = query.value(1).toString();
    i++;
    }
    db.close();
    return i;
    } else {
        qDebug()<<"Data.db is not open";
        return -1;
}
}

int TextOut::maxConTime()
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("Data.db");
    if (db.open()){
    //Осуществляемзапрос
    QSqlQuery query;
    query.exec("SELECT * FROM conditions");
    int i=0;
    while (query.next()){
    QString numCon = query.value(0).toString();
    QString timeCon = query.value(1).toString();
    i=timeCon.toInt();
    }
    db.close();
    return i+20;
    } else {
        qDebug()<<"Data.db is not open";
        return -1;
    }
}
```

Приложение 4

Листинг 6.2 файлmain.qml.

```
import QtQuick.Window 2.2
import QtQuick 2.10
import QtQuick.Controls 2.2
import QtQuick.Layouts 1.3
import ModuleName 1.0
import BasaGet 1.0


ApplicationWindow {
    id:window
    visible: true
    width: 840
    height: 480
    onHeightChanged: {
        console.log(height);
linii.clearline();//при каждом изменение экрана перерисовываем линии связи
    }
onWidthChanged: {
linii.clearline();//при каждом изменение экрана перерисовываем линии связи
    }
TypeName{//с помощью этого объекта переносим функции из textout.cpp
id:obh
    kol:0
    }
Yeah{//с помощью этого объекта переносим функции из baseout.cpp
id: qwer
    }
property real secHod : (aroundSh.width-4)/allTime//ходшкалывсекунду
    property int nowKol: 0
    property int rangeMax: obh.maxR//максимальноеколличестворанговв
property int kolInRangeMax: obh.maxKolRange();//максимальноеколличествовранге
propertyintallKol: obh.kolAll()//колличество всех PTP приёмников и источников
property string dannie: obh.readFile(obh.kol)
property int time: 1000 //времяциклатаймера
propertyintkolTime: 0//время пройденное с начала отсчёта
propertyboolflag: true //флаг для смены цвета линий
property color lineColor: "slategray"; //цветпоумолчанию
propertyintstandart: {//стандарт высоты и ширины изображений приёмников
if (window.height<window.width){
        if (rangeMax>=kolInRangeMax){
            return (window.height-200)/rangeMax;}
        else{
            return (window.height-200)/kolInRangeMax;}
    }
    else{
        if (rangeMax>=kolInRangeMax){
            return (window.width-200)/rangeMax;}
        else{
            return (window.width-200)/kolInRangeMax;}

}
    }
propertyintstanX: {//расстояние по x между приёмниками
return (window.width-20-rangeMax*standart)/rangeMax-1;
    }
    signal changeNowKol();
    onChangeNowKol: {
        nowKol++;
    }
```

```
function drawLines(ctx){//функцияотрисовкилинийсвязи
varsumall=obh.kolAll();//получаем кол-во всех Источников, РТР и Получателей
var tFlag=true;
        for (var i=0;i<sumall;i++){
for (var j=0;j<sumall;j++){//проходимся по всем возможным связям
if (obh.isSvaz(i,j)){
/**/
var s=obh.rasprRange();//считываем как распределились ранги
var obshSum=0;
                    var kolu=0;
                    var h=0;
                    var flagN=false;
                    var str="";
                    while (obshSum-1<i){
if((s[h]!=' ')&&(flagN))
                        {
str+=s[h];//считываем по одному символу количества объектов на ранге
                        }

if (s[h]==' '){//если пробел не считываем данные
if (flagN){
obshSum+=parseInt(str);//сколько всего объектов до этого ранга
kolu=parseInt(str);//объекты на ранге
str="";
                            }
flagN=!flagN;
                        }
                        h++;
                    }
if ((j<obshSum)&&(j>=(obshSum-kolu))){//если объекты попадают в один ранг
ctx.beginPath();
                        ctx.strokeStyle=obh.sendColor(i,j);
ctx.setLineDash([5, 3]);//задача на отрисовку пунктирных линий
ctx.lineWidth = 2;

ctx.moveTo(allSquare.itemAt(i).x+standart/2,allSquare.itemAt(i).y+standart);

ctx.lineTo(allSquare.itemAt(j).x+standart/2,allSquare.itemAt(j).y);
ctx.stroke();
                    }
elseif(obh.dalSvaz(i,j)){//если между объектами есть 1 или более рангов
ctx.beginPath();
                        ctx.strokeStyle=obh.sendColor(i,j);
ctx.setLineDash([5, 3]);//задача на отрисовку пунктирных линий
ctx.lineWidth = 2;
                        if (tFlag){
if (i<j){//еслиобъект i ближе

ctx.moveTo(allSquare.itemAt(i).x+standart/2,allSquare.itemAt(i).y);

ctx.bezierCurveTo(allSquare.itemAt(i).x+standart,allSquare.itemAt(i).y-
standart*3,allSquare.itemAt(j).x,allSquare.itemAt(j).y-
standart*3,allSquare.itemAt(j).x+standart/2,allSquare.itemAt(j).y);
                        }else{

ctx.moveTo(allSquare.itemAt(j).x+standart/2,allSquare.itemAt(j).y);

ctx.bezierCurveTo(allSquare.itemAt(j).x+standart/2,allSquare.itemAt(j).y-
standart*3,allSquare.itemAt(i).x+standart/2,allSquare.itemAt(i).y-
standart*3,allSquare.itemAt(i).x+standart/2,allSquare.itemAt(i).y);
                        }
```

```
                        tFlag=!tFlag;
                        }else{
if (i<j){//еслиобъект i ближе

ctx.moveTo(allSquare.itemAt(i).x+standart/2,allSquare.itemAt(i).y+standart);

ctx.bezierCurveTo(allSquare.itemAt(i).x+standart,allSquare.itemAt(i).y+standa
rt*4,allSquare.itemAt(j).x,allSquare.itemAt(j).y+standart*4,allSquare.itemAt(
j).x+standart/2,allSquare.itemAt(j).y+standart);
                            }else{

ctx.moveTo(allSquare.itemAt(j).x+standart/2,allSquare.itemAt(j).y);

ctx.bezierCurveTo(allSquare.itemAt(j).x+standart/2,allSquare.itemAt(j).y+stan
dart*4,allSquare.itemAt(i).x+standart/2,allSquare.itemAt(i).y+standart*4,allS
quare.itemAt(i).x+standart/2,allSquare.itemAt(i).y+standart);
                            }
                            tFlag=!tFlag;
                        }
                        ctx.stroke();
                }
                    else{
                        ctx.beginPath();
                        ctx.strokeStyle=obh.sendColor(i,j);
ctx.setLineDash([5, 3]);//задача на отрисовку пунктирных линий
ctx.lineWidth = 2;
if(i>j){//еслиобъект i дальше

ctx.moveTo(allSquare.itemAt(i).x,allSquare.itemAt(i).y+standart/2);

ctx.lineTo(allSquare.itemAt(j).x+standart,allSquare.itemAt(j).y+standart/2);}
                            if(i<j){

ctx.moveTo(allSquare.itemAt(i).x+standart,allSquare.itemAt(i).y+standart/2);

ctx.lineTo(allSquare.itemAt(j).x,allSquare.itemAt(j).y+standart/2);}
                        ctx.stroke();
                    }
                }
            }
        }
    }
    Canvas {//площадьотрисовки
        x:0
        y:0
        id: linii
        width: window.width; height: window.height
functionclearline() {//функция перерисовки пунктирных линий
var ctx = getContext("2d");
                    ctx.reset();
                    linii.requestPaint();
                }
        onPaint:{
            var ctx = getContext("2d");
drawLines(ctx) ;
        }
    }
Repeater{//Все объекты на карте
id: allSquare
    model:allKol
    Image{
x:{
var s=obh.rasprRange();//считываем как распределились ранги
```

```
var obshSum=0;
            var kolRange=0;
            var i=0;
            var flagN=false;
            var str="";
while (s[i]!='.'){//пока не достигнут конец строки
if((s[i]!=' ')&&(flagN))//если данный символ не пробел и он является
количеством рангов
{
                    str+=s[i];
                }

                if (s[i]==' '){
                    if (flagN){
                        obshSum+=parseInt(str);
                        str="";
                        if (index<obshSum){
                            return 10+stanX*(kolRange)+standart*(kolRange);
                        } else{
                            kolRange++;

                        }
                    }
                    flagN=!flagN;
                }
                i++;
            }

            return 10+stanX*(kolRange)+standart*(kolRange);

        }
        y:{
            var s=obh.rasprRange();
            var obshSum=0;
            var kolRange=0;
            var i=0;
            var flagN=false;
            var str="";
            while (s[i]!='.'){
                if((s[i]!=' ')&&(flagN))
                {
                    str+=s[i];
                }

                if (s[i]==' '){
                    if (flagN){
                        obshSum+=parseInt(str);
                        if (index<obshSum){
                            return (window.height-
(standart+10)*(parseInt(str)))/2+(obshSum-parseInt(str)-
index)*(standart+10)*-1-(obshSum-parseInt(str)-index)*20;
                        } else{
                            kolRange++;

                        }

                        str="";
                    }
                    flagN=!flagN;
                }
                i++;
            }
```

```
            return 10+stanX*(kolRange)+standart*(kolRange);


        }
        width: standart
        height: standart
source:
"qrc:/img/"+obh.vidVariant(index);//выставляемнужноеизображениекотороестоитвБ
Д
    }
    }
Repeater{//Отображениеверхнихтабло
        id:upTablo
        model:allKol
    Text{
        x:{
            var s=obh.rasprRange();
            var obshSum=0;
            var kolRange=0;
            var i=0;
            var flagN=false;
            var str="";
            while (s[i]!='.'){
                if((s[i]!=' ')&&(flagN))
                {
                    str+=s[i];
                }

                if (s[i]==' '){
                    if (flagN){
                        obshSum+=parseInt(str);
                        str="";
                        if (index<obshSum){
                            return 10+stanX*(kolRange)+standart*(kolRange);
                        } else{
                            kolRange++;

                        }
                    }
                    flagN=!flagN;
                }
                i++;
            }

            return 10+stanX*(kolRange)+standart*(kolRange);

        }
        y:{
            var s=obh.rasprRange();
            var obshSum=0;
            var kolRange=0;
            var i=0;
            var flagN=false;
            var str="";
            while (s[i]!='.'){
                if((s[i]!=' ')&&(flagN))
                {
                    str+=s[i];
                }

                if (s[i]==' '){
                    if (flagN){
                        obshSum+=parseInt(str);
```

```
                        if (index<obshSum){
                            return (window.height-
(standart+10)*(parseInt(str)))/2+(obshSum-parseInt(str)-
index)*(standart+10)*-1-(obshSum-parseInt(str)-index)*20-10;
                        } else{
                            kolRange++;

                        }

                        str="";
                    }
                    flagN=!flagN;
                }
                i++;
            }

            return stanX*(kolRange)+standart*(kolRange)+320;


}
text: (index+1).toString()+'| '+'|'+qwer.timeBroken(index);//текст
формируется из индекса объекта, тоесть его номера, пустое место под
радиоданные и время до конца работы
Rectangle{
            x:-2
            y:-2
            width: standart
            height:parent.height+4
            color:"transparent"
            border.color: "black"
            border.width: 2

        }
    }

    }
Repeater{//отображениеранговсверху
    model:obh.maxR
    Text {
        x:10+stanX*index+standart*index+standart/4
        y:5
text: qsTr((index).toString()+" ранг")
    }
    }
    Timer {
        interval: time
        repeat: true
        running: true
        onTriggered: {
            for (var i=0;i<upTablo.model;i++){
var tablo=upTablo.itemAt(i).text.split('|');//смотримэлементывтабло
tablo[2]-=1;//уменьшаем время работы на 1
if (tablo[2]>=0){//если не достигнут 0 просто отрисовываем табло по новой
upTablo.itemAt(i).text=tablo[0]+'|'+tablo[1]+'|'+tablo[2];
}
elseif(tablo[2]<0)//если 0 достигнут заменяем объект на крест
{
                    allSquare.itemAt(i).source="qrc:/img/5.png";
                }
            }
            kolTime++;//общеевремя
            var sumall=obh.kolAll();
            for (var i=0;i<sumall;i++){
```

```
                for (var j=0;j<sumall;j++){
                    if (obh.isSvaz(i,j)){
                        if
((obh.sendColor(i,j)=="red")||(obh.sendColor(i,j)=="blue")){//еслиполученынеу
дачнаясинхронизацияилипростопоиск
obh.getColor(i,j,"");//перекращиваем линию в цвет по умолчанию
                        }
obh.perehodPoisk(i,kolTime);//осуществляем переход поиска

                    }
                }
            }
vararr=dannie.split(' ');//получаем данные из журнала данных
while(kolTime==parseInt(arr[3])){//пока время в данных соответствует текущему
делаем операции
var tablico=upTablo.itemAt(parseInt(arr[2])).text.split('|');//считываем
данные из табло

upTablo.itemAt(parseInt(arr[2])).text=tablico[0]+'|'+arr[1]+'|'+tablico[2];//
записываем обновлённые данные в табло
obh.kol++;//ставим счётчик на следующие данные
obh.getColor(parseInt(arr[1]),parseInt(arr[2]),arr[0]);//меняем цвет в
соответствие с полученными данными
dannie=obh.readFile(obh.kol);//считываем следующие данные
arr=dannie.split(' ');
            }

            linii.clearline();
    }
}
Rectangle{//отображениесчётчикавремени
    id:schet
    x:standart
    y:parent.height-standart/2-10
    height: 20
    width: {
    return (17)*10}
    Text {
        anchors.centerIn: parent
        id: timeText
text: qsTr("Прошловремени: "+kolTime.toString())
    }
    border.color: "black"
    border.width: 3
}
Dial{//Изменение скоротечности времени
id:dial
x:0
    y:parent.height-standart
    width: standart
    height: standart
    value: 0.75
    stepSize: 0.1
    onValueChanged: {time=value*1000;}
}
Rectangle{//большая шкала в которую помещаются промежутки с условиями
id:aroundSh
    x:schet.x+schet.width+10
    y:schet.y
    width: window.width/2
    height:schet.height
    border.color: "black"
    border.width: 2
```

```
    }
    property int allTime:obh.maxConTime()
Repeater{
id:shkala
model:obh.kolCon();//сколько условий столько и делений
Rectangle{//разделяющие области
x:{
if (index==0)
return aroundSh.x+2;
                else
                    return shkala.itemAt(index-1).x+shkala.itemAt(index-
1).width;
            }
            y:aroundSh.y+2;
            width: {
                return (obh.timeCondition(index+1)-
obh.timeCondition(index))*secHod;}
            height: aroundSh.height-4
            border.color: "blue"
            Rectangle{
                x:1
                y:0
                width: {
                    var sum=0;
                    for (var i=0;i<=index;i++){
                        sum+=(obh.timeCondition(i+1)-obh.timeCondition(i));

                    }
                    if ((kolTime>=(sum-(obh.timeCondition(index+1)-
obh.timeCondition(index))))&&(kolTime<=sum)){
                        return secHod*(kolTime+(obh.timeCondition(index+1)-
obh.timeCondition(index))-sum);
                    } else if(sum<=kolTime)
                        return secHod*(obh.timeCondition(index+1)-
obh.timeCondition(index));
                }
                height: parent.height
                color: {
                    if (parseInt(qwer.numCondition(index))==10){
                        return "tomato";
                    }
                    else if (parseInt(qwer.numCondition(index))==20){
                        return "orangered";
                    }
                    else if (parseInt(qwer.numCondition(index))==30){
                        return "red";
                    }
                    else {
                        return "darkred";
                    }

                }
                Text {
                    text: qsTr(qwer.numCondition(index))
                }
            }
        }
    }
}
```

Приложение 5

Листинг 6.3 файлbaseout.cpp.

```
#include "baseout.h"
#include "textout.h"

Baseout::Baseout(QObject *parent)
{

}

QString Baseout::numCondition(int a)//кодусловиявзапрошенномместе
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("Data.db");
    if (db.open()){
    //Осуществляемзапрос
    QSqlQuery query;
    query.exec("SELECT * FROM conditions");
    int i=0;
    while (query.next()){
    QString numCon = query.value(0).toString();
if(i==a){//когда дошли до нужного номера возвращаем код
db.close();
        return numCon;
    }
    i++;
    }
    db.close();
    } else {
        qDebug()<<"Data.db is not open";
        return "-1";
    }
}

QString Baseout::timeBroken(int a)//Возвращаетвремяполомкиобъекта
{
    QSqlDatabase db;
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("Data.db");
    if (db.open()){
    //Осуществляемзапрос
    QSqlQuery query;
    query.exec("SELECT BrokenTime FROM Devices");
    int i=0;
    while (query.next()){
        QString timeBr=query.value(0).toString();
        if (a==i){
            db.close();
            return timeBr;}
        i++;
    }
    }else {
        qDebug()<<"Data.db is not open";
return "-1";
}

}
```