# Kursovaya Rabota

AUTHOR
Версия 1.0
Чт 26 Янв 2023

# Оглавление

Table of contents

# Иерархический список классов

## Иерархия классов

Иерархия классов.

# Алфавитный указатель классов

## Классы

Классы с их кратким описанием.

# Список файлов

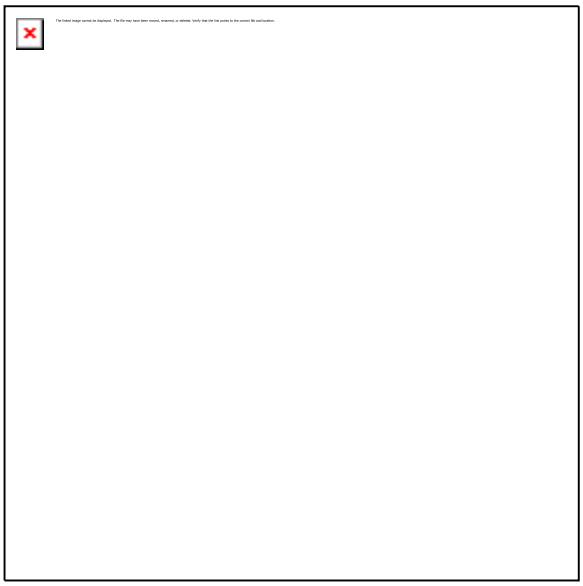
## Файлы

Полный список документированных файлов.

# Классы

## Класс Authors

Граф наследования:Authors:



### Открытые члены

- **Authors** (QWidget *parent=nullptr)

---

Объявления и описания членов классов находятся в файлах:

- autors.h
- autors.cpp

# Класс Edges

Класс ребра
`#include <graph_parser.h>`

## Открытые члены

- std::string **to_str** (const std::string &s)
  *функция преобразающая стоку в набор подряд идущих символов*

- std::pair< std::vector< **Edges** >, std::vector< **Vertices** > > **get_ver_edges** (const std::string &url)
  *функция получения списка ребер по файлу и списка вершин*

- std::vector< std::string > **split** (const std::string &s, const char &delimber)
  *функция разбивающая строку по разделителю*

- **Vertices get_vertices_from** ()
  *методы получения полей узла*

- **Vertices get_vertices_to** ()
  *методы получения полей узла*

- **Edges** (**Vertices** from, **Vertices** to)
  *конструктор ребра*

- **Edges** ()
  *конструктор по умолчвнию*

- **~Edges** ()=default
  *дуструктор ребра*

---

## Подробное описание

Класс ребра

---

## Методы

### std::pair< std::vector< Edges >, std::vector< Vertices > > Edges::get_ver_edges (const std::string & *url*)

функция получения списка ребер по файлу и списка вершин

функция получаения списка ребер по файлу

**Аргументы**

| | |
|---|---|
| *url* | сама ссылка на файл |

**Возвращает**

список ребер

### std::vector< std::string > Edges::split (const std::string & *s*, const char & *delimber*)

функция разбивающая строку по разделителю

**Аргументы**

| | |
|---|---|
| *s* | получаемая стока |
| *delimber* | разделитель |

**Возвращает**

список ребер

### std::string Edges::to_str (const std::string & *s*)

функция преобразающая стоку в набор подряд идущих символов

**Аргументы**

| | |
|---|---|
| *s* | строку которую нужно разбить |

**Возвращает**

стоку которую нужно было преобразовать

---

**Объявления и описания членов классов находятся в файлах:**

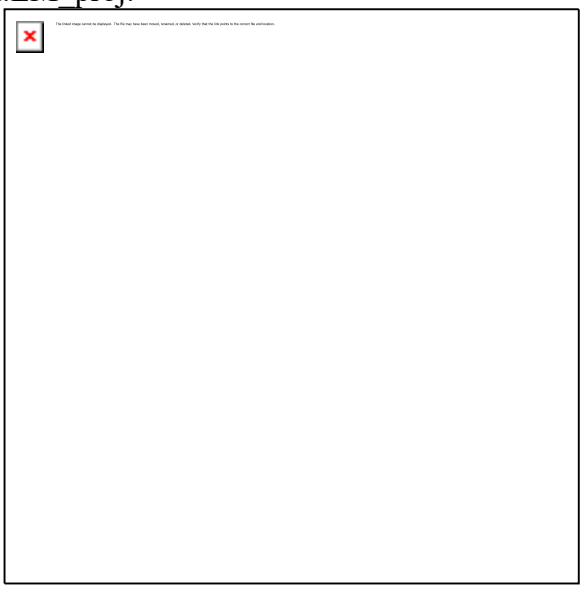
- graph_parser.h
- graph_parser.cpp

# Класс EM_proj

класс моего проекта унаследованный от исходного Widget

`#include <em_proj.h>`

Граф наследования:EM_proj:



## Открытые члены

- QStackedWidget * **getSteck** ()
  *функция получения стека*

- void **pushStack** (QWidget *wgt)
  *добавление в стек*

- void **popStack** ()
  *удаление из стека*

- **EM_proj** (QWidget *parent=nullptr)
  *конструктор проекта*

---

## Подробное описание

класс моего проекта унаследованный от исходного Widget

---

## Методы

### QStackedWidget * EM_proj::getSteck ()

функция получения стека

**Возвращает**

возвращает стек

### void EM_proj::pushStack (QWidget * *wgt*)

добавление в стек

**Аргументы**

| | |
|---|---|
| *wgt* | виджет для добавления |

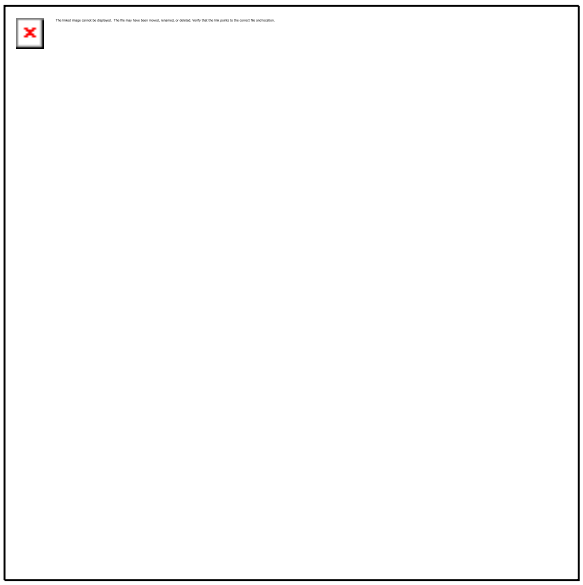
---

**Объявления и описания членов классов находятся в файлах:**

- em_proj.h
- em_proj.cpp

## Класс MainWindow

Граф наследования:MainWindow:



### Открытые члены

- **MainWindow** (QWidget *parent=nullptr)

---

Объявления и описания членов классов находятся в файлах:

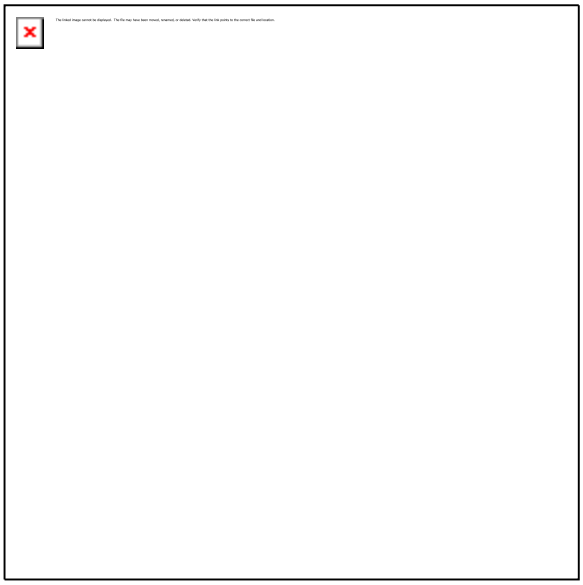
- mainwindow.h
- mainwindow.cpp

# Класс Menu

класс стартового меня унаследованного от QWidget

```
#include <menu.h>
```

Граф наследования:Menu:



## Открытые члены

- **Menu** (QWidget *parent=nullptr)
  *конструктор меню начального*

---

## Подробное описание

класс стартового меня унаследованного от QWidget

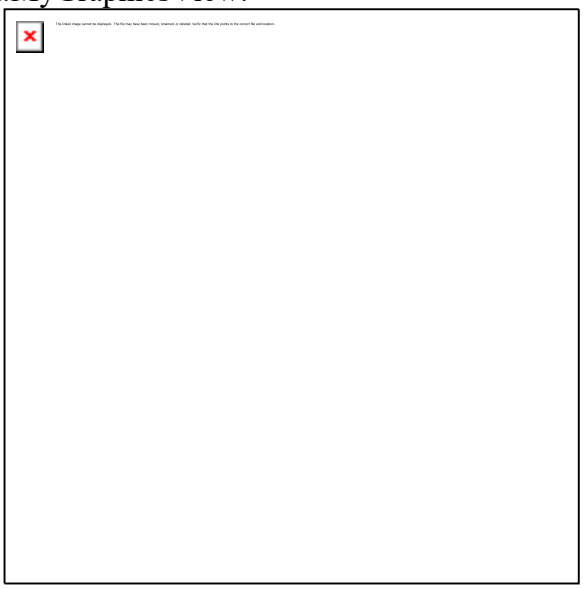
---

Объявления и описания членов классов находятся в файлах:

- menu.h
- menu.cpp

# Класс MyGraphicsView

класс моего отображения для графа унаследованного от QGraphicsView
`#include <mygraphicsview.h>`
Граф наследования:MyGraphicsView:



## Открытые члены

- **MyGraphicsView** (QGraphicsScene *scen, QWidget *parent=nullptr)
  *конструктор создания моего поля для отображения графа*

- void **wheelEvent** (QWheelEvent *event) override
  *переопределение базового метода прокрутки колеса мыши*

---

## Подробное описание

класс моего отображения для графа унаследованного от QGraphicsView

---

## Конструктор(ы)

### MyGraphicsView::MyGraphicsView (QGraphicsScene * *scen*, QWidget * *parent* = nullptr)[explicit]

конструктор создания моего поля для отображения графа

**Аргументы**

| | |
|---|---|
| *scen* | указатель на сцену на которую выставлены объекты для отображения |

---

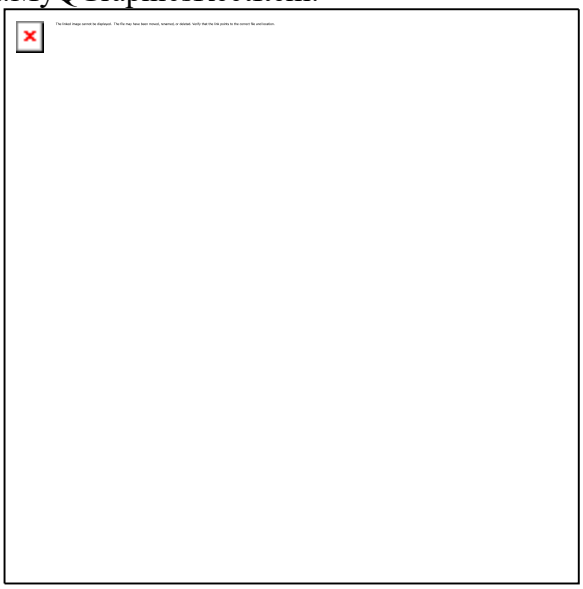
**Объявления и описания членов классов находятся в файлах:**

- mygraphicsview.h

- mygraphicsview.cpp

# Класс MyQGraphicsRectItem

класс моего отображения для узла графа унаследованного от базового прямоугольника
`#include <myqgraphicsrectiten.h>`
Граф наследования:MyQGraphicsRectItem:



## Открытые члены

- **MyQGraphicsRectItem** (const qreal &x, const qreal &y, const int &r)
  *констуктор*

- **~MyQGraphicsRectItem** ()=default
  *деструктор узла графа*

- QRectF **boundingRect** () const
  *выделение места на сцене под узел графа*

- void **paint** (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
  *функция отрисовки узла графа*

- void **dead** ()
  *функция которая логически убивает узел*

- void **sick** ()
  *функция которая логически инфицирует узел*

- void **recover** ()
  *функция которая логически выздоравлевает узел*

- void **infected** ()
  *функция которая заражает узел*

- void **unrecovered** ()

- bool **isSick** ()
  *функция проверяющая на болезнь*

- bool **isDead** ()
  *функция проверяющая на смерть узел*

- bool **isInfectd** ()
  *функция проверяющая на инфицированность*

- void **print_id** ()
- bool **isRecover** ()
  *функция проверяющая на вызжоровление*

- int **get_day** ()
  *функция получающая количнство дней узла*

- virtual void **advance** (int phase)

## Открытые атрибуты

- int **day_sick** =0
- int **day_recovered** = 0

---

## Подробное описание

класс моего отображения для узла графа унаследованного от базового прямоугольника

---

## Методы

### int MyQGraphicsRectItem::get_day ()

функция получающая количнство дней узла

**Возвращает**

количнство дней

### bool MyQGraphicsRectItem::isDead ()

функция проверяющая на смерть узел

**Возвращает**

результат проверки

### bool MyQGraphicsRectItem::isInfectd ()

функция проверяющая на инфицированность

**Возвращает**

результат проверки

**bool MyQGraphicsRectItem::isRecover ()**

функция проверяющая на вызжоровление

**Возвращает**

результат проверки

**bool MyQGraphicsRectItem::isSick ()**

функция проверяющая на болезнь

**Возвращает**

результат проверки

**void MyQGraphicsRectItem::paint (QPainter *  *painter*, const QStyleOptionGraphicsItem *  *option*, QWidget *  *widget*)**

функция отрисовки узла графа

**Аргументы**

| | |
|---|---|
| *painter* | стиль отрисовки узла |
| *option* | не используемый параметр |

**Возвращает**

widget родительский класс

---

**Объявления и описания членов классов находятся в файлах:**

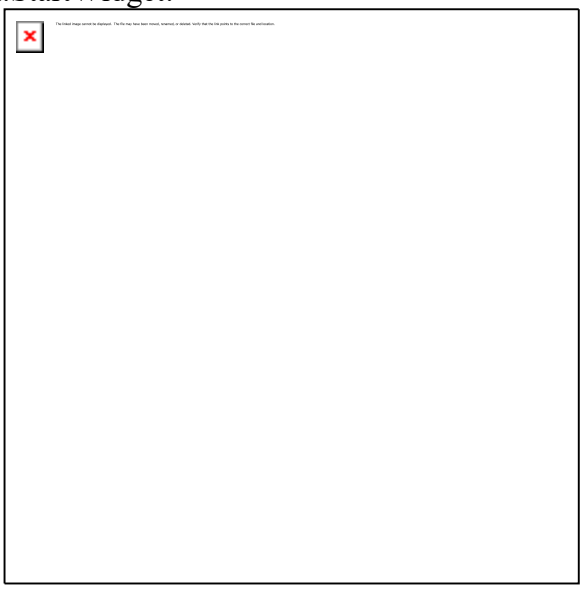
- myqgraphicsrectiten.h
- myqgraphicsrectiten.cpp

## Класс StartWidget

класс окна в котором настраивается симуляция

```
#include <startwidget.h>
```

Граф наследования:StartWidget:



### Открытые слоты

- void **slotButtonChoose** ()
  *слот кнопки выбора файла*

- void **onButtonRepeat** ()
  *слот кнопки повтора*

- void **onStepTimer** ()
  *слот кнопки шага таймера*

- void **onStartButton** ()
  *слот кнопки старта*

- void **onSliderdistribution** ()
  *слот прокрутки ползунков*

- void **onSliderMortalityRate** ()
- void **onSliderHealthEfficiency** ()
- void **onChanceSick** ()
  *слот шанса заболеть в такт таймера*

- void **ChangeTimer** (int)
  *слот начала таймера*

## Сигналы

- void **signalFromButton** (QString str)
  *сигнал отсылающийся при нажатии на кнопку*

## Открытые члены

- **StartWidget** (QWidget *parent=nullptr)
- void **sendButtonQuite** ()
- void **loadfile** (const std::string &url)
  *функция загрузки файла в программу (память)*

- void **parseTgf** (const std::string &url, std::vector< std::string > &edges_name, std::vector< std::pair< size_t, size_t > > &edges)
- void **addScengraph** (std::pair< std::vector< **Edges** >, std::vector< **Vertices** > > &mas)
  *функция отрисовывающая граф на сцене*

- std::vector< std::pair< int, int > > **findMinPointMaxPoint** (std::pair< std::vector< **Edges** >, std::vector< **Vertices** > > &mas)
  *нахождение крайних точек для графа, чтобы изменить рзамер сцены и поставить его по середине*

- QGraphicsLineItem * **CreateItamEdges** (const int &x1, const int &y1, const int &x2, const int &y2, const QPen &pen)
  *конструктор ребра*

- **MyQGraphicsRectItem** * **CreateMyItamVerties** (const int &x, const int &y, const qreal &r, const QPen &pen, const QBrush &brush)
  *конструктор узла*

- std::string **readFile** (const std::string &str)

---

## Подробное описание

класс окна в котором настраивается симуляция

---

## Методы

### void StartWidget::addScengraph (std::pair< std::vector< Edges >, std::vector< Vertices > > & *mas*)

функция отрисовывающая граф на сцене

#### Аргументы

| | |
|---|---|
| *mas* | список вершин и ребер |

### QGraphicsLineItem * StartWidget::CreateItamEdges (const int & *x1*, const int & *y1*, const int & *x2*, const int & *y2*, const QPen & *pen*)

конструктор ребра

**Аргументы**

| | |
|---|---|
| *x1* | координата x начала |
| *x2* | координата x конца |
| *y1* | координата y начала |
| *y2* | координата y конца |
| *pen* | стиль каисточки для отрисовки |

**Возвращает**

ребро

### MyQGraphicsRectItem * StartWidget::CreateMyItamVerties (const int & *x*, const int & *y*, const qreal & *r*, const QPen & *pen*, const QBrush & *brush*)

конструктор узла

**Аргументы**

| | |
|---|---|
| *x* | координата x центра узла |
| *y* | координата y центра узла |
| *r* | радиус узла |
| *brush* | кисть для заливки |
| *pen* | стиль каисточки для отрисовки |

**Возвращает**

ребро

### void StartWidget::loadfile (const std::string & *url*)

функция загрузки файла в программу (память)

**Аргументы**

| | |
|---|---|
| *url* | ссылка на логический граф |

---

**Объявления и описания членов классов находятся в файлах:**

- startwidget.h
- startwidget.cpp

# Класс Vertices

Класс вершины
`#include <graph_parser.h>`

## Открытые члены

- qreal **get_x** ()
  *методы получения значений вершин*

- qreal **get_y** ()
- std::string **get_m_name** ()
- int **get_id** ()
- **Vertices** ()
  *понтструкторы вершины*

- **Vertices** (const qreal &x, const qreal &y, const int &name, const int &id)
- **Vertices** (const qreal &x, const qreal &y, const std::string &m_name, const int &id)
- **Vertices** (const qreal &x, const qreal &y, const std::string &m_name, const int &id, const qreal &r)
- **Vertices** (const **Vertices** &other)
- **~Vertices** ()=default
  *деструктор вершины*

- **Vertices** & **operator=** (const **Vertices** &other)

---

## Подробное описание

Класс вершины

---

Объявления и описания членов классов находятся в файлах:

- graph_parser.h
- graph_parser.cpp

# Файлы

## autors.h

```
#ifndef AUTORS_H
#define AUTORS_H
#include <QPushButton>
#include <QMainWindow>
#include <QVBoxLayout>
#include <QComboBox>
#include <QLayout>
#include <QStackedWidget>
#include <QWidget>


class Authors : public QWidget
{
    Q_OBJECT
public:

    explicit Authors(QWidget *parent = nullptr);

private:

    QPushButton* buttonBack_a;

signals:

};

#endif // AUTORS_H
```

## em_proj.h

```
1 #ifndef EM PROJ H
2 #define EM PROJ H
3 //подключаемые библиотеки
4 #include <menu.h>
5 #include <startwidget.h>
6 #include <QFileDialog>
7 #include <autors.h>
8 #include <QtCore>
9 #include <QDebug>
10 #include <myqgraphicsrectiten.h>
11 #include <QPushButton>
12 #include <QMainWindow>
13 #include <QVBoxLayout>
14 #include <QComboBox>
15 #include <QLayout>
16 #include <QStackedWidget>
17
21 class EM proj : public QWidget
22 {
23     Q OBJECT
24 public:
29     QStackedWidget* getSteck();
30
35     void pushStack(QWidget* wgt);
39     void popStack();
43     explicit EM proj(QWidget *parent = nullptr);
44
45 private:
47     QStackedWidget* stack;
48     Menu* startmenu;
49     StartWidget* startWidget;
50     Authors* authorsWidget;
51
52 private slots:
53
55     void slotButtonStart();
56     void SlotButtonAutors();
57     void SlotButtonQuite();
58     void slotButtonBack();
59 };
60
61 #endif // EM_PROJ_H
```

## graph_parser.h

```
1 #ifndef GRAPH H
2 #define GRAPH H
3
4 #include <iostream>
5 #include <string>
6 #include <fstream>
7 #include <vector>
8 #include <sstream>
9 #include <QWidget>
10
14 //классы вершины и ребра
15 class Vertices
16 {
17
18 public:
20     qreal get x();
21     qreal get_y();
22     std::string get m name();
23     int get id();
24
26     Vertices();
27     Vertices(const qreal& x, const qreal& y,const int& name, const int& id);
28     Vertices(const qreal& x, const qreal& y,const std::string& m_name, const int& id);
29     Vertices(const qreal& x, const qreal& y,const std::string& m name, const int& id,
const qreal& r);
30     Vertices(const Vertices& other);
31
33     ~Vertices() = default;
34
35     Vertices& operator=(const Vertices& other);
36
37 private:
38     std::string m name="";
39     qreal x = 0;
40     qreal y = 0;
41     int id =0;
42     qreal r = 0;
43 };
44
48 class Edges
49 {
50 public:
51
57     std::string to str(const std::string& s);
58
60
65     std::pair<std::vector<Edges>,std::vector<Vertices>> get ver edges(const
std::string& url);
66
73     std::vector<std::string> split(const std::string& s, const char& delimber );
74
78     Vertices get_vertices_from();
82     Vertices get vertices to();
83
87     Edges(Vertices from, Vertices to);
91     Edges();
95     ~Edges() = default;
96
97 private:
99     Vertices from;
100     Vertices to;
101
102 };
103
104 #endif // GRAPH_H
```

## mainwindow.h

```
1 #ifndef MAINWINDOW H
2 #define MAINWINDOW H
3
4 #include <QMainWindow>
5 #include <QObject>
6 #include <QWidget>
7
8 class MainWindow : public QMainWindow
9 {
10     Q OBJECT
11 public:
12     explicit MainWindow(QWidget *parent = nullptr);
13
14 signals:
15
16 };
17
18 #endif // MAINWINDOW_H
```

## menu.h

```
1 #ifndef MENU H
2 #define MENU H
3 #include <QPushButton>
4 #include <QMainWindow>
5 #include <QVBoxLayout>
6 #include <QComboBox>
7 #include <QLayout>
8 #include <QStackedWidget>
9 #include <QWidget>
10 #include <QDebug>
14 class Menu : public QWidget
15 {
16     Q OBJECT
17
18 public:
19
23     explicit Menu(QWidget *parent = nullptr);
24
25 signals:
26
27
28 private:
30     QPushButton* start;
31     QPushButton* authors;
32     QPushButton* quit;
33 };
34
35 #endif // MENU_H
```

## mygraphicsview.h

```
1
2
3 #ifndef MYGRAPHICSVIEW H
4 #define MYGRAPHICSVIEW H
5
6 #include <QPushButton>
7 #include <QMainWindow>
8 #include <QVBoxLayout>
9 #include <QComboBox>
10 #include <QLayout>
11 #include <QStackedWidget>
12 #include <QGraphicsView>
13 #include <QWheelEvent>
14 #include <QGraphicsItem>
15
19 class MyGraphicsView : public QGraphicsView
20 {
21     Q OBJECT
22 public:
23
28     explicit MyGraphicsView(QGraphicsScene* scen, QWidget *parent = nullptr);
29
33     void wheelEvent(QWheelEvent* event) override;
34
35 signals:
36
37 };
38
39 #endif // MYGRAPHICSVIEW_H
```

## myqgraphicsrectiten.h

```
1 #ifndef MYQGRAPHICSRECTITEN H
2 #define MYQGRAPHICSRECTITEN H
3
4 #include <QPainter>
5 #include <QWidget>
6 #include <QGraphicsRectItem>
7 #include <QAbstractGraphicsShapeItem>
8 #include <QGraphicsItem>
9
13 class MyQGraphicsRectItem : public QGraphicsRectItem
14 {
15     //Q_OBJECT
16 public:
20     explicit MyQGraphicsRectItem(const qreal& x,const qreal& y,const int& r);
24     ~MyQGraphicsRectItem() = default;
28     QRectF boundingRect() const;
36     void paint(QPainter* painter,
37                const QStyleOptionGraphicsItem* option, QWidget* widget);
38
42     void dead();
46     void sick();
50     void recover();
54     void infected();
55
56     void unrecovered();
57
62     bool isSick();
67     bool isDead();
72     bool isInfectd();
73     void print id();
78     bool isRecover();
79
84     int get day();
85     int day_sick =0 ;
86     int day_recovered = 0;
87 public:
88     virtual void advance(int phase);
89
90 signals:
91
92
93 private:
95     bool m infected = false;
96     bool m dead = false;
97     bool m_recover = false;
98     bool m_sick = false;
99     qreal m x=0;
100     qreal m y=0;
101     int r =0;
102 };
103
104 #endif // MYQGRAPHICSRECTITEN_H
```

## startwidget.h

```
1 #ifndef STARTWIDGET H
2 #define STARTWIDGET H
3 #include <graph parser.h>
4 #include <myqgraphicsrectiten.h>
5 #include <QFileDialog>
6 #include <mygraphicsview.h>
7 #include <QPushButton>
8 #include <QMainWindow>
9 #include <QVBoxLayout>
10 #include <QComboBox>
11 #include <QLayout>
12 #include <QStackedWidget>
13 #include <QGraphicsView>
14 #include <QGraphicsEllipseItem>
15 #include <QTimer>
16 #include <nlohmann json/include/nlohmann/json.hpp>
17 #include <QLabel>
18 #include <QSpinBox>
19 #include <QCheckBox>
20 #include <QMenu>
21 #include <QComboBox>
22
23 //#include <matplot/matplot.h>
24 #include<gvc.h>
25
26
27
31 class StartWidget: public QWidget
32 {
33     Q OBJECT
34 public:
35
36     explicit StartWidget(QWidget *parent = nullptr);
37
38 public:
39
40     void sendButtonQuite();
41
46     void loadfile(const std::string& url);
47     void parseTgf(const std::string& url,std::vector<std::string>& edges_name,
48                   std::vector<std::pair<size t,size t>>& edges);
49
54     void addScengraph(std::pair<std::vector<Edges>,std::vector<Vertices>>& mas);
55
57     std::vector<std::pair<int,int>> findMinPointMaxPoint(
58               std::pair<std::vector<Edges>,std::vector<Vertices>>& mas);
59
60     //методы создания вершин и ребер
61     //QGraphicsRectItem* CreateItamRectverties(const int& x,const int& y,const int&
r,const QPen& pen, const QBrush& brush);
71     QGraphicsLineItem* CreateItamEdges(const int& x1,const int& y1,
72                                        const int& x2,const int& y2,
73                                        const QPen& pen);
83     MyQGraphicsRectItem* CreateMyItamVerties(const int& x,const int& y,const qreal& r,
84                                              const QPen& pen, const QBrush& brush);
85     std::string readFile(const std::string& str);
86 signals:
88     void signalFromButton(QString str);
89
90 public slots:
91
93     void slotButtonChoose();
95     void onButtonRepeat();
97     void onStepTimer();
99     void onStartButton();
101     void onSliderdistribution();
102     void onSliderMortalityRate();
103     void onSliderHealthEfficiency();
105     void onChanceSick();
107     void ChangeTimer(int);
108
109 private:
110
```

```
112     int countDead=0;
113     int daySimulation = 0;
114     QLabel* countDay = nullptr;
115     QComboBox* BoxStepQuarantine = nullptr;
116     QComboBox* Boxlayout=nullptr;
117     QSpinBox* BoxstepTimer = nullptr;
118     QCheckBox* quarantine = nullptr;
119     QSpinBox* sickDays= nullptr;
120     QSpinBox* dayToRecover= nullptr;
121     QSpinBox* quarantinePros= nullptr;
122     QSpinBox* dayToUnrecover= nullptr;
123     QSlider* chance_sick= nullptr;
124     QSlider* mortality rate= nullptr;
125     QSlider* health efficiency= nullptr;
126     QSlider* distribution c= nullptr;
127     std::vector<MyQGraphicsRectItem*> Vec Item;
128     //std::vector<QGraphicsLineItem*> Vec_Imem_edges;
129     std::vector<std::vector<MyQGraphicsRectItem*>> adjacency_list;
130     QTimer* stepTimer= nullptr;
131     QGraphicsScene* scen= nullptr;
132     MyGraphicsView* view= nullptr;
133     QPushButton* buttonRepeat= nullptr;
134     QPushButton* buttonBack= nullptr;
135     QPushButton* buttonStart= nullptr;
136     QPushButton* buttonChooseFile= nullptr;
137     QTimer* animatiomTimer= nullptr;
138     int m chance sick=0;
139     int m_sick=0;
140     int m_mortality_rate=0;
141     int m health efficiency=0;
142     int m distribution c=0;
143     std::string URL= "";
144
145 };
146
147 #endif // STARTWIDGET_H
```

# Алфавитный указатель

INDEX