МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСИС»

Институт информационных технологий и компьютерных наук
Кафедра инженерной кибернетики

Курсовая работа

по дисциплине «Технологии программирования»
на тему
«Приложение для моделирования и визуализации распространение вируса (графовая модель)»

Выполнил:
студент 2-го курса,
гр. БПМ-21-3 Попечителев М.К.

Проверил:
доцент, к.т.н. Полевой Д. В

Москва, 2023

Оглавление

Техничское задание

Нужно создать кросплатформенное приложение, позволяющее пользователю сгенерировать и пронаблюдать распространение вируса на абстрактном графе.

Требования к функциональности приложения:

1. Приложение должно иметь графический интерфейс пользователя
2. Приложение должно генерировать граф на основе предоставляемого ему файла
3. Приложение должно позволять настраивать вирус, для дальнейшей его симуляции.

Интерфейс

1. Кнопки “Начать” , “Повторить” , “Выбрать файл” , “Назад” для старта симуляции, повтора симуляции , выбора файла для генерации абстрактного графа, и возвращения в меню соответственно.
2. Ползунки “Коэффициент распространения” , ”Шанс заболеть” , ”Коэффициент летальности” отвечающие за соответственные параметры симуляции.
3. Поля ввода параметра для количества дней болезни, инкубационного периода, дней до потери иммунитета ,скорость шага симуляции в миллисекундах, а так же процент заражения общего числа людей перед вводом карантина.
4. Элементы выбора конкретного расположения графа в пространстве и варианты вводимых карантинов.
5. Элемент выделения “Карантин” который показывает проходит ли симуляция с карантином или нет.
6. Поле отображающее количество прошедших дней с начала симуляции
7. Графическая сцена, отображающая сам граф.

Установка программы

Требования для сборки и тестирования программы:

1. Cmake минимальной версии 3.5.

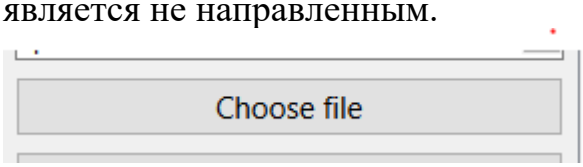
2. Язык C++ стандарт 17.
3. Qt минимальной версии 5.15.2.
4. Библиотека graphviz минимальной версии 4.0.0

Установка и запуск программы

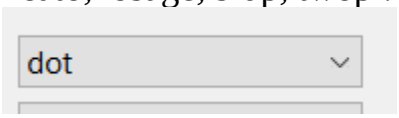
1. Выгрузите репозиторий по ссылке https://github.com/mikhailpopechitelev/Project_PI
2. Скачайте откомпилированную библиотеку с сайта https://graphviz.org/download/ в зависимости от вашей ОС в папку Project_PI\EM_project. При установке при возможности добавьте папку bin из библиотеки Graphviz в PATH.
3. Соберите и установите программу с помощью Cmake. По умолчанию программа устанавливается в директорию Project_PI\EM_project\install. Если хотите поменять папку поменяйте значение CMAKE_INSTALL_PREFIX на необходимую вам в CMakeLists.txt
4. Откройте папку, в которую была произведена установка и запустите EM_project.exe

Инструкция по использованию

1. Расположение абстрактного графа.
   Подготовьте файлы с прописанным Graphviz языком. Файл не должен содержать больше ничего лишнего, так же граф является не направленным.

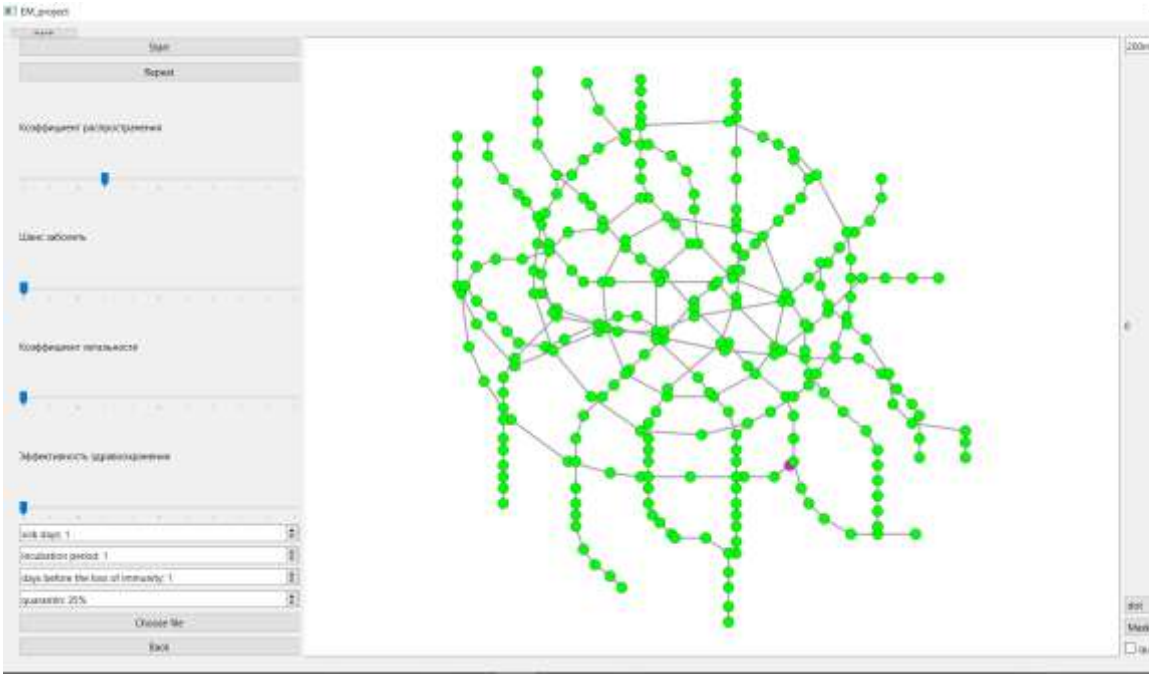


   К примеру файлы (txt или dot). Пример такого графа есть в репозитории в папке examples. Так же необходимо выбрать вариант расположения графа с помощью кнопки dot и выбрать один из возможных вариантов компоновки графа: dot, fdp, neato, osage, sfdp, twopi.



   Описание графической сцены:

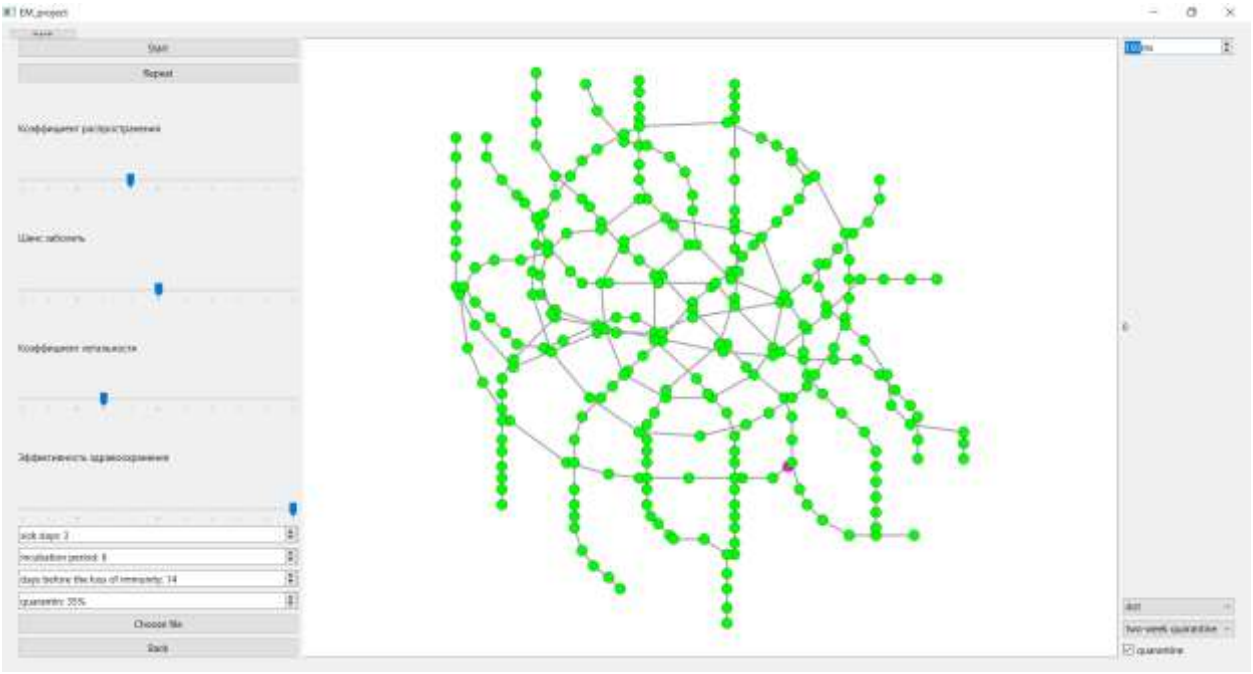
- Зеленые – здоровые узлы
- Розовые – инфицированные узлы
- Красные – больные узлы
- Синие – получившие иммунитет



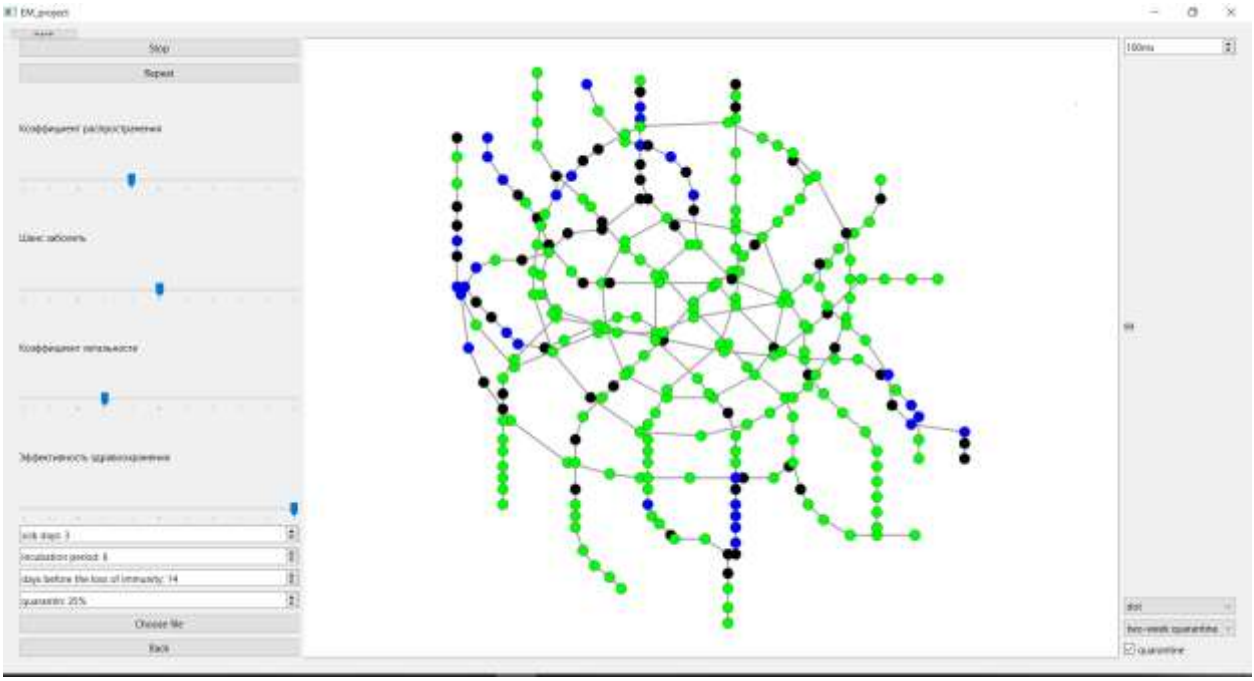
2. Настройка коэффициентов симуляции:

- Коэффициент распространения показывает какой шанс заразиться и стать переносчиком инфекции.
- Шанс заболеть (после того как человек стал переносчиком в конце инкубационного периода у него есть шанс либо получить иммунитет, либо заболеть)
- Коэффициент летальности определяет с каким шансом происходит смерть при окончании болезни
- Эффективность здравоохранения немного влияет на коэффициент летальности
- “Дни болезни” показывает сколько дней узел будет болеть
- “Инкубационный период” количество дней сколько узел будет переносчиком
- “Карантин:”- процентное значение, которое показывает со скольки процентов заражения узлов(от всей части) вводится карантин
- “dot” показывает возможные компановки графа
- “Масочный режим” показывает все возможные варианты карантина

- Поле “Карантин” показывает в каком варианте будет проходить симуляция (с карантином или без).
- “Начать” начинает симуляцию
- “Повторить” возобновляет симуляцию



3. Начало симуляции:
   Если граф расположен правильно и все параметры выставлены, то для начала симуляции достаточно нажать на “Начать”.



В правой части будет поле с цифрой, означающей количество пройденных итераций (дней).

Техническая документация

Модули

Модуль для разбора и вычисления математических выражений заданных строкой

Классы

class **RVar***клас хранит переменные для формул*

class **ROperation***класс обрабатывает строки с математическими выражениями*

class **RFunction***класс для работы с функциями*

Определения типов

- typedef **RVar** * **PRVar**
- typedef **ROperation** * **PROperation**
- typedef **RFunction** * **PRFunction**

Перечисления

- enum **ROperator** { **ErrOp**, **Juxt**, **Num**, **Var**, **Add**, **Sub**, **Opp**, **Mult**, **Div**, **Pow**, **Sqrt**, **NthRoot**, **Abs**, **Sin**, **Cos**, **Tg**, **Ln**, **Exp**, **Acos**, **Asin**, **Atan**, **E10**, **Fun** }
  перечисление - хранит возможные математические операции

Функции

- typedef **void** ((*pfoncld)(double *&))
- char * **MidStr** (const char *s, int i1, int i2)
  выделяет подстроку из строки в новую строку

- char * **CopyStr** (const char *s)
  копирование строка

- char * **InsStr** (const char *s, int n, char c)
  проверка находится ли символ в строке

- signed char **EqStr** (const char *s, const char *s2)
  проверка равенства строк

- signed char **CompStr** (const char *s, int n, const char *s2)
  сравнение части строки

- char * **DelStr** (const char *s, int n)
  удалегие части строки

Подробное описание

Модуль для разбора и вычисления математических выражений заданных строкой

Функции

signed char CompStr (const char * *s*, int *n*, const char * *s2*)

сравнение части строки

Аргументы

s	- строка
n	- подстрока
s2	- строка

Возвращает

signed char

char * CopyStr (const char * *s*)

копирование строка

Аргументы

s	- строка

Возвращает

char*

char * DelStr (const char * *s*, int *n*)

удалегие части строки

Аргументы

s	- строка
n	- подстрока

Возвращает

char*

signed char EqStr (const char * *s*, const char * *s2*)

проверка равенства строк

Аргументы

s	- строка
s2	- строка

Возвращает

signed char

char * InsStr (const char * *s*, int *n*, char *c*)

проверка находится ли символ в строке

Аргументы

s	- строка
n	- построка которая проверяется
c	- символ

Возвращает

char*

char * MidStr (const char * *s*, int *i1*, int *i2*)

выделяет подстроку из строки в новую строку

Аргументы

s	- строка
i1	- начало подстроки
i2	- конец подстроки

Возвращает

char*

Классы

Класс MainWindow

основное окно программы
`#include <mainwindow.h>`

Открытые члены

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

- **~MainWindow** ()
  деструктор

Подробное описание

основное окно программы

Объявления и описания членов классов находятся в файлах:

- D:/Books/Test/C++/GraphicOpengl/mainwindow.h
- D:/Books/Test/C++/GraphicOpengl/mainwindow.cpp

Класс myglwidget

класс отрисовывает график на форме Класс - наследник QOpenGLWidget и QOpenGLFunctions, позволяет работать с функциями OpenGL

`#include <myglwidget.h>`

Открытые слоты

- void **cleanup** ()

Открытые члены

- **myglwidget** (**Shape** shape, QWidget *parent=nullptr)
  конструктор

Защищенные члены

- void **initializeGL** () override
  создание контекста и инициализация шейдеров

- void **paintGL** () override
  вывод на экран

- void **resizeGL** (int w, int h) override
  корретировка вида при изменении окна

- void **mousePressEvent** (QMouseEvent *event) override
  обработчик нажатия кнопок мыши

- void **mouseMoveEvent** (QMouseEvent *event) override
  обработчик движения мыши

- void **keyPressEvent** (QKeyEvent *e) override
  обработчик нажатия кнопок клавиатуры

- void **wheelEvent** (QWheelEvent *event) override
  обработчик колесика мыши

Подробное описание

класс отрисовывает график на форме Класс - наследник QOpenGLWidget и QOpenGLFunctions, позволяет работать с функциями OpenGL

Конструктор(ы)

myglwidget::myglwidget (Shape *shape*, QWidget * *parent* = `nullptr`)

конструктор

Аргументы

shape	- объект класса **Shape**
parent	- родительский виджет

Объявления и описания членов классов находятся в файлах:

- D:/Books/Test/C++/GraphicOpengl/myglwidget.h
- D:/Books/Test/C++/GraphicOpengl/myglwidget.cpp

Класс RFunction

класс для работы с функциями
`#include <mathexpr.h>`

Открытые члены

- **double** ((*pfuncval)(double))
  указатель на функцию

- **RFunction** ()
  конструктор по умолчанию

- **RFunction** (**double**((*)(**double**)))
  конструктор

- **RFunction** (const **ROperation** &opp, **RVar** *pvarp)
  конструктор

- **RFunction** (const **ROperation** &opp, int nvarsp, **RVar** **ppvarp)
  конструктор

- **RFunction** (const **RFunction** &)
  копирующий конструктор

- **~RFunction** ()
  деструктор

- **RFunction** & **operator=** (const **RFunction** &)
- void **SetName** (const char *s)
  задает имя

- **double Val** (**double**) const
  задает значение

- **double Val** (**double** *) const
  задает массив значений

- **ROperation operator()** (const **ROperation** &)

Открытые атрибуты

- signed char **type**
  тип функциии

- **ROperation op**
- int **nvars**
- **RVar** ** **ppvar**
- char * **name**

Друзья

- int **operator==** (const **RFunction** &, const **RFunction** &)

Подробное описание

класс для работы с функциями

использует введенные выражения как отдельные функции

Конструктор(ы)

RFunction::RFunction (double((*)(double)))

конструктор

Аргументы

-	указатель на функцию

RFunction::RFunction (const ROperation & *opp*, RVar * *pvarp*)

конструктор

Аргументы

ROperation	- выражение
RVar	- переменная

RFunction::RFunction (const ROperation & *opp*, int *nvarsp*, RVar ** *ppvarp*)

конструктор

Аргументы

ROperation	- выражение
nvarsp	- количество переменных
RVar	- массив переменных

Методы

void RFunction::SetName (const char * s)

задает имя

Аргументы

s	- имя

double RFunction::Val (double * *pv*) const

задает массив значений

Аргументы

double	- массив значений

double RFunction::Val (double *x*) const

задает значение

Аргументы

double	- значение

Объявления и описания членов классов находятся в файлах:

- D:/Books/Test/C++/GraphicOpengl/mathexpr.h
- D:/Books/Test/C++/GraphicOpengl/mathexpr.cpp

Класс ROperation

класс обрабатывает строки с математическими выражениями
`#include <mathexpr.h>`

Открытые члены

- **ROperation** ()
  конструктор по умолчанию

- **ROperation** (const **ROperation** &)
  копирующий конструктор

- **ROperation** (double)
  конструктор

- **ROperation** (const **RVar** &)
  конструктор

- **ROperation** (char *sp, int nvarp=0, **PRVar** *ppvarp=NULL, int nfuncp=0, **PRFunction** *ppfuncp=NULL)
  конструктор

- **~ROperation** ()
  деструктор

- double **Val** () const
  созвращает вычислнное значение функции

- signed char **ContainVar** (const **RVar** &) const
  проверяет содержит ли выражение переменную

- signed char **ContainFunc** (const **RFunction** &) const
  проверяет содержит ли выражение функцию

- signed char **ContainFuncNoRec** (const **RFunction** &) const
  проверяет содержит ли выражение рекурсивную функцию

- **ROperation NthMember** (int) const
  возвращает указанную по счету операцию в выражении

- int **NMembers** () const
- signed char **HasError** (const **ROperation** *=NULL) const
  возвращает ошибку, если она есть

- **ROperation** & **operator=** (const **ROperation** &)
- **ROperation operator+** () const
- **ROperation operator-** () const
- **ROperation Diff** (const **RVar** &) const

дифференцирует по переменной

- char * **Expr** () const
  возвращает выражение после обработки

- **ROperation Substitute** (const **RVar** &, const **ROperation** &) const
  подставляет переменную строку с выражением

Открытые атрибуты

- **ROperator op**
- **PROperation mmb1**
- **PROperation mmb2**
- double **ValC**
- const **RVar** * **pvar**
- double * **pvarval**
- **RFunction** * **pfunc**

Друзья

- int **operator==** (const **ROperation** &, const double)
- int **operator==** (const **ROperation** &, const **ROperation** &)
- int **operator!=** (const **ROperation** &, const **ROperation** &)
- **ROperation operator,** (const **ROperation** &, const **ROperation** &)
- **ROperation operator+** (const **ROperation** &, const **ROperation** &)
- **ROperation operator-** (const **ROperation** &, const **ROperation** &)
- **ROperation operator*** (const **ROperation** &, const **ROperation** &)
- **ROperation operator/** (const **ROperation** &, const **ROperation** &)
- **ROperation operator^** (const **ROperation** &, const **ROperation** &)
- **ROperation sqrt** (const **ROperation** &)
- **ROperation abs** (const **ROperation** &)
- **ROperation sin** (const **ROperation** &)
- **ROperation cos** (const **ROperation** &)
- **ROperation tan** (const **ROperation** &)
- **ROperation log** (const **ROperation** &)
- **ROperation exp** (const **ROperation** &)
- **ROperation acos** (const **ROperation** &)
- **ROperation asin** (const **ROperation** &)
- **ROperation atan** (const **ROperation** &)
- **ROperation ApplyOperator** (int, **ROperation** **, **ROperation**(*)(const **ROperation** &, const **ROperation** &))

Подробное описание

класс обрабатывает строки с математическими выражениями

разбирает строки с математическими выражениями, подставляет значения в переменные и вычисляет значения

Конструктор(ы)

ROperation::ROperation (double *x*)

конструктор

Аргументы

double	- значение

ROperation::ROperation (const RVar & *varp*)

конструктор

Аргументы

RVar	- объект класса **RVar**

ROperation::ROperation (char * *sp*, int *nvarp* = `0`, PRVar * *ppvarp* = `NULL`, int *nfuncp* = `0`, PRFunction * *ppfuncp* = `NULL`)

конструктор

Аргументы

sp	- строка с выражением
nvarp	- количество переменных
sp	- массив переменных
nfuncp	- количество функций
ppfuncp	- массив функций

Методы

signed char ROperation::ContainFunc (const RFunction & *func*) const

проверяет содержит ли выражение функцию

Аргументы

RFunction	- функция

signed char ROperation::ContainFuncNoRec (const RFunction & *func*) const

проверяет содержит ли выражение рекурсивную функцию

Аргументы

RFunction	- функция

signed char ROperation::ContainVar (const RVar & *varp*) const

проверяет содержит ли выражение переменную

Аргументы

RVar	- переменная

ROperation ROperation::Diff (const RVar & *var*) const

дифференцирует по переменной

Аргументы

RVar	- переменная

Возвращает

ROperation

char * ROperation::Expr () const

возвращает выражение после обработки

Возвращает

char*

signed char ROperation::HasError (const ROperation * *pop* = `NULL`) const

возвращает ошибку, если она есть

Аргументы

ROperation	- объект выражения

Возвращает

signed char

ROperation ROperation::NthMember (int *n*) const

возвращает указанную по счету операцию в выражении

Аргументы

int	- номер

Возвращает

ROperation

ROperation ROperation::Substitute (const RVar & *var*, const ROperation & *rop*) const

подставляет переменную строку с выражением

Аргументы

RVar	- переменная
ROperation	- выражение

Возвращает

ROperation

Объявления и описания членов классов находятся в файлах:

- D:/Books/Test/C++/GraphicOpengl/mathexpr.h
- D:/Books/Test/C++/GraphicOpengl/mathexpr.cpp

Класс RVar

клас хранит переменные для формул

```
#include <mathexpr.h>
```

Открытые члены

- **RVar** ()
  конструктор

- **RVar** (const **RVar** &)
  копирующий конструктор

- **RVar** (const char *, double *)
  конструктор

- **~RVar** ()
  деструктор

Открытые атрибуты

- char * **name**
  название переменной

- double * **pval**
  указател на переменную

Друзья

- int **operator==** (const **RVar** &, const **RVar** &)
  оператор сравнения

Подробное описание

клас хранит переменные для формул

Конструктор(ы)

RVar::RVar () `[inline]`

конструктор

Аргументы

name	- имя переменной
pval	- переменная

RVar::RVar (const RVar & *rvarp*)

копирующий конструктор

Аргументы

RVar	- объект класса **RVar**

RVar::RVar (const char * *namep*, double * *pvalp*)

конструктор

Аргументы

*char**	- имя переменной
*double**	- переменная

Объявления и описания членов классов находятся в файлах:

- D:/Books/Test/C++/GraphicOpengl/mathexpr.h
- D:/Books/Test/C++/GraphicOpengl/mathexpr.cpp

Класс Shape

класс хранит список точек графика и возвращает их в виде массива Класс хранит координаты точек графика в виде QList отдельных чисел, позволяет получить данные в виде массива, полное количество чисел, количество координат точек которое хранится в массиве

`#include <shape.h>`

Открытые члены

- **Shape** ()
  конструктор по умолчанию

- **Shape** (QList< GLfloat > data)
  конструктор

- const GLfloat * **constData** () const
  возвращает указатель на константный массив GLfloat

- int **count** () const
  возвращает количество элементов в QList

- int **vertexCount** () const
  возвращает количество точек координаты которых хранятся в QList

---

Подробное описание

класс хранит список точек графика и возвращает их в виде массива Класс хранит координаты точек графика в виде QList отдельных чисел, позволяет получить данные в виде массива, полное количество чисел, количество координат точек которое хранится в массиве

Конструктор(ы)

Shape::Shape (QList< GLfloat > *data*)

конструктор

Аргументы

data	- QList координат точек графика

Методы

const GLfloat * Shape::constData () const `[inline]`

возвращает указатель на константный массив GLfloat

Возвращает

GLfloat*

int Shape::count () const [inline]

возвращает количество элементов в QList

Возвращает

int

int Shape::vertexCount () const [inline]

возвращает количество точек координаты которых хранятся в QList

Возвращает

int

Объявления и описания членов классов находятся в файлах:

- D:/Books/Test/C++/GraphicOpengl/shape.h
- D:/Books/Test/C++/GraphicOpengl/shape.cpp

Файлы

D:/Books/Test/C++/GraphicOpengl/mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QLabel>
#include <QGroupBox>
#include <QLineEdit>
#include <QPushButton>
#include <QMessageBox>
#include "myglwidget.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE
// основное окно программы
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
  void handleButton();

private:
    Ui::MainWindow *ui;
    QLineEdit *lineEdit;
    void createInputForm(QOpenGLWidget* wid);
};
#endif // MAINWINDOW_H
```

D:/Books/Test/C++/GraphicOpengl/mathexpr.h

```
1 #ifndef _MATHEXPR_H
2 #define _MATHEXPR_H
3
4 #include<string.h>
5 #include<stdio.h>
6 #include<stdlib.h>
7 #include<math.h>
8 #include<float.h>
9
10 #define atanl atan
11 #define asinl asin
12 #define acosl acos
13 #define expl exp
14 #define logl log
15 #define powl pow
16 #define pow10l(x) pow(10,x)
17 #define fabsl fabs
18 #define cosl cos
19 #define sinl sin
20 #define tanl tan
21 #define fmodl fmod
22 #define sqrtl sqrt
23
24
25 const double ErrVal=DBL_MAX;
26
33 class RVar{
34  public:
36   char*name;
38   double*pval;
44   RVar(){name=NULL;pval=NULL;};
49   RVar(const RVar&);
55   RVar(const char*,double*);
57   ~RVar();
59   friend int operator==(const RVar&,const RVar&);
60 };
61 typedef RVar* PRVar;
63 enum ROperator{ErrOp,Juxt,Num,Var,Add,Sub,Opp,Mult,Div,Pow,Sqrt,
64             NthRoot,Abs,Sin,Cos,Tg,Ln,Exp,Acos,Asin,Atan,E10,Fun};
65
66 typedef void ((*pfoncld)(double*&));
67
68 class ROperation;
69 typedef ROperation* PROperation;
70 class RFunction;
71 typedef RFunction* PRFunction;
72
78 class ROperation{
79   pfoncld*pinstr;double**pvals;double*ppile;RFunction**pfuncpile;
80   mutable signed char containfuncflag;
81   void BuildCode();
82     void Destroy();
83  public:
84   ROperator op;
85   PROperation mmb1,mmb2;
86   double ValC;
87   const RVar* pvar;
88   double*pvarval;
89   RFunction* pfunc;
93   ROperation();
97   ROperation(const ROperation&);
102    ROperation(double);
107    ROperation(const RVar&);
116    ROperation(char*sp,int nvarp=0,PRVar*ppvarp=NULL,int
nfuncp=0,PRFunction*ppfuncp=NULL);
118    ~ROperation();
120   double Val() const;
125   signed char ContainVar(const RVar&) const;
130   signed char ContainFunc(const RFunction&) const;
135   signed char ContainFuncNoRec(const RFunction&) const; // No recursive test on
subfunctions
141   ROperation NthMember(int) const;int NMembers() const;
147   signed char HasError(const ROperation* =NULL) const;
```

```
148   ROperation& operator=(const ROperation&);
149   friend int operator==(const ROperation& ,const double);
150   friend int operator==(const ROperation& ,const ROperation&);
151   friend int operator!=(const ROperation& ,const ROperation&);
152   ROperation operator+() const;ROperation operator-() const;
153   friend ROperation operator,(const ROperation&,const ROperation&);
154   friend ROperation operator+(const ROperation&,const ROperation&);
155   friend ROperation operator-(const ROperation&,const ROperation&);
156   friend ROperation operator*(const ROperation&,const ROperation&);
157   friend ROperation operator/(const ROperation&,const ROperation&);
158   friend ROperation operator^(const ROperation&,const ROperation&);  // Caution:
wrong associativity and precedence
159   friend ROperation sqrt(const ROperation&);
160   friend ROperation abs(const ROperation&);
161   friend ROperation sin(const ROperation&);
162   friend ROperation cos(const ROperation&);
163   friend ROperation tan(const ROperation&);
164   friend ROperation log(const ROperation&);
165   friend ROperation exp(const ROperation&);
166   friend ROperation acos(const ROperation&);
167   friend ROperation asin(const ROperation&);
168   friend ROperation atan(const ROperation&);
169   friend ROperation ApplyOperator(int,ROperation**,ROperation (*)(const
ROperation&,const ROperation&));
175   ROperation Diff(const RVar&) const;
180   char* Expr() const;
187   ROperation Substitute(const RVar&,const ROperation&) const;
188 };
189
195 class RFunction{
196   double*buf;
197 public:
199   signed char type;
201   double ((*pfuncval)(double));
202   ROperation op;
203   int nvars;
204   RVar** ppvar;
205   char*name;
209   RFunction();
214   RFunction(double ((*)(double)));
220   RFunction(const ROperation& opp,RVar* pvarp);
227   RFunction(const ROperation& opp,int nvarsp,RVar**ppvarp);
231   RFunction(const RFunction&);
233   ~RFunction();
234   RFunction& operator=(const RFunction&);
239   void SetName(const char*s);
244   double Val(double) const;
249   double Val(double*) const;
250   friend int operator==(const RFunction&,const RFunction&);
251   ROperation operator()(const ROperation&);
252 };
253
261 char* MidStr(const char*s,int i1,int i2);
267 char* CopyStr(const char*s);
275 char* InsStr(const char*s,int n,char c);
282 signed char EqStr(const char*s,const char*s2);
290 signed char CompStr(const char*s,int n,const char*s2);
297 char* DelStr(const char*s,int n);
298
301 #endif
```

D:/Books/Test/C++/GraphicOpengl/myglwidget.h

```
1 #ifndef MYGLWIDGET_H
2 #define MYGLWIDGET_H
3
4 #include <QOpenGLWidget>
5 #include <QOpenGLFunctions>
6 #include <QOpenGLVertexArrayObject>
7 #include <QOpenGLBuffer>
8 #include <QMatrix4x4>
9 #include <QVector3D>
10 #include <QtMath>
11 #include "shape.h"
12
13 QT_FORWARD_DECLARE_CLASS(QOpenGLShaderProgram)
14 // для работы с opengl создаем свой виджет на основе QOpenGLWidget и protected
QOpenGLFunctions
15
16
21 class myglwidget : public QOpenGLWidget, protected QOpenGLFunctions
22 {
23     Q_OBJECT
24 public:
25     // конструтор и деструктор
31     myglwidget( Shape shape, QWidget *parent = nullptr);
32     ~myglwidget();
33 // методы унаследованные от QOpenGLWidget
34 protected:
36     void initializeGL() override;
38     void paintGL() override;
40     void resizeGL(int w, int h) override;
42     void mousePressEvent(QMouseEvent *event) override;
44     void mouseMoveEvent(QMouseEvent *event) override;
46     void keyPressEvent(QKeyEvent *e) override;
48     void wheelEvent(QWheelEvent *event) override;
49 // функциия-слот вызывается при уничтожении виджета
50 public slots:
51     void cleanup();
52
53 private:
54     // объект класса с данными графика
55     Shape m_shape;
56     // Vertex Array Object (или VAO) - специальный тип объектов, который
инкапсулирует все данные,
57     // связанные с вершинным процессором. Вместо хранения текущих данных они
содержат ссылки
58     // на вершинный буфер, буфер индексов и указания для слоев самих вершин.
Преимущество в том, что
59     // единожды настроив VAO для меша вы можете привести внутренние состояния меша
просто привязав VAO.
60     QOpenGLVertexArrayObject m_vao;
61     // буфер для передачи данных графика в OpenGL
62     QOpenGLBuffer m_vbo;
63     // программа которая обрабатывает шейдер
64     QOpenGLShaderProgram *m_program = nullptr;
65     // матрица отвечает за перспективную проекцию
66     QMatrix4x4 m_projection;
67     // матрица отвечает за расположение модели
68     QMatrix4x4 m_model;
69     // буферы и списки точек для осей
70     QOpenGLBuffer m_xvbo;
71     QList<GLfloat> m_x_axis_data;
72     QOpenGLBuffer m_yvbo;
73     QList<GLfloat> m_y_axis_data;
74     QOpenGLBuffer m_zvbo;
75     QList<GLfloat> m_z_axis_data;
76     // работа с камерой, где находится. куда смотрит, где у нее верх
77     QVector3D cameraPos;
78     QVector3D cameraFront;
79     QVector3D cameraUp;
80     // переменные для доступа к переменным в шейдерах
81     int m_projectionMatrixLoc = 0;
82     int m_viewMatrixLoc = 0;
83     int m_modelMatrixLoc = 0;
84     int m_colorLoc = 0;
```

```
85      // размеры виджета
86      int width;
87      int height;
88      // переменные для работы с мышью
89      bool firstMouse = true;
90      float yaw   = -90.0f;
91      float pitch =  0.0f;
92      float lastX =  0.0;
93      float lastY =  0.0;
94      float fov   =  45.0f;
95      // шейдеры, программы для opengl, нужны для вывода на экран и окрашивания в
указанный цвет
96      const char *vertexShaderSource = "#version 330 core\n"
97          "layout (location = 0) in vec3 aPos;\n"
98          "layout (location = 1) in highp vec3 color;\n"
99          "uniform mat4 model;\n"
100          "uniform mat4 view;\n"
101          "uniform mat4 projection;\n"
102          "void main() {\n"
103          "   gl_Position = projection * view * model * vec4(aPos, 1.0f);\n"
104          "}\n";
105
106      const char *fragmentShaderSource = "#version 330 core\n"
107          "layout(location=0) out vec4 fragColor;\n"
108          "uniform vec3 color;\n"
109          "void main() {\n"
110          "   fragColor = vec4(color, 1.0);\n"
111          "}\n";
112  };
113
114  #endif // MYGLWIDGET_H
```

D:/Books/Test/C++/GraphicOpengl/shape.h

```
1 #ifndef SHAPE_H
2 #define SHAPE_H
3
4 #include <qopengl.h>
5 #include <QList>
6
13 class Shape
14 {
15 public:
16     // конструкторы
18     Shape(){}
23     Shape(QList<GLfloat> data);
24     // возвращает QList в виде простого массива
29     const GLfloat *constData() const { return m_data.constData(); }
30     // возвращает количество элементов
35     int count() const { return m_count; }
36     // для каждой точки нужны 3 координаты, поэтому точек в 3 раза меньше
41     int vertexCount() const { return m_count / 3; }
42 private:
43     QList<GLfloat> m_data;
44     int m_count = 0;
45 };
46
47 #endif // SHAPE_H
```