|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

«Программная имитация i-го прибора»

Студент ИУ7-68Б(В) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Бахолдин

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель темы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В. Рудаков

(Подпись, дата) (И.О.Фамилия)

*2020 г*

В лабораторной работе необходимо сделать: генератор, очередь и ОА. Закон генерации заявок выбирается равномерный (параметры настраиваются и варьируются). Закон в ОА из 1 лабораторной работы. Определить оптимальную длину очереди, т.е. ту длину, при которой ни одно сообщение необработанным не исчезает, т.е. нет отказа.

Рассмотрим имитатор одноканальной СМО с неограниченной очередью. Пусть на вход системы поступает случайный поток заявок, интервал времени *t* который является случайной величиной, распределенной по равномерному закону в интервале от *t1* до *t2* (∆*t=R*(*t1*, *t2*)). Время выполнения заявки в канале так же случайно и выбирается пользователем из равномерного, нормального, пуассона, экспоненционального распределений.

Выполним декомпозицию данной системы. В качестве отдельных эле-ментов можно выделить следующие:

1. Генератор заявок, характеризуется общим количеством сгенерированых заявок за весь период времени наблюдения. Законом распределения на основании которого происходит генерирование очередной заявки.

2. Обработчик заявок, характеризуется количеством обработаных заявок за весь период времени наблюдения. Законом распределения на основании которого происходит обработка очередной заявки.

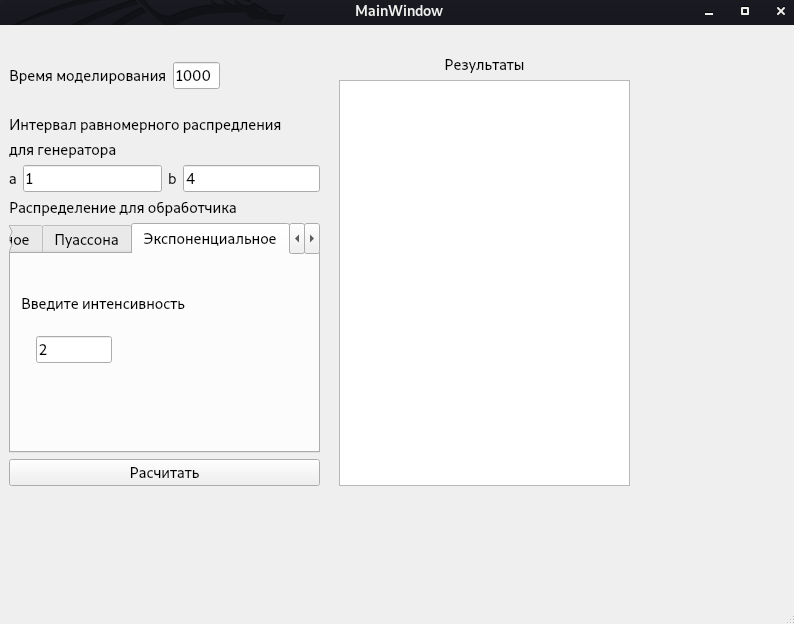
Программа состоит из 6 классов :

1. Структуры Task
2. Структуры Queue
3. Класса DistributionLaw
4. Класса Generator
5. Класса Processing
6. Класса Modeling

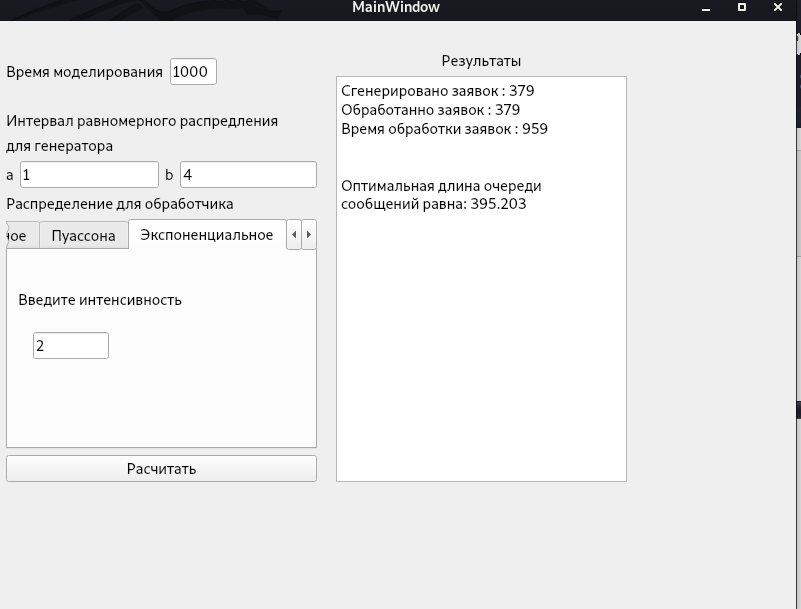
Структура Task хранит случайное число распределенное по равномерному закону распределения из выбранного пользователем интервала ( по смыслу это число соответствует времени появления заявки ). Структура Queue хранит в себе очередь созданых заявок. Класс DistributionLaw предназначен для выбора закона распределения. Класс Generator генерирует заявки. Класс Processing предназначен для обработки заявок. Класс Modeling моделирует работу СМО.

Для выполнения работы использовалась среда разработки Qt Creator 4.11.0 Based on Qt 5.12.5 ( GCC 9.2.1 20200104, 64 bit ), язык программирования Qt C++( стандарт языка С++17 ). Программа собрана под ОС Linux Debian 5.4.8-1kali1 ( 2020-01-06 ) x86\_64.

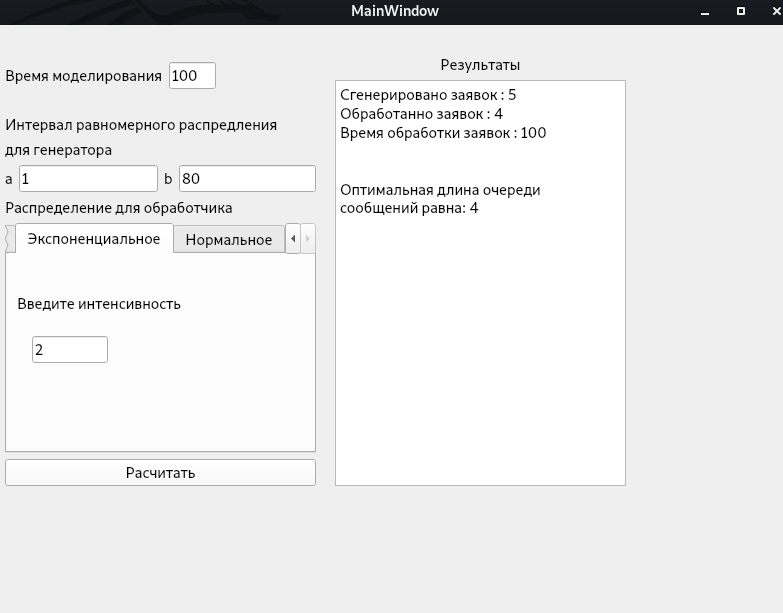
Пользователь вводить в программу время моделирования, интервал для генератора заявок( равномерное распределение), выбирает закон распределения(Равномерный, Пуассона, Экспоненциональный, Нормальный) для обработчика заявок вводит параметр для соответсвующего закона и нажимает кнопку «Расчитать».

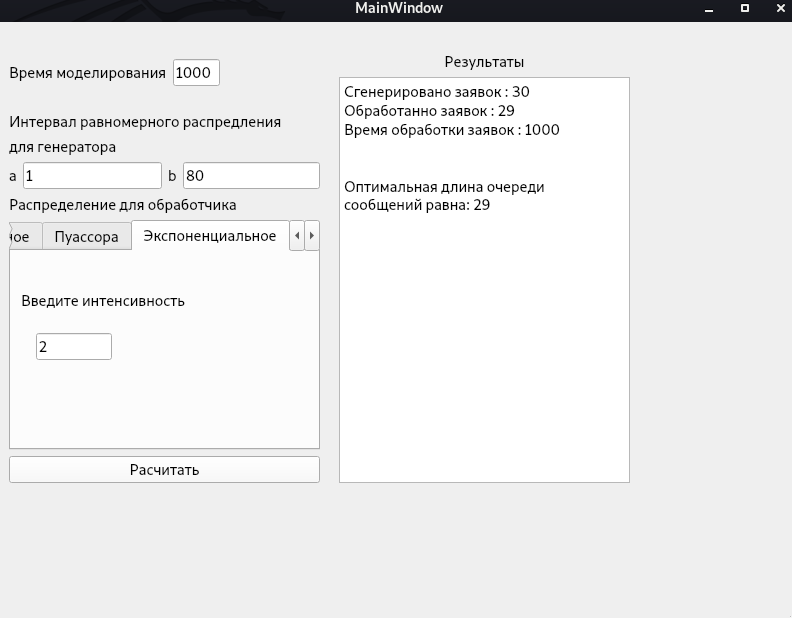


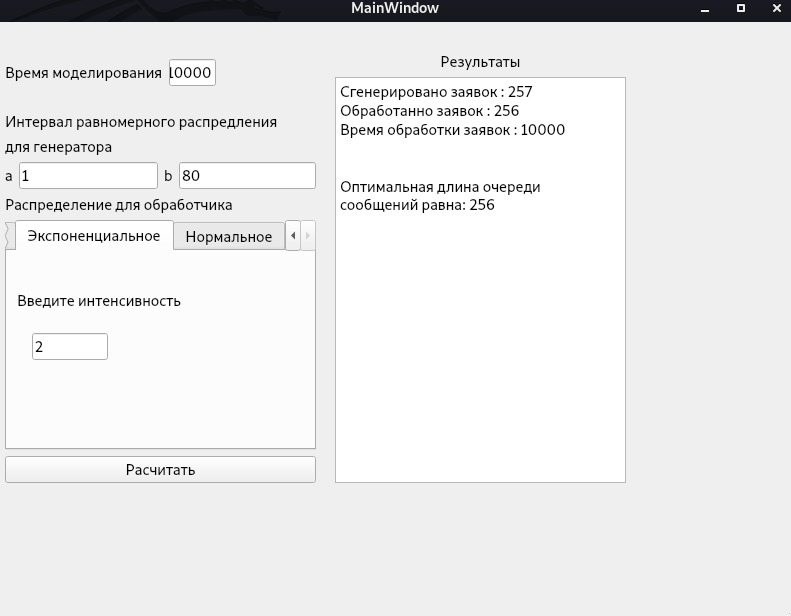
Получаем результаты: количество сгенерированых заявок, количесво обработанных заявок, время затраченное на обработку заявок.



В результате моделирования вычисляется оптимальная длина очереди сообщений, которая обрабатывалась бы полностью без отказов. Так как очередь по условию задачи неограниченна, то вероятность отказа обслуживания заявки равна 0, а значит полностью зависит от интенсивности обработки заявок обработчиком и зависит от времени моделирования СМО. В процессе обработки заявок интенсивность обработки постоянная и не меняется со временем, поэтому оптимальная длина очереди будет зависеть только от времени моделирования СМО ( по смыслу это означает успеет ли обработчик заявок, работающий с постояной интенсивностью, за время моделирования обработать все заявки или нет). Оптимальная длина вычисляется:







Листинг программы.

**model.h**

#ifndef MODEL\_H

#define MODEL\_H

#include<QString>

#include<QQueue>

#include<QTime>

#include<QMap>

#include<QDebug>

#include<algorithm>

#include<random>

*namespace* MDL {

*//--------------------------------------------------------------------------------------------------------//*

*struct* Task {

*private*:

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

double id\_task = 0 ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Task() = *delete* ;

Task( *const* Task &\_copy ) = *delete* ;

Task& *operator*=( *const* Task &task ) = *delete* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Task( double &id\_task ) : id\_task(id\_task) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

double Get\_id\_task() { *return* id\_task ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

} ;

*//--------------------------------------------------------------------------------------------------------//*

*//--------------------------------------------------------------------------------------------------------//*

*struct* Queue {

*private* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

QList< Task \* > queue ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

void Push( Task \*task ) { *this*->queue.append( task ) ; }

void Pop() { *this*->queue.removeFirst() ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

bool isEmpty() *const* { *return* *this*->queue.isEmpty() ? *true* : *false* ; }

size\_t GetSize() *const* { *return* *this*->queue.size() ; }

double Get\_head() *const* { *return* *this*->queue.first()->Get\_id\_task() ; }

double Get\_tail() *const* { *return* *this*->queue.front()->Get\_id\_task() ; }

void Print() { *for*( *const* *auto* &itm : *this*->queue ) qDebug() << itm->Get\_id\_task() << "\n" ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

} ;

*//--------------------------------------------------------------------------------------------------------//*

*//--------------------------------------------------------------------------------------------------------//*

*class* DistributionLaw {

*private*:

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*struct* AbsLaw {

*private* :

*const* double \_a = 0 ;

*const* double \_b = 0 ;

*const* int \_i = 0 ;

*public* :

double Get\_a() { *return* \_a ; }

double Get\_b() { *return* \_b ; }

int Get\_i() { *return* \_i ; }

AbsLaw( *const* double &a, *const* double &b ) : \_a(a), \_b(b) {}

AbsLaw( *const* int &i ) : \_i( i ) {}

*virtual* ~*AbsLaw*() {}

*virtual* double *GetRandom*() = 0 ;

};

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*struct* Uniform : *public* AbsLaw {

Uniform( *const* double &a, *const* double &b ) : AbsLaw( a, b ) {}

double *GetRandom*() *override* {

std::random\_device rd ;

std::mt19937 gen( rd() ) ;

std::uniform\_real\_distribution<> dis( Get\_a(), Get\_b() ) ;

*return* dis( gen ) ;

}

} ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*struct* Poisson : *public* AbsLaw {

Poisson( *const* int &l ) : AbsLaw( l ) {}

double *GetRandom*() *override* {

std::random\_device rd ;

std::mt19937 gen( rd() ) ;

std::poisson\_distribution<> dis( Get\_i() ) ;

*return* dis( gen ) ;

}

} ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*struct* Exponential : *public* AbsLaw {

Exponential( *const* int &l ): AbsLaw( l ) {}

double *GetRandom*() *override* {

std::random\_device rd ;

std::mt19937 gen( rd() ) ;

std::exponential\_distribution<> dis ( Get\_i() ) ;

*return* dis( gen ) ;

}

} ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*struct* Normal : *public* AbsLaw {

Normal( *const* double &m = 0, *const* double &s = 1 ) : AbsLaw( m, s ) {}

double *GetRandom*() *override* {

std::random\_device rd ;

std::mt19937 gen( rd() ) ;

std::normal\_distribution<> dis( Get\_a(), Get\_b() ) ;

*return* std::round( dis(gen) );

}

} ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*private* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

AbsLaw \*pLaw = *nullptr* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*enum* Law { *uniform* = 0, *poisson* = 1, *exponential* = 2, *normal* = 3 } law ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

DistributionLaw( DistributionLaw &\_copy ) = *delete* ;

DistributionLaw& *operator*=( *const* DistributionLaw &DL ) = *delete* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

DistributionLaw() {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

DistributionLaw( *const* Law &law, *const* int &i ) : law( law ) {

*if*( law == *poisson* ) pLaw = *new* Poisson( i ) ;

*else* *if*(law == *exponential*) pLaw = *new* Exponential( i ) ;

}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

DistributionLaw( *const* Law &law, *const* double &d1, *const* double &d2 ) : law( law ) {

*if*( law == *uniform* ) pLaw = *new* Uniform( d1, d2 ) ;

*else* *if*( law == *normal* ) pLaw = *new* Normal( d1, d2 ) ;

}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

~DistributionLaw() { *if*( pLaw != *nullptr* ) *delete* pLaw ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

double GetRandom() *const* { *return* pLaw->*GetRandom*() ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

} ;

*//--------------------------------------------------------------------------------------------------------//*

*//--------------------------------------------------------------------------------------------------------//*

*class* Generator {

*private* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

DistributionLaw \*DL = *nullptr* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Generator() = *delete* ;

Generator( *const* Generator &\_copy ) = *delete* ;

Generator& *operator*=( *const* Generator &generator ) = *delete* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Generator( *const* DistributionLaw::Law &l, *const* int &i )

: DL( *new* DistributionLaw( l , i ) ) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Generator( *const* DistributionLaw::Law &l, *const* double &d1, *const* double &d2 )

: DL( *new* DistributionLaw( l , d1, d2 ) ) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*virtual* ~*Generator*() { *if*( DL != *nullptr* ) *delete* DL ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

double Generate() *const* { *return* DL->GetRandom() ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

} ;

*//--------------------------------------------------------------------------------------------------------//*

*//--------------------------------------------------------------------------------------------------------//*

*class* Processing {

*private* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

DistributionLaw \*DL = *nullptr* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Processing() = *delete* ;

Processing( *const* Processing &\_copy ) = *delete* ;

Processing& *operator*=( *const* Processing &processing ) = *delete* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Processing( *const* DistributionLaw::Law &l, *const* int &i )

: DL( *new* DistributionLaw( l , i ) ) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Processing( *const* DistributionLaw::Law &l, *const* double &d1, *const* double &d2 )

: DL( *new* DistributionLaw( l , d1, d2 ) ) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*virtual* ~*Processing*() { *if*( DL != *nullptr* ) *delete* DL ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

double Generate() *const* { *return* DL->GetRandom() ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

} ;

*//--------------------------------------------------------------------------------------------------------//*

*//--------------------------------------------------------------------------------------------------------//*

*class* Modeling {

*private* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*struct* Statistics {

*private*:

QMap< QString , uint64\_t > stats ;

*public* :

void Add( *const* QString &s, *const* uint64\_t &i ) { stats[s] = i ; }

uint64\_t GetInfoWithKey( *const* QString &s ) { *return* stats[s] ; }

} STAT ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Generator \*GENERATOR = *nullptr* ;

Processing \*PROCESSING = *nullptr* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Statistics& GetStat() { *return* STAT ; }

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*public* :

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Modeling( *const* DistributionLaw::Law &l\_g, *const* double &d1\_g, *const* double &d2\_g,

*const* DistributionLaw::Law &l\_p, *const* int &i\_p )

: GENERATOR( *new* Generator ( l\_g, d1\_g, d2\_g ) ),

PROCESSING( *new* Processing( l\_p, i\_p ) ) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Modeling( *const* DistributionLaw::Law &l\_g, *const* int &i\_g,

*const* DistributionLaw::Law &l\_p, *const* int &i\_p )

: GENERATOR( *new* Generator ( l\_g, i\_g ) ),

PROCESSING( *new* Processing( l\_p, i\_p ) ) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Modeling( *const* DistributionLaw::Law &l\_g, *const* double &d1\_g, *const* double &d2\_g,

*const* DistributionLaw::Law &l\_p, *const* double &d1\_p, *const* double &d2\_p )

: GENERATOR( *new* Generator ( l\_g, d1\_g, d2\_g ) ),

PROCESSING( *new* Processing( l\_p, d1\_p, d2\_p ) ) {}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

Modeling() = *delete* ;

Modeling( *const* Modeling &\_copy ) = *delete* ;

*const* Modeling& *operator*=( *const* Modeling &modeling ) = *delete* ;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*virtual* ~*Modeling*() {

*if*( GENERATOR != *nullptr* ) *delete* GENERATOR ;

*if*( PROCESSING != *nullptr* ) *delete* PROCESSING ;

}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

void Run( *const* uint64\_t &\_max\_time ) {

Queue \*QUEUE = *new* Queue() ;

*const* double max\_time = *static\_cast*<double>(\_max\_time) ;

*const* double step\_time = 0.1 ;

*//* *моделируем* *процесс* *работы* *генератора* *заявок*

double time = 0 ;

*while*( time < max\_time ) {

time += step\_time ;

double res = GENERATOR->Generate() ;

*if*( res < time ) {

*if*( time < max\_time ) { *if*(time+res < max\_time) QUEUE->Push( *new* Task(*res*) ) ; time += res ; }

} *else* {

*while*( res > time ) { time += step\_time ; }

*if*( time < max\_time ) { *if*(time+res < max\_time) QUEUE->Push( *new* Task(*res*) ) ; time += res ; }

}

}

STAT.Add( "genereted\_tasks", QUEUE->GetSize() ) ;

*//* *моделируем* *процесс* *работы* *генератора* *заявок*

*//* *моделируем* *процесс* *работы* *обработчика* *заявок*

time = 0 ;

uint64\_t completed\_task = 0 ;

double time\_in\_queue = QUEUE->Get\_head() ;

double time\_in\_processing = 0 ;

*while*( time < max\_time && (!QUEUE->isEmpty()) ) {

time\_in\_processing += PROCESSING->Generate() ;

*if*( time > time\_in\_processing ) {

*if*( time\_in\_processing >= time\_in\_queue ) {

completed\_task++ ;

QUEUE->Pop() ;

*if*( !QUEUE->isEmpty() ) time\_in\_queue += QUEUE->Get\_head() ;

}

*else* *continue* ;

}*else* *if*( time < time\_in\_processing ) {

*while*( time < time\_in\_processing ) time += step\_time ;

*if*( time\_in\_processing >= time\_in\_queue ) {

completed\_task++ ;

QUEUE->Pop() ;

*if*( !QUEUE->isEmpty() ) time\_in\_queue += QUEUE->Get\_head() ;

}

*else* *continue* ;

}

*if*(time > time\_in\_processing) time += step\_time ;

}

STAT.Add( "completed\_task", completed\_task ) ;

STAT.Add( "time", time ) ;

*//* *моделируем* *процесс* *работы* *обработчика* *заявок*

*delete* QUEUE ;

}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//*

} ;

*//--------------------------------------------------------------------------------------------------------//*

}*//* *namespace* *MDL*

#endif *//* *MODEL\_H*

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

*namespace* Ui { *class* MainWindow; }

QT\_END\_NAMESPACE

*class* MainWindow : *public* QMainWindow

{

Q\_OBJECT

*public*:

MainWindow(QWidget \*parent = *nullptr*);

~*MainWindow*();

*private* slots:

void on\_pushButton\_clicked();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**mainwindow.cpp**

#include<QStringList>

#include<QPushButton>

#include<QDebug>

#include<QVariant>

#include<QMessageBox>

#include<sstream>

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "model.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent) , ui(*new* Ui::MainWindow) {

ui->setupUi(*this*);

}

MainWindow::~*MainWindow*() {

*delete* ui;

}

void MainWindow::on\_pushButton\_clicked() {

MDL::DistributionLaw \*DL\_g = *new* MDL::DistributionLaw() ;

MDL::DistributionLaw::Law law\_g = DL\_g->*uniform* ;

MDL::DistributionLaw \*DL\_p = *new* MDL::DistributionLaw() ;

MDL::DistributionLaw::Law law\_p ;

int index\_of\_tab = ui->tabWidget->currentIndex() ;

int time\_modeling = (ui->lineEdit\_time->text()).toInt() ;

double a\_g = (ui->lineEdit\_a->text()).toDouble() ;

double b\_g = (ui->lineEdit\_b->text()).toDouble() ;

double a\_p, b\_p, i\_p ;

bool isErr = *true* ;

*if*( a\_g < 0 || b\_g <= 0 ) QMessageBox::information( NULL, QObject::tr( "Информация" ),

tr("Введите интервал для генератора заявок") ) ;

*else* {

*switch*( index\_of\_tab ) {

*case* 0 :

a\_p = (ui->lineEdit\_a\_u->text()).toDouble() ;

b\_p = (ui->lineEdit\_b\_u->text()).toDouble() ;

*if*( a\_p < 0 || b\_p <= 0 ) QMessageBox::information( NULL, QObject::tr( "Информация" ),

tr("Введите интервал равномерного распределения для обработчика заявок") ) ;

*else* { law\_p = DL\_p->*uniform* ; isErr = *false* ; }

*break* ;

*case* 1 :

i\_p = (ui->lineEdit\_i\_p->text()).toDouble() ;

*if*( i\_p == 0 ) QMessageBox::information( NULL, QObject::tr( "Информация" ),

tr("Введите Мат.ожидание для обработчика заявок") ) ;

*else* { law\_p = DL\_p->*poisson* ; isErr = *false* ; }

*break* ;

*case* 2 :

i\_p = (ui->lineEdit\_i\_e->text()).toDouble() ;

*if*( i\_p == 0 ) QMessageBox::information( NULL, QObject::tr( "Информация" ),

tr("Введите Мат.ожидание для обработчика заявок") ) ;

*else* { law\_p = DL\_p->*exponential* ; isErr = *false* ; }

*break* ;

*case* 3 :

a\_p = (ui->lineEdit\_a\_n->text()).toDouble() ;

b\_p = (ui->lineEdit\_b\_n->text()).toDouble() ;

*if*( a\_p < 0 || b\_p < 0) QMessageBox::information( NULL, QObject::tr( "Информация" ),

tr("Введите Мат.ожидание и дисперсию для обработчика заявок") ) ;

*else* { law\_p = DL\_p->*normal* ; isErr = *false* ; }

*break* ;

}

*if*( isErr ) QMessageBox::information( NULL, QObject::tr( "Информация" ),

tr("Вы ввели неверные параметры!") ) ;

*else* {

MDL::Modeling \*MODEL = *nullptr* ;

*if*( law\_p == DL\_p->*uniform* || law\_p == DL\_p->*normal* ) {

MODEL = *new* MDL::Modeling( law\_g, a\_g, b\_g, law\_p, a\_p, b\_p ) ;

}

*else* *if*( law\_p == DL\_p->*poisson* || law\_p == DL\_p->*exponential* ) {

MODEL = *new* MDL::Modeling( law\_g, a\_g, b\_g, law\_p, i\_p ) ;

}

MODEL->Run( time\_modeling ) ;

std::stringstream ss;

QString str = "genereted\_tasks" ;

ss << "Сгенерировано заявок : " << MODEL->GetStat().GetInfoWithKey(str) << "\n" ;

QString str2 = "completed\_task" ;

ss << "Обработанно заявок : " << MODEL->GetStat().GetInfoWithKey(str2) << "\n" ;

QString str3 = "time" ;

ss << "Время обработки заявок : " << MODEL->GetStat().GetInfoWithKey(str3) << "\n\n\n" ;

ss << "Оптимальная длина очереди сообщений равна: " <<

*static\_cast*<double>(MODEL->GetStat().GetInfoWithKey(str2) \* time\_modeling ) /

*static\_cast*<double>(MODEL->GetStat().GetInfoWithKey(str3)) ;

QString s = QString::fromStdString(ss.str());

ui->textBrowser->setText( s ) ;

*delete* MODEL ;

}

}

*delete* DL\_g, DL\_p ;

}

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}

**mainwindow.ui**

*<?xml version="1.0" encoding="UTF-8"?>*

*<ui version="4.0">*

*<class>MainWindow</class>*

*<widget class="QMainWindow" name="MainWindow">*

*<property name="geometry">*

*<rect>*

*<x>0</x>*

*<y>0</y>*

*<width>800</width>*

*<height>600</height>*

*</rect>*

*</property>*

*<property name="windowTitle">*

*<string>MainWindow</string>*

*</property>*

*<widget class="QWidget" name="centralwidget">*

*<widget class="QWidget" name="horizontalLayoutWidget">*

*<property name="geometry">*

*<rect>*

*<x>10</x>*

*<y>30</y>*

*<width>211</width>*

*<height>41</height>*

*</rect>*

*</property>*

*<layout class="QHBoxLayout" name="horizontalLayout\_23">*

*<item>*

*<widget class="QLabel" name="label\_23">*

*<property name="text">*

*<string>Время моделирования</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_time"/>*

*</item>*

*</layout>*

*</widget>*

*<widget class="QWidget" name="verticalLayoutWidget\_3">*

*<property name="geometry">*

*<rect>*

*<x>340</x>*

*<y>30</y>*

*<width>291</width>*

*<height>431</height>*

*</rect>*

*</property>*

*<layout class="QVBoxLayout" name="verticalLayout\_5">*

*<item>*

*<widget class="QLabel" name="label\_24">*

*<property name="text">*

*<string>Результаты</string>*

*</property>*

*<property name="alignment">*

*<set>Qt::AlignCenter</set>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QTextBrowser" name="textBrowser"/>*

*</item>*

*</layout>*

*</widget>*

*<widget class="QWidget" name="layoutWidget">*

*<property name="geometry">*

*<rect>*

*<x>10</x>*

*<y>90</y>*

*<width>311</width>*

*<height>371</height>*

*</rect>*

*</property>*

*<layout class="QVBoxLayout" name="verticalLayout\_3">*

*<item>*

*<layout class="QVBoxLayout" name="verticalLayout">*

*<item>*

*<widget class="QLabel" name="label\_3">*

*<property name="text">*

*<string>Интервал равномерного распредления</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLabel" name="label\_5">*

*<property name="text">*

*<string>для генератора</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_3">*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_2">*

*<item>*

*<widget class="QLabel" name="label\_2">*

*<property name="text">*

*<string>a</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_a"/>*

*</item>*

*</layout>*

*</item>*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout">*

*<item>*

*<widget class="QLabel" name="label">*

*<property name="text">*

*<string>b</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_b"/>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</item>*

*<item>*

*<widget class="QLabel" name="label\_4">*

*<property name="text">*

*<string>Распределение для обработчика</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<layout class="QVBoxLayout" name="verticalLayout\_2">*

*<item>*

*<widget class="QTabWidget" name="tabWidget">*

*<property name="currentIndex">*

*<number>1</number>*

*</property>*

*<widget class="QWidget" name="tab\_1">*

*<attribute name="title">*

*<string>Равномерное</string>*

*</attribute>*

*<widget class="QWidget" name="layoutWidget">*

*<property name="geometry">*

*<rect>*

*<x>0</x>*

*<y>60</y>*

*<width>251</width>*

*<height>31</height>*

*</rect>*

*</property>*

*<layout class="QHBoxLayout" name="horizontalLayout\_4">*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_5">*

*<item>*

*<widget class="QLabel" name="label\_6">*

*<property name="text">*

*<string>a</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_a\_u"/>*

*</item>*

*</layout>*

*</item>*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_6">*

*<item>*

*<widget class="QLabel" name="label\_7">*

*<property name="text">*

*<string>b</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_b\_u"/>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</widget>*

*<widget class="QLabel" name="label\_8">*

*<property name="geometry">*

*<rect>*

*<x>0</x>*

*<y>20</y>*

*<width>271</width>*

*<height>19</height>*

*</rect>*

*</property>*

*<property name="font">*

*<font>*

*<pointsize>10</pointsize>*

*</font>*

*</property>*

*<property name="text">*

*<string>Интервал равномерно распределения</string>*

*</property>*

*</widget>*

*</widget>*

*<widget class="QWidget" name="tab\_2">*

*<attribute name="title">*

*<string>Пуассона</string>*

*</attribute>*

*<widget class="QWidget" name="layoutWidget\_3">*

*<property name="geometry">*

*<rect>*

*<x>10</x>*

*<y>80</y>*

*<width>91</width>*

*<height>31</height>*

*</rect>*

*</property>*

*<layout class="QHBoxLayout" name="horizontalLayout\_19">*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_20">*

*<item>*

*<widget class="QLabel" name="label\_17">*

*<property name="text">*

*<string/>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_i\_p"/>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</widget>*

*<widget class="QLabel" name="label\_19">*

*<property name="geometry">*

*<rect>*

*<x>10</x>*

*<y>40</y>*

*<width>181</width>*

*<height>19</height>*

*</rect>*

*</property>*

*<property name="text">*

*<string>Введите интенсивность</string>*

*</property>*

*</widget>*

*</widget>*

*<widget class="QWidget" name="tab\_3">*

*<attribute name="title">*

*<string>Экспоненциальное</string>*

*</attribute>*

*<widget class="QWidget" name="layoutWidget\_4">*

*<property name="geometry">*

*<rect>*

*<x>10</x>*

*<y>80</y>*

*<width>91</width>*

*<height>31</height>*

*</rect>*

*</property>*

*<layout class="QHBoxLayout" name="horizontalLayout\_21">*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_22">*

*<item>*

*<widget class="QLabel" name="label\_18">*

*<property name="text">*

*<string/>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_i\_e"/>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</widget>*

*<widget class="QLabel" name="label\_20">*

*<property name="geometry">*

*<rect>*

*<x>10</x>*

*<y>40</y>*

*<width>181</width>*

*<height>19</height>*

*</rect>*

*</property>*

*<property name="text">*

*<string>Введите интенсивность</string>*

*</property>*

*</widget>*

*</widget>*

*<widget class="QWidget" name="tab\_4">*

*<attribute name="title">*

*<string>Нормальное</string>*

*</attribute>*

*<widget class="QWidget" name="layoutWidget">*

*<property name="geometry">*

*<rect>*

*<x>0</x>*

*<y>10</y>*

*<width>294</width>*

*<height>83</height>*

*</rect>*

*</property>*

*<layout class="QVBoxLayout" name="verticalLayout\_4">*

*<item>*

*<widget class="QLabel" name="label\_21">*

*<property name="text">*

*<string>Введите Мат. ожидание</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLabel" name="label\_22">*

*<property name="text">*

*<string>Введите Дисперсию</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_16">*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_17">*

*<item>*

*<widget class="QLabel" name="label\_15">*

*<property name="text">*

*<string>E</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_a\_n"/>*

*</item>*

*</layout>*

*</item>*

*<item>*

*<layout class="QHBoxLayout" name="horizontalLayout\_18">*

*<item>*

*<widget class="QLabel" name="label\_16">*

*<property name="text">*

*<string>S</string>*

*</property>*

*</widget>*

*</item>*

*<item>*

*<widget class="QLineEdit" name="lineEdit\_b\_n"/>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</widget>*

*</widget>*

*</widget>*

*</item>*

*<item>*

*<widget class="QPushButton" name="pushButton">*

*<property name="text">*

*<string>Расчитать</string>*

*</property>*

*</widget>*

*</item>*

*</layout>*

*</item>*

*</layout>*

*</widget>*

*</widget>*

*<widget class="QMenuBar" name="menubar">*

*<property name="geometry">*

*<rect>*

*<x>0</x>*

*<y>0</y>*

*<width>800</width>*

*<height>24</height>*

*</rect>*

*</property>*

*</widget>*

*<widget class="QStatusBar" name="statusbar"/>*

*</widget>*

*<resources/>*

*<connections/>*

*</ui>*