САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ

Кафедра Информационных и управляющих систем

|  |  |
| --- | --- |
| **К У Р С О В О Й П Р О Е К Т** | |
| МОДЕЛИРОВАНИЕ СИСТЕМЫ, ФОРМАЛИЗОВАННОЙ КАК СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ | |
| по дисциплине «Алгоритмизация и основы программирования» | |
| Выполнил |  |
| студент гр.13534/6 | Исаев В.С. |
| Руководитель |  |
| доцент, к.т.н. | И. А. Веренинов |
|  | « » мая 2018 г. |

Санкт – Петербург 2016

### ЗАДАНИЕ

### НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

### студенту группы 13534/6 Исаев В.С.

#### (номер группы)

#### (фамилия, имя, отчество)

## Тема проекта (работы)

#### Разработка программной системы моделирования дискретной стохастической системы, формализованной как одноканальная система массового обслуживания с многомерным входным потоком и памятью заявок с применением технологии ООП

#### 29 мая

## Срок сдачи студентом законченного проекта (работы) 2018года

1. ***Исходные данные к проекту (работе)***  *Перечень индивидуальных заданий на проект с указанием конкретного номера задания Методическое пособие по курсовому проектированию*

***4.Содержание пояснительной записки*** (перечень подлежащих разработке вопросов):

### введение, основная часть (раскрывается структура основной части), заключение, список использованных источников, приложения)*.*

## Введение с формулировкой конкретного задания

## Инструкция системного программиста с указанием состава, назначения модулей, особенностей размещения драйверов и т.д.

## Инструкция по запуску, особенностей интерфейса, выводу результатов Примеры результатов в графической и табличной форме

## Полные исходные тексты всех модулей с комментариями полей данных и методов

***типов объектов*** Примерный объем пояснительной записки ***21*** страница машинописного текста

### Перечень графического материала ( с указанием обязательных чертежей и

### плакатов) Скриншоты

### Консультанты Нет

### Дата получения задания: «10» февраля 2016 г.

|  |  |  |
| --- | --- | --- |
| Руководитель | *(подпись)* | *Веренинов И. А. (инициалы, фамилия)* |
| Задание принял к исполнению | *(подпись)* | *Исаев В.С. (инициалы, фамилия)* |

#### (дата)

**СОДЕРЖАНИЕ**

[Введение 4](#_bookmark0)

1. [Инструкция программиста 5](#_bookmark1)
2. [Инструкция оператора 6](#_bookmark2)

[Список использованных источников 8](#_bookmark3)

[Приложение 2. Исходный код головной программы GOLOVN.pas 8](#_bookmark4)

[Приложение 3. Исходный код функционального модуля FUNCMDLE.tpu 9](#_bookmark5)

[Приложение 4. Исходный код графического модуля GRPH.tpu 12](#_bookmark6)

[Приложение 5. Модуль интерфейса INTFMDL.tpu 15](#_bookmark7)

# Введение

### Задача состоит в разработке программной системы моделирования,

### формализованной как система массового обслуживания (СМО) с двумя входными потоками, с применением технологии ООП.

**Вариант 7**

##### ДАНО :

##### Количество источников - 3

Потоки заявок - простейшие с 2=2, 3 =2 1=3

##### Объем буфера - 3

##### Дисциплина записи в буфер - в порядке поступления Дисциплина выборки из буфера - без приоритета

##### Поток обслуживания - простейший с \* =2    \*\* =3 с  =0.1 Точность моделирования - =20%, Р=0.90

4 4 4 4

##### НАЙТИ :

Вероятность отказа Ротказа i = f 1 i () i=1,2,3

##### Среднее время пребывания в системе М ( Тпреб i ) = f 2 i ( 4 ) i=1,2,3

* 1. **Инструкция программиста**

### Данный программный продукт состоит из 4 файлов: funcmdle.tpu, intfmdl.tpu, grph.tpu, golovn.pas, helpme.txt.

### golovn.pas – головная программа, запускает интерфейсный модуль.

### funcmdle.tpu – функциональный модуль, осуществляет все расчеты моделирования СМО. Модуль содержит описание объектов machine, generator, buffer. Блок объектов machine, generator, buffer прогоняет 11 реализаций с предоставленными параметрами и выводит результаты в виде графиков и таблиц.

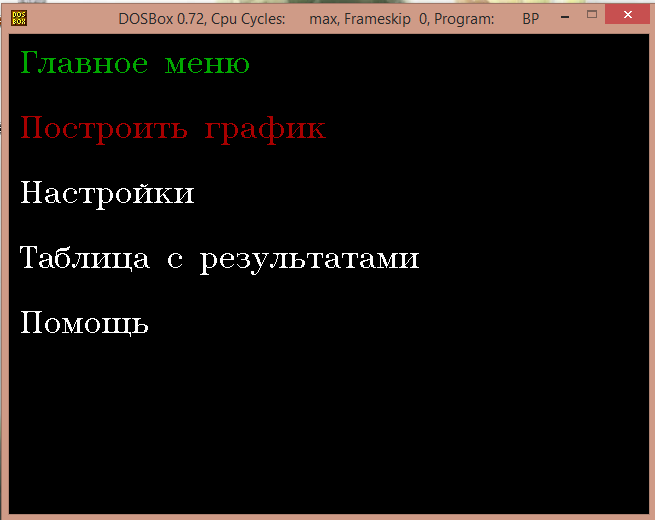
### grph.tpu - графический модуль, осуществляет рисование графиков в СМО. При этом используются вспомогательные модули Graph.

### intfmdl.tpu – интерфейсный модуль, хранящий описания объектов действий по нажатию стрелок, enter, esc. Модуль содержит объект menu. При этом используются вспомогательные модули Graph,crt,funcmdle,grph.

### HelpMe.txt – файл руководства к программе, выводимый из диалогового окна запущенной программы.

### Более подробные комментарии приведены в исходном коде программы в приложениях 2-5.

* 1. **Инструкция оператора**



##### Главное меню

Перемещение по меню осуществляется с помощью стрелок вверх-вниз, а нажатие на кнопку - с помощью Enter. Пользователь может задать такие

параметры как: минимальное количество заявок, начальное и конечное состояние прибора, его инкремент.

Моделирование проходит с построением графиков в режиме реального времени.

Интерактивное моделирование

По окончанию моделирования, пользователь может увидеть результаты в виде графика или в виде таблицы. Для просмотра таблицы следует выбрать подменю «Таблица с результатами». Для просмотра графиков следует

выбрать подменю «Построить график» и выбрать интересующий. Также все инструкции можно найти в подменю «Помощь».

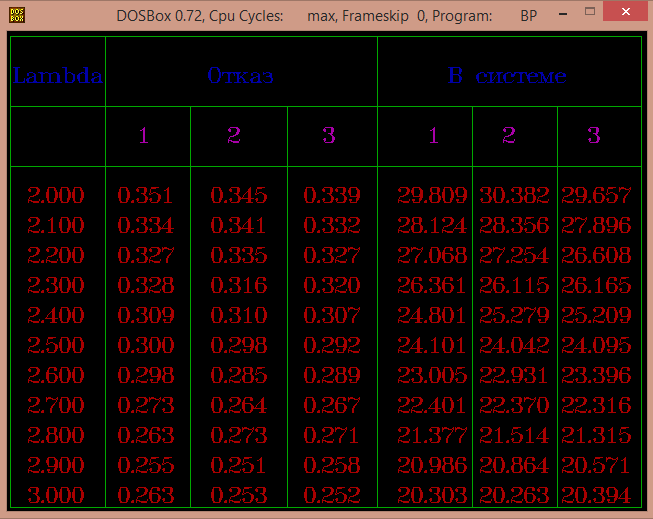
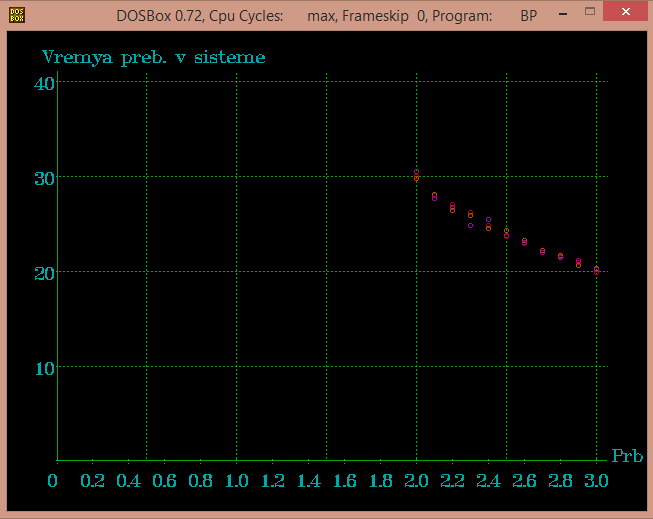
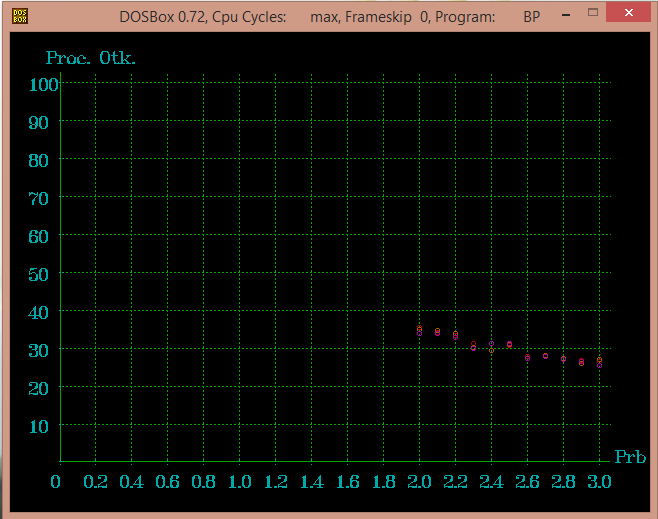


Таблица результатов





Графики

**Список использованных источников**

Веренинов И.А. Методические указания к курсовому проектированию по курсам "Основы программирования", “Структуры и алгоритмы обработки данных”, “Технология программирования для студентов кафедры ИУС факультета технической кибернетики. – СПб.: 2012. – 26с.

**Приложение 2. Исходный код головной программы GOLOVN.pas**

program glavn;

uses crt,Graph,funcmdle,grph,intfmdl; var

code : integer; key : char; Canvas : draw; Main : menu;

Gen : generator;

Pribor : machine;

Sourse : integer;

Stream : integer;

Time, PrebTime1,PrebTime2,PrebTime3 : real; Byfer : buffer;

Grap : Draw;

count, kmin : integer; f : text;

w:integer; otkz1,otkz2,otkz3:real;

BEGIN

Canvas.Init;

Main.Init;

Code := 1;

Main.DrawMenu(code); while key <> #27 do begin

Key := ReadKey; case key of

#72 : if code > 1 then

code := code - 1; #80 : if code < 4 then

code := code + 1; #13 : Main.Task(code);

end;

Main.DrawMenu(code); end;

END.

**Приложение 3. Исходный код функционального модуля FUNCMDLE.tpu**

unit funcmdle; interface const

NumOfSources = 3;{кол - во источников} SizeOfBuffer = 5;{размер буфера}

type

order = record{завка}

stream : integer;{номер источника с данной заявкой} time : real;{время генерации заявки}

end;

machine = object{прибор}

Start : real;{начальное значение}

Inc : real;{ускорение работы прибора} Edge : real;{конечное значение}

constructor Init;{создание машины со стандартными параметрами} procedure next;{ускорение прибола}

end;

generator = object{генератор заявок}

TGen : array [1..NumOfSources] of real;{время генерации заявки от каждого источника}

Lambda1, Lambda2, Lambda3 : integer;{коэфиценты для генератора заявок} procedure giveorder(var NOS : integer; var STime : real);{определение

заявки с наименьшим временем генерации и генерация новой заявки от нужного источника}

constructor Init;{инициализация генератора}

function TAY(Lambda : integer) : real;{вычисление времени генерации заявки}

end;

buffer = object{буфер}

buff : array[1..SizeOfBuffer] of order;{ячейки буфера} serviceTime : real;{время работы прибора}

timeLine : real;{временная линия прибора}

total\_1, total\_2, total\_3, denied\_1, denied\_2, denied\_3 : integer; {общее кол - во заявок и кол - во отказов для каждого источника}

TotalTime1,TotalTime2,TotalTime3 : real;{общее время ожидания в буфере заявок от каждого источника}

PriborLambda : real;{коэффициент прибора} constructor Init;{Инициализация буфера}

procedure addToBuffer(Bstream : integer; BTime : real);{Обработка поступления заявки в буфер}

procedure GetFromBuffer(InStream : integer; InTime : real);{Выборка из буфера в прибор}

procedure getPriborTime(var MLambda : real);{Получение лямбды прибора и определение времени работы прибора}

end; implementation

constructor machine.Init; begin

Inc := 0.1;

Start := 2;

Edge := 3; end;

procedure machine.next; begin

Start := Start + Inc; end;

function generator.TAY(Lambda : integer) : real; begin

TAY := (-1)/Lambda \* ln(random); end;

constructor generator.Init; begin

randomize;

Lambda1 := 3;

Lambda2 := 2;

Lambda3 := 2;

TGen[1] := generator.TAY(Lambda1); TGen[2] := generator.TAY(Lambda2); TGen[3] := generator.TAY(Lambda3);

end;

procedure generator.giveorder(var NOS : integer; var STime : real); var

i : integer; begin

NOS := 1;

STime := TGen[1];

for i := 2 to NumOfSources do if TGen[i] < STime then begin

NOS := i;

STime := TGen[i]; end;

case NOS of

1. : begin

TGen[NOS] := TGen[NOS] + generator.TAY(Lambda1);

end;

1. : begin

TGen[NOS] := TGen[NOS] + generator.TAY(Lambda2);

end;

1. : begin

TGen[NOS] := TGen[NOS] + generator.TAY(Lambda3);

end; end; end;

constructor Buffer.Init;

var i : integer; begin

for i := 1 to SizeOfBuffer do buff[i].stream := 0;

serviceTime := 0;

timeLine := 0;

total\_1 := 0; total\_2 := 0; total\_3 := 0;

denied\_1 := 0; denied\_2 := 0; denied\_3 := 0;

TotalTime1 := 0; TotalTime2:=0; TotalTime3:=0;TotalTime:=0; end;

procedure Buffer.GetFromBuffer(InStream : integer; InTime : real); var i, OutStream, Cell : integer;

OutTime : real;

R : real;

begin

OutStream := buff[1].stream; OutTime := buff[1].time; cell := 1;

if buff[1].stream = 0 then

if (SizeOfBuffer > 1) and (buff[1].stream <> 0) then for i := 2 to SizeOfBuffer do

begin

if buff[i].stream = 0 then begin

cell := i; break;

end;

if OutStream = buff[i].stream then if OutTime < buff[i].time then

begin

OutStream := buff[i].stream; OutTime := buff[i].time; Cell := i;

end;

if OutStream < buff[i].stream then begin

OutStream := buff[i].stream; OutTime := buff[i].time;

Cell := i;

end;

end;

if (cell > 0) and (cell <= SizeOfBuffer) then begin

buff[cell].stream := InStream; buff[cell].time := InTime;

end;

if OutStream = 1 then

TotalTime1 := TotalTime1 + (OutTime - TimeLine); if OutStream = 2 then

TotalTime2 := TotalTime2 + (OutTime - TimeLine); if OutStream = 3 then

TotalTime3 := TotalTime3 + (OutTime - TimeLine); TimeLine := OutTime + ServiceTime;

While R <= 0 do R := random;

ServiceTime := (-1)/PriborLambda \* ln(R); end;

procedure Buffer.AddToBuffer(Bstream : integer; BTime : real); var

info : integer; i : integer;

begin

info := -1; Case Bstream of

1 : total\_1 := total\_1 + 1; 2 : total\_2 := total\_2 + 1; 3 : total\_3 := total\_3 + 1;

end;

if BTime <= timeLine then begin

for i := 1 to SizeOfBuffer do if buff[i].stream = 0 then begin

buff[i].time := BTime; buff[i].stream := Bstream; info := i;

break; end;

if info = -1 then case BStream of

1. : denied\_1 := denied\_1 + 1;
2. : denied\_2 := denied\_2 + 1;
3. : denied\_3 := denied\_3 + 1; end;

end

else begin

GetFromBuffer(BStream, BTime); end;

end;

procedure Buffer.GetPriborTime(var MLambda : real); var

R : real; begin

while R <= 0 do R := random;

ServiceTime := (-1)/MLambda \* ln(R); PriborLambda := MLambda;

end; begin end.

**Приложение 4. Исходный код графического модуля GRPH.tpu**

unit grph;

INTERFACE

uses

Graph; type

Draw = object{объект для рисования}

Gd, Gm, x0, y0, x, y, xLeft, yLeft, xRight, yRight, n: integer; a, b, fmin, fmax, x1, y1, mx, my, dx, dy, num: real;

s: string;

constructor Init;{инициализация}

procedure Time\_MakeUp(finish,inc:real); {подготовка вывода среднего времени ожидания заявок от каждого источника в буфере}

procedure Denied\_MakeUp(finish,inc:real);{подготовка вывода процентов отказов для каждого источника}

procedure Time\_Draw(time, lambda : real);{вывод времени ожидания} procedure Denied\_Draw(d1, d2, d3, lambda : real);{вывод поцентов отказов} end;

IMPLEMENTATION

constructor Draw.Init; begin

x0 := 0;

Gd := Detect; InitGraph(Gd, Gm, ''); xLeft := 50;

yLeft := 50;

xRight := GetMaxX - 50; yRight := GetMaxY - 50; clearviewport;

end;

procedure Draw.Time\_MakeUp; var i : integer;

c,d:real; begin

a := 0; b := finish ;

if inc<0.2 then dx := 0.2; if inc>=0.2 then dx:=inc;

fmin := 0; fmax := 40; dy := 10; mx := (xRight - xLeft) / (b - a);

my := (yRight - yLeft) / (fmax - fmin); x0 := trunc(abs(a) \* mx) + xLeft;

y0 := yRight - trunc(abs(fmin) \* my); setcolor(2);

line(xLeft, y0, xRight + 10, y0); line(x0, yLeft - 10, x0, yRight); setcolor(2);

setlinestyle(1,0,1); c:=xLeft;

d:=xRight; i:=1;

line(x0+89, yLeft-10, x0+89, yRight); line(x0+179, yLeft-10, x0+179, yRight); line(x0+269, yLeft-10, x0+269, yRight); line(x0+359, yLeft-10, x0+359, yRight); line(x0+449, yLeft-10, x0+449, yRight); line(x0+539, yLeft-10, x0+539, yRight); line(x0, yLeft, x0+550, yLeft); line(x0, yLeft+95, x0+550, yLeft+95); line(x0, yLeft+190, x0+550, yLeft+190); line(x0, yLeft+285, x0+550, yLeft+285); SetColor(3);

SetTextStyle(1, 0, 1);

OutTextXY(xRight + 15, y0 - 15, 'Prb');

OutTextXY(x0 - 15, yLeft - 35, 'Vremya preb. v sisteme'); SetColor(3);

n := round((b - a) / dx) + 1; for i := 1 to n do

begin

num := a + (i - 1) \* dx;

x := xLeft + trunc(mx \* (num - a)); Line(x, y0 - 3, x, y0 + 3);

str(Num:0:1, s);

if abs(num) > 1E-15 then

OutTextXY(x - TextWidth(s) div 2, y0 + 10, s) end;

n := round((fmax - fmin) / dy) + 1; for i := 1 to n do

begin

num := fMin + (i - 1) \* dy;

y := yRight - trunc(my \* (num - fmin)); Line(x0 - 3, y, x0 + 3, y);

str(num:0:0, s);

if abs(num) > 1E-15 then

OutTextXY(x0 -23, y - TextHeight(s) div 2, s)

end;

OutTextXY(x0 - 10, y0 + 10, '0');

x1 := a; end;

procedure Draw.Time\_Draw(time, lambda : real); begin

x1 := lambda; y1 := time;

x := x0 + round(x1 \* mx); y := y0 - round(y1 \* my); Circle(x, y, 2);

end;

procedure Draw.Denied\_MakeUp; var i : integer;

begin

a := 0; b := finish;

if inc<0.2 then dx := 0.2; if inc>=0.2 then dx:=inc;

fmin := 0; fmax := 100; dy := 10; mx := (xRight - xLeft) / (b - a);

my := (yRight - yLeft) / (fmax - fmin); x0 := trunc(abs(a) \* mx) + xLeft;

y0 := yRight- trunc(abs(fmin) \* my); setcolor(2);

line(xLeft, y0, xRight + 10, y0); line(x0, yLeft - 10, x0, yRight); setlinestyle(1,0,1);

line(x0+35, yLeft-10, x0+35, yRight); for i:=1 to 14 do

line(x0+35+36\*i, yLeft-10, x0+35+36\*i, yRight); line(x0, yLeft, x0+550, yLeft);

for i:=1 to 9 do

line(x0, yLeft+38\*i, x0+550, yLeft+38\*i); SetColor(3);

SetTextStyle(1, 0, 1);

OutTextXY(xRight + 15, y0 - 15, 'Prb'); OutTextXY(x0 - 15, yLeft - 35, 'Proc. Otk.'); SetColor(3);

n := round((b - a) / dx) + 1; for i := 1 to n do

begin

num := a + (i - 1) \* dx;

x := xLeft + trunc(mx \* (num - a)); Line(x, y0 - 3, x, y0 + 3);

str(Num:0:1, s);

if abs(num) > 1E-15 then

OutTextXY(x - TextWidth(s) div 2, y0 + 10, s) end;

n := round((fmax - fmin) / dy) + 1; for i := 1 to n do

begin

num := fMin + (i - 1) \* dy;

y := yRight - trunc(my \* (num - fmin)); Line(x0 - 3, y, x0 + 3, y);

str(num:0:0, s);

if abs(num) > 1E-15 then

OutTextXY(x0 -32, y - TextHeight(s) div 2, s) end;

OutTextXY(x0 - 10, y0 + 10, '0');

x1 := a; end;

procedure Draw.Denied\_Draw(d1, d2, d3, lambda : real); begin

x1 := lambda; y1 := d1;

x := x0 + round(x1 \* mx); y := y0 - round(y1 \* my); SetColor(4);

Circle(x, y, 2); y1 := d2;

x := x0 + round(x1 \* mx); y := y0 - round(y1 \* my); SetColor(6);

Circle(x, y, 2); y1 := d3;

x := x0 + round(x1 \* mx); y := y0 - round(y1 \* my); SetColor(5);

Circle(x, y, 2); end;

BEGIN END.

**Приложение 5. Модуль интерфейса INTFMDL.tpu**

unit intfmdl; interface

uses Graph, crt, grph, funcmdle; type

Menu = object{объект для работы с интерфейсом}

kmin : integer;{минимальное кол - во заявок от каждого источника}

Start, Inc, Finish : real;{начальная и конечная лямбда, инкремент лямбды} constructor Init;{Инициализация}

procedure DrawMenu(kod : integer);{Рисование основного меню} procedure Task(kod : integer);

procedure Increment(kod : integer);{увеличение значений в настройках и проверка их корректности}

procedure Decrement(kod : integer);{уменьшение значений в настройках и проверка их корректности}

end;

procedure Output(ButtCode : integer; kmin : integer; start, inc, finish : real);{вывод результатов}

procedure convert(var s:string);{преобразование букв} implementation

procedure Table(s : string);{вывод результатов на экран} var f : text;

i : integer; begin

Assign(f, s);

Reset(f); setcolor(green);

line(3, 5, getmaxx-5, 5);

line(3, 75, getmaxx-5,75);

line(3, 135, getmaxx-5, 135);

line(3, getmaxy-3, getmaxx-5, getmaxy-3); line(3, 5, 3, getmaxy-3);

line(getmaxx-5, 5, getmaxx-5, getmaxy-3); line(98, 5, 98, getmaxy-3);

line(370, 5, 370, getmaxy-3);

line(183, 75, 183, getmaxy-3);

line(280, 75, 280 , getmaxy-3);

line(465, 75, 465, getmaxy-3);

line(550, 75, 550, getmaxy-3); Settextstyle(1,0,3); Setcolor(blue);

OutTextXY(5 ,30,'Lambda');

OutTextXY(200 ,30,'Отказ'); OutTextXY(440 ,30,'В системе'); SetTextStyle(1, 0, 3); setcolor(magenta); OutTextXY(130 ,90, '1');

OutTextXY(220 ,90, '2');

OutTextXY(315 ,90, '3');

OutTextXY(420 ,90, '1');

OutTextXY(495 ,90, '2');

OutTextXY(580 ,90, '3');

i := 1;

while not eof(f) do begin setcolor(red);

Readln(f, s);

OutTextXY(20 ,(120 + 30 \* i), s);

Readln(f, s);

OutTextXY(110 ,(120 + 30 \* i), s);

Readln(f, s);

OutTextXY(390 ,(120 + 30 \* i), s); i := i + 1;

end; setcolor(white); Close(f);

SetTextStyle(1, 0, 4); end;

constructor Menu.Init; begin

Start := 2;

Inc := 0.1;

Finish := 3;

kmin := 1500; end;

procedure Menu.DrawMenu(kod : integer); var

MainCode, AddCode, MButtX, MButtY : integer; SizeX, SizeY, i : integer;

begin

SetTextStyle(1, 0, 4); setcolor(green); Outtextxy(10,10,'Главное меню'); SetColor(White);

if kod = 1 then SetColor(Red);

OutTextXY(10,75,'Построить график'); SetColor(White);

if kod = 2 then SetColor(Red);

OutTextXY(10, 140,'Настройки'); SetColor(White);

if kod = 3 then SetColor(Red);

OutTextXY(10, 205,'Таблица с результатами'); SetColor(White);

if kod = 4 then SetColor(Red);

OutTextXY(10, 270,'Помощь'); SetColor(White);

end;

procedure Menu.Increment(kod : integer); var

koef : real;

begin

koef := 0.1; ClearViewPort; case kod of

1. : kmin := kmin + 100;
2. : if (start < finish - koef) then start := start + koef;
3. : if (finish - start > inc) then inc := inc + koef;
4. : if (trunc(finish - start) >= trunc(inc)) and (trunc(finish) >= trunc(inc)) then

finish := finish + koef;

end; end;

procedure Menu.Decrement(kod : integer); var koef : real;

begin

koef := 0.1; ClearViewPort; case kod of

1. : if kmin > 100 then kmin := kmin - 100;
2. : if (start > koef) then start := start - koef;
3. : if (finish - start > koef) and (inc > 2\*koef) then inc := inc - 0.1;
4. : if (finish - start > inc) and (finish > inc) then finish := finish - koef;

end; end;

procedure convert(var s:string); var i:integer;

begin

for i:=1 to length(s) do

if ord(s[i]) in [192..239] then s[i]:=chr(ord(s[i])-64)

else if ord(s[i]) in [240..255] then s[i]:=chr(ord(s[i])-16) else if ord(s[i]) =168 then s[i]:=chr(ord(240))

else if ord(s[i]) =184 then s[i]:=chr(ord(241));

end;

procedure Menu.Task(kod : integer); var Butt : char;

s : string; t : text; i:integer;

begin ClearViewPort;

case kod of 1 :

begin

While (Butt <> #27) do begin

Setcolor(Green); OutTextXY(10,10,'Выберите график'); setcolor(white);

if kod = 1 then SetColor(Red);

OutTextXY(10,75,'Процент отказов'); SetColor(White);

if kod = 2 then SetColor(Red);

OutTextXY(10,140,'Среднее время в системе'); SetColor(White);

if kod = 3 then SetColor(Red);

OutTextXY(10,205,'Просто промоделируйте'); SetColor(White);

Butt := ReadKey; case Butt of

#72 : if kod > 1 then

kod := kod - 1; #80 : if kod < 3 then

kod := kod + 1; #13 : begin

ClearViewPort;

Output(kod, kmin, Start, Inc, Finish); Break;

end; #27 : break; end;

end;

ClearViewPort; end;

2 :

begin

While (Butt <> #27) do begin

Setcolor(Green); OutTextXY(10,20,'Изменение параметров'); setcolor(white);

if kod = 1 then SetColor(Red); OutTextXY(10,100,'Kmin'); SetColor(White); Str(kmin, s); OutTextXY(200, 100, s);

if kod = 2 then SetColor(Red); OutTextXY(10,140,'Start'); SetColor(White);

Str(Start : 1 : 1, s); OutTextXY(200,140, s);

if kod = 3 then SetColor(Red); OutTextXY(10,180,'Lambda'); SetColor(White);

Str(Inc : 1 : 1, s); OutTextXY(200,180, s);

if kod = 4 then SetColor(Red); OutTextXY(10,220,'Finish'); SetColor(White);

Str(Finish : 1 : 1, s); OutTextXY(200,220, s);

Butt := ReadKey; case Butt of

#72 : if kod > 1 then

kod := kod - 1; #80 : if kod < 4 then

kod := kod + 1; #77 : Increment(kod);

#75 : Decrement(kod);

#27 : break; end;

end;

ClearViewPort;

end;

1. : begin

ClearViewPort; Table('output.txt'); readkey; ClearViewPort;

end;

1. : begin

ClearViewPort; assign(t,'C:\\24\HelpMe.txt'); reset(t);

settextstyle(1,0,3); setcolor(green); i:=0;

while not eof(t) do begin Readln(t, s); convert(s);

OutTextXY(10 ,10+20\*i,s); i:=i+1 ;

end; close(t); readkey;

clearviewport; end;

end;

end;

procedure Output(ButtCode : integer; kmin : integer; start, inc, finish : real);

var

Gen : generator;

Pribor : machine;

Sourse : integer;

Stream : integer;

Time, HammerTime : real; Byfer : buffer;

count : integer; f : text;

Canvas : draw; PrebTime1,PrebTime2,PrebTime3:real;

begin Canvas.Init; Gen.Init; Byfer.Init;

Assign(f, 'output.txt'); Rewrite(f);

Pribor.Start := Start; Pribor.Inc := Inc; Pribor.Edge := Finish;

Byfer.GetPriborTime(Pribor.Edge);

while (Byfer.total\_1 < kmin) or (Byfer.total\_2 < kmin) or (Byfer.total\_3 < kmin) do

begin

Gen.giveorder(Stream, Time); Byfer.AddToBuffer(Stream, Time);

end;

Byfer.Init;

if ButtCode = 2 then begin

Canvas.Time\_MakeUp(Pribor.Edge, Pribor.Inc); while Pribor.Start < Pribor.Edge do

begin randomize;

if Pribor.Edge < Pribor.start then break;

Byfer.GetPriborTime(Pribor.Start);

while (Byfer.total\_1 < kmin) or (Byfer.total\_2 < kmin) or (Byfer.total\_3 < kmin) do

begin

Gen.giveorder(Stream, Time); Byfer.AddToBuffer(Stream, Time);

end; PrebTime1:=Byfer.totaltime1/(Byfer.total\_1-

Byfer.denied\_1)+Pribor.Start+(Gen.TGen[1]/Byfer.total\_1); PrebTime2:=Byfer.totaltime2/(Byfer.total\_2-

Byfer.denied\_2)+Pribor.Start+(Gen.TGen[2]/Byfer.total\_2); PrebTime3:=Byfer.totaltime3/(Byfer.total\_3-

Byfer.denied\_3)+Pribor.Start+(Gen.TGen[3]/Byfer.total\_3); setcolor(red);

Canvas.Time\_Draw(PrebTime1, Pribor.Start); setcolor(6);

Canvas.Time\_Draw(PrebTime2, Pribor.Start); setcolor(5);

Canvas.Time\_Draw(PrebTime3, Pribor.Start); Writeln(f, Pribor.start : 1 : 3);

Write(f, Byfer.denied\_1/Byfer.total\_1 : 1 : 3, ' '); Write(f, Byfer.denied\_2/Byfer.total\_2 : 1 : 3, ' '); Writeln(f, Byfer.denied\_3/Byfer.total\_3 : 1 : 3, ' '); Write(f, PrebTime1 : 1 : 3, ' ');

Write(f, PrebTime2 : 1 : 3, ' ');

Writeln(f, PrebTime3 : 1 : 3, ' '); Byfer.Init;

Pribor.next; end;

readln; end;

if ButtCode = 1 then begin

Canvas.Denied\_MakeUp(Pribor.Edge, Pribor.Inc); while Pribor.Start < Pribor.Edge do

begin

randomize; Byfer.GetPriborTime(Pribor.Start);

while (Byfer.total\_1 < kmin) or (Byfer.total\_2 < kmin) or (Byfer.total\_3 < kmin) do

begin

Gen.giveorder(Stream, Time); Byfer.AddToBuffer(Stream, Time);

end;

Canvas.Denied\_Draw(Byfer.denied\_1/Byfer.total\_1 \* 100,

Byfer.denied\_2/Byfer.Total\_2 \* 100, Byfer.denied\_3/Byfer.total\_3 \* 100, Pribor.Start); Writeln(f, Pribor.start : 1 : 3);

Write(f, Byfer.denied\_1/Byfer.total\_1 : 1 : 3, ' '); Write(f, Byfer.denied\_2/Byfer.total\_2 : 1 : 3, ' '); Writeln(f, Byfer.denied\_3/Byfer.total\_3 : 1 : 3, ' '); PrebTime1:=Byfer.totaltime1/(Byfer.total\_1-

Byfer.denied\_1)+Pribor.Start+(Gen.TGen[1]/Byfer.total\_1); PrebTime2:=Byfer.totaltime2/(Byfer.total\_2-

Byfer.denied\_2)+Pribor.Start+(Gen.TGen[2]/Byfer.total\_2); PrebTime3:=Byfer.totaltime3/(Byfer.total\_3-

Byfer.denied\_3)+Pribor.Start+(Gen.TGen[3]/Byfer.total\_3); Write(f, PrebTime1 : 1 : 3, ' ');

Write(f, PrebTime2 : 1 : 3, ' ');

Writeln(f, PrebTime3 : 1 : 3, ' '); Byfer.Init;

Pribor.next; end;

readln; end;

if ButtCode = 3 then begin

SetTextStyle(1, 0, 4); setcolor(green);

OutTextXY(round(GetMaxX/3) + 20, round(GetMaxY/3) + 20, 'Њ®¤Ґ«ЁагҐ¬...'); setcolor(white);

while (Pribor.Start < Pribor.Edge ) do begin

if Pribor.Edge < Pribor.Start then break;

randomize; Byfer.GetPriborTime(Pribor.Start);

while (Byfer.total\_1 < kmin) or (Byfer.total\_2 < kmin) or (Byfer.total\_3 < kmin) do

begin

Gen.giveorder(Stream, Time); Byfer.AddToBuffer(Stream, Time);

end; PrebTime1:=Byfer.totaltime1/(Byfer.total\_1-

Byfer.denied\_1)+Pribor.Start+(Gen.TGen[1]/Byfer.total\_1); PrebTime2:=Byfer.totaltime2/(Byfer.total\_2-

Byfer.denied\_2)+Pribor.Start+(Gen.TGen[2]/Byfer.total\_2); PrebTime3:=Byfer.totaltime3/(Byfer.total\_3-

Byfer.denied\_3)+Pribor.Start+(Gen.TGen[3]/Byfer.total\_3); Writeln(f, Pribor.start : 1 : 3);

Write(f, Byfer.denied\_1/Byfer.total\_1 : 1 : 3, ' '); Write(f, Byfer.denied\_2/Byfer.total\_2 : 1 : 3, ' '); Writeln(f, Byfer.denied\_3/Byfer.total\_3 : 1 : 3, ' '); Write(f, PrebTime1 : 1 : 3, ' ');

Write(f, PrebTime2 : 1 : 3, ' ');

Writeln(f, PrebTime3 : 1 : 3, ' '); Byfer.Init;

Pribor.next; end;

ClearViewPort; setcolor(green);

OutTextXY(round(GetMaxX/3) + 60, round(GetMaxY/3) + 20, 'ѓ®в®ў®'); Delay(500);

setcolor(white); end;

Close(f); end;

begin END.