|  |  |  |
| --- | --- | --- |
| Санкт-Петербургский политехнический университет  Институт компьютерных наук и технологий  **Кафедра «Информационные и управляющие системы»** | | |
| **КУРСОВОЙ ПРОЕКТ** | | |
| МОДЕЛИРОВАНИЕ СИСТЕМЫ, ФОРМАЛИЗОВАННОЙ КАК СИСТЕМА МАССОВОГО ОБСЛУЖИВАНИЯ | | |
| по дисциплине «Алгоритмизация и основы программирования» | | |
| Выполнил | | |
| студент гр.13534/3 |  | А. А. Логинов |
| Руководитель | | |
| доцент, к.т.н. |  | И.А.Веренинов |
| 28 мая 2019 г. | | |
| Санкт-Петербург | | |
| 2019 | | |

ЗАДАНИЕ

НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| студенту группы | | 13534/3 | |  | Логинову Аркадию Александровичу | | | | | | | | |
|  | | *(номер группы)* | |  | *(фамилия, имя, отчество)* | | | | | | | | |
| ***1. 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* | | | | | | | | | | | | | |
| ***2. Срок сдачи студентом законченного проекта (работы)*** | | | | | | | | | | | | | *28 мая 2019 года* |
| ***3. Исходные данные к проекту (работе)*** | | | | | | | | |  | | | | |
| *Перечень индивидуальных заданий на проект с указанием конкретного номера задания*  *Методическое пособие по курсовому проектированию*  *Конспект лекций по программированию* | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| ***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 | | | | | | ***40*** | | | | страниц машинописного | | | |
| Текста | | | | | | | | | | | | | |
| 5. Перечень графического материала (с указанием обязательных чертежей и | | | | | | | | | | | | | |
| плакатов) | Нет | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 6. Консультанты | | | Нет | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| 7. Дата получения задания: «\_12\_\_\_» \_февраля\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | |
| Руководитель | | | | |  | | |  | | | *Веренинов И. А.* | | |
|  | | | | | *(подпись)* | | |  | | | *(инициалы, фамилия)* | | |
| Задание принял к исполнению | | | | |  | |  | | | | *Логинов А. А.* | | |
|  | | | | | *(подпись)* | |  | | | | *(инициалы, фамилия)* | | |
|  | | | | | | | | | | | |  | |
|  | | | | | | | | | | | | *(дата)* | |

СОДЕРЖАНИЕ

[Введение 5](#_Toc9885257)

[1. Инструкция программиста 6](#_Toc9885258)

[2. Инструкция оператора 8](#_Toc9885259)

[Список использованных источников 13](#_Toc9885260)

[Приложение 1. Исходный код модуля FUNC\_MOD 14](#_Toc9885261)

[Приложение 2. Исходный код головной программы Main 19](#_Toc9885262)

[Приложение 3. Исходный код модуля Screen 22](#_Toc9885263)

[Приложение 4. Исходный код модуля MyGraph. 26](#_Toc9885264)

[Приложение 5. Исходный код модуля MyTable 28](#_Toc9885265)

[Приложение 6. Исходный код модуля OnGraph 30](#_Toc9885266)

[Приложение 7. Исходный код модуля Menu 32](#_Toc9885267)

[Приложение 8. Исходный код модуля Tools 34](#_Toc9885268)

[Приложение 9. Исходный код модуля TypeAns 38](#_Toc9885269)

# Введение

Задача № 14.

Задача состоит в разработке программной системы моделирования, формализованной как система массового обслуживания (СМО) с тремя входными потоками с применением технологии ООП.

ДАНО:

* Количество источников – 3 (I – высший приоритет)
* Потоки заявок - Эрланга 2 порядка с λ1==2, λ2 и λ3 =1 - порождающ. простейшего потока
* Буфер - отсутствует
* Поток обслуживания – простейший с

НАЙТИ:

* Вероятность отказа Ротказа i = f 1 i ( λ2 ) λ2\* =1 ≤ λ2≤λ2\*\* =3 с Δλ2=0.2
* Среднее время ожидания М (Тожид i) = f 2 i (λ1)
* Среднее время обслуживания заявки от второго источника:

М (Тобсл 2) = f( λ2)

# Инструкция программиста

Данный программный продукт состоит из 9 файлов: Main.pas, FuncUnit.pas, Screen.pas, MyGraph.pas, OnGraph.pas, Menu.pas, MyTable.pas, Tools.pas, TypeAns.pas.

FuncUnit.pas представляет собой функциональный модуль программы, который содержит описание используемых в нем типов объектов, таких как «источники» (generator), «исполнитель» (isp), а также процедуру emulate, запускающей моделирование работы системы массового обслуживания. Все объекты статические. При этом используются вспомогательные модули OnGraph, Tools. Методы функционального модуля обеспечивают формирование результатов одной длинной реализации, т.е. зависимость вероятностей отказов от количества заявок. Эти данные в процессе моделирования сразу же в графическом режиме выводятся на экран, с оцифровкой по осям координат. Кроме того, данный модуль обеспечивает фиксацию окончательных результатов моделирования при варьировании основного параметра системы – интенсивности второго источника, который изменяется при переходе к другой длинной реализации. Эти результаты записываются в отдельный файл.

MyGraph.PAS представляет собой графический модуль программы (graf\_module). Он содержит методы, которые обеспечивают вывод результирующих зависимостей.

OnGrpah.pas – модуль, который обеспечивает отрисовку графиков, показывающих зависимость отказа в обработке поступившей заявки от количества cгенерированных заявок. При этом графики строятся в режиме online в процессе моделирования.

Screen.pas является интерфейсным модулем. Он является контроллером между данными и отображением их на экране. После эмуляции в модуле Main.pas данные передаются в данный модуль, который в свою очередь передает их в нужный момент нужному модулю для того, чтобы показать новый экран.

Menu.pas является модулем, который хранит и умеет показывать пользователю вкладки меню на экране. В нем описан статический объект tab (вкладка).

MyTable.pas – модуль, который показывает все результаты на экран в виде таблицы.

TypeAns.pas – модуль, который хранит в типы результатов. Написан для того, чтобы код стал более читабельным и понятным.

Tools.pas – модуль, который хранит данные о всех пользовательских настройках, также позволяет пользователю их изменять во время работы программы. Умеет показывать хранящиеся данные на экране.

В файле Main.pas представлена головная программа, к которой подключен интерфейсный модуль и запускается меню из этого модуля.

# Инструкция оператора

При запуске данной программы первоначально эмулируется сам процесс со всеми стандартными настройками. На экране пользователь видит каждую реализацию, в режиме онлайн.

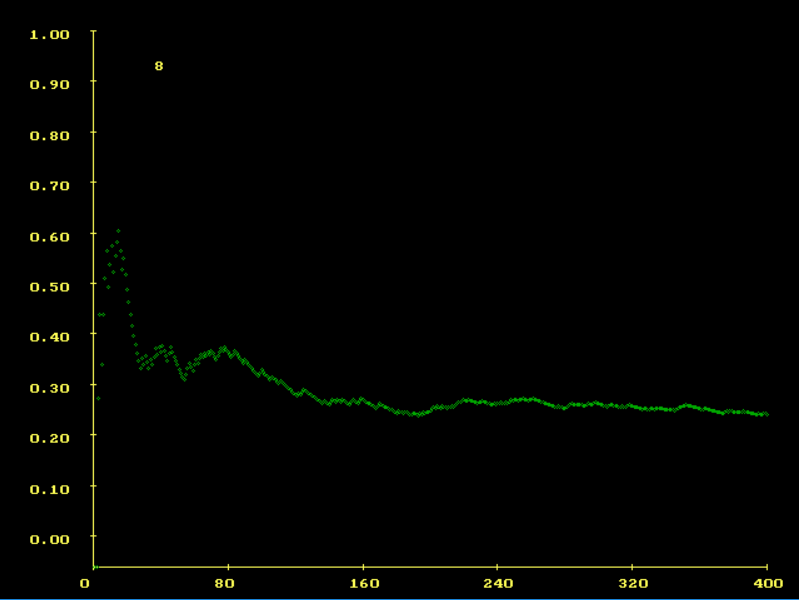


Рисунок 4. Онлайн график в процессе моделирования

Подменю параметров (Settings) содержит кнопки с числовыми значениями: KMIN – длина реализации, Answers – количество реализаций, Step – шаг изменения скорости работы второго источника, Speed\_1, Speed\_3 – скорости работы 1 и 3 источников соответственно, SpeedIsp – скорость работы исполнителя, Min – первоначальное значение скорости второго источника. При запуске программы параметры моделирования установлены по умолчанию. Их можно менять, вертикальными стрелками клавиатуры выбрав в подменю параметров нужный и изменить горизонтальными стрелками. При этом значения на экране будут меняться. После того как процесс закончился пользователь автоматически переходит на главное меню приложения. На экран с одним из графиков. Теперь горизонтальными стрелками листая вкладки он может посмотреть все графики

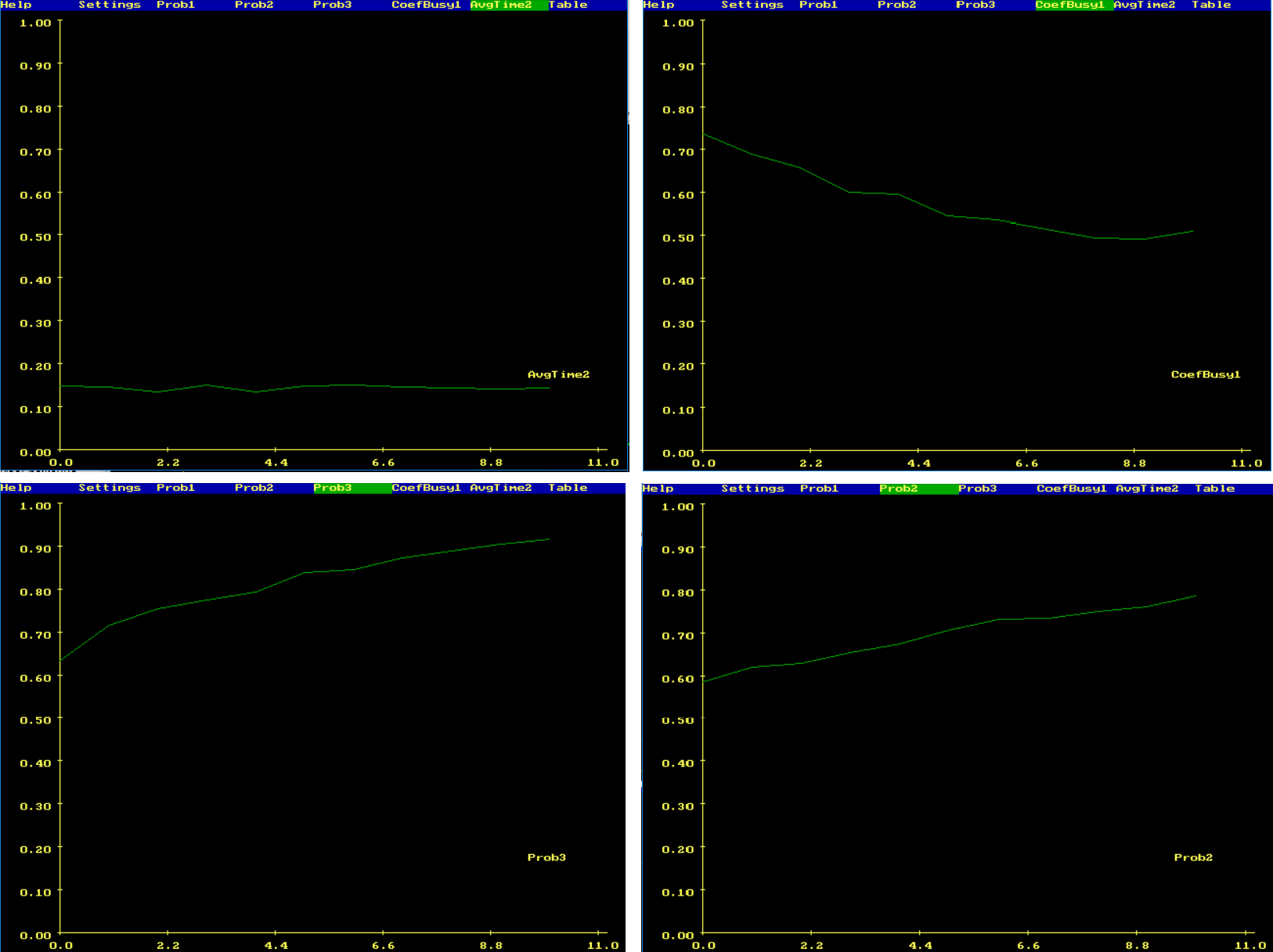


Рисунок 2. Окна графиков

Также можно перейти на вкладку Table, где пльзователю анимированно покажется таблица всех результатов



Рисунок 2. Окно настроек

Подменю параметров (Settings) содержит кнопки с числовыми значениями: KMIN – длина реализации, Answers – количество реализаций, Step – шаг изменения скорости работы второго источника, Speed\_1, Speed\_3 – скорости работы 1 и 3 источников соответственно, SpeedIsp – скорость работы исполнителя, Min – первоначальное значение скорости второго источника. При запуске программы параметры моделирования установлены по умолчанию. Их можно менять, вертикальными стрелками клавиатуры выбрав в подменю параметров нужный и изменить горизонтальными стрелками. При этом значения на экране будут меняться.

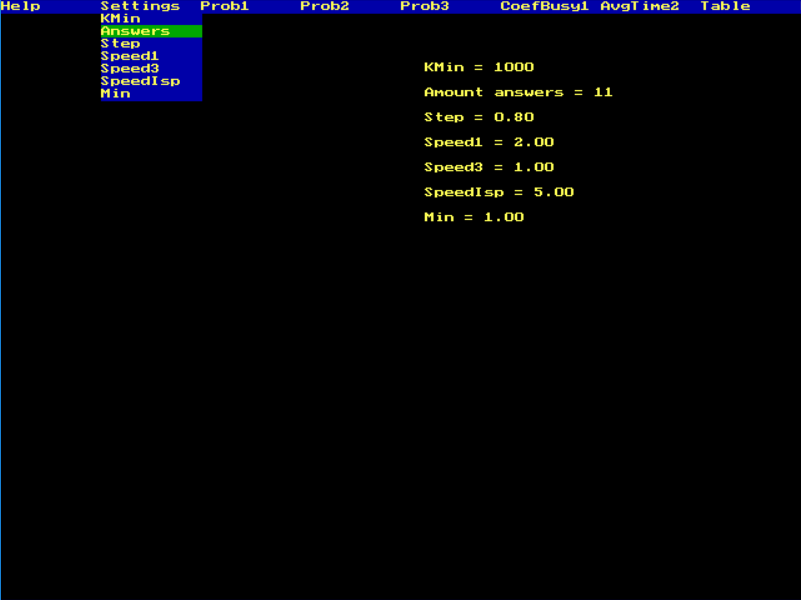


Рисунок 2. Окно настроек

Установка новых значений параметров пользователем сопровождается их проверкой. Так, для всех значений нельзя устанавливать отрицательные числа.

В подменю Help пользователь может увидеть краткую информацию об управлении программой

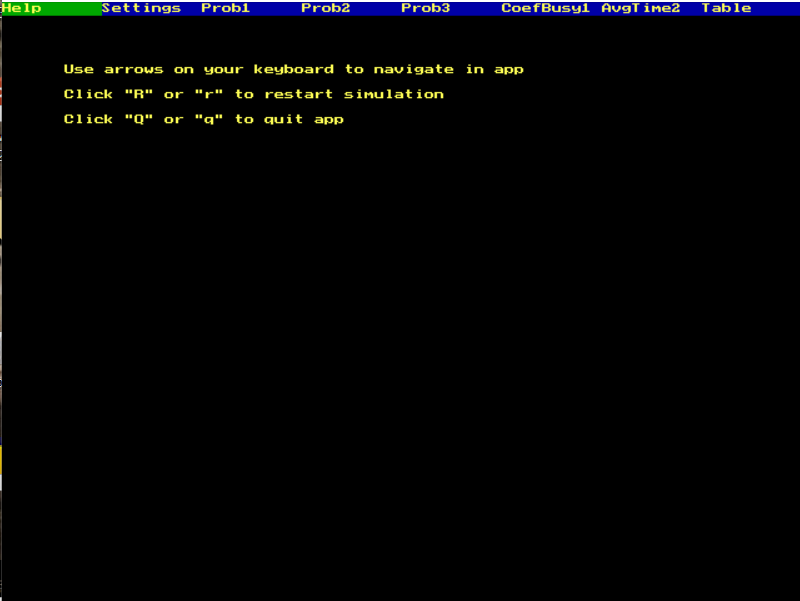


Рисунок 2. Окно настроек

# Список использованных источников

Веренинов И.А. Программирование на языке высокого уровня: учеб. пособие / И. А. Веренинов. – СПб.: Изд-во Политехн. ун-та, 2013. – 212с.

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 1. Исходный код модуля FUNC\_MOD

**unit** FuncUnit;

**interface**

**uses** OnGraph, Tools;

**type** person = object

speed, nextTime: real;

priority: integer;

**procedure** setNextTime;

**end**;

generator = object(person)

countGood, countBad, count: integer;

fullTime: real;

**constructor** initGenerator(number: integer);

**procedure** showResults(number: integer);

**procedure** showProbablity(number : integer);

**end**;

isp = object(person)

number : integer;

oldTime : real;

**constructor** initIsp;

**end**;

arr = **array** [1..3] **of** generator;

**procedure** emulate(num: integer; p : isp; **var** a : arr; speed2 : real; kMin : integer);

**procedure** writeAnswers2(a : arr; speed2 : real);

**function** getTime(a : arr): real;

**implementation**

**procedure** person.setNextTime;

**begin**

nextTime := nextTime - Ln(Random) / speed;

**end**;

**constructor** generator.initGenerator(number: integer);

**begin**

nextTime := 0;

priority := number;

count := 0;

countGood := 0;

countBad := 0;

fullTime := 0;

**case** priority **of**

1: speed := getSpeed1;

2: speed := 1;

3: speed := getSpeed3;

**end**;

**if** number <> 2 **then** setNextTime;

**end**;

**procedure** generator.showResults(number : integer);

**begin**

writeln(number, '. real = ', countBad + countGood, ' || teor = ', count);

**end**;

**constructor** isp.initIsp;

**begin**

priority := 10;

speed := getSpeedIsp;

number := 0;

nextTime := 1000000.0;

**end**;

**procedure** generator.showProbablity(number : integer);

**begin**

write(countBad / count:0:3, ' | ');

**end**;

**function** getMinIndexTime(**var** a: arr; **var** p : isp): integer;

**var** minIndex, i : integer;

**begin**

minIndex := 1;

**for** i := 2 **to** 3 **do**

**if**(a[minIndex].nextTime > a[i].nextTime) **then**

minIndex := i;

**if**(a[minIndex].nextTime > p.nextTime) **then**

minIndex := -1;

getMinIndexTime := minIndex;

**end**;

**function** getMinCount(a: arr):integer;

**var** min, i : integer;

**begin**

min:= a[1].count;

**for** i := 2 **to** 3 **do**

**if**(min > a[i].count) **then**

min := a[i].count;

getMinCount := min;

**end**;

**function** getProbability(a : arr): real;

**var** count, countBad, i : integer;

**begin**

count := 0;

countBad := 0;

**for** i := 1 **to** 3 **do begin**

count := count + a[i].count;

countBad := countBad + a[i].countBad;

**end**;

getProbability := countBad / count;

**end**;

**function** getSum(a : arr): integer;

**var** sum, i : integer;

**begin**

sum := 0;

**for** i := 1 **to** 3 **do begin**

sum := sum + a[i].count;

**end**;

getSum := sum;

**end**;

**function** getTime(a : arr): real;

**var** i : integer;

time : real;

**begin**

time := 0;

**for** i := 1 **to** 3 **do**

time := time + a[i].fullTime;

getTime := time;

**end**;

**function** getAvgTime(a: arr; i : integer): real;

**begin**

**if**(a[i].countGood = 0) **then**

getAvgTime := 0

**else**

getAvgTime := a[i].fullTime / a[i].countGood;

**end**;

**procedure** writeAnswers2(a : arr; speed2 : real);

**var** i: integer;

**begin**

write(speed2:0:2, ' | ');

**for** i := 1 **to** 3 **do**

a[i].showProbablity(i);

write(a[1].fullTime / getTime(a):0:3, ' | ');

**for** i := 1 **to** 3 **do**

write(getAvgTime(a, i):0:3, ' | ');

writeln;

**end**;

**procedure** emulate(num : integer; p : isp; **var** a : arr; speed2 : real; kMin : integer);

**var**

i, actualIndex: integer;

**begin**

randomize;

p.initIsp;

**for** i := 1 **to** 3 **do**

a[i].initGenerator(i);

a[2].speed := speed2;

a[2].setNextTime;

initMyGraph(num);

**while** (getMinCount(a) < kMin) **do begin**

actualIndex := getMinIndexTime(a, p);

{Isp end the work}

**if**(actualIndex < 0) **then begin**

inc(a[p.number].countGood);

a[p.number].fullTime := a[p.number].fullTime + p.nextTime - p.oldTime;

p.initIsp;

**continue**;

**end**;

inc(a[actualIndex].count);

**if**(actualIndex = 1) **then**

drawNextPoint(a[1].countBad / a[1].count, a[1].count);

{New Request change old}

**if** (a[actualIndex].priority < p.priority) **then begin**

p.priority := a[actualIndex].priority;

**if** (p.number > 0) **then** inc(a[p.number].countBad);

p.number := actualIndex;

p.nextTime := a[actualIndex].nextTime;

p.oldTime := p.nextTime;

p.setNextTime;

**end else**{This request was closed}

inc(a[actualIndex].countBad);

a[actualIndex].setNextTime;

**end**;

**end**;

**end**.

# Приложение 2. Исходный код головной программы Main

**program** main;

**uses** FuncUnit, TypeAns, Screen, Tools, crt;

**var**

mIsp : isp; a : arr;

speed2 : real;

i, j, index : integer;

ch: char;

needRestart: boolean;

probs: **array**[1..3] **of** answers;

coefs1 : answers;

avgTimes2 : answers;

**procedure** init(**var** mIsp: isp; **var** a : arr);

**var** i : integer;

**begin**

assign(output, 'outsmo.txt');

rewrite(output);

setDefaultSettigns;

**end**;

**procedure** quit;

**begin**

close(output);

**end**;

**procedure** work;

**var** i, j : integer;

**begin**

speed2 := getLeft;

writeln('-----------------------------------------');

**for** i := 1 **to** getCountAnswers **do begin**

emulate(i, mIsp, a, speed2, getKMin);

writeAnswers2(a, speed2);

**for** j := 1 **to** 3 **do begin**

probs[j][i] := a[j].countBad / a[j].count;

**end**;

coefs1[i] := a[1].fullTime / getTime(a);

avgTimes2[i] := a[2].fullTime / a[2].countGood;

speed2 := speed2 + getStep;

**end**;

**end**;

**procedure** life;

**begin**

createMenu2(1);

createMenu2(2);

createMenu(3, probs[1]);

createMenu(4, probs[2]);

createMenu(5, probs[3]);

createMenu(6, coefs1);

createMenu(7, avgTimes2);

createMenu2(8);

initScreen;

needRestart := false;

**while** true **do begin**

ch := readKey;

**if**(ch = #77) **then** toRight;

**if**(ch = #75) **then** toLeft;

**if**(ch = #72) **then** toDown;

**if**(ch = #80) **then** toUp;

**if**(ch = #13) **then** select;

**if** (ch = 'r') **or** (ch = 'R') **or** (ch = 'Љ') **or** (ch = 'Є') **then begin**

needRestart := true;

**break**;

**end**;

**if** (ch = 'q') **or** (ch = 'Q') **or** (ch = '©') **or** (ch = '‰') **then break**;

**end**;

**end**;

**begin**

init(mIsp, a);

needRestart := true;

**while** needRestart **do begin**

work;

life;

**end**;

quit;

**end**.

# Приложение 3. Исходный код модуля Screen

**unit** Screen;

**INTERFACE**

**uses** Graph, Menu, MyGraph, TypeAns, Tools, MyTable;

**const**

names: **array** [1..COUNT\_TABS] **of** string = ('Help', 'Settings', 'Prob1',

'Prob2', 'Prob3', 'CoefBusy1', 'AvgTime2', 'Table');

**procedure** createMenu2(i: integer);

**procedure** createMenu(i: integer; ans: answers);

**procedure** toLeft;

**procedure** toRight;

**procedure** toUp;

**procedure** toDown;

**procedure** select;

**procedure** initScreen;

**IMPLEMENTATION**

**var**

gd, gm, indexTab: integer;

a: arrTabs;

**procedure** drawHelp;

**begin**

setColor(Yellow);

outTextXY(50, 50, 'Use arrows on your keyboard to navigate in app');

outTextXY(50, 70, 'Click "R" or "r" to restart simulation');

outTextXY(50, 90, 'Click "Q" or "q" to quit app');

**end**;

**procedure** drawMenu;

**var** i: integer;

**begin**

**for** i := 1 **to** COUNT\_TABS **do**

a[i].draw(i, round(GetMaxX / COUNT\_TABS), indexTab);

**end**;

**procedure** redraw;

**var** table : CellArray;

i, indexResult : integer;

speed2 : real;

**begin**

clearDevice;

drawMenu;

**if** (a[indexTab].isGraph) **then** drawGraph(a[indexTab].tabs[0], a[indexTab].ans);

**if**(indexTab = 1) **then** drawHelp;

**if**(indexTab = 2) **then** drawSettings(getMaxX - 300);

**if**(indexTab = 8) **then begin**

indexResult := 1;

{set data}

table[1].name := 'Speed2';

speed2 := getLeft;

**for** i := 1 **to** getCountAnswers **do begin**

table[1].a[i] := speed2;

speed2 := speed2 + getStep;

**end**;

indexResult := 2;

**for** i := 1 **to** COUNT\_TABS **do begin**

**if**(a[i].isGraph) **then begin**

table[indexResult].name := a[i].tabs[0];

table[indexResult].a := a[i].ans;

inc(indexResult);

**end**;

**end**;

drawTable(table);

**end**;

**end**;

**procedure** createMenu(i: integer; ans: answers);

**begin**

**if** i **in** [1..COUNT\_TABS] **then**

a[i].init2(names[i], ans);

**end**;

**procedure** createMenu2(i: integer);

**begin**

**if** i **in** [1..COUNT\_TABS] **then**

a[i].init1(names[i]);

**if**(i = 2) **then begin**

a[i].setNext('KMin');

a[i].setNext('Answers');

a[i].setNext('Step');

a[i].setNext('Speed1');

a[i].setNext('Speed3');

a[i].setNext('SpeedIsp');

a[i].setNext('Min');

**end**;

**end**;

**procedure** initScreen;

**begin**

gd := Detect;

InitGraph(gd, gm, '');

indexTab := 3;

redraw;

**end**;

**procedure** toLeft;

**begin**

**if**(a[indexTab].index <> 0) **and** (indexTab = 2) **then begin**

decrease(a[indexTab].index);

redraw;

**exit**;

**end**;

**if**(indexTab > 1) **then begin**

a[indexTab].close;

indexTab := indexTab - 1;

drawMenu;

**end**;

**end**;

**procedure** toRight;

**begin**

**if**(a[indexTab].index <> 0) **and** (indexTab = 2) **then begin**

increase(a[indexTab].index);

redraw;

**exit**;

**end**;

**if**(indexTab < COUNT\_TABS ) **then begin**

a[indexTab].close;

indexTab := indexTab + 1;

drawMenu;

**end**;

**end**;

**procedure** toUp;

**begin**

**if**(a[indexTab].isOpen) **and** (a[indexTab].index < a[indexTab].n - 1) **then begin**

inc(a[indexTab].index);

drawMenu;

**end**;

**end**;

**procedure** toDown;

**begin**

**if**(a[indexTab].isOpen) **and** (a[indexTab].index > 0) **then begin**

dec(a[indexTab].index);

drawMenu;

**end**;

**end**;

**procedure** select;

**begin**

**if not** (a[indexTab].isOpen) **then begin**

a[indexTab].open;

**end else begin**

**if**(a[indexTab].index = 0) **then** a[indexTab].close;

**if**(indexTab = 2) **then**

**case** a[indexTab].index **of**

1:

**end**;

**end**;

redraw;

**end**;

**END**.

# Приложение 4. Исходный код модуля MyGraph.

**unit** MyGraph;

**INTERFACE**

**uses** Graph, TypeAns, crt, Tools;

**const** COUNT\_HATCH = 10;

**procedure** drawGraph(name : string; a : answers);

**IMPLEMENTATION**

**const** k = 20;

**var** h, w, x0, y0 : integer;

gd, gm : integer;

dx, dy, y1, x1: real;

speed2, max, ms, min, dry , ds: real;

s : string;

**function** getY(ry : real):integer;

**begin**

getY := y0 - round( ry \* ms );

**end**;

**procedure** drawGraph(name : string; a : answers);

**var** i : integer;

**begin**

h := getMaxY;

w := getMaxX;

x0 := 3 \* k;

y0 := h - k;

setColor(Yellow);

line(x0, y0, w - k, y0);

line(x0, y0, x0, k);

dry := (1.0 - 0.0) / COUNT\_HATCH;

dx := (w - 4 \* k) **div** getCountAnswers;

dy := (h - 2 \* k) / COUNT\_HATCH;

ms :=(h - 2 \* k) / (1.0 - 0.0);

y1 := y0;

speed2 := 0.0;

outTextXY(w - 5 \* k, h - 5 \* k , name);

{horizontal lines}

**for** i := 0 **to** COUNT\_HATCH **do begin**

y1 := y0 - dy \* i;

line(x0 - 2, round(y1), x0 + 2, round(y1));

str(speed2:0:2, s);

outTextXY(x0 - 2 \* k, round(y1), s);

speed2 := speed2 + dry;

**end**;

x1 := x0;

speed2 := 0.0;

ds := getCountAnswers / 5;

{vertical lines }

**for** i := 1 **to** getCountAnswers **do begin**

line(round(x1), y0 + 2, round(x1), y0 - 2);

str(speed2:0:1, s);

outTextXY(round(x1 - 10), y0 + 10, s);

x1 := x1 + dx \* ds;

speed2 := speed2 + ds;

**end**;

x1 := x0 + dx;

setColor(Green);

**for** i := 2 **to** getCountAnswers **do begin**

line(round(x1 - dx),

getY(a[i-1]),

round(x1),

getY(a[i]));

delay(15);

x1 := x1 + dx;

**end**;

**end**;

**END**.

# Приложение 5. Исходный код модуля MyTable

**unit** MyTable;

**INTERFACE**

**uses** TypeAns, Tools, Graph, crt;

**const** COUNT\_COLUMN = 6;

SPACE = 30;

CELL\_HEIGHT = 20;

**type** Column = object

name : string;

a : answers;

**end**;

**type** CellArray = **array** [1..COUNT\_COLUMN] **of** column;

**procedure** drawTable(columns : CellArray);

**IMPLEMENTATION**

**procedure** drawCell(xCentre, yTop : integer; text : string);

**var** width, height : integer;

**begin**

width := textWidth(text);

height := textHeight(text);

outTextXY(xCentre - width **div** 2,

yTop + CELL\_HEIGHT **div** 2 - height **div** 2,

text);

**end**;

**procedure** drawColumn(col : Column; x1, x2 : integer);

**var** XCentre, y, i: integer;

text : string;

**begin**

xCentre := x1 + (x2 - x1) **div** 2;

y := SPACE;

drawCell(xCentre, y, col.name);

{vertical lines}

line(x1, y, x1, y + (getCountAnswers + 1) \* CELL\_HEIGHT);

line(x2, y, x2, y + (getCountAnswers + 1) \* CELL\_HEIGHT);

**for** i := 1 **to** getCountAnswers **do begin**

y := y + CELL\_HEIGHT;

str(col.a[i]:0:3, text);

drawCell(xCentre, y, text);

**end**;

**end**;

**procedure** drawTable(columns : CellArray);

**var** widthColumn, x , width, i: integer;

**begin**

width := getMaxX;

widthColumn := (width - 2 \* SPACE) **div** COUNT\_COLUMN;

x := SPACE;

{horizontal lines}

line(SPACE, SPACE, x + widthColumn \* COUNT\_COLUMN, SPACE);

line(SPACE, SPACE + CELL\_HEIGHT, x + widthColumn \* COUNT\_COLUMN, SPACE + CELL\_HEIGHT);

line(SPACE,

SPACE + CELL\_HEIGHT \* (getCountAnswers + 1),

x + widthColumn \* COUNT\_COLUMN,

SPACE + CELL\_HEIGHT \* (getCountAnswers + 1));

**for** i := 1 **to** COUNT\_COLUMN **do begin**

delay(50);

drawColumn(columns[i], x, x + widthColumn);

x := x + widthColumn;

**end**;

**end**;

**END**.

# Приложение 6. Исходный код модуля OnGraph

**unit** OnGraph;

**INTERFACE**

**uses** Graph, crt, TypeAns;

**const** COUNT\_HATCH = 10;

**procedure** drawNextPoint(ry : real; nk : integer);

**procedure** initMyGraph(num : integer);

**IMPLEMENTATION**

**const** k = 25;

COUNT\_POINT = 400;

**var** h, w, x0, y0 : integer;

gd, gm : integer;

dx, dy, y1, x1: real;

speed2, max, ms, min, dry, ds : real;

s : string;

**function** getY(ry : real):integer;

**begin**

getY := y0 - round(ry \* ms);

**end**;

**procedure** drawNextPoint(ry: real; nk : integer);

**begin**

**if** (nk **mod** 5 = 0) **then** delay(1);

**if**(nk > COUNT\_POINT) **then exit**;

setColor(Green);

circle(round(x1), getY(ry), 1);

x1 := x1 + dx;

**end**;

**procedure** initMyGraph(num : integer);

**var** i : integer;

**begin**

gd := Detect;

initGraph(gd, gm, '');

clearDevice;

h := getMaxY;

w := getMaxX;

x0 := 3 \* k;

y0 := h - k;

setColor(Yellow);

line(x0, y0, w - k, y0);

line(x0, y0, x0, k);

dry := (1.0 - 0.0) / COUNT\_HATCH;

dx := (w - 4 \* k) / COUNT\_POINT;

dy := (h - 3 \* k) / COUNT\_HATCH;

ms :=(h - 3 \* k) / (1.0 - 0.0);

y1 := y0 - k;

speed2 := 0;

str(num, s);

outTextXY(5 \* k, 2 \* k , s);

{horizontal lines}

**for** i := 1 **to** COUNT\_HATCH + 1 **do begin**

line(x0 - 2, round(y1), x0 + 2, round(y1));

str(speed2:0:2, s);

outTextXY(x0 - 2 \* k, round(y1), s);

speed2 := speed2 + dry;

y1 := y1 - dy;

**end**;

x1 := x0;

speed2 := 0.0;

ds := COUNT\_POINT / 5;

{vertical lines }

**for** i := 1 **to** COUNT\_ANSWERS **do begin**

line(round(x1), y0 + 2, round(x1), y0 - 2);

str(round(speed2), s);

outTextXY(round(x1 - 10), y0 + 10, s);

x1 := x1 + dx \* ds;

speed2 := speed2 + ds;

**end**;

x1 := x0 + dx;

setColor(Green);

**end**;

**END**.

# Приложение 7. Исходный код модуля Menu

**unit** Menu;

**interface**

**uses** Graph, TypeAns;

**const** COUNT\_TABS = 8;

**type**

tab = object

n, index : integer;

isOpen, isGraph : boolean;

ans : answers;

tabs : **array** [0..10] **of** string;

**constructor** init2(name : string; ans0 : answers);

**constructor** init1(name: string);

**procedure** setNext(name : string);

**procedure** draw(i, width, indexTab : integer);

**procedure** open;

**procedure** close;

**end**;

arrTabs = **array** [1..COUNT\_TABS] **of** tab;

**IMPLEMENTATION**

**const**

height = 10;

**constructor** tab.init1(name : string);

**begin**

tabs[0] := name;

n := 1;

isGraph := false;

**end**;

**constructor** tab.init2(name : string; ans0 : answers);

**begin**

tabs[0] := name;

isGraph := true;

ans := ans0;

n := 1;

**end**;

**procedure** tab.setNext(name : string);

**begin**

tabs[n] := name;

n := n + 1;

**end**;

**procedure** tab.open;

**begin**

isOpen := true;

**end**;

**procedure** tab.close;

**begin**

isOpen := false;

index := 0;

**end**;

**procedure** tab.draw(i, width, indexTab : integer);

**var** j : integer;

**begin**

**for** j := 0 **to** n - 1 **do begin**

**if** (indexTab = i) **and** (index = j) **then begin**

setFillStyle(SolidFill, Green);

Bar((i - 1) \* width, j \* height, i \* width, (j + 1) \* height);

**end else begin**

setFillStyle(SolidFill, Blue);

Bar((i - 1) \* width, j \* height, i \* width, (j + 1) \* height);

**end**;

setColor(Yellow);

outTextXY((i - 1) \* width, j \* height + 1, tabs[j]);

**if not**(isOpen) **then break**;

**end**;

**end**;

**END**.

# Приложение 8. Исходный код модуля Tools

**unit** Tools;

**INTERFACE**

**uses** Graph;

**procedure** drawSettings(x : integer);

**procedure** setLeft(left0 : real);

**procedure** setKMin(kMin0 : integer);

**procedure** setCountAnswers(countAnswers0 : integer);

**procedure** setStep(step0 : real);

**procedure** setSpeed1(speed : real);

**procedure** setSpeed3(speed : real);

**procedure** setSpeedIsp(speed : real);

**function** getKMin : integer;

**function** getCountAnswers : integer;

**function** getStep : real;

**function** getSpeed1 : real;

**function** getSpeed3 : real;

**function** getSpeedIsp: real;

**function** getLeft : real;

**procedure** setDefaultSettigns;

**procedure** increase(index: integer);

**procedure** decrease(index: integer);

**IMPLEMENTATION**

**var** KMin, countAnswers : integer;

step, speed1, speed3, speedIsp, left : real;

**procedure** setDefaultSettigns;

**begin**

setKMin(1000);

setCountAnswers(11);

setStep(0.8);

setSpeed1(2.0);

setSpeed3(1.0);

setSpeedIsp(5.0);

setLeft(1.0);

**end**;

**procedure** drawSettings(x : integer);

**var** s: string;

**begin**

str(KMin, s);

outTextXY(x, 50, 'KMin = ' + s);

str(countAnswers, s);

outTextXY(x, 70, 'Amount answers = ' + s);

str(step:0:2, s);

outTextXY(x, 90, 'Step = ' + s);

str(speed1:0:2, s);

outTextXY(x, 110, 'Speed1 = ' + s);

str(speed3:0:2, s);

outTextXY(x, 130, 'Speed3 = ' + s);

str(speedIsp:0:2, s);

outTextXY(x, 150, 'SpeedIsp = ' + s);

str(left:0:2, s);

outTextXY(x, 170, 'Min = ' + s);

**end**;

**procedure** setKMin(kMin0 : integer);

**begin**

**if** (kMin0 >= 500) **and** (kMin0 <= 15000) **then**

kMin := kMin0;

**end**;

**procedure** setCountAnswers(countAnswers0 : integer);

**begin**

**if**(countAnswers0 **in**[10..50]) **then**

countAnswers := countAnswers0;

**end**;

**procedure** setStep(step0 : real);

**begin**

**if**(step0 >= 0.1) **and** (step0 <= 5.1) **then begin**

step := step0;

**end**;

**end**;

**procedure** setSpeed1(speed : real);

**begin**

**if**(speed > 0.05) **and** (speed <= 5.0) **then begin**

speed1 := speed;

**end**;

**end**;

**procedure** setSpeed3(speed : real);

**begin**

**if**(speed > 0.05) **and** (speed <= 5.0) **then begin**

speed3 := speed;

**end**;

**end**;

**procedure** setSpeedIsp(speed : real);

**begin**

**if**(speed > 0.05) **and** (speed <= 10.0) **then begin**

speedIsp := speed;

**end**;

**end**;

**procedure** setLeft(left0 : real);

**begin**

**if**(left0 > 0.05) **then**

left := left0;

**end**;

**function** getStep : real;

**begin**

getStep := step;

**end**;

**function** getKMin : integer;

**begin**

getKMin := kMin;

**end**;

**function** getCountAnswers : integer;

**begin**

getCountAnswers := countAnswers;

**end**;

**function** getSpeed1 : real;

**begin**

getSpeed1 := speed1;

**end**;

**function** getSpeed3 : real;

**begin**

getSpeed3 := speed3;

**end**;

**function** getSpeedIsp : real;

**begin**

getSpeedIsp := speedIsp;

**end**;

**function** getLeft : real;

**begin**

getLeft := left;

**end**;

**procedure** increase(index: integer);

**begin**

**case** index **of**

1: setKMin(kMin + 100);

2: setCountAnswers(countAnswers + 1);

3: setStep(step + 0.1);

4: setSpeed1(speed1 + 0.1);

5: setSpeed3(speed3 + 0.1);

6: setSpeedIsp(speedIsp + 0.1);

7: setLeft(left + 0.1);

**end**;

**end**;

**procedure** decrease(index: integer);

**begin**

**case** index **of**

1: setKMin(kMin - 100);

2: setCountAnswers(countAnswers - 1);

3: setStep(step - 0.1);

4: setSpeed1(speed1 - 0.1);

5: setSpeed3(speed3 - 0.1);

6: setSpeedIsp(speedIsp - 0.1);

7: setLeft(left - 0.1);

**end**;

**end**;

**END**.

# Приложение 9. Исходный код модуля TypeAns

**unit** TypeAns;

**INTERFACE**

**const** COUNT\_ANSWERS = 20;

**type** answers = **array** [1..50] **of** real;

**function** getMax(a : answers):real;

**function** getMin(a : answers) : real;

**IMPLEMENTATION**

**function** getMax(a : answers) : real;

**var** i, maxIndex : integer;

**begin**

maxIndex := 1;

**for** i := 2 **to** COUNT\_ANSWERS **do begin**

**if** (a[maxIndex] < a[i] ) **then**

maxIndex := i;

**end**;

getMax := a[maxIndex];

**end**;

**function** getMin(a : answers) : real;

**var** i, minIndex : integer;

**begin**

minIndex := 1;

**for** i := 2 **to** COUNT\_ANSWERS **do begin**

**if** (a[minIndex] > a[i] ) **then**

minIndex := i;

**end**;

getMin := a[minIndex];

**end**;

**END**.