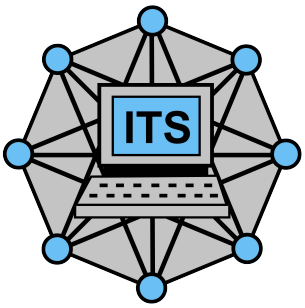
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ



Михальов О.І., Бабенко Ю.В., Царик В.Ю.

РОБОЧА ПРОГРАМА,
методичні вказівки та індивідуальні завдання до вивчення дисципліни «Основи наукових досліджень інформаційних комп’ютерних технологій (ІКТ)»

для студентів спеціальності 122 «Комп'ютерні науки»

Дніпро НМетАУ 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

Михальов О.І., Бабенко Ю.В., Царик В.Ю.

РОБОЧА ПРОГРАМА,
методичні вказівки та індивідуальні завдання до вивчення дисципліни «Основи наукових досліджень інформаційних комп'ютерних технологій (ІКТ)» для студентів спеціальності 122 «Комп'ютерні науки» заочної форми навчання

Дніпро НМетАУ 2019

УДК 621.65

Робоча програма, методичні вказівки та індивідуальні завдання до вивчення дисципліни «Основи наукових досліджень інформаційних комп’ютерних технологій (ІКТ)» для студентів спеціальності 122 «Комп'ютерні науки» заочної форми навчання / Укл. О.І Михальов, Ю.В. Бабенко, В.Ю. Царик . – Дніпро: НМетАУ, 2019. – 31 с.

Наведена робоча програма дисципліни «Основи наукових досліджень інформаційних комп’ютерних технологій (ІКТ)» з коротким викладом змісту її розділів і методичними вказівками до вивчення навчального матеріалу, а також завдання до контрольної роботи, на основі якої студент засвоює принципи наукових досліджень систем різної природи, аналізу отриманих результатів моделювання інформаційних систем та розширює знання програмних пакетів Matlab, Maple та Anylogic.

Призначена для студентів спеціальності 122 «Комп'ютерні науки» заочної форми навчання.

Укладачі: О.І. Михальов, д-р. техн. наук, професор

Ю. В. Бабенко, канд. техн. наук
В. Ю. Царик, аспірант

Відповідальний за випуск О.І. Михальов, д-р техн. наук, професор

Методичні вказівки розглянуто та ухвалено на засіданні кафедри інформаційних технологій і систем, протокол № 9 від 06.03.2019.

Методичні вказівки розглянуто та ухвалено на засіданні навчально-методичної комісії НМетАУ зі спеціальності 122 Комп’ютерні науки, протокол № 7/18-19 від 26.03.2019.

Друкується за авторською редакцією.

Підписано до друку 27.03.2019. Формат 60х841/16. Папір типогр. Друк різограф.
Облік.-вид. арк. 0,75. Умов. друк. арк. 0,857.
Тираж 100 пр. Замовл. № 10/19.

Національна металургійна академія України.
49600, м. Дніпро, пр. Гагаріна, 4

_____

Редакційно-видавничий відділ НМетАУ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Учебная дисциплина «Основы научных исследований ИКТ» посвящена вопросам организации научно-исследовательской работы и ее этапам, методологии научных исследований и оформлении научных результатов, основам научной этики, а также рекомендациям по подготовке, написанию и оформлению научно-исследовательских работ и методических указаний по подготовке и оформлению научных докладов, курсовых и дипломных работ и индивидуальных заданий.

Цель изучения дисциплины – изучение методологии и методов научных исследований, а также способов их организации.

В результате изучения теоретического курса и выполнения заданий по лабораторным работам студент **должен**:

- освоить методологию и методику научных исследований,
- уметь формулировать цель и задачи исследования,
- планировать и проводить эксперимент,
- обрабатывать результаты измерений,
- сопоставлять результаты эксперимента с теоретическими моделями
- и формулировать выводы научного исследования,
- составлять реферат, доклад, курсовую работу или статью по результатам научного исследования.

Сегодня согласно действующего государственного образовательного стандарта «Основы научных исследований ИКТ» изучается как самостоятельная дисциплина студентами специальности 122 «Компьютерные науки». Данная дисциплина состоит из таких основных тем:

- обзор методологий и методов научных исследований, а также способов их организации;
- использование языка программирования Matlab для моделирования работы системы массового обслуживания;
- использование пакета Maple для решения различных математических задач с использованием встроенных графических редакторов;
- использование пакета Anylogic в задачах моделирования работы сложных информационных систем.

1 ОБЩИЕ СВЕДЕНИЯ

Учебная дисциплина «Основы научных исследований ИКТ» является обязательной и входит в цикл профессиональных дисциплин по специальности.

В результате изучения дисциплины студент **получит:**

– *знания и понимание* методов научных исследований информационных систем и процессов; основ научной этики; этапов научно-исследовательской работы; методики оформления результатов исследований в виде научных работ;

– *применение знаний:* владение методами составления научной документации (тезисов, статей, рефератов и т.п.); владение методами использования стандартного программного и аппаратного обеспечения ИКТ; владение методами планирования, разработки и тестирования сложных информационных систем;

– *формирование суждений*: способность использовать существующие методологии научных исследований для проведения исследований информационных систем разной природы; способность использовать существующее программное обеспечение по математическому моделированию технологических систем и обработки экспериментальных данных информационной среды.

Для успешной сдачи экзамена по дисциплине необходимо: проработать вопросы контрольной работы, подготовиться к экзамену по вопросам дисциплины.

Связь с другими дисциплинами: навыки и умения этой дисциплины базируются на дисциплинах «Технология создания программных продуктов», «Системный анализ», «Моделирование систем».

Приобретенные знания и умения обеспечивают необходимую подготовку студентов для выполнения дипломной работы, написания исследовательских публикаций.

2 РАБОЧАЯ ПРОГРАММА И МЕТОДИЧЕСКИЕ УКАЗАНИЯ

2.1. Рабочая программа

Распределение учебных часов и формы контроля дисциплины «Основы научных исследований ИКТ» для студентов заочной формы обучения специальности 122 «Компьютерные науки» представлена в таблице 2.1.

Таблица 2.1– Распределение учебных часов и формы контроля

Всего часов согласно учебному плану	60
в том числе: Аудиторные занятия	14
из них: – лекции	8
– лабораторные занятия	6
– практические занятия	
– семинары	
Самостоятельная работа	46
– выполнение контрольной работы	
– подготовка к экзамену	
– изучение отдельных разделов, которые не вошли в лекционный курс	
Итоговый контроль (экзамен, зачет)	Диф. зачет

2.2. Методические указания

Лабораторная работа № 1

MATLAB (сокращение от англ. *«Matrix Laboratory*) — пакет прикладных программ для решения задач технических вычислений и одноимённый язык программирования, используемый в этом пакете. Пакет используют более миллиона инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, Mac OS, и Microsoft Windows.

Язык MATLAB является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования.

Система MATLAB - это одновременно и операционная среда и язык программирования. Одна из наиболее сильных сторон системы состоит в том, что на языке MATLAB могут быть написаны программы для многократного использования. Пользователь может сам написать специализированные функции и программы, которые оформляются в виде М-файлов. По мере увеличения количества созданных программ возникают проблемы их классификации и тогда можно попытаться собрать родственные функции в специальные папки. Это приводит к концепции пакетов прикладных программ (ППП), которые представляют собой коллекции М-файлов для решения определенной задачи или проблемы.

Часть 1. Моделирование системы массового обслуживания по времени

Для примера рассмотрим решение задачи системы массового обслуживания (СМО):

В магазине самообслуживания установлено, что поток покупателей является простейшим с интенсивностью 2 покупателя в минуту. В магазине установлен один кассовый аппарат, позволяющий добиться интенсивности потока обслуживания 2 покупателя в минуту. Определить характеристики СМО при условии, что очередь ограничена пятью покупателями при входе в зал самообслуживания.

При использовании метода с постоянным шагом отсчет системного времени ведется через фиксированные, выбранные исследователем интервалы времени. События в модели считаются наступившими в момент окончания этого интервала. Тогда, чтобы смоделировать простейший поток покупателей и обслуживания, надо воспользоваться пуассоновским распределением, которое моделирует количество событий в единицу времени (за единицу времени возьмем 1 мин).

Ход работы

1. Построение модели

1.1. Соберите модель в Matlab Simulink, взяв за образец схему на рис. 1.1.

1.2. Задайте в блоке *MATLAB Fcn* функцию для моделирования потока покупателей с интенсивностью 2 чел/мин – это будет функция *poissrnd(2).* Поскольку касса работает с такой же интенсивностью, то во втором блоке *MATLAB Fcn* тоже поставьте такую же функцию *poissrnd(2).*

1.3. Из количества пришедших покупателей вычтем количество обслуженных кассиром в блоке *Sum*, оставшихся будем накапливать в блоке *Discrete-Time Integrator*. Но, чтобы не накапливать «отрицательную» очередь, настройте блок *Discrete-Time Integrator* так, как показано на рис. 1.2. Поставьте флажок в окошке «Ограничение выходных сигналов», потом поставьте *0* в

параметре «Нижнее предельное значение». Все остальные блоки *Discrete-Time Integrator* просто суммируют входные сигналы.

1.4. Чтобы организовать ограничение по очереди, работают несколько блоков. Блок *Relational Operator* сравнивает размер очереди с ограничением в 5 покупателей, и, если очередь больше 5, то отправляет управляющий сигнал на блок *Switch – Переключатель*, который ограничивает вход покупателей, не пропуская сигнал с блока *MATLAB Fcn*, а пропуская *0*. Настройте блок *Switch*, установив в нем пороговое значение *1*.

1.5. В блоках *Fcn* и *Fcn1* введите указанные функции пользователя.

1.6. Запустите на выполнение построенную модель для проверки, указав время 100.

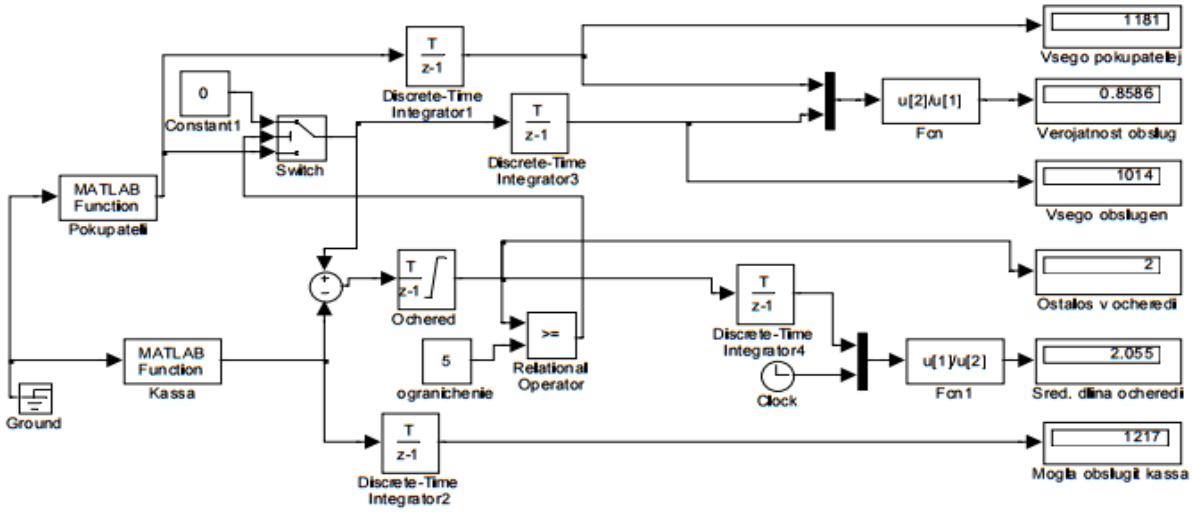


Рисунок 1.1. Схема моделирования СМО

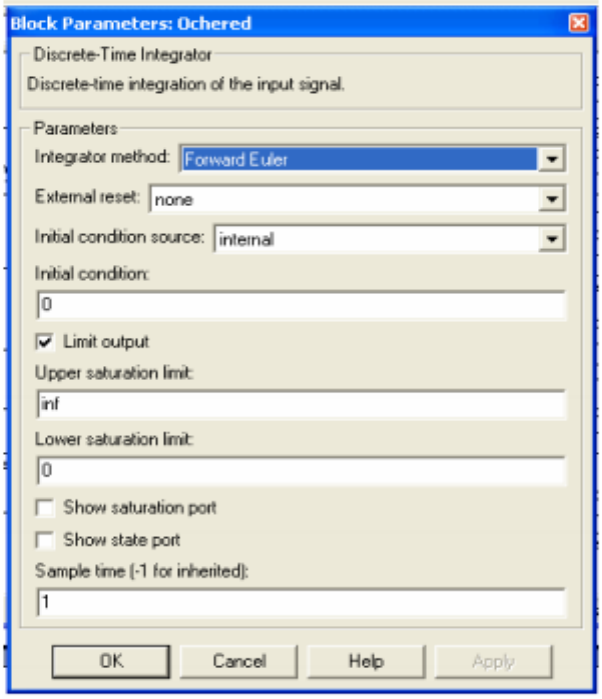


Рисунок 1.2. Настройка блока *Discrete-Time Integrator*

2. Решение собственной задачи

2.1. Соберите модель в Matlab Simulink, взяв за образец схему на рис. 1.1. При этом СМО будет иметь следующие параметры:

1. поток покупателей с интенсивностью *(N+2)* чел/мин;

2. размер очереди с ограничением в (*N+5)* покупателей для вариантов 1-10, для вариантов 11-30 – (*N-5)* покупателей,

где *N* – номер варианта.

2.2. Для каждого значения времени выполните 6 прогонов работы вашей модели, сделайте выводы.

2.3. Проработайте задачу согласно номеру своего варианта для разного времени работы модели: 600, 720, 1060 минут.

2.4. Построить графики параметров «Осталось в очереди» и «Средняя длина в очереди» при разном времени работы модели.

2.5. Сделайте выводы по результатам.

Часть 2. Моделирование системы массового обслуживания по особым состояниям

При моделировании по особым состояниям системное время каждый раз изменяется на величину, строго соответствующую интервалу времени до момента наступления очередного события. Тогда, чтобы смоделировать простейший поток покупателей и обслуживания, надо воспользоваться экспоненциальным распределением, которое моделирует временной промежуток между наступившими событиями (за событие возьмем приход покупателя).

Ход работы

1. Построение модели

1.1. Соберите модель, взяв за образец схему на рис. 1.3.

1.2. Задайте в блоке *MATLAB Fcn* функцию для моделирования потока покупателей с интервалом 1/2 мин – это будет функция *exprnd(1/2)*. Поскольку касса работает с такой же регулярностью, то во втором блоке *MATLAB Fcn* поставьте такую же функцию *exprnd(1/2).*

1.3. Теперь в блоке *Sum* происходит вычитание из времени прихода покупателя времени работы кассира. Чтобы посчитать длину очереди, будем складывать не остаток времени, а 1 со знаком + или – (блок *Sign*). Настройте блок *Discrete-TimeIntegrator* для очереди так же, как на рис. 1.2).

1.4 Настройте блок *Switch*, установив в нем пороговое значение 1. Теперь блок *Switch* пропускает или время прихода покупателя или, если очередь

ограничена, то 0. Накапливаем случаи, когда пропускается ненулевое значение – это количество обслуженных покупателей.

1.5. В блоках *Fcn* и *Fcn1* введите указанные функции пользователя. Имейте в виду, что блок *Clock* теперь показывает не время, а количество пришедших покупателей.

1.6. Настройте данную модель: шаг – фиксированный размером 1, логические сигналы – выключить.

1.7. Запустите на выполнение построенную модель для проверки, указав время 100, что означает 100 покупателей.

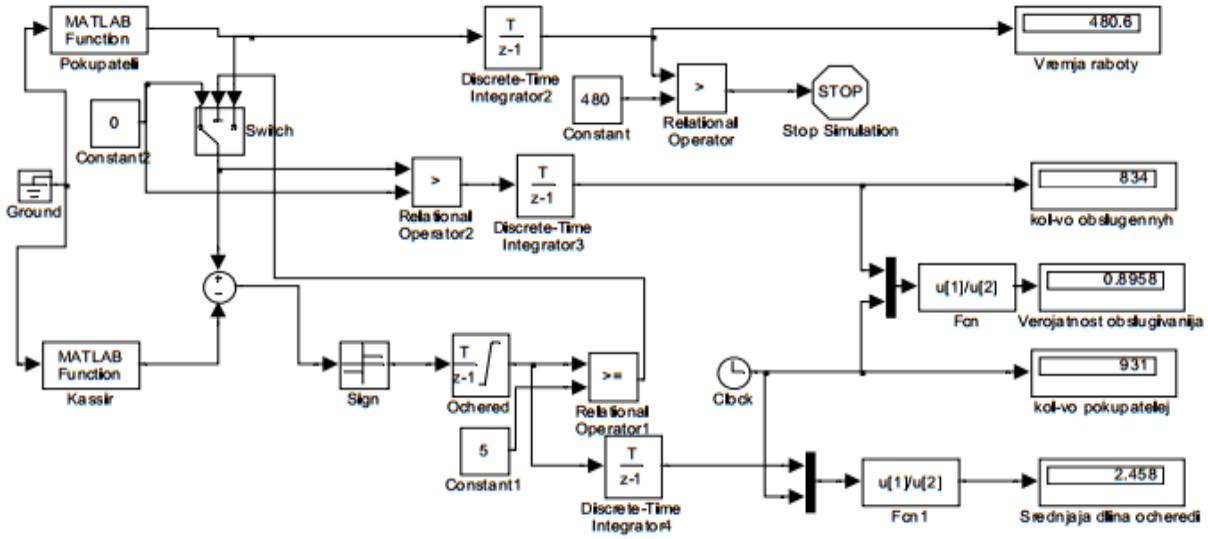


Рисунок 1.3. Схема моделирования СМО по особым состояниям

2. Решение собственной задачи

2.1. Соберите модель в Matlab Simulink, взяв за образец схему на рис. 1.3. При этом СМО будет иметь следующие параметры:

1. поток покупателей с интервалом *1/(N+2)* мин для вариантов 1-10, для вариантов 11-30 – 1/(*N-9)* мин;

2. регулярность работы кассы аналогична потоку покупателей,

где *N* – номер варианта.

2.2. Проработайте задачу согласно номеру своего варианта для разного времени работы модели: 600, 720, 1060 минут. Поскольку моделирование происходит по особым состояниям, и время мы получаем экспериментально, ограничение по времени мы можем задать только в модели явно. Для этого накапливаем время прихода покупателей, сравниваем с заданным количеством минут, и останавливаем моделирование (блок *Stop Simulation*). Чтобы не переделывать настройки параметров модели при каждом изменении времени, задайте в параметрах настройки остановку по времени очень большую, например

5000, – это количество покупателей, которое не должно играть роли, потому что остановка модели произойдет раньше.

2.3. Построить графики параметра «Средняя длина очереди» при разном времени работы модели.

2.4. Сделайте выводы по результатам.

Лабораторная работа № 2

Maple — система компьютерной математики, разработанная канадской фирмой Waterloo Maple, которая ориентирована на широкий круг пользователей: от школьников до преподавателей вузов и научных работников. Она известна благодаря своей высокой эффективности при решении самых разных математических задач и великолепной графике.

Пример 2.1. Загрязнение реки

В результате аварийного сброса промышленных стоков концентрация c_0 вредных веществ в реке сильно увеличилась. Через сутки после аварии уровень загрязнения снизился и стал равным $c_1 = kc_0$, где k — коэффициент естественного самоочищения, $0 < k < 1$. Предположим, что дальнейшее самоочищение реки будет проходить так же, как и в первые сутки. Тогда через двое суток $c_2 = kc_1 = k^2 c_0$. Каким будет уровень загрязнения реки вредными веществами (см. таблицу) через сутки, двое и в последующие дни, до тех пор, пока их концентрация не станет меньше предельно допустимой d ? Провести исследование для веществ, указанных в следующей таблице.

Таблица 2.1 Вредные вещества, попавшие в реку

Вещество	c_0	d	k
Свинец	2.5	0.03	0.83
Мышьяк	1.5	0.05	0.95
Фтор	0.2	0.05	0.98

Ограничимся реализацией модели для свинца. Определим исходные данные:

```
> restart:
> НачальнаяКонцентрация:= 2.5: ПредельнаяКонцентрация:= 0.03:
  КоэффициентУменьшения:= 0.83:
```

Построим таблицу, которая отражает уровень загрязнения реки по суткам. Таблица формируется в виде списка:

```
> УровеньЗагрязнения:= НачальнаяКонцентрация:
  Свинец:= НачальнаяКонцентрация:
  Сутки:= 1:
```

```
> while Свинец > ПредельнаяКонцентрация do
    Сутки:= Сутки+1:
    Свинец:= КоэффициентУменьшения*Свинец:
    УровеньЗагрязнения:= УровеньЗагрязнения,Свинец:
  end do:
```

```
> Сутки;
                                  25
```

Таким образом, загрязнение реки свинцом снизится до допустимых пределов через 25 суток. Ход изменения концентрации свинца в течение этого времени:

```
> Таблица:= [УровеньЗагрязнения]:
```

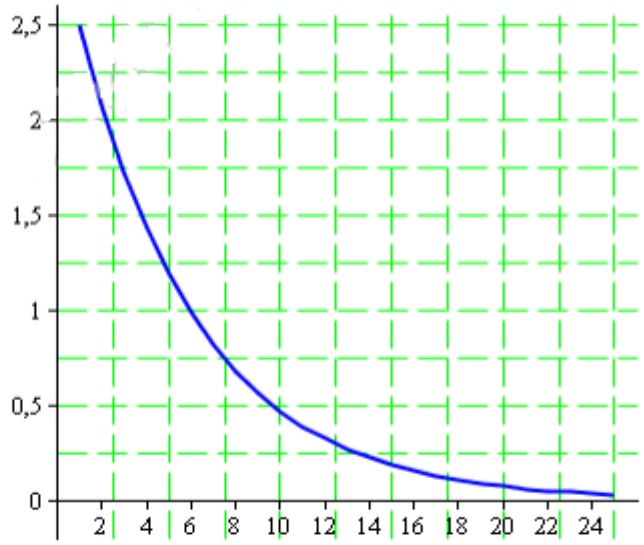
```
> evalf(Таблица[1..12],2); evalf(Таблица[13..25],2);

                [2.5, 2.1, 1.7, 1.4, 1.2, 0.98, 0.82, 0.68, 0.56, 0.47, 0.39, 0.32]
    [0.27, 0.22, 0.18, 0.15, 0.13, 0.11, 0.087, 0.073, 0.060, 0.050, 0.041, 0.034, 0.029]
```

Для большей наглядности полученных результатов покажем их на графике изменения концентрации свинца от времени:

```
> with(plots): with(plottools):
  gr1:= listplot(Таблица,color=blue,thickness=2):
```

```
> display([gr1],view=[0..Сутки+1,-0.2..2.6],
    xtickmarks=[seq(0..Сутки,2)],ytickmarks=[seq(0..3,0.5
    axis=[gridlines=[10,color=green,linestyle=3]]);
```



Экологические модели позволяют прогнозировать изменения состояния различных экосистем в течение некоторого времени и в случае необходимости принимать меры по улучшению природопользования и охраны живых организмов.

Пример 2.2. Модель эпидемии

Пусть в городке живет N человек. В начальный период эпидемии инфицировано b_0 человек. Остальные $z_0 = N - b_0$ человек здоровы. Будем считать, что все, перенесшие заболевание, приобретают иммунитет к нему и больше не болеют. Поэтому число заболевших в первый день пропорционально произведению числа больных на число еще не болевших: $z_1 = pb_0\, z_0$, где p — коэффициент пропорциональности, зависящий от различных мер профилактики. Обозначим через b_i — число человек, заболевших в течении *i-го* дня, через z_i — число человек еще не болевших к концу этого дня. Тогда математическую модель эпидемии можно описать двумя уравнениями:

$$b_i = pb_{i-1}z_{i-1},$$
$$z_i = z_{i-1} - b_i\,.$$

Спрашивается, как развивается эпидемия, т.е. как ежедневно изменяется число b больных и число z не болевших? Пусть в городке с населением $N = 20\,000$ человек первоначально заболели 50, а коэффициент $p = 0{,}0001$.

Определим исходные данные:

```
> n:= 13: p:= 0.0001: N:= 20000:
  B:= array(0..n): Z:= array(0..n):
  B[0]:= 50.0: Z[0]:= N-50:
```

Развитие эпидемии:

```
> for i from 1 to n do
    B[i]:= p*B[i-1]*Z[i-1]:
    Z[i]:= Z[i-1]-B[i]:
  end do:
```

```
> Больные:= seq(trunc(B[i]),i=1..n):
  НеБолевшие:= seq(trunc(Z[i]),i=1..n):
  [Больные[1..n]]; [НеБолевшие[1..n]];

                [99, 198, 389, 749, 1387, 2376, 3505, 3941, 2878, 1273, 401, 110, 29]
        [19850, 19652, 19263, 18513, 17125, 14749, 11243, 7302, 4424, 3150, 2749, 2639, 2610]
```

При расчете количества больных и неболевших округляли получающиеся числа до целых значений. Построим графики изменения количества больных и неболевших по дням:

```
> with(plots):
  gr1:= listplot([Больные],color=red):
  gr2:= listplot([ЕщеЗдоровые],color=blue):
```

```
> display([gr1,gr2],view=[1..n,0..N],
    thickness=2,axis=[gridlines=[15,color=green]]);
```

Нижняя кривая показывает число больных, а верхняя — оставшихся здоровыми. Как видно из рисунка, на тринадцатый день наблюдается спад эпидемии — только 29 заболевших:

```
> Больные[13], НеБолевшие[13];

                                  29, 2610
```

Критическая точка — восьмой день — 3941 заболевший:

```
> Больные[8], НеБолевшие[8];

                                  3941, 7302
```

Чтобы определить всех, перенесших заболевание, нужно просуммировать число больных по дням:

```
> trunc(sum(B[j],j=0..n));
                                  17389
```

Прогнозирование развития эпидемии позволит лучше подготовиться к ней и принять необходимые профилактические меры.

Задания к лабораторной работе № 2

1. Реализовать Примеры 2.1 и 2.2.
2. Реализовать модель загрязнения реки для всех вредных веществ (см. таблицу примера 1). Показать все графики на одном рисунке. Сделать подписи осей графика.

3. Построить модель эпидемии, учитывающую, что на распространение болезни влияют все заболевшие, а не только те, кто заболели в последний день. При этом параметры задачи:

- население N = (Z^3+20 000) человек
- первоначально заболели Z+50,
- коэффициент p = 0,0001
- Z – номер варианта.

Лабораторная работа № 3

AnyLogic — программное обеспечение для имитационного моделирования, разработанное компанией The AnyLogic Company. Инструмент обладает современным графическим интерфейсом и позволяет использовать язык Java для разработки моделей. Продукт получил название AnyLogic, потому что он поддерживал все три известных метода моделирования:

- системная динамика;
- дискретно-событийное (процессное) моделирование;
- агентное моделирование.

AnyLogic является кросс-платформенным программным обеспечением, работает как под управлением операционной системы Windows, так и под Mac OS и Linux.

Пример 3.1. Модель пешеходного перехода

Необходимо построить модель регулируемого пешеходного перехода со светофором, разрешающим или запрещающим движение транспорта.

Рассмотрим состояния, в которых может находиться светофор:

1. движение транспорта разрешено (зеленый),
2. приготовиться к запрещающему сигналу (мигающий зеленый),
3. приготовиться к остановке (желтый),
4. движение запрещено (красный)
5. приготовиться к движению (красный и желтый).

Светофор работает в автоматическом режиме. В каждом состоянии светофор находится определенный постоянный период времени.

1. Постороение модели в AnyLogic

Создайте новый проект под названием ***Svetofor*** и назовите класс корневого активного объекта ***Model***.

Наша модель будет иметь только один активный объект, представляющий светофор, поэтому корневой объект ***Model*** будет единственным активным объектом нашей модели. На диаграмму класса активного объекта ***Model*** поместите **Начало диаграммы состояний** из панели **Диаграмма состояний** и назовите ее ***Для_автомобилей***. Перетащите мышью элемент **Состояние** под стрелочку начала диаграммы (рис. 3.1).

Рисунок 3.1

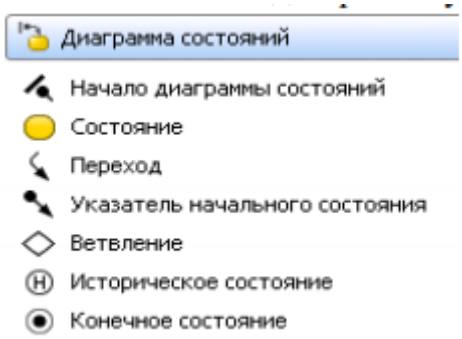


Рисунок 3.2

Имя состояния, как и все другие его параметры, можно редактировать в окне его свойств.

Для того чтобы построить стейтчарт, следует использовать элементы из палитры **Диаграмма состояний**, рис. 3.2:

1. Элемент «**Начало диаграммы состояний**» определяет начальное состояние всего стейтчарта,

2. с помощью кнопки «**Состояние**» рисуются состояния (как простые, так и гиперсостояния),

3. Элемент «**Переход**» используется для рисования переходов между состояниями,

4. Элемент «**Указатель начального состояния**» определяет начальное состояние внутри сложного состояния,

5. Элемент «**Конечное состояние** » используется для рисования состояния, являющегося "финальным" в поведении активного объекта.

Для любого выделенного объекта внизу появляется панель его свойств, в котором можно изменить параметры и, в частности, имя объекта, если это необходимо. Структурные ошибки при рисовании стейтчарта - повисшие переходы, дублированные указатели начального состояния и т.п. - выделяются в панели **Проекты** значком X красного цвета и записью в панели **Ошибки**. Выделенные переходы должны иметь на концах зеленые точки, если точки белые, это значит, что переход не соединен с состоянием – висит.

В соответствии с алгоритмом работы светофора помимо начального состояния в модель нужно ввести дополнительные состояния (рис. 3.3). Начальное

состояние назовите ***движение*** – движение автомобилям разрешено (зеленый свет), затем светофор переходит в состояние ***внимание*** – внимание (мигающий зеленый), ***медленно*** – приготовиться к остановке (желтый свет), остановка транспорта ***stop*** – запрет движения (красный свет) и ***приготовиться*** – приготовиться к движению (красный и желтый свет горят одновременно).

Состояние ***внимание*** представим гиперсостоянием с парой переключающихся элементарных состояний: в одном из них зеленый горит (состояние ***A***), в другом – нет (состояние ***B***). Для построения гиперсостояния сначала создайте обычное состояние, увеличьте его (растянув мышью) и поместите внутрь другое состояние. Постройте эти состояния и соедините их переходами, как показано на рис. 3.3. Зададим условия срабатывания переходов. Переходы в нашем автоматическом светофоре выполняются по таймауту, т. е. по истечении интервала времени, который прошел с момента прихода системы в данное состояние.

Рисунок 3.3

1. В состоянии ***движение*** светофор находится 10 секунд,
2. затем 7 секунд зеленый сигнал мигает,
3. в состоянии ***медленно*** 4 секунды горит желтый,
4. в течение 10 секунд движение запрещено и
5. 4 секунды светофор находится в состоянии ***приготовиться***.

В нашей модели единица модельного времени соответствует 1 секунде реального времени.

Для задания условий срабатывания переходов, выделите переход ***t1***, и в поле **Происходит** оставьте без изменения вариант **По таймауту**, а в

поле **По таймауту** введите 10 (рис. 3.4). Аналогично задайте условия срабатывания других переходов.

Между состояниями ***A*** и ***B*** переходы должны срабатывать через 1 секунду. Запустите модель. Активное в данный момент состояние подсвечивается красным. Переход, ожидающий истечения таймаута подсвечивается синим.

В каждый момент светофора должен гореть определенный сигнал: в состоянии ***движение*** должен гореть зеленый, в состоянии ***приготовиться*** должны гореть красный и желтый одновременно и т. п. Перейдите на диаграмму класса активного объекта ***Model***. Создайте три параметра логического типа: ***красный, желтый*** и ***зеленый***, которые будут принимать истинное значение тогда, когда у светофора горит соответствующий сигнал: красный, желтый или зеленый (рис. 3.5). Начальные значения этих булевых параметров можно не задавать: по умолчанию они будут равны ***false***.

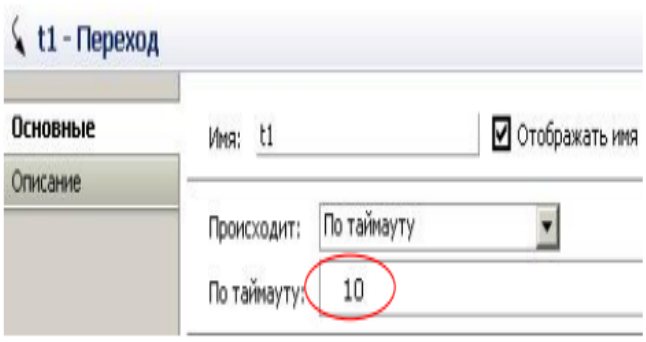


Рисунок 3.4

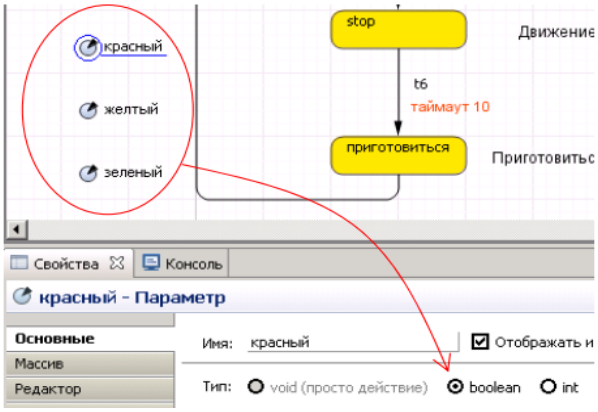


Рисунок 3.5

Мы создали для управления значениями этих параметров, каждое состояние отвечает за зажигание своего света или их комбинации. Например, в состоянии ***медленно*** должен гореть желтый, в состоянии ***stop*** должен загореться красный свет, а в состоянии ***приготовиться*** должны гореть красный и желтый одновременно. Запрограммируем эти действия.

1. В свойствах состояния ***движение*** в поле **Действие при входе** запишите ***зеленый=true;,*** а в поле **Действие при выходе** запишите ***зеленый = false***; (рис. 3.6).

2. То же самое запишите для состояния ***B*** гиперсостояния ***внимание***, а у состояния ***A*** эти поля нужно оставить пустыми – когда светофор находится в этом состоянии, он не горит.

3. Аналогично в состоянии ***медленно*** нужно включить желтый сигнал, т. е. при входе в это состояние установить параметр желтый в ***true***, а при выходе из этого состояния установить его в ***false.***

4. Для состояния ***stop*** опишите состояния параметра ***красный.***

5. Для состояния ***приготовиться*** оба параметра ***красный*** и ***желтый*** нужно установить в ***true*** при входе, в ***false*** при выходе.

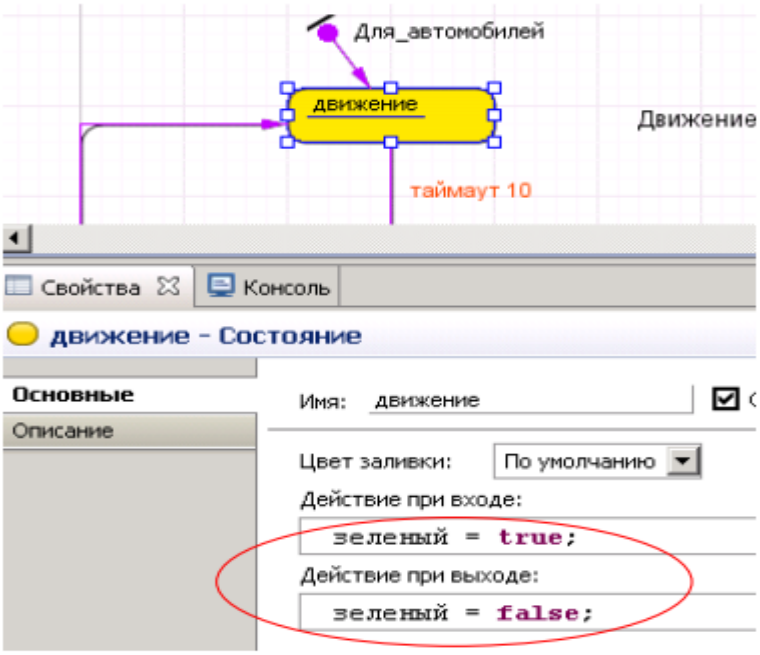


Рисунок 3.6

Запустите модель на выполнение. Вы увидите, что параметры ***зеленый, желтый*** и ***красный*** будут переключаться между значениями ***истина*** и ***ложь*** в соответствии с алгоритмом переключения светофора.

2. Презентация модели

Презентация модели рисуется в той же диаграмме, в которой задается и диаграмма моделируемого процесса, (рис. 7). Графические объекты цвета сигналов светофора в презентации имеют динамические параметры, все остальные – статические. Проезжую часть удобно нарисовать с помощью графических примитивов типа **Прямоугольник**, а светофор с помощью примитива - **Ломаная**. Сигнальные элементы светофора строятся из трех овалов, повернутых на 45 градусов (поле **Поворот** вкладки **Дополнительные** окна свойств овала).

Установим динамическое значение цвета верхнего сигнала светофора: если переменная красный истинна, то цвет должен быть ***red*** (красный), в противном случае его цвет нужно установить ***gray*** (серый). Это записывается следующим условным выражением на языке Java:

красный? red: gray.

Цвет среднего и нижнего овалов, следует установить в поле их динамических значений соответственно так:

желтый? yellow: gray

зеленый? green: gray

red, yellow, green и ***gray*** – предопределенный цвета, обозначающие стандартные цвета. Запустите модель и проверьте ее работу.

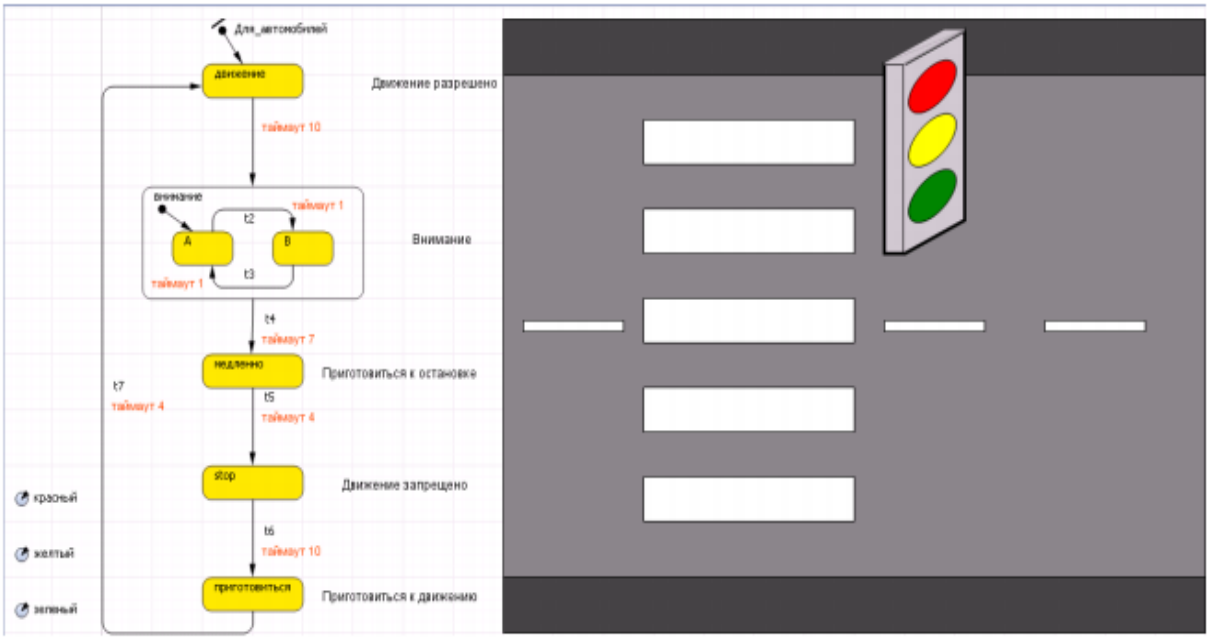


Рисунок 3.7

3. Срабатывание перехода по сигналу

Добавим к нашей модели второй светофор, для пешеходов. Он будет иметь два сигнала, зеленый и красный, и три состояния: ***идите*** (зеленый), ***внимание*** (мигающий зеленый) и ***стойте*** (красный). Добавим в модель два булевских параметра ***стойте*** и ***идите***, их значениями будет управлять второй стейтчарт – для пешеходов. Создадим этот стейтчарт на той же диаграмме класса ***Model***, назвав его ***Для_пешеходов***, (рис. 3.8).

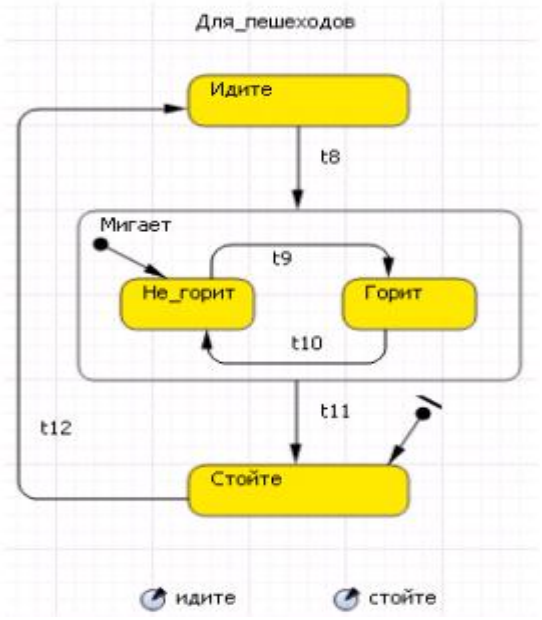


Рисунок 3.8

Поскольку управление светофором пешеходов похоже на управление светофором автомобилей, новый стейтчарт можно построить копированием и изменением уже построенного стейтчарта ***Для_автомобилей***. Выделите 3 состояния стейтчарта ***Для_автомобилей*** и вставьте их в друг место диаграммы. Эти элементы скопируются, и им автоматически будут даны уникальные имена, чтобы не было конфликта имен.

Переименуйте состояния стейтчарта ***Для_пешеходов***, дорисуйте недостающий переход ***t12*** и перенесите начало диаграммы состояние ***Стойте*** (рис. 3.8).

Измените параметры в полях **Действие при входе** и **Действие при выходе** в свойствах состояний стейтчарта ***Для_пешеходов***. Теперь наш стейтчарт должен управлять параметрами ***идите, стойте***, которые, в свою очередь, будут управлять зажиганием света именно пешеходного светофора.

Настроим условия срабатывания переходов стейтчартов между состояниями. Для обеспечения безопасной работы пешеходного перехода необходимо синхронизировать срабатывания стейтчартов так, чтобы всегда, когда светофор пешеходов находится в состояниях ***Идите*** или ***Мигает***, светофор автомобилей обязательно находился бы в состоянии ***stop***. Для этого можно подобрать подходящие таймауты срабатывания переходов стейтчарта, но при каждом изменении модели, придется эти таймауты подбирать снова и снова. Более разумно синхронизировать стейтчарты, посылая специальные разрешающие сигналы из одного стейтчарта в другой.

В нашей модели стейтчарты будут обмениваться следующими сигналами: ***АВТОМОБИЛИ*** и ***ПЕШЕХОДЫ***. В стейтчарте ***Для_пешеходов*** переход ***t12*** будет срабатывать когда получен сигнал ***ПЕШЕХОДЫ***, который будет генерироваться в стейтчарте ***Для_автомобилей*** при переходе ***t5*** в состояние ***stop***. В свою очередь, в стейтчарте ***Для_автомобилей*** переход ***t6*** будет срабатывать когда получен сигнал ***АВТОМОБИЛИ***, который генерируется в стейтчарте ***Для_пешеходов*** при переходе ***t11*** в состояние ***Стойте***, рис. 3.9.

В AnyLogic есть несколько способов передачи сообщения в диаграмму состояний. В нашей модели мы будем использовать метод ***fireEvent()***, который должен вызываться в том стейтчарте, которому предназначено сообщение. То есть, если из некоего объекта мы хотим послать сообщение стейтчарту, то нужно в этом объекте написать команду:

стейтчарт.fireEvent(сообщение)

Поэтому, в поле **Действие** перехода ***t5*** стейтчарта ***Для_автомобилей*** нужно вставить команду:

Для_пешеходов.fireEvent("ПЕШЕХОДЫ"),

в такое же поле перехода ***t11*** стейтчарта ***Для_пешеходов*** вставьте команду:

Для_автомобилей.fireEvent("АВТОМОБИЛИ")

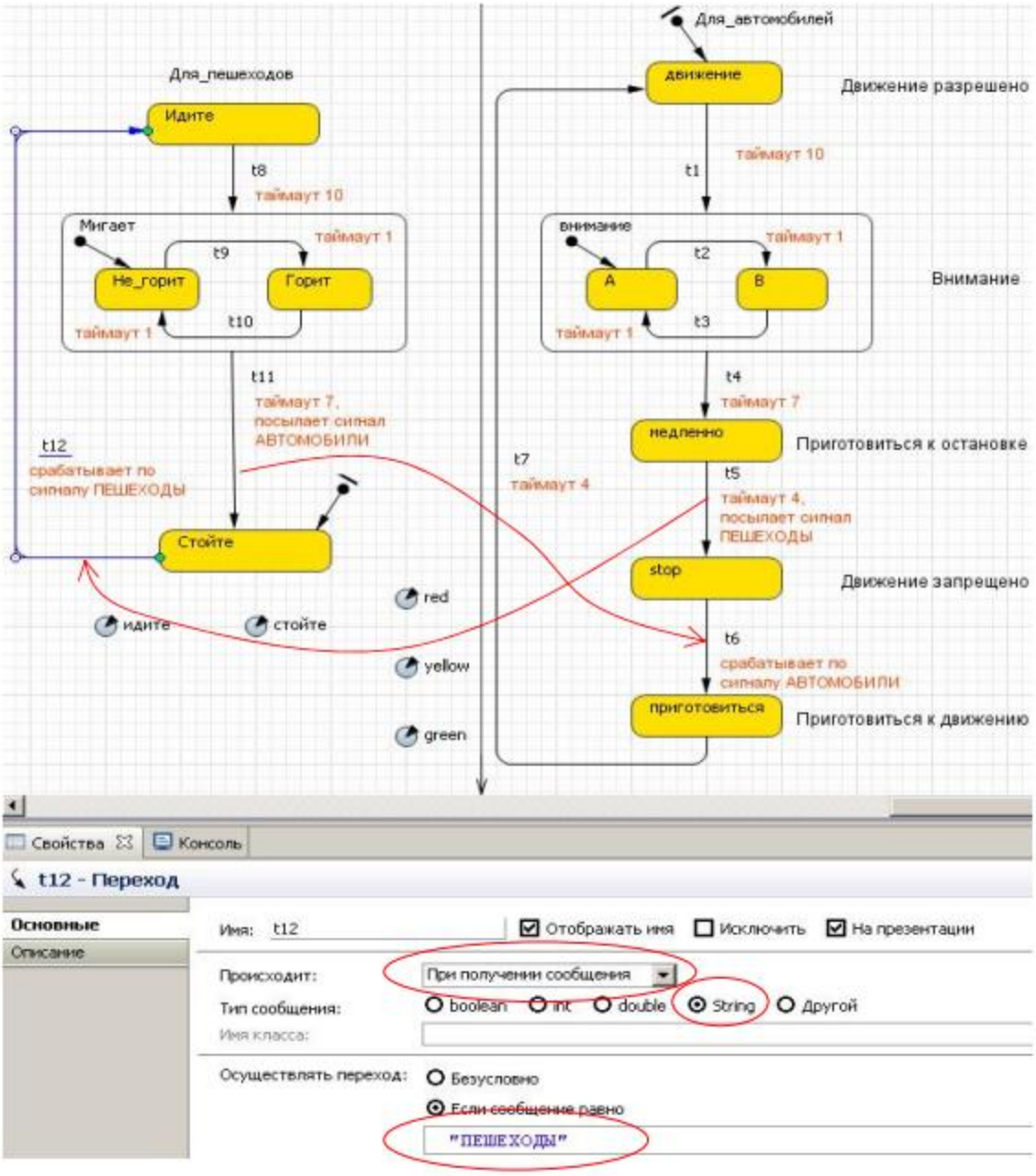


Рисунок 3.9

Таким образом, каждый из светофоров будет информировать другого о своем переходе в состояние запрещения движения, как пешеходов, так и автомобилей.

Для срабатывания перехода стейтчарта при получении нужного сообщения, в стейтчарте ***Для_пешеходов*** в поле **Происходит** окна свойств перехода ***t12*** выберите вариант **При получении сообщения**, укажите тип сообщения ***String***, а в поле **Осуществлять переход** выберите ***Если сообщение равно*** и введите "***ПЕШЕХОДЫ***" (рис. 3.9).

Аналогично, для срабатывания перехода автомобильного стейтчарта по сигналу от пешеходного стейтчарта в стейтчарте ***Для_автомобилей*** в поле **Происходит** окна свойств перехода ***t6*** выберите вариант **При получении**

сообщения, укажите тип сообщения ***String***, а в поле **Осуществлять переход** выберите ***Если сообщение равно*** и введите "***АВТОМОБИЛИ***".

Остальные переходы этих стейтчартов будут срабатывать по таймаутам, как и прежде. Проверьте по рис. 3.9 установленные параметры переходов стейтчартов. Запустите модель на выполнение.

На презентации модели, следует нарисовать светофор для пешеходов с двумя сигналами: красной надписью ***СТОЙТЕ*** и зеленой ***ИДИТЕ***. Динамикой цвета этих надписей будут управлять логические параметры ***стоите*** и ***идите,*** которые нужно создать на диаграмме по аналогии с параметрами ***красный, желтый*** и ***зеленый***.

4. Срабатывание перехода по условию

Если пешеходы переходят улицу достаточно редко, то автоматическое переключение светофора будет неоправданно. В этом случае разумно установить специальную кнопку, при нажатии которой пешеходом, светофор автомобилей остановит на некоторое время движение автомашин. Для этого перетащите мышью элемент **Кнопка** с палитры **Элементы управления** на диаграмму класса активного объекта ***Model*** рядом с пешеходным светофором, рис 3.10. Назовите кнопку и ее метку: **переход.**

Определите действие при нажатии на кнопку так, чтобы устанавливался в истину булевый параметр ***ожидание.*** Для этого нужно записать в поле **Действие** окна свойств этой кнопки команду:

ожидание = true.

Логический параметр ***ожидание,*** конечно же, нужно ввести на диаграмме активного объекта ***Model*** с начальным значением ***false***. Этот параметр будет определять, собирается ли пешеход перейти дорогу. Значение этого параметра будем переводить в ***false*** каждый раз как только пешеходный светофор перейдет в состояние ***Мигает***. Для этого нужно записать в поле **Действие при входе** окна свойств состояния ***Мигает*** стейтчарта ***Для_пешеходов*** команду: ***ожидание=false***

При нажатии кнопки **переход,** стейтчарт автомобилей «узнает» об ожидающих на переходе пешеходах и переключится из состояния в состояние ***Внимание*** и затем в состояние запрещения движения автомобилей.

В случае, если некий злоумышленник будет постоянно нажимать кнопку **переход,** это может полностью парализовать движение автомобилей. Для того чтобы этого не произошло, в стейтчарте ***Для_автомобилей*** сделайте иерархическим состояние ***Движение с двумя состояниями непрерывное и обычное*** (рис 3.11).

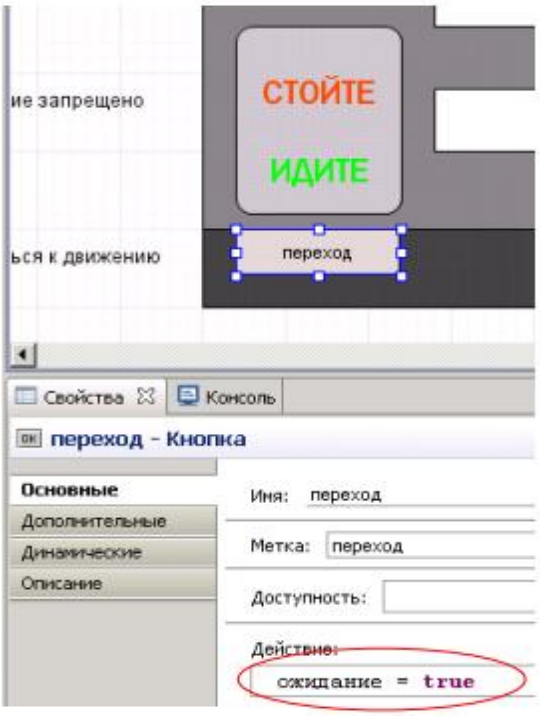


Рисунок 3.10

В состоянии ***непрерывное*** автомобили будут двигаться до истечения таймаута 10 секунд невзирая на состояние параметра ***ожидание*** т.е. автомобилям будет гарантированно предоставлено некоторое время для движения, даже если кнопка **переход** будет всегда нажата.

После истечения таймаута 10 секунд светофор перейдет в обычное состояние движения, которое может прерываться. Переход ***T*** из состояния ***обычное*** сработает, когда будет нажата кнопка **переход**, т.е. параметр ***ожидание*** будет истинен.

Если кнопка **переход** не нажата (т. е. параметр ***ожидание*** имеет значение ***false***), автомобили будут продолжать движение до нажатия этой кнопки. Если нажать кнопку один или более раз, то параметр ***ожидание*** станет истинным и автомобильный светофор из состояния ***обычное*** перейдет в состояние ***внимание*** и затем остановит движение автомобилей.

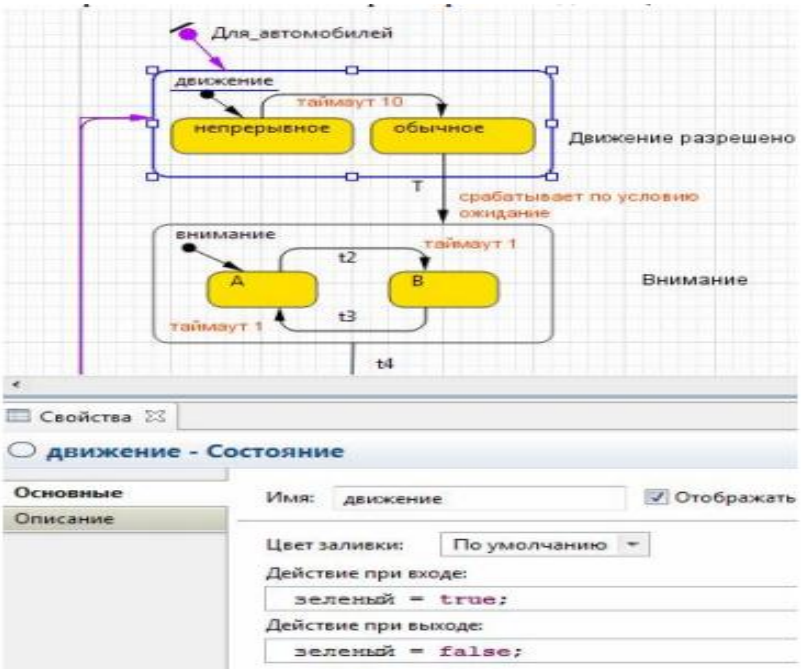


Рисунок 3.11

Задания к лабораторной работе № 3

1. Реализуйте пример, приведенный в лабораторной работе № 3.
2. Согласно своему варианту выполните задание:

№ варианта	Задание
1	Измените презентацию модели таким образом, чтобы рядом со светофором высвечивалось время в секундах, оставшееся до смены сигнала.
2	Измените модель таким образом, чтобы регулировалось движение двух пересекающихся дорожных потоков в автоматическом режиме (без кнопки) т.е. модель двух трехцветных светофоров со следующей последовательностью сигналов: зеленый – зеленый мигающий – желтый – красный – (красный +желтый) – зеленый.
3	Измените модель светофора с кнопкой таким образом, чтобы пешеход видел, нажата уже кнопка или нет, например, с помощью надписи.
4	Измените модель таким образом, чтобы моделировать железнодорожный переезд: по кнопке **ПОЕЗД** включается сигнал и через 5 секунд начинает мигать сдвоенный красный светофор (попеременно правый – левый), опускается шлагбаум. По истечении случайного промежутка времени в диапазоне от 30 до 60 сек. (прохождение поезда), переезд переводится в исходное состояние.
5	Измените модель таким образом, чтобы 3-х цветный светофор переключался в полуавтоматическом режиме по нажатию кнопок регулировщиком: по кнопке **КРАСНЫЙ** начинает мигать зеленый свет, затем зажигается желтый и через несколько секунд – красный; по кнопке **ЗЕЛЕНЫЙ** зажигается одновременно красный и желтый, затем горит зеленый.
6	Измените модель таким образом, чтобы частота мигания зеленого света регулировалась с помощью слайдера
7	Измените модель автоматического светофора автомобилей и пешеходов (без кнопки), удалив переходы по сигналу и подберите таймауты для согласованной работы светофоров
8	Измените модель светофора с кнопкой таким образом, чтобы с помощью слайдера можно было регулировать время, через которое повторное нажатие кнопки **ПЕРЕХОД** приведет к переключению

	светофора автомобилей в режим ожидания.
9	Измените модель таким образом, чтобы время, в течение которого разрешено движение пешеходов, было 10 секунд и перед окончанием запрещающего сигнала пешеходам мигал красный свет.
10	Измените модель таким образом, чтобы зеленый сигнал светофора не мигал.
11	Измените модель автоматического светофора таким образом, чтобы по кнопке СТОП включался красный свет у все светофоров, а по кнопке ДВИЖЕНИЕ светофор переходил в нормальный режим работы.
12	Измените модель светофора таким образом, чтобы на модели присутствовал реверсивный светофор.
13	Измените модель светофора таким образом, чтобы на модели присутствовал светофор для регулирования движения трамваев.
14	Измените модель таким образом, чтобы моделировать железнодорожный переезд: по кнопке **ПОЕЗД** включается сигнал и через 10 секунд начинает мигать сдвоенный красный светофор (попеременно правый – левый). По истечении случайного промежутка времени в диапазоне от 30 до 60 сек. (прохождение поезда), переезд переводится в исходное состояние.
15	Измените модель таким образом, чтобы моделировать железнодорожный переезд: по кнопке **ПОЕЗД** включается сигнал и через 10 секунд начинает мигать сдвоенный красный светофор (попеременно правый – левый). Рядом со светофором высвечивалось время в секундах, оставшееся до проезда поезда.

Номер варианта x рассчитывается по формуле $x = N \bmod 15$, где N – номер студента согласно списку группы, mod – операция "остаток от деления".

3 КОНТРОЛЬНАЯ РАБОТА

3.1 Задание 1

Дать развернутые ответы на теоретические вопросы согласно своему варианту. Вариант определяется согласно порядковому номеру из списка группы.

Таблица 3.1. Теоретические вопросы

№ варианта	Вопросы
1	1. Синектика 2. В чем состоит различие фундаментальных и прикладных научных исследований?
2	1. Метод мозгового штурма 2. Этапы научно-исследовательской работы
3	1. Научное исследование (НИ) как творческий процесс 2. Методы эмпирического исследования
4	1. Творчество и научное творчество 2. Методы теоретического исследования
5	1. Что такое наука? 2. Суть требования эвристичности
6	1. Обыденное знание. 2. Что такое научная проблема
7	1. Научное знание. Классификация наук. 2. Конструктивность теории
8	1. Метод. 2. Основные источники научной информации
9	1. Методология. 2. Выбор темы научного исследования
10	1. Основные принципы познания. 2. Методика планирования научно-исследовательской работы
11	1. Принцип отражения. 2. Изучение источников научной информации
12	1. Принцип активности. 2. Научные результаты и их обнародование
13	1. Принцип всесторонности. 2. Схема создания научной публикации
14	1. Принцип восхождения от единичного к общему и обратно. 2. Работа над статьей
15	1. Принцип единства индукции и дедукции. 2. Особенности написания заключения и выводов научной

	статьи.
16	1. Принцип взаимосвязи качественных и количественных характеристик. 2. Основные принципы этики научного сообщества
17	1. Принцип детерминизма. 2. Нормы научной этики
18	1. Принцип историзма. 2. Нарушения научной этики
19	1. Принцип противоречия. 2. Подготовка и повышение квалификации научно-педагогических и научных кадров в Украине
20	1. Принцип диалектического отрицания. 2. Докторантура
21	1. Принцип восхождения от абстрактного к конкретному. 2. Аспирантура
22	1. Принцип единства исторического и логического. 2. Рефераты и доклады
23	1. Принцип анализа и синтеза. 2. Внедрение научных исследований
24	1. Объект познания. 2. Эффективность научных исследований
25	1. Генеральная совокупность. 2. Рецензирование научно-исследовательских работ
26	1. Выборка. 2. Изучение литературы
27	1. Предмет исследования. 2. Составление и оформление библиографического списка использованных источников
28	1. Основные признаки научного знания. 2. Особенности индивидуальной научной деятельности
29	1. Критерий истинности научного знания 2. Особенности коллективной научной деятельности

3.2 Задание 2

Выполнить задания к лабораторным работам № 1-3 согласно своему варианту. Отчет должен содержать коды программ, графики и результаты работы программ. Сделать выводы о полученных результатах

ВЫВОДЫ

Рабочая программа, методические указания и индивидуальные задания по изучению дисциплины «Основы научных исследований ИКТ» предназначены для студентов специальности 122 «Компьютерные науки» заочной формы обучения.

Приведена технологическая карта дисциплины «Основы научных исследований ИКТ».

Рассмотрены основные вопросы дисциплины «Основы научных исследований ИКТ»:

— задачи моделирования работы системы массового обслуживания, различные математические задачи;

— инструментальные средства моделирования сложных информационных систем.

Приведены краткие сведения о работе в таких программных средствах как Matlab, Maple та Anylogic.

Материал, который представлен, предназначен для самостоятельной работы студентов заочной формы обучения и может быть использован для выполнения контрольной работы по дисциплине «Основы научных исследований ИКТ».

РЕКОМЕНДОВАННАЯ ЛИТЕРАТУРА

Для теоретического курса:

1. Крутов В.И, Грушко И.М., Попов В.В, и др. Основы научных исследований: Учебник для вузов / Под ред. В.И. Крутова, В.В. Попова. - М.: Высшая школа,1989. - 400 с.
2. Новиков А.М., Новиков Д.А. Методология научного исследования. – М.: Либроком. – 280 с.
3. Альтшуллер Г. С. Творчество как точная наука. - М.: Сов. радио, 1979
4. Голдовский Б. И., Вайнерман М. И. Комплексный метод поиска решений технических проблем. - М.: «Речной транспорт», 1990.. - 112 с.
5. Лудченко А.А., Лудченко Я.А., Примак Т.А. Основы научных исследований: Учеб. пособие / Под ред. А.А. Лудченко. — 2-е изд., стер. — К.: О-во "Знания", КОО, 2001. — 113 с.

Для выполнения лабораторных работ:

1. Потемкин В.Г. Система инженерных и научных расчетов MATLAB 5.x: - В 2-х т. Том 1. – М.: ДИАЛОГ-МИФИ, 1999. – 366с.
2. Потемкин В.Г. Система инженерных и научных расчетов MATLAB 5.x: - В 2-х т. Том 2. – М.: ДИАЛОГ-МИФИ, 1999. – 304с.
3. Дьяконов В., Круглов В. MATLAB. Анализ, идентификация и моделирование систем. Специальный справочник. Питер. 2001.
4. Гультяев А. Визуальное моделирование в среде Matlab: Учебный курс. Питер. 2000.
5. Чен К., Джиблин П., Ирвинг А. MATLAB в математических исследованиях. Мир. 2001.
6. Потемкин В. Инструментальные средства Matlab 5.x. Диалог-МИФИ. 2000.
7. Методичні вказівки до виконання лабораторних робіт з дисципліни "Системи штучного інтелекту". Для студентів напряму 050101 – "Комп'ютерні науки". - / Укл.: О. І. Михальов, Вік. В. Гнатушенко, Вол. В. Гнатушенко, К.Ю. Новікова, Ю.В. Бабенко. Під ред. О.І. Михальова. – Дніпропетровськ: НМетАУ, 2013. – 40 с.
6. Мельников О. И. Математическое моделирование с применением системы MAPLE. – Мн.: , 2009.– 100 с.
7. М. Н. Кирсанов. "Практика программирования в системе Maple" М.: Издательский дом МЭИ, 2011, 208с
8. В.П. Дьяконов. Maple 9 в математике, физике и образовании. М.: СОЛОН-Пресс, 2004.
9. AnyLogic 6. Лабораторный практикум / А.Е.Осоргин. – Изд. 2-е, перераб. и доп. – Самара: ПГК, 2012.
10. Карпов Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5.0. – СПб.: БХВ-Петербург, 2005. – 400 с.: ил.
11. Интернет ресурс anylogic.ru.