МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Факультет компьютерных наук и информационных технологий

УТВЕРЖДАЮ
Декан факультета компьютерных наук
и информационных технологий
_____ / С. В. Миронов
"01" сентября 2021 г.

Рабочая программа дисциплины

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Направление подготовки бакалавриата
44.03.01 – Педагогическое образование

Профиль подготовки бакалавриата
Информатика

Квалификация (степень) выпускника
Бакалавр

Форма обучения
Очная

Саратов,
2021

Статус	ФИО	Подпись	Дата
Преподаватель-разработчик	Векслер Виталий Абрамович		24.08.21
Председатель НМК	Кондратова Юлия Николаевна		24.08.21
Заведующий кафедрой	Александрова Наталья Алексеевна		24.08.21
Специалист Учебного управления			

1. Цели освоения дисциплины

Целями освоения дисциплины «Компьютерное моделирование и системы программирования» являются формирование представлений о целях и методах теории моделирования, и о возможности применения этой теории к решению разнообразных задач с помощью систем программирования.

2. Место дисциплины в структуре ООП

Данная учебная дисциплина относится к вариативной части Блока 1 «Дисциплины (Модули)» ООП (часть, формируемая участниками образовательных отношений), является дисциплиной по выбору и направлена на формирование у обучающихся общекультурных и специальных компетенций (Б1.В.ДВ.04.02).

Для изучения дисциплины необходимы компетенции, сформированные в результате изучения дисциплин «Теоретические основы информатики», «Высшая математика», «Программирование».

Компетенции, сформированные при изучении данной дисциплины, используются при изучении курсов «Преподавание робототехники в образовательной организации», «Преподавание машинного обучения в образовательной организации», «Цифровая образовательная среда».

3. Результаты обучения по дисциплине

Код и наименование компетенции	Код и наименование индикатора (индикаторов) достижения компетенции	Результаты обучения
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	1.1_Б.УК-1. Анализирует задачу, выделяя ее базовые составляющие. Осуществляет декомпозицию задачи. 2.1_Б.УК-1. Находит и критически анализирует информацию, необходимую для решения поставленной задачи. 3.1_ Б.УК-1. Рассматривает различные варианты решения задачи, оценивая их достоинства и недостатки. 4.1_ Б.УК-1. Грамотно, логично, аргументированно формирует собственные суждения и оценки. Отличает факты от мнений, интерпретаций, оценок и т.д. в рассуждениях других участников деятельности. 5.1_ Б.УК-1. Определяет и оценивает практические последствия возможных решений задачи.	Знать - предмет и задачи дисциплины; – различные способы классификации моделей; – современные мировоззренческие, социально и личностно значимые философские проблемы; закономерности и этапы исторического процесса, Уметь – уметь проводить компьютерный эксперимент и определять степень адекватности модели оригиналу; – выбирать, строить и

		анализировать математические и компьютерные модели в различных областях деятельности; – реализовывать отдельные этапы компьютерного моделирования с помощью основных систем программирования; Владеть - знаниями о моделировании, как о методе познания. - культурой мышления, способность к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения - навыками анализа современных мировоззренческих, социальных и личностно значимых проблем.
ПК - 7. Способен использовать математический аппарат, методы программирования и современные информационно-коммуникационные технологии для решения практических задач получения, хранения, обработки и передачи информации	ПК - 7.1. Решает практические задачи получения, хранения, обработки и передачи информации. ПК - 7.2. Использует математический аппарат, методы программирования и современные информационно-коммуникационные технологии для решения учебных задач.	Знать – теоретические основы компьютерного моделирования; – основные математические понятия и методы решения базовых математических задач, рассматриваемые в рамках компьютерного моделирования; Уметь – выбирать и анализировать существующие проблемы в различных областях деятельности; критически

		воспринимать, анализировать и оценивать историческую информацию; – подобрать программное обеспечение для построения компьютерной модели; – работать с выбранными программными средствами; – применять системный подход и математические методы в формализации решения прикладных задач, реализовать компьютерный эксперимент при решении задач, где возникает потребность в компьютерном математическом моделировании. Владеть - способностью самостоятельно осваивать и применять в профессиональной деятельности современные языки программирования, операционные системы, электронные библиотеки и пакеты прикладных программ, сетевые технологи

4. Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 4 зачетные единицы 144 часа.

№ п/п	Раздел дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)					Формы текущего контроля успеваемости *(по неделям семестра)* Формы промежуточной аттестации *(по семестрам)*
				Всего часов	Лекции	Лабораторные		СР	
						Общая трудоемкость	Из них – практическая подготовка		
1	Моделирование как метод познания	6	1	6	2	2		2	
2	Языки программирования, их свойства. Среды программирования.	6	2-4	16	2	4		10	
3	Модели динамических систем. Сети Петри.	6	5-6	16	4	4	2	8	Реферат
4	Имитационное моделирование.	6	7-9	24	4	6	2	14	
5	Вычислительный эксперимент.	6	10-11	19	4	4	2	11	
6	Моделирование стохастических систем.	6	12	16	6	2		8	Реферат
7	Учебные компьютерные модели.	6	13-16	20	4	4	2	12	Контрольная работа на 15 неделе
	Промежуточная аттестация								Экзамен **27**
	ИТОГО			**144**	**26**	**26**	**8**	**65**	

4.1 Содержание дисциплины

Моделирование как метод познания Цели и задачи моделирования. Понятие «модель». Виды моделирование в естественных и технических науках. Натурные и абстрактные модели. Информационные модели. Системный подход в научных исследованиях. Примеры информационных моделей. Основные структуры в информационном моделировании. Компьютерная реализация моделей. Абстрактные модели и их классификация. Объекты и их связи. Математические модели. Классификация моделей.

Языки программирования, их свойства. Среды программирования. Особенности сред программирования. Проект. Решение. Компиляция, интерпретация, трансляция. Основы алгоритмизации и программирования задач на языке высокого уровня. Статические и динамические данные; сложные структуры данных (списки, деревья, сети); потоки ввода-вывода. Основные принципы и подходы проектирования структурированных алгоритмов.

Модели динамических систем. Сети Петри. Динамические модели. Инструментальные программные средства для моделирования динамических систем. Модель популяции. Сети Петри: тезисы, основные условно-событийные модели, матричное представление, граф разметки, решение задач о достижимости и сохраняемости. Моделирование в Pascal, VBA, Lazarus.

Имитационное моделирование. Виды имитационного моделирования. Обзор современных средств поддержки имитационного моделирования. Моделирование в Pascal, VBA, Lazarus.

Вычислительный эксперимент. Системный подход в научных исследованиях. Численный эксперимент. Его взаимосвязи с натурным экспериментом и теорией. Достоверность численной модели. Анализ и интерпретация модели. Моделирование в Pascal, VBA, Lazarus.

Моделирование стохастических систем. Достоверность модели. Моделирование стохастических систем. Метод статистических испытаний. Моделирование последовательностей независимых и зависимых случайных испытаний. Общий алгоритм моделирования дискретной случайной величины. Модели, методы и алгоритмы двумерной и трёхмерной машинной графики. Построение компьютерных моделей. Моделирование в Pascal, VBA, Lazarus.

Учебные компьютерные модели. Примеры математических моделей в химии, биологии, экологии, экономике. Моделирование в Pascal, VBA, Lazarus.

Лабораторные работы

Наряду с прослушиванием лекций по курсу «Компьютерное моделирование и системы программирования» важное место в учебном процессе занимают лабораторные работы, призванные закреплять полученные студентами теоретические знания.

№ занятия	Тема
1	**2**
1	Задание 1
2	Задание 2
3	Задание 2
4	Задание 2
5	Задание 3
6	Задание 4
7	Задание 5
8	Задание 6
9	Задание 7-8
10	Задание 9-10
11	Задание 11-12
12	Задание 13

13	Задание 14
14	Задание 15
15	Задание 15
16	Задание 16

Задание 1.

а.) Разработка модели идеального жениха, модели самолёта, модели броуновского движения.

б.) Объекты и модели: бухгалтерия, карты.

в.) Примеры различных видов моделей в естественных науках.

г.) Примеры различных видов моделей в технических науках.

Задание 2.

Изучение основ работы в программе Lazarus. Рабочий интерфейс и работа со стандартными программами моделирования.

Lazarus - свободная среда быстрой разработки программного обеспечения для компилятора Free Pascal, аналогичную Delphi. Данный проект базируется на оригинальной кроссплатформенной библиотеке визуальных компонентов Lazarus Component Library (LSL), также совместимых с Visual Component Library (VCL) – объектно-ориентированная библиотека для разработки программного обеспечения. Кроссплатформенное программное обеспечение – программное обеспечение, работающее более чем на одной аппаратной платформе и/или операционной системе. Free Pascal - кроссплатформенный язык на уровне компиляции, т.е. для него существуют компиляторы под различные платформы. Таким образом, разработанные приложения могут функционировать практически под любой ОС. Все что мы видим на экране во время работы различных приложений, все элементы (кнопки, бегунки, меню и т.п.) можно реализовать в Lazarus. В Lazarus используется технология визуального программирования, т.е. пользователь оформляет будущую программу и видит результаты своей работы еще до запуска самой программы. Визуальное программирование – способ создания программы путем манипулирования графическими объектами вместо написания ее текста. Процесс создания приложения можно разделить на следующие этапы:

1. Формирование окна программы – расположение необходимых элементов, задание размеров, изменение свойств;

2. Написание программного кода, описание свойств элементов, доступных только во время работы приложения, описание реакций на событие появления окна, нажатие на кнопку и других;

3. Отладка программы.

Запуск программыПуск→Все программы→Lazarus→Lazarus – запускается оболочка создания приложений называемая интегрированной средой разработки IDE (Integrated Development Enviroment).

Интерфейс программы: Проект Lazarus представляет собой набор программных единиц — модулей.

1. Главное окно. Здесь располагаются меню, панель инструментов и палитра компонентов. На палитре компонентов, представляющую множество тематических вкладок, располагаются визуальные и невизуальные компоненты будущей программы. Невизуальные компоненты видны только на первом этапе создания приложения – при проектировании. Панель инструментов находится под главным меню и содержит наборы компонентов, заготовок будущих объектов. Для удобства эти компоненты объединены в различные группы — стандартные компоненты (Standart), расширенные компоненты (Additional), диалоги (Dialogs) и другие. Главное окно остается открытым все время работы IDE. Закрывая его, мы закрываем Lazarus и все открытые в нем окна.

2. Окно Инспектор объектов содержит 4 страницы:

1. Свойства – отображает доступные свойства выбранного компонента. В левой колонке список существующих для выделенного объекта свойства, в правой – текущие значения по умолчанию. Общие для большинства компонентов свойства: цвет – Color, имя – Name, размер

(Width – ширина, Height – высота) и т.п. Например, для будущего окна вашего приложения (формы) свойство Name имеет значение Form1, и его можно изменить в Инспекторе объектов

2. События – содержит возможные обработчики событий для выбранного компонента. В левой колонке расположены названия события, в правой соответствующие процедуры. Реакция на событие – это результат произошедшего системного события, например, щелчок мыши, нажатие на кнопку.. Например если пользователь выполняет клик по кнопке, производится копирование файла. 3. Избранное.

4. Ограничения.

3. Окно Редактор кода.

На момент первого запуска имеет заголовок Unit 1. В редакторе кода могут быть открытыми сразу несколько файлов, размещенных на отдельных страницах. В окне кода пишется текстовая часть программы, и само окно похоже на обычные текстовый редактор. Строки пронумерованы, все служебные слова выделяются жирным цветом, знаки препинания становятся красными, строки с ошибками выделяются коричневым цветом, комментарии могут заключаются в {} или (**), начинаться с // и выделяются синим. Текст программы разбивается на процедуры и функции, которые работают независимо. Основная работа программиста происходит именно здесь.

4. Окно Проектировщик форм. При запуске Lazarus автоматически предлагает пользователю новый проект, окно под названием Form 1, и назначает его главным окном. Перенося на него элементы из палитры компонетов, тем самым оформляете его. Главное окно в проекте может быть только одно, все другие создаваемые окна будут дочерними.

5. Окно Сообщения. Это окно выводит диагностические сообщения, сообщения об ошибке, сообщения об удачной компиляции и др.

Для создания нового проекта выполните команду Файл→Создать→Проект и выберите Приложение, Или выполните команду Проект →Создать проект.

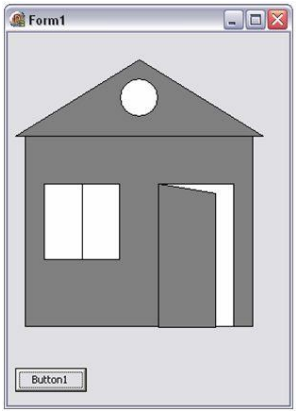
Для сохранения проекта Файл→Сохранить все. Каждый проект сохраняется в отдельный каталог.

Файлы проекта:
Модуль программы - Unit1.pas
Проект – project1.lpi
Файл Unit.lfm – файл с данными о проектировщике форм
Резервные копии файлов – папка backup

Запуск Приложения на выполнение можно сделать нажатием клавиши F9, или через пункт в меню Запуск или выбором соответствующей кнопки на панели инструментов

Вопросы и задания.
1. В чем принцип визуального программирования?
2. Назовите основные окна программы.
3. Назовите этапы создания приложения.
4. Как сохраняются проекты?
5. Как запускается созданное приложение?

1. Запустите программу Lazarus с компьютера.
2. Измените имя формы «Моя первая программа».
3. Поместите компонент Button на форму, измените его имя, свойства.
4. Поместите компонент Label. Введите текст «Я программист!».
5. Измените шрифт, цвет, расположение.
6. Сохраните проект.

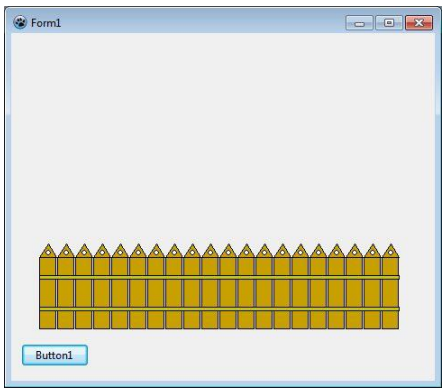


Среда программирования:
Delphi (Lazarus)
Построение повторяющихся элементов изображения имеет смысл включать в операторы цикла. Операторы цикла условно можно разделить на циклы по условию и циклы по количеству повторений (циклы-счетчики).
Когда точно известно количество повторяющихся элементов изображения удобно использовать цикл **for**.

```
for счетчик:=значение to конечное_значение do
    тело_цикла;

for счетчик:=значение downto конечное_значение do
    тело_цикла;
```

При переходе к обработке оператора цикла **for** управляющей переменной *счетчик* присваивается заданное начальное значение. Затем в цикле выполняется исполнительный оператор (или составной оператор begin..end). Каждый раз при выполнении исполнительного оператора управляющая переменная увеличивается на 1 (для for...to) или уменьшается на 1 (для for...downto). Цикл завершается при достижении управляющей переменной своего конечного значения.
Пример использования оператора цикла **for** при построении забора из 20 элементов.

```
procedure TForm1.Button1Click(Sender: TObject);
Var maxX, maxY : Integer;
   i : Integer;
begin
  maxX := PaintBox1.Width;
  maxY := PaintBox1.Height;
  // Количество повторений
  for i := 1 to 20 do  begin
    // Определение цвета досок забора
    PaintBox1.Canvas.Brush.Color := RGBToColor(200,160,0);
    // Отрисовка досок забора
    PaintBox1.Canvas.Rectangle(i*20, maxY, 19 + i*20, maxY-80);
    // Верхняя шапка забора
    PaintBox1.Canvas.Polygon([Point(i*20, maxY-80),
                        Point(10 + i*20, maxY-95), Point(19 + i*20, maxY-80)]);
    // Круглые отверстия в шапке
    PaintBox1.Canvas.Brush.Color := clWhite;
    PaintBox1.Canvas.Ellipse(7 + i*20, maxY-82, 13 + i*20, maxY-88);
  end;
  // Поперечные перекладины
  PaintBox1.Canvas.Brush.Color := RGBToColor(200,200,0);
  PaintBox1.Canvas.Rectangle(0, maxY-20, 620, maxY-25);
  PaintBox1.Canvas.Rectangle(0, maxY-55, 620, maxY-60);
end;
```



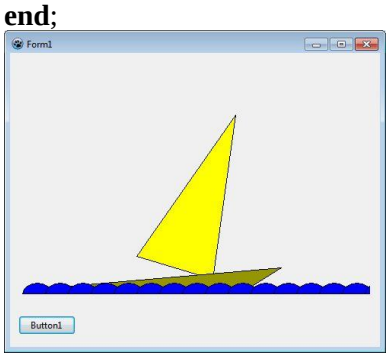
Если количество элементов не известно, то можно воспользоваться циклом по условию. Например, строить элементы изображения пока не будет достигнут край экрана (или поля для рисования). Различают циклы с предусловием и с постусловием. Цикл с предусловием

while expression **do** statement;

При выполнении этого оператора вначале вычисляется значение логического выражения expression. Если это значение истинно, выполняется оператор statement, затем значение выражения проверяется вновь и т. д., до тех пор, пока выражение не примет значение «ложь». Если выражение принимает значение «ложь» при первой же проверке, то оператор statement не выполняется вообще.

Пример использования оператора цикла **while** при построении морских волн.

```
procedure TForm1.Button1Click(Sender: TObject);
Var maxX, maxY : Integer;
    x : Integer;
begin
  maxX := PaintBox1.Width;
  maxY := PaintBox1.Height;
  // Построение паруса
  PaintBox1.Canvas.Brush.Color := clYellow;
  PaintBox1.Canvas.Polygon([Point(250,maxY-35), Point(280, maxY-250), Point(150, maxY-65)]);
  // Построение корпуса лодки
  PaintBox1.Canvas.Brush.Color := RGBToColor(150, 150, 0);
  PaintBox1.Canvas.Polygon([Point(50,maxY-25),
                    Point(300,maxY-25), Point(340,maxY-50)]);
  // Построение волн в виде полуокружностей.
  PaintBox1.Canvas.Brush.Color := clBlue;
  x:=0;
  while x < maxX do begin
    PaintBox1.Canvas.Pie(0 + x, maxY, 40 + x, maxY-30,
                  40 + x, maxY-15, 0 + x, maxY-15 );
    x := x+30;
  end;
end;
```



Понятие процедуры, функции и метода класса

Важной составной частью программирования в Object Pascal является использование подпрограмм - специальным образом оформленных и логически законченных блоков инструкций. Подпрограмму можно вызывать любое число раз из других мест программы, или из других

подпрограмм. Таким образом, использование подпрограмм позволяет сделать исходный код более стройным и наглядным.

Подпрограммы в Pascal делят на процедуры и функции. Процедуры - это такие подпрограммы, которые выполняют предназначенное действие и возвращают выполнение в точку вызова. Функции в целом аналогичны процедурам, за тем исключением, что они еще и возвращают результат своего выполнения.

Процедуры и функции, объявленные внутри класса называют методами класса.

Процедуры и функции могут иметь свой собственный набор переменных, объявленных внутри нее и называемых локальными переменными. По завершению работы значения таких переменных теряются.

```
procedure Stars (N : Integer); // N - параметр, определяющий количество звезд
Var
  i : Integer; // Локальная переменная, используемая в операторе цикла
begin
 for i:=1 to N do
  Form1.PaintBox1.Canvas.Pixels[Random(PaintBox1.Width),
Random(PaintBox1.Height)]:=clRed;
end;
```

Обмен значениями между различными процедурами возможен через параметры, глобальные переменные и возвращаемые в функциях значения.

Переменные объявленные в разделе **Var** модуля являются глобальными для всех процедур и классов модуля и доступны в любой процедуре, функции и методе класса данного модуля. Однако, работа с такими переменными напрямую внутри процедур считается "не красивым" и усложняет отлов ошибок. Поэтому удобно использовать передачу параметров и возвращаемые значения.

Передача параметров

Pascal позволяет передавать параметры в функции и процедуры либо по значению, либо по ссылке. Передаваемый параметр может иметь любой встроенный или пользовательский тип либо являться открытым массивом. Параметр также может быть константой, если его значение в процедуре или функции не меняется.

Передача параметров по значению

Этот режим передачи параметров применяется по умолчанию. Если параметр передается по значению, создается локальная копия данной переменной, которая и предоставляется для обработки в процедуру или функцию. Посмотрите на следующий пример:

```
procedure Test(s: string);
```

При вызове указанной процедуры будет создана копия передаваемой ей в качестве параметра строки s, с которой и будет работать процедура Test. При этом все внесенные в строку изменения никак не отразятся на исходной переменной s.

Этот способ передачи является у большинства самым излюбленным, но в тоже время является и самым не практичным, т.к. для выполнения метода выделяется дополнительная память для создания точной копией передаваемой переменой. Для решения этой проблемы следует использовать один из способов описанных ниже.

Примечание: в случае если в процедуру передается переменная объекта, то в данном случае произойдет передача по ссылке (даже если это не указано явно).

Передача параметров по ссылке

Pascal позволяет также передавать параметры в функции или процедуры по ссылке — такие параметры называются параметрами-переменными. Передача параметра по ссылке означает, что функция или процедура сможет изменить полученные значения параметров. Для передачи параметров по ссылке используется ключевое слово **var**, помещаемое в список параметров вызываемой процедуры или функции.

```
procedure ChangeMe(var x: longint);
 begin
 x := 2; // Параметр x изменен вызванной процедурой
 end;
```

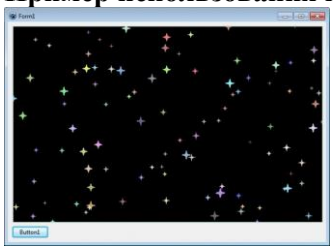
Вместо создания копии переменной **x**, ключевое слово **var** требует передачи адреса самой переменной **x**, что позволяет процедуре непосредственно изменять ее значение.

Передача параметров констант

Если нет необходимости изменять передаваемые функции или процедуре данные, можно описать параметр как константу. Ключевое слово const не только защищает параметр от изменения, но и позволяет компилятору сгенерировать более оптимальный код передачи строк и записей. Вот пример объявления параметра-константы:

procedure Test(**const** s: **string**);

Пример использования процедуры Star для построения звездного неба



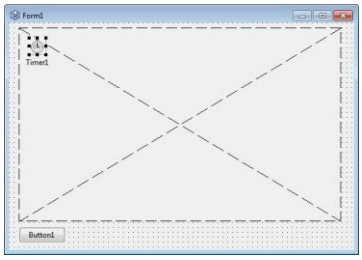
Процедура **Star** объявлена внутри класса **TForm1** и по сути является методом этого класса, позволяющим рисовать звезды на форме. При создании процедуры удобно использовать горячие клавиши (Ctrl-Shift+C), которые автоматически создадут тело процедуры внутри блока **implementation**. Для этого необходимо прописать заголовок процедуры в разделе **public** или **private** и нажать горячие клавиши Ctrl-Shift+C.

В процедуру передаются параметры x,y - координаты расположения звезды, Size - размер и Color - цвет отрисовки. Используя цикл задается 100 звезд разного цвета и размера.

```
unit Unit1;

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,
  ExtCtrls;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    PaintBox1: TPaintBox;
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
    // Рисование звезды
    procedure Star(x,y: Integer; Size: Integer; Colour: TColor);
    // Получение случайного цвета. A - параметр, определяющий яркость случайного цвета
    function GetRandomColor( A: Integer): TColor;
  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
// Локальные переменные i и c используются для задания количества и цвета звезд
```

```
Var i : Integer;
   c : TColor;
begin
  // Закрашиваем поле для рисования в черный цвет
  PaintBox1.Canvas.Brush.Color:=clBlack;
  PaintBox1.Canvas.Rectangle(0,0,PaintBox1.Width, PaintBox1.Height);
  // Вызываем процедуру Star 100 раз, тем самым рисуя 100 звезд
  for i := 1 to 100 do begin
    // Случайным образом формируем цвет звезды, с помощью функции
    c := GetRandomColor(100);
    // Вызываем процедуру отрисовки звезды с разными случайными параметрами
    Star( Random(PaintBox1.Width), Random(PaintBox1.Height), Random(7)+3, c);
  end;
end;

//Процедура прорисовки звезды с разными параметрами расположения, размера и цвета
procedure TForm1.Star(x, y: Integer; Size: Integer; Colour: TColor);
begin
  // Задаем цвет в соответствии с полученным параметром
  PaintBox1.Canvas.Pen.Color:= Colour;
  PaintBox1.Canvas.Brush.Color:= Colour;
  // Рисуем звезду, используя переданные координаты и размер
  Paintbox1.Canvas.Polygon( [Point(x, y-size),
  Point(x-size div 4, y-size div 4), Point(x-size, y),
  Point(x-size div 4, y+size div 4), Point(x, y+size),
  Point(x+size div 4, y+size div 4), Point(x+size, y),
  Point(x+size div 4, y-size div 4), Point(x, y-size)]);
end;

function TForm1.GetRandomColor( A: Integer): TColor;
begin
 // Проверка на корректность задания параметра A (не больше 255)
 if A > 255 then A := 255;
 // Получение случайного цвета, в зависимости от значения A
 Result := RGBToColor(Random(256-A)+A, Random(256-A)+A, Random(256-A)+A);
end;

end.
```

Линейное движение по однородному фону является довольно простым в плане программной реализации. Достаточно закрашивать объект цветом фона, изменять его координаты и прорисовывать в новом месте, повторяя эти действия через определенный интервал времени. Для реализации анимации, помимо двух уже известных компонентов TPaintBox (поле для рисования) и TButton (кнопка запуска), понадобится компонент TTimer со вкладки System. Компонент Timer имеет единственное событие OnTimer, которое выполняется пока Timer включен с интервалом по времени, установленным в свойстве Interval.
Расположите компонент Timer1 на форме. Установите его свойства Timer1.Interval := 100 и Timer1.Enabled := false

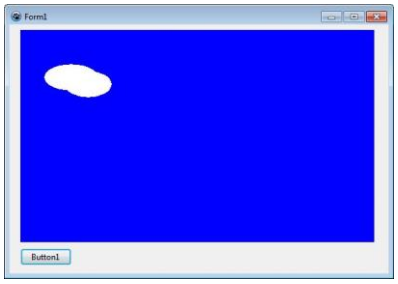


В коде программы необходимо прописать три процедуры (см. урок "Процедуры и функции при построении изображений"). Процедуру отрисовки объекта procedure TForm1.Cloud, процедуру,

отрабатывающую на событие OnTimer, - procedure TForm1.Timer1Timer и процедуру запуска анимации, срабатывающую на нажатие кнопки, procedure TForm1.Button1Click.

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, Buttons,
  ExtCtrls, StdCtrls;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    PaintBox1: TPaintBox;
    Timer1: TTimer;
    procedure Button1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { private declarations }
     // координаты прорисовки объекта. Доступны всем процедурам класса TForm1
     x1, y1 : Integer;
  public
    { public declarations }
    // процедура прорисовки облака
    procedure Cloud (x, y: Integer; ColorCloud: TColor);
  end;

var
  Form1: TForm1;


implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.Cloud(x, y: Integer; ColorCloud: TColor);
begin
  // прорисовка облака из двух эллипсов
  with PaintBox1.Canvas do begin
    Pen.Style := psClear;
    Brush.Color := ColorCloud;
    Ellipse(x,y,x+80,y+40);
    Ellipse(x+30,y+10,x+100,y+50);
  end;
end;


procedure TForm1.Button1Click(Sender: TObject);
begin
  // установка начальных значений
```

```
  x1:=0;
  y1:=50;
  Timer1.Interval:=100;
  // прорисовка картинки по которой двигается объект
  PaintBox1.Canvas.Brush.Color := clBlue;
  PaintBox1.Canvas.Rectangle(0,0, PaintBox1.Width, PaintBox1.Height);
  // Включение таймера - запуск анимации
  Timer1.Enabled := true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  // Закраска объекта цветом фона
  Cloud(x1,y1,clBlue);
  // Изменение координат прорисовки
  x1:=x1+1;
  // Прорисовка объекта в новом месте
  Cloud(x1,y1,clWhite);
end;

end.
```



Основное отличие движения по неоднородному фону от движения по однородному фону состоит в том, что движущийся объект уже не закрашивается цветом фона, а восстанавливается часть изображения по которому движется объект и лишь затем происходит его прорисовка.
Для сохранения части изображения используется класс TBitMap (битовая карта изображения). Используя методы этого класса вырезаем часть изображения и по таймеру производим восстановление изображения и прорисовку объекта на новом месте.

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, Buttons,
  ExtCtrls, StdCtrls;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    PaintBox1: TPaintBox;
    Timer1: TTimer;
    procedure Button1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { private declarations }
```

```
    // координаты прорисовки объекта. Доступны всем процедурам класса TForm1
    x1, y1 : Integer;
    ARect : TRect;
    BitMap : TBitMap;
  public
    { public declarations }
    // процедура прорисовки облака
    procedure Cloud (x, y: Integer; ColorCloud: TColor);
  end;

var
  Form1: TForm1;


implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.Cloud(x, y: Integer; ColorCloud: TColor);
begin
  // прорисовка облака из трех эллипсов
  with PaintBox1.Canvas do begin
    Pen.Style := psClear;
    Brush.Color := ColorCloud;
    Ellipse(x,y,x+80,y+40);
    Ellipse(x+30,y+10,x+120,y+50);
    Ellipse(x+40,y-10,x+150,y+35);
  end;
end;


procedure TForm1.Button1Click(Sender: TObject);
begin
  // установка начальных значений
  x1:=0;
  y1:=50;
  Timer1.Interval:=30;
  // прорисовка картинки по которой двигается объект
  PaintBox1.Canvas.Brush.Color := clBlue;
  PaintBox1.Canvas.Rectangle(0,0, PaintBox1.Width, PaintBox1.Height);
  PaintBox1.Canvas.Brush.Color := clYellow;
  PaintBox1.Canvas.Ellipse(50,50, 150, 150);
  Cloud(200,100,clWhite);
  Cloud(400,10,clWhite);
  // Сохранение части изображения в BitMap-массив
  BitMap := TBitmap.Create;
  BitMap.Width := PaintBox1.Width;
  BitMap.Height := 100;
  ARect := Rect(0,y1-10,PaintBox1.Width,y1+90);
  BitMap.Canvas.CopyRect(ARect,PaintBox1.Canvas, ARect);
  // Включение таймера - запуск анимации
  Timer1.Enabled := true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
```

```
begin
  // Восстановление изображения из BitMap-массива
  ARect := Rect(0,y1-10,PaintBox1.Width,y1+90);
  PaintBox1.Canvas.CopyRect(ARect, BitMap.Canvas, ARect);
  // Прорисовка объекта
  Cloud(x1,y1,clWhite);
  x1 := x1+1;
  // Если объект вышел за пределы окна, то начать движение объекта сначала
  if x1 > PaintBox1.Width then x1:= -100;
end;

end.
```

OpenGL (Open Graphics Library — открытая графическая библиотека) — спецификация, определяющая независимый от языка программирования кросс-платформенный программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

Включает более 250 функций для рисования сложных трёхмерных сцен из простых примитивов. Используется при создании компьютерных игр, САПР, виртуальной реальности, визуализации в научных исследованиях. На платформе Windows конкурирует с Direct3D.

Основные возможности OpenGL

Что предоставляет библиотека в распоряжение программиста? Основные возможности:

- Геометрические и растровые примитивы. На основе геометрических и растровых примитивов строятся все объекты. Из геометрических примитивов библиотека предоставляет: точки, линии, полигоны. Из растровых: битовый массив(bitmap) и образ(image)
- Использование B-сплайнов. B-сплайны используются для рисования кривых по опорным точкам.
- Видовые и модельные преобразования. С помощью этих преобразований можно располагать обьекты в пространстве, вращать их, изменять форму, а также изменять положение камеры из которой ведётся наблюдение.
- Работа с цветом. OpenGL предоставляет программисту возможность работы с цветом в режиме RGBA (красный-зелёный-синий-альфа) или используя индексный режим, где цвет выбирается из палитры.
- Удаление невидимых линий и поверхностей. Z-буферизация.
- Двойная буферизация. OpenGL предоставляет как одинарную так и двойную буферизацию. Двойная буферизация используется для того, чтобы устранить мерцание при мультипликации, т.е. изображение каждого кадра сначала рисуется во втором(невидимом) буфере, а потом, когда кадр полностью нарисован, весь буфер отображается на экране.
- Наложение текстуры. Позволяет придавать объектам реалистичность. На объект, например шар, накладывается текстура(просто какое-то изображение), в результате чего наш объект теперь выглядит не просто как шар, а как разноцветный мячик.
- Сглаживание. Сглаживание позволяет скрыть ступенчатость, свойственную растровым дисплеям. Сглаживание изменяет интенсивность и цвет пикселей около линии, при этом линия смотрится на экране без всяких зигзагов.
- Освещение. Позволяет задавать источники света, их расположение, интенсивность, и т.д.
- Атмосферные эффекты. Например туман, дым. Всё это также позволяет придать объектам или сцене реалистичность, а также "почувствовать" глубину сцены.
- Прозрачность объектов.

- Использование списков изображений.

Дополнительные библиотеки

Для OpenGL существуют так называемые вспомогательные библиотеки.

Первая из этих библиотек называется GLU. Эта библиотека уже стала стандартом и поставляется вместе с главной библиотекой OpenGL. В состав этой библиотеки вошли более сложные функции, например для того чтобы определить цилиндр или диск потребуется всего одна команда. Также в библиотеку вошли функции для работы со сплайнами, реализованы дополнительные операции над матрицами и дополнительные виды проекций.

Следующая библиотека, также широко используемая - это GLUT. Это также независимая от платформы библиотека. Она реализует не только дополнительные функции OpenGL, но и предоставляет функции для работы с окнами, клавиатурой и мышкой. Для того чтобы работать с OpenGL в конкретной операционной системе (например Windows или X Windows), надо провести некоторую предварительную настройку и эта предварительная настройка зависит от конкретной операционной системы. С библиотекой GLUT всё намного упрощается, буквально несколькими командами можно определить окно, в котором будет работать OpenGL, определить прерывание от клавиатуры или мышки и всё это не будет зависеть от операционной системы. Библиотека предоставляет также некоторые функции, с помощью которых можно определять некоторые сложные фигуры, такие как конусы, тетраэдры, и даже можно с помощью одной команды определить чайник!

Есть ещё одна библиотека похожая на GLUT, называется она GLAUX. Это библиотека разработана фирмой Microsoft для операционной системы Windows. Она во многом схожа с библиотекой GLUT, но немного отстаёт от неё по своим возможностям. И ещё один недостаток заключается в том, что библиотека GLAUX предназначена только для Windows, в то время как GLUT поддерживает очень много операционных систем.

Синтаксис команд

type glCommand_name[1 2 3 4][b s i f d ub us ui][v]
(type1 arg1,…,typeN argN)

Таким образом, имя состоит из нескольких частей: Gl это имя библиотеки, в которой описана эта функция: для базовых функций OpenGL, функций из библиотек GLU, GLUT, GLAUX это gl, glu, glut, glaux соответственно

Command_name - имя команды

[1 2 3 4] – число аргументов команды

[b s i f d ub us ui] – тип аргумента:

b - GLbyte байт,

s - GLshort короткое целое,

i - GLint целое,

f - GLfloat дробное,

d - GLdouble дробное с двойной точностью,

ub - GLubyte беззнаковый байт,

us - GLushort беззнаковое короткое целое,

ui - GLuint беззнаковое целое.

Полный список типов и их описание можно посмотреть в файле gl.h.

[v] – наличие этого символа показывает, что в качестве параметров функции используется указатель на массив значений.

Рисование геометрических объектов

Очистка окна.

glClearColor (clampf r, clampf g, clampf b, clampf a)

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

Вершины и примитивы

Вершина является атомарным графическим примитивом OpenGL и определяет точку, конец отрезка, угол многоугольника и т.д. Все остальные примитивы формируются с помощью задания вершин, входящих в данный примитив. Например, отрезок определяется двумя вершинами, являющимися концами отрезка.

С каждой вершиной ассоциируются ее атрибуты. В число основных атрибутов входят положение вершины в пространстве, цвет вершины и вектор нормали. Положение вершины в пространстве

Положение вершины определяются заданием ее координат в двух-, трех-, или четырехмерном пространстве (однородные координаты). Это реализуется с помощью нескольких вариантов команды **glVertex***:

void glVertex[2 3 4][s i f d] (type coords)

void glVertex[2 3 4][s i f d]v (type *coords)

Каждая команда задает четыре координаты вершины: x, y, z, w. Команда **glVertex2*** получает значения x и y. Координата z в таком случае устанавливается по умолчанию равной 0, координата w – равной 1. Vertex3* получает координаты x, y, z и заносит в координату w значение 1. Vertex4* позволяет задать все четыре координаты.

Для ассоциации с вершинами цветов, нормалей и текстурных координат используются текущие значения соответствующих данных, что отвечает организации OpenGL как конечного автомата. Эти значения могут быть изменены в любой момент с помощью вызова соответствующих команд.

Цвет вершины

Для задания текущего цвета вершины используются команды :

void glColor[3 4][b s i f] (GLtype components)

void glColor[3 4][b s i f]v (GLtype components)

Первые три параметра задают R, G, B компоненты цвета, а последний параметр определяет коэффициент непрозрачности (так называемая альфа-компонента). Если в названии команды указан тип ‘f’ (float), то значения всех параметров должны принадлежать отрезку [0,1], при этом по умолчанию значение альфа-компоненты устанавливается равным 1.0, что соответствует полной непрозрачности. Тип ‘ub’ (unsigned byte) подразумевает, что значения должны лежать в отрезке [0,255].

Вершинам можно назначать различные цвета, и, если включен соответствующий режим, то будет проводиться линейная интерполяция цветов по поверхности примитива.

Для управления режимом интерполяции используется команда

void glShadeModel (GLenum mode)

вызов которой с параметром GL_SMOOTH включает интерполяцию (установка по умолчанию), а с GL_FLAT – отключает.

Нормаль

Определить нормаль в вершине можно, используя команды

void glNormal3[b s i f d] (type coords)

void glNormal3[b s i f d]v (type coords)

Для правильного расчета освещения необходимо, чтобы вектор нормали имел единичную длину. Командой **glEnable(GL_NORMALIZE)** можно включить специальный режим, при котором задаваемые нормали будут нормироваться автоматически.

Режим автоматической нормализации должен быть включен, если приложение использует модельные преобразования растяжения/сжатия, так как в этом случае длина нормалей изменяется при умножении на модельно-видовую матрицу.

Однако применение этого режима уменьшает скорость работы механизма визуализации OpenGL, так как нормализация векторов имеет заметную вычислительную сложность (взятие квадратного корня и т.п.). Поэтому лучше сразу задавать единичные нормали.

Отметим, что команды

void glEnable (GLenum mode)

void glDisable (GLenum mode)

производят включение и отключение того или иного режима работы конвейера OpenGL. Эти команды применяются достаточно часто, и их возможные параметры будут рассматриваться в каждом конкретном случае.

Операторные скобки **glBegin / glEnd**

Мы рассмотрели задание атрибутов одной вершины. Однако, чтобы задать атрибуты графического примитива, одних координат вершин недостаточно. Эти вершины надо объединить в одно целое, определив необходимые свойства. Для этого в OpenGL используются так называемые операторные скобки, являющиеся вызовами специальных команд OpenGL Определение примитива или последовательности примитивов происходит между вызовами команд

void glBegin (GLenum mode);

void glEnd (void);

Параметр mode определяет тип примитива, который задается внутри и может принимать следующие значения: GL_POINTS каждая вершина задает координаты некоторой точки.

- **GL_LINES** каждая отдельная пара вершин определяет отрезок; если задано нечетное число вершин, то последняя вершина игнорируется.
- **GL_LINE_STRIP** каждая следующая вершина задает отрезок вместе с предыдущей.
- **GL_LINE_LOOP** отличие от предыдущего примитива только в том, что последний отрезок определяется последней и первой вершиной, образуя замкнутую ломаную.
- **GL_TRIANGLES** каждая отдельная тройка вершин определяет треугольник; если задано не кратное трем число вершин, то последние вершины игнорируются.
- **GL_TRIANGLE_STRI**P каждая следующая вершина задает треугольник вместе с двумя предыдущими.
- **GL_TRIANGLE_FAN** треугольники задаются первой и каждой следующей парой вершин (пары не пересекаются).
- **GL_QUADS** каждая отдельная четверка вершин определяет четырехугольник; если задано не кратное четырем число вершин, то последние вершины игнорируются.
- **GL_QUAD_STRIP** четырехугольник с номером n определяется вершинами с номерами 2n-1, 2n, 2n+2, 2n+1.
- **GL_POLYGON** последовательно задаются вершины выпуклого многоугольника.

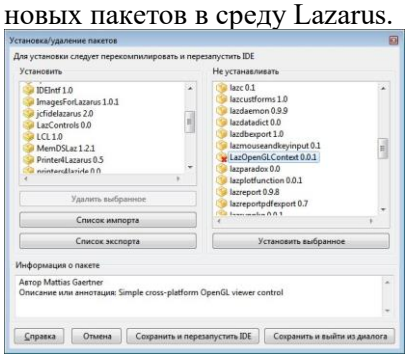
Например, чтобы нарисовать треугольник с разными цветами в вершинах, достаточно написать:

```
glBegin(GL_TRIANGLES);
glColor3f(1.0, 0.0, 0.0); glVertex3f(0.0, 0.0, 0.0);
glColor3ub(0,255,0); glVertex3f(1.0, 0.0, 0.0);
glColor3fv(BlueCol); glVertex3f(1.0, 1.0, 0.0);
glEnd();
```

Установка пакета получения контекста устройства в Lazarus

Как и в любой среде программирования, для того чтобы начать работать с графикой, необходимо получить контекст устройства и связать его с контекстом воспроизведения библиотеки OpenGL. В Lazarus этот процесс осуществляется довольно просто, за счет использования встроенной библиотеки **OpenGLContext** с готовым компонентом **TOpenGLControl**.

Однако компонент **TOpenGLControl**, по — умолчанию, не установлен в среде Lazarus, поэтому необходимо установить данный компонент. Выберите пункт меню "Пакет -> Установить/Удалить пакеты" (Package -> Install/Uninstall Packages), откроется окно установки новых пакетов в среду Lazarus.



В окне установки новых пакетов, в списке неустановленных пакетов, необходимо найти пакет с именем lazopenglcontext 0.0.1, выбрать его и нажать кнопку "Установить выбранное" (Install selection). После этого необходимо нажать кнопку "Сохранить и перезапустить IDE" (Save and Rebuild IDE), для пересборки Lazarus уже с компонентом **TOpenGLControl**. В окне подтверждения установки нового пакета, нажмите кнопку "Продолжить" (Continue).

Если все проделано верно, то в панели инструментов появится вкладка OpenGl с компонентом TOpenGLControl.

Можно начинать работать с библиотекой.

Простейшая программа на OpenGL в Lazarus

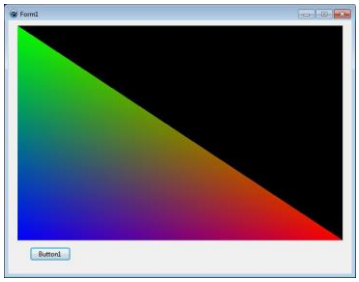
Расположите на форме два компонента

Panel1: TPanel

Button1: TButton
пропишите следующий код на события **OnCreate** и **ButtonClick**

```
unit Unit1;

{$mode objfpc}{$H+}

interface

// Подключение библиотек. Для работы С OpenGL необходимы OpenGLContext, GL, GLU
uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,
  ExtCtrls, OpenGLContext, GL, GLU;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    Panel1: TPanel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
    OpenGLControl1: TOpenGLControl; // Контекст воспроизведения OpenGL
  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);

begin
  // Создание контекста воспроизведения OpenGL и привязка его к панели на форме
  OpenGLControl1:=TOpenGLControl.Create(Self);
  with OpenGLControl1 do begin
    Name:='OpenGLControl1';
    Align:=alClient;
    Parent:=Panel1;
  end;

end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 glClearColor (0, 0, 0, 0); // цвет фона
 glClear (GL_COLOR_BUFFER_BIT);      // очистка буфера цвета
 glMatrixMode(GL_MODELVIEW);  // Выбор видовой матрицы
 glLoadIdentity();   //  Установка в единичные значения
```

```
 glOrtho(0, 1, 0, 1, -1 ,1); // Установка проекции окна
 //glTranslatef(0.5,0.5,0);  // Сдвиг
 //glRotatef(10, 0, 0, 0);   // Вращение
 glBegin(GL_TRIANGLES);      // Рисование треугольника
      glColor3f(1.0,0.0,0.0);     // Красный
      glVertex3f(1,0,0 ); // Верх треугольника (Передняя)
      glColor3f(0.0,1,0.0);
      glVertex3f( 0,1,0);
      glColor3f(0,0,1);
      glVertex3f( 0,0,0);
 glEnd;
 OpenGLControl1.SwapBuffers; // Отрисовка из буффера
end;

end.
```



Среда программирования:
Delphi (Lazarus)

В OpenGL используются основные три системы координат: левосторонняя, правосторонняя и оконная. Первые две системы являются трехмерными и отличаются друг от друга направлением оси z: в правосторонней она направлена на наблюдателя, в левосторонней – в глубину экрана. Ось x направлена вправо относительно наблюдателя, ось y – вверх.

Системы координат в OpenGL

Строго говоря, OpenGL позволяет путем манипуляций с матрицами моделировать как правую, так и левую систему координат. Но на данном этапе лучше пойти простым путем и запомнить: основной системой координат OpenGL является правосторонняя система.

РАБОТА С МАТРИЦАМИ

Для задания различных преобразований объектов сцены в OpenGL используются операции над матрицами, при этом различают три типа матриц: модельно-видовая, матрица проекций и матрица текстуры. Все они имеют размер 4x4. Видовая матрица определяет преобразования объекта в мировых координатах, такие как параллельный перенос, изменение масштаба и поворот. Матрица проекций определяет, как будут проецироваться трехмерные объекты на плоскость экрана (в оконные координаты), а матрица текстуры определяет наложение текстуры на объект.

Умножение координат на матрицы происходит в момент вызова соответствующей команды OpenGL, определяющей координату (как правило, это команда glVertex*)

Для того чтобы выбрать, какую матрицу надо изменить, используется команда:

void glMatrixMode (GLenum mode)

вызов которой со значением параметра mode равным

GL_MODELVIEW,

GL_PROJECTION,

GL_TEXTURE

включает режим работы с модельно-видовой матрицей, матрицей проекций, или матрицей текстуры соответственно. Для вызова команд, задающих матрицы того или иного типа, необходимо сначала установить соответствующий режим.

Для определения элементов матрицы текущего типа вызывается команда

void glLoadMatrix[f d] (GLtype *m)

где **m** указывает на массив из 16 элементов типа float или double в соответствии с названием команды, при этом сначала в нем должен быть записан первый столбец матрицы, затем второй, третий и четвертый. Еще раз обратим внимание: в массиве m матрица записана по столбцам.

Команда

void glLoadIdentity (void)

заменяет текущую матрицу на единичную.

Часто бывает необходимо сохранить содержимое текущей матрицы для дальнейшего использования, для чего применяются команды

void glPushMatrix (void)

void glPopMatrix (void)

Они записывают и восстанавливают текущую матрицу из стека, причем для каждого типа матриц стек свой. Для модельно-видовых матриц его глубина равна как минимум 32, для остальных – как минимум 2.

МОДЕЛЬНО-ВИДОВЫЕ ПРЕОБРАЗОВАНИЯ

К модельно-видовым преобразованиям будем относить перенос, поворот и изменение масштаба вдоль координатных осей. Для проведения этих операций достаточно умножить на соответствующую матрицу каждую вершину объекта и получить измененные координаты этой вершины.

Сама матрица может быть создана с помощью следующих команд:

void glTranslate[f d] (GLtype x, GLtype y, GLtype z)

void glRotate[f d] (GLtype angle, GLtype x, GLtype y, GLtype z)

void glScale[f d] (GLtype x, GLtype y, GLtype z)

glTranlsate*() производит перенос объекта, прибавляя к координатам его вершин значения своих параметров.

glRotate*() производит поворот объекта против часовой стрелки на угол angle (измеряется в градусах) вокруг вектора (*x*,*y*,*z*).

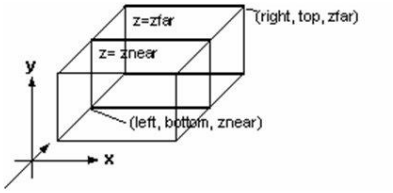
glScale*() производит масштабирование объекта (сжатие или растяжение) вдоль вектора (*x*,*y*,*z*), умножая соответствующие координаты его вершин на значения своих параметров.

Все эти преобразования изменяют текущую матрицу, а поэтому применяются к примитивам, которые определяются позже. В случае, если надо, например, повернуть один объект сцены, а другой оставить неподвижным, удобно сначала сохранить текущую видовую матрицу в стеке командой **glPushMatrix()**, затем вызвать **glRotate()** с нужными параметрами, описать примитивы, из которых состоит этот объект, а затем восстановить текущую матрицу командой **glPopMatrix()**.

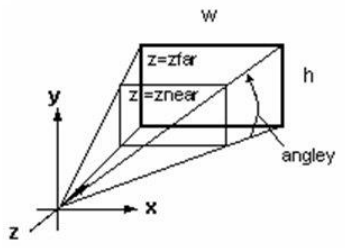
Чтобы изображенная фигура выглядела пространственной, систему координат разворачивают вокруг оси *X* и вокруг оси *Y*.

ПРОЕКЦИИ

В OpenGL существуют стандартные команды для задания ортографической (параллельной) и перспективной проекций. Первый тип проекции может быть задан командами



void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top,
GLdouble **near**, GLdouble **far**)



void glFrustum (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top,
GLdouble **near**, GLdouble **far**)

Место команд в программе.

Для того чтобы при каждой последующей перерисовке экрана не происходило изменение размеров сцены, связанное с переносом и проецированием, следует использовать видовые команды по определенным правилам.

Первый способ – использование команды **glLoadIdentity**:

```
glLoadIdentity;                    //Сброс всех матриц в 1
glFrustum(-1, 1, -1, 1, 3, 10);    //видовые параметры
glTranslatef(0.0, 0.0, -5.0);      //начальный сдвиг системы координат
glRotatef(30.0, 1.0, 0.0, 0.0);    //поворот относительно оси X
glRotatef(70.0, 0.0, 1.0, 0.0);    //поворот относительно оси Y
glBegin(…);
  …..                              // команды рисования
glEnd;
```

Второй способ – использование команд **glPushMatrix** и **glPopMatrix**:

```
glPushMatrix();                    //запоминаем текущую матрицу
glFrustum(-1, 1, -1, 1, 3, 10);    //видовые параметры
glTranslatef(0.0, 0.0, -5.0);      //начальный сдвиг системы координат
glRotatef(30.0, 1.0, 0.0, 0.0);    //поворот относительно оси X
glRotatef(70.0, 0.0, 1.0, 0.0);    //поворот относительно оси Y
glBegin(…);
  …..                              // команды рисования
glEnd;
glPopMatrix();                                  //восстанавливаем текущую матрицу
```

ПРОГРАММА - ПРИМЕР ВРАЩЕНИЯ КУБА В LAZARUS НА OPENGL

Разместите на форме три компонента

OpenGLControl1: TOpenGLControl;

Button1: TButton;

Timer1: TTimer;

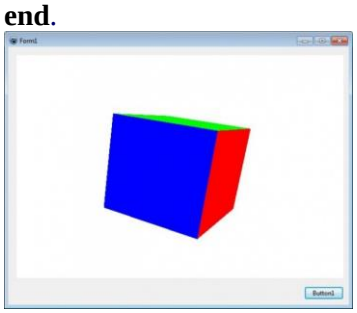
Примечание. Обратите внимание для установки компонента TOpenGLControl смотрите урок "Подключение и работа с OpenGL в Lazarus под Windows".

Пропишите на события создания формы Form1: OnFormCreate, нажатия кнопки Form1: OnButton1Click и работы таймера Form1: OnTimer1Timer следующий код:

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,
  ExtCtrls, OpenGLContext, GL, GLU;

type

  { TForm1 }

  TForm1 = class(TForm)
```

```
    Button1: TButton;
    OpenGLControl1: TOpenGLControl;
    Timer1: TTimer;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
    cube_rotation: GLFloat;
    Speed:         Double;
  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin
  Timer1.Enabled := true;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Enabled  := false;
  Timer1.Interval := 100;
  Speed := 1;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  glClearColor(1.0, 1.0, 1.0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
  glEnable(GL_DEPTH_TEST);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(45.0, double(width) / height, 0.1, 100.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();

  glTranslatef(0.0, 0.0,-6.0);
  glRotatef(cube_rotation, 1.0, 1.0, 1.0);

  glBegin(GL_QUADS);
        glColor3f(0.0,1.0,0.0);                 // Set The Color To Green
        glVertex3f( 1.0, 1.0,-1.0);             // Top Right Of The Quad (Top)
        glVertex3f(-1.0, 1.0,-1.0);             // Top Left Of The Quad (Top)
        glVertex3f(-1.0, 1.0, 1.0);             // Bottom Left Of The Quad (Top)
        glVertex3f( 1.0, 1.0, 1.0);             // Bottom Right Of The Quad (Top)
  glEnd();
```

```
  glBegin(GL_QUADS);
        glColor3f(1.0,0.5,0.0);                 // Set The Color To Orange
        glVertex3f( 1.0,-1.0, 1.0);             // Top Right Of The Quad (Bottom)
        glVertex3f(-1.0,-1.0, 1.0);             // Top Left Of The Quad (Bottom)
        glVertex3f(-1.0,-1.0,-1.0);             // Bottom Left Of The Quad (Bottom)
        glVertex3f( 1.0,-1.0,-1.0);             // Bottom Right Of The Quad (Bottom)
  glEnd();
  glBegin(GL_QUADS);
        glColor3f(1.0,0.0,0.0);                 // Set The Color To Red
        glVertex3f( 1.0, 1.0, 1.0);             // Top Right Of The Quad (Front)
        glVertex3f(-1.0, 1.0, 1.0);             // Top Left Of The Quad (Front)
        glVertex3f(-1.0,-1.0, 1.0);             // Bottom Left Of The Quad (Front)
        glVertex3f( 1.0,-1.0, 1.0);             // Bottom Right Of The Quad (Front)
  glEnd();
  glBegin(GL_QUADS);
        glColor3f(1.0,1.0,0.0);                 // Set The Color To Yellow
        glVertex3f( 1.0,-1.0,-1.0);             // Bottom Left Of The Quad (Back)
        glVertex3f(-1.0,-1.0,-1.0);             // Bottom Right Of The Quad (Back)
        glVertex3f(-1.0, 1.0,-1.0);             // Top Right Of The Quad (Back)
        glVertex3f( 1.0, 1.0,-1.0);             // Top Left Of The Quad (Back)
  glEnd();
  glBegin(GL_QUADS);
        glColor3f(0.0,0.0,1.0);                 // Set The Color To Blue
        glVertex3f(-1.0, 1.0, 1.0);             // Top Right Of The Quad (Left)
        glVertex3f(-1.0, 1.0,-1.0);             // Top Left Of The Quad (Left)
        glVertex3f(-1.0,-1.0,-1.0);             // Bottom Left Of The Quad (Left)
        glVertex3f(-1.0,-1.0, 1.0);             // Bottom Right Of The Quad (Left)
  glEnd();
  glBegin(GL_QUADS);
        glColor3f(1.0,0.0,1.0);                 // Set The Color To Violet
        glVertex3f( 1.0, 1.0,-1.0);             // Top Right Of The Quad (Right)
        glVertex3f( 1.0, 1.0, 1.0);             // Top Left Of The Quad (Right)
        glVertex3f( 1.0,-1.0, 1.0);             // Bottom Left Of The Quad (Right)
        glVertex3f( 1.0,-1.0,-1.0);             // Bottom Right Of The Quad (Right)
  glEnd();

  cube_rotation += 5.15 * Speed;


  OpenGLControl1.SwapBuffers;
end;

end.
```



Задание 3.

а.) Разработать примеры древовидных структур данных из окружающей реальности.

б.) Разработать информационную модель для процесса «Зачёт».

в.) Разработать информационную модель для объекта «Студент».

г.) Разработать информационную модель для объекта «ВУЗ».

Задание 4.

а.) Разработать графическую схему переноса начала координат.

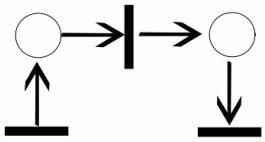
б.) Разработать вербальную схему сжатия координат.

в.) Разработать математическую схему переноса начала координат.

Задание 5.

Построение компьютерной модели сети Петри

Сети Петри — математический аппарат для моделирования динамических дискретных систем. Впервые описаны Карлом Петри в 1962 году.



Сеть Петри представляет собой двудольный ориентированный граф, состоящий из вершин двух типов — позиций и переходов, соединённых между собой дугами. Вершины одного типа не могут быть соединены непосредственно. В позициях могут размещаться метки (маркеры), способные перемещаться по сети.

Событием называют срабатывание перехода, при котором метки из входных позиций этого перехода перемещаются в выходные позиции. События происходят мгновенно, либо разновременно, при выполнении некоторых условий.

Матричный метод основан на выражении, связывающим разметки сети, которые были до и после срабатывания некоторого перехода и матрице D, описывающей работу сети.

Пусть начальная разметка сети равна M_0. Если в сети допустима последовательность срабатывания переходов $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$, то выполняются следующие соотношения

$$M_1 = M_0 + D * t_k$$

$$M_1 = M_l + D * t_{i_2} = M_0 + D * t_{i_1} + D * t_{i_2}$$

$$M_3 = M_2 + D * t_{i_3} = M_1 + D * t_{i_2} + D * t_{i_3} = M_0 + D * t_{i_1} + D * t_{i_2} + D * t_{i_3}$$

$$\ldots\ldots\ldots\ldots$$

$$M_k = M_0 + D * t_{i_1} + D * t_{i_2} + \cdots + D * t_{i_k} = M_0 + D(t_{i_1} + t_{i_2} + \cdots + ti_k)$$

Обозначив $\tau = t_{i_1} + t_{i_2} + \cdots + t_{i_k}$,получим

$$M_k = M_0 + L \quad \tau$$

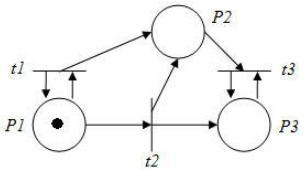
$$M_k - M_0 = L \ 1)\tau$$

Из-за того, что вектор является суммой векторов он должен быть целочисленным неотрицательным вектором. Выражение (1) является системой линейных неоднородных уравнений относительно неизвестных компонент вектора τ. Следует заметить, что целочисленное неотрицательное решение уравнения (1), как правило, не определяет однозначно порядок срабатывания переходов, потому что от порядка суммирования векторов t_{i_k} сумма $\tau = t_{i_1} + t_{i_2} + \cdots + t_{i_k}$ не зависит. Для нахождения

требуемого порядка срабатывания переходов необходимо проводить дополнительные исследования.

Задача 1

Представлена сеть Петри в начальном состоянии.



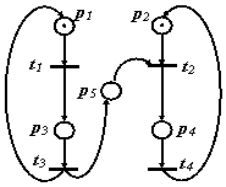
1. Достижимы ли состояния:

1. 021
2. 111
3. 011
4. 223
5. 501

Задача 2

Представлена сеть Петри в начальном состоянии

Найдите вектор весов, для которых данная сеть будет сохраняющейся



Задание 6.

а.) Моделирование блок-схемы алгоритма решения дифференциального уравнения методом Эйлера.

б.) Имитационное моделирование процессов по теории происхождения Вселенной.

Задание 7.

а.) Разработка динамической модели популяции без внутривидовой конкуренции.

б.) Разработка динамической модели популяции с внутривидовой конкуренции.

Задание 8.

Моделирование процесса распределения длительности ожидания в очереди и длительности простоя «продавца» и соответственно средние времена ожидания в системе с одним «прилавком» при различных комбинациях распределений промежутков времен между приходами «покупателей» и времен обслуживания, используя следующие распределения:

а) равновероятное; б) нормальное; в) случайное.

Задание 9.

С помощью пакета Pascal ABC (Lazarus) рассмотреть полет зенитного снаряда длиной L=0,2 м и диаметром D=0,045 м, выпущенного с начальной скоростью v0 = 800 м/с под углом α = 85° к поверхности Земли. Определить, какое расстояние пролетит снаряд, на какую максимальную высоту он поднимется и как изменяется скорость полета со временем.

На заданном расстоянии от пушки находится стена. Известны угол наклона пушки и начальная скорость снаряда. Попадет ли снаряд в стену?

Постановка задачи

Цель моделирования — пользуясь знакомыми физическими законами движения тела, брошенного под углом к горизонту, исследовать данную ситуацию при различных значениях исходных данных.

Объектом моделирования является система, состоящая из двух компонентов: снаряд, брошенный под углом к горизонту, и стена. Подобрать начальную скорость и угол бросания так, чтобы брошенное тело (снаряд) достигло цели.

Разработка модели

Снаряд считаем материальной точкой.

Сопротивлением воздуха и размерами пушки пренебрегаем.

Исходные данные:

α - угол наклона пушки, $0<\alpha<90$ градусов;

V - начальная скорость снаряда (м/с), $0<V<1000$;

S - расстояние от пушки до стены (м), $S>0$;

h - высота стены (м), $h>0$.

Результатом является одно из сообщений: “Снаряд попал в стену”, “Снаряд не попал в стену”.

Для определения попадания снаряда в стену надо найти высоту L снаряда на расстоянии S от пушки: ведь попадание снаряда в стену означает, что $0<L<h$. Перемещение снаряда по горизонтали и вертикали:

$x=V*t*\cos\alpha$

$y=V*t*\sin\alpha-g*t^2/2$, где g-ускорение свободного падения (9,8 м/с^2).

Определим, сколько времени понадобится снаряду, чтобы преодолеть расстояние S:

$t=S/(V*\cos\alpha)$.

Подставив это значение t в выражение для y, получим значение:

$L=S*\operatorname{tg}\alpha-g*S^2/(2*V^2*\cos^2\alpha)$.

Если $L<0$, то снаряд до стены не долетит. Если $L>h$, то снаряд перелетит через стену.

Задание 10.

Схема функционирования автоматизированных обучающих систем (АОС) включает последовательность шагов, каждый из которых направлен на усвоение учащимися определенной порции учебного материала. В структуру типового шага обычно входят три основных функциональных компонента: предъявление определённого объёма теоретической информации, подлежащей усвоению; выполнение упражнений для закрепления теории; оказание помощи учащемуся при выполнении упражнений. На основе перечисленных условий, с помощью пакета Lazarus, произвести лабораторное моделирование данной АОС.

Задание 11. Методы исследования объектов, динамика которых описывается дифференциальными уравнениями с использованием языков программирования

Цель занятия:

1 Получить практические навыки исследования систем (объектов), динамика которых описывается дифференциальными уравнениями 1-го порядка

2. Научиться разрабатывать алгоритм и программу с использованием языков программирования

3. Практически усвоить численные методы Эйлера и Рунге-Кутта для решения дифференциальных уравнений 1 –го порядка.

Задачи занятия:
1. Разработка алгоритма в виде блок - схемы
2. Построение графиков кривых y(x), dy/dx при параметрах a-const и var,
3. Анализ результатов исследований.

Модели объектов исследования

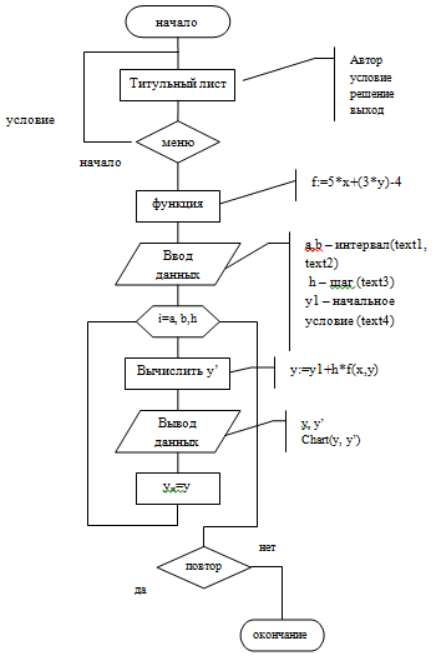
$$a\acute{y}+b\acute{y}+cy=f$$

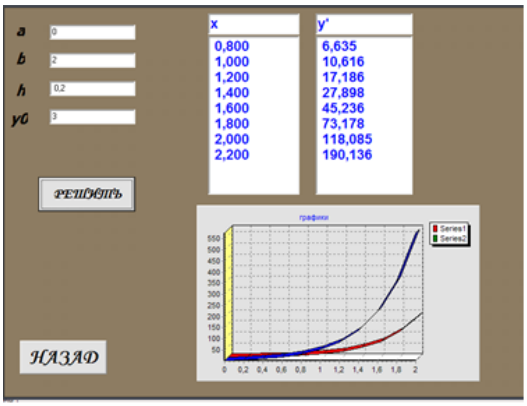
Программа исследования
1. a,c,f - const, t – var (t0 – tk, h= 0.1, 0.01)
2. a,f - const, t -var(t0 – tk, h= 0.1, 0.01), b- var

Условие задачи: составить алгоритм и проект моделирования объекта, динамика которого описывается дифференциальным уравнением 1 –го порядка методом Эйлера

$$5\acute{y}+3y=4$$

Блок-схема алгоритма





Листинг программы (язык программирования Delphi)

```
procedure TForm3.Button2Click(Sender: TObject);
var: x:real; i:integer;
function F(x,y:real):real;
begin
f:=5*x+(3*y)-4;
end;
begin
 a:=StrToFloat(Edit1.text);
 b:=StrToFloat(Edit2.text);
 h:=StrToFloat(Edit3.text);
 y1:=strtofloat(edit4.text);  //шаг
memo1.lines.add(floattostr(x));
memo2.lines.add(floattostr(y));
```

```
x:=a;
  Chart1.Series[0].clear; Chart1.Series[1].clear;
 repeat
  begin
  y:=y1+h*f(x,y);
   Chart1.Series[0].Add(y,FloatToStr(x),clRed);
    Chart1.Series[1].Add(f(x,y),FloatToStr(x),clblue);
   x:=x+h;   y1:=y;
    memo1.lines.add(floattostrf(x,fffixed,7,3));
    memo2.lines.add(floattostrf(y,fffixed,7,3));
   end;
  until x>b;
end;
```

Язык программирования C#

```
private void button1_Click(object sender, EventArgs e)
       {
          double h = 0.01;
          double y = 0;
          double b = 2;
          double x = 0;
          double y1 = 4;
          do
          {
             double f = y + 2*Math.Sin(x);
             x = x + h;
             y = y1 + f * h;
             chart1.Series[0].Points.AddXY(x, y);
             chart1.Series[1].Points.AddXY(x, f);
             y1 = y;
          }
          while (x <= b);
       }
                  // foreach (DataPoint p in chart1.Series[0].Points)
         // {
            //  Вывожу X в лог
            // textBox1.AppendText("X=" + p.XValue.ToString());
            // textBox1.AppendText(Environment.NewLine);
           //  listBox1.Items.Add("X=" + p.XValue.ToString());
         // }
          //   Y является массивом, поэтому пробегаю по массиву
         // foreach (DataPoint yp in chart1.Series[0].Points)
         // {
             //Вывожу Y
            // textBox1.AppendText("Y=" + yp.ToString());
            //  textBox1.AppendText(Environment.NewLine);
             // listBox1.AppendText("Y=" + yp.ToString());
             // listBox1.AppendText(Environment.NewLine);
            // listBox2.Items.Add("Y=" + yp.ToString());
         // }
     // }
           private void chart1_Click(object sender, EventArgs e)
```

```
{
}
private void button2_Click_1(object sender, EventArgs e)
{
   Close();
}        }}
```

Задание 12. Исследовать падение шарика радиуса R с высоты h

Задачи исследования:

1. Получить зависимости h=f(t), v=f(t) при начальных значениях r,v0,h0,k1,k2.
2. Получить семейство графиков h=f(t,r), v=f(t,r).
3. Получить зависимость h=f(r), v=f(r) при t - const

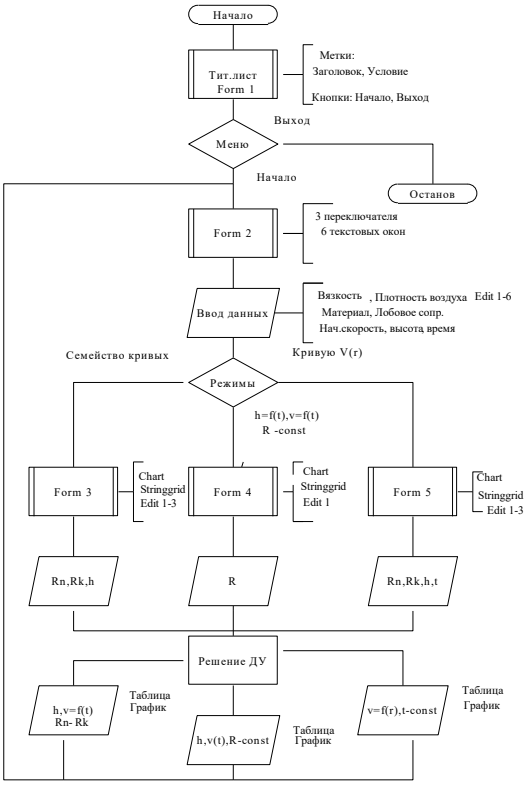
Исходные данные:

1. Модель объекта падения, R - радиус шарика, h - высота падения, v0 - начальная скорость падения, k1 k2 - коэффициенты внешней среды.

$$\begin{cases} \frac{dh}{dt} = \upsilon, \\ \frac{d\upsilon}{dt} = \frac{m \cdot g - k_1 \cdot \upsilon - k_2 \cdot \upsilon^2}{m}, \end{cases}$$

$$k_1 = 6 \cdot \pi \cdot \mu \cdot r, k_2 = 0.5 \cdot c \cdot S \cdot \rho,$$

2. - площадь сечения тела, поперечного по отношению к потоку
3. $\mu = 0.0182 \frac{H \cdot c}{м^2}$ плотность шарика;
4. $\rho_{шарика} = 800 \frac{кг}{м^3}$ – плотность воздуха.
5.
6. c = 0.4 – коэффициент лобового сопротивления шарика;

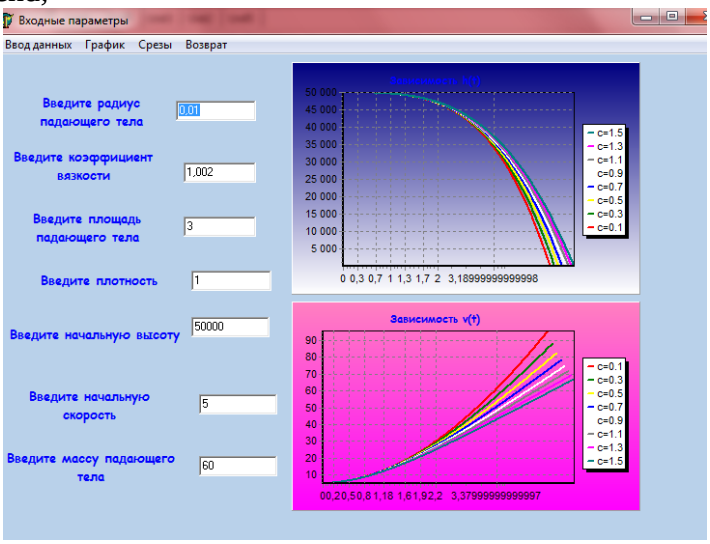



```
var
  Form4: TForm4;
km,r,m,vo,ho,s,p,v,a,dv,da,c:double ;
h1,h2,cn,ck,dc,t,dt,y1,y2,dy1,dy2:double;
ind,k:integer;
arrt,arrv,arra,arrh:array[1..10000] of double;
implementation
uses Unit1, Unit5;
//описание функции
function ff(m,km,r,c,s,p,v:double):double;
begin
   ff:=(m*9.8-6*km*Pi*r*v-0.5*c*s*p*v*v)/m;
end;
//Кнопка возврата к титульному листу
procedure TForm4.N3Click(Sender: TObject);
begin
  form5.show;
  for ind:=0 to 7 do
   begin
     form5.Chart1.SeriesList[ind].Clear;
```

```
    form5.Chart2.SeriesList[ind].Clear;
  end;
  //Входные параметры
  r:=strtofloat(edit1.Text);//радиус падующего тела
  km:=strtofloat(edit2.text);//коэффициент вязкости
  s:=strtofloat(edit3.Text);//площадь падующего тела
  p:=strtofloat(edit4.Text);//плотность среды
  ho:=strtofloat(edit5.Text);//начальная высота
  vo:=strtofloat(edit6.Text);//начальная скорость
  m:=strtofloat(edit7.Text);//масса падающего тела
  c:=0.1;//коэффициент лобового сопротивления среды
  k:=0;//номер графика
  // 1
  while c<=1.5 do
  begin
    ind:=0;
    t:=0;dt:=0.1;
    y1:=vo;
    y2:=ff(m,km,r,c,s,p,vo);
    while t<=10 do
    begin
       form5.Chart1.SeriesList[k].Add(y1,floattostr(t));
       dy1:=dt*y2;
       dy2:=dt*ff(m,km,r,c,s,p,y1);
       y1:=y1+dy1;
       y2:=y2+dy2;
       t:=t+dt;
       end;
    c:=c+0.2;
    k:=k+1;
  end;
  //2
  k:=0;
  t:=30;y1:=vo;y2:=ff(m,km,r,c,s,p,vo);
  while t<=44 do
  begin
     c:=0.1;
     while c<=2 do
     begin
       form5.Chart2.SeriesList[k].Add(y2,floattostr(c));
       dy1:=dt*y2;
       dy2:=dt*ff(m,km,r,c,s,p,y1);
       y1:=y1+dy1;
       y2:=y2+dy2;
       c:=c+0.1;
     end;
     t:=t+2;
     k:=k+1;
  end;
end;
//Кнопка ввода данных
procedure TForm4.N1Click(Sender: TObject);
```

```
//Кнопка построения графика
procedure TForm4.N2Click(Sender: TObject);
begin
  //очистка старых графиков
  for ind:=0 to 7 do
  begin
    form4.Chart1.SeriesList[ind].Clear;
    form4.Chart2.SeriesList[ind].Clear;
  end;
  //Входные параметры
  r:=strtofloat(edit1.Text);//радиус падующего тела
  km:=strtofloat(edit2.text);//коэффициент вязкости
  s:=strtofloat(edit3.Text);//площадь падующего тела
  p:=strtofloat(edit4.Text);//плотность среды
  ho:=strtofloat(edit5.Text);//начальная высота
  vo:=strtofloat(edit6.Text);//начальная скорость
  m:=strtofloat(edit7.Text);//масса падающего тела
  cn:=0.1;ck:=1.5;//интервал лобового сопротивления среды
  dc:=(1.5-0.1)/7;//шаг изменения лобового сопротивления
  k:=0;//номер графика
  // цикл по лобовому сопротивлению
  while cn<=ck do
  begin
      v:=vo;//первая производная расстояния по времени- начальная скорость
      a:=1; //вторая производная расстояния по времени-ускорение
      ind:=1;//счетчик записи данных в массивы
      t:=0;//счетчик подчета времени падения тела
      h1:=ho;//высота с которой падает тело
      h2:=0; // тело лежит на земле
      //цикл времени падения тела
      while h1>=h2 do
      begin
          //запись в массивы полученных данных
          arrh[ind]:=h1;//высота
          arrv[ind]:=v;//скорость
          arrt[ind]:=t;//время
          //вывод графиков
          form4.Chart1.SeriesList[k].Add(arrh[ind],floattostr(arrt[ind]));
          form4.Chart2.SeriesList[k].Add(arrv[ind],floattostr(arrt[ind]));
          // решение дифференциального уравнения методом Эйлера
          dv:=0.01*a;
          da:=0.01*ff(m,km,r,cn,s,p,a);
          v:=v+dv;
          a:=a+da;
          h1:=h1-arrv[ind]*arrt[ind];//изменение высоты за время t
          t:=t+0.01;//изменение времени
          ind:=ind+1;
      end;
    //переход к следующему коэффициенту лобового сопротивления
    // и соответсвенно к следующему графику
    k:=k+1;
    cn:=cn+dc;
```

```
  end;
end;
procedure TForm4.N4Click(Sender: TObject);
begin
form4.Hide;
form5.hide;
form1.Show;
end;
```



Задание 13. Исследовать движение исследовательского зонда вертикально вверх с летящего самолета

Условие

Промоделировать движение исследовательского зонда, «выстреленного» вертикально вверх с летящего над землей самолета. В верхней точке траектории над зондом раскрывается парашют, и он плавно спускается на землю.

Входные параметры.

Математическая модель свободного падения тела - уравнение второго закона Ньютона с учетом двух сил, действующих на тело -силы тяжести и силы сопротивления среды. Движение является одномерным; проецируя силу, скорость и перемещение на ось, направленную вертикально вверх, получаем

$$\frac{dh}{dt} = v,$$
$$\frac{dv}{dt} = \frac{mg - k_1 v - k_2 v^2}{m}.$$

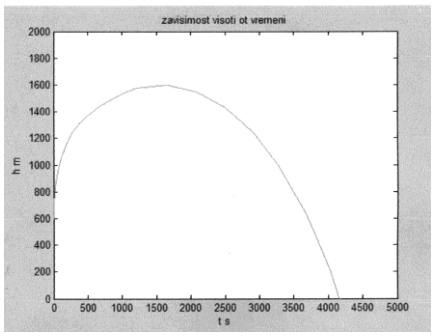
Входные параметры модели:

- начальная высота тела;
- начальная скорость тела;
- величины, определяющие коэффициенты сопротивления среды

```
function FmO=fun(t,y)
tv=10;
a=100;
g=9,8
mn=10000;
 mk=3000;
m=mn-a*t;
```

```
 ifm>mk
else
m=mk end; mm=m; if t>=tv
p=0;
m=500; else
p=1000000;
m=mam; end;
FmO=[(p*cos(y(2))-cos(y(2))-g*sin(y(2)))/m;
((p*sin(y(2))+sin(y(2)))/m-
g*cos(y(2)))/y(l);-y(l)*cos(y(2))/10;y(l)*sin(y(2))/10];
Y0=[1000 pi/2 0 600];
[T,Y]=odel5s(@fun,[0 1000],Y0);
plot(Y(:53),Y(:,4),'g');
hold on;
title("zavisimost visoti ot vremeni");
xlabel(fts');
y1aЬel(Ът');
axis([0,5000,0,2000]);
```

Результат.



Задание 14. Математическая модель. Метод наименьших квадратов (Линейная регрессия) Создайте компьютерную модель в среде VBA, которая на основе метода наименьших квадратов, для приведенной пары измерительных значений, находит значения R и b.

Метод наименьших квадратов (МНК) – один из наиболее часто используемых методов при обработке эмпирических данных, построении и анализе физических, биологических, технических, экономических и социальных моделей.

С помощью МНК решают задачу выбора параметров функции (заранее заданного вида) для приближённого описания зависимости величины *y* от величины *x*.

Исходные данные могут носить самый разнообразный характер и относиться к различным отраслям науки или техники, например:

- ✓ зависимость продолжительности службы электрических ламп (y) от поданного на них напряжения (x);
- ✓ зависимость пробивного напряжения конденсаторов (y) от температуры окружающей среды (x);
- ✓ зависимость предела прочности стали (y) от содержания углерода (x);
- ✓ зависимость показателей безработицы (y) и инфляции (x);
- ✓ зависимость роста преступности (y),% и роста безработицы (x),%
- ✓ зависимость цен товара (y) от спроса (x) на этот товар;
- ✓ зависимость частного потребления (y) от располагаемого дохода (x);

- ✓ зависимость температура воздуха (y) от высоты над уровнем моря (x) и другие зависимости.

Индивидуальные задания

<table>
<tr><th>Вариант</th><th>З а д а н и е</th></tr>
<tr><td>1</td><td>В таблице приведены данные численности занятого населения (x, млн.) и валового выпуска продукции (y, у.е.).
<table>
<tr><td>x_i</td><td>80</td><td>82</td><td>83</td><td>84</td><td>85</td><td>86</td><td>88</td><td>89</td><td>90</td><td>91</td></tr>
<tr><td>y_i</td><td>32</td><td>34</td><td>35</td><td>36</td><td>36</td><td>37</td><td>38</td><td>40</td><td>39</td><td>40</td></tr>
</table>
В предположении, что между x и y существует линейная зависимость, определить параметры линейной регрессии $y = kx + b$ методом наименьших квадратов. Спрогнозировать валовой выпуск продукции в случае, если занятое население увеличится на 10% по сравнению с последними данными (90 млн.)</td></tr>
<tr><td>2</td><td>В таблице приведены данные об уровне безработицы (x) и уровне преступности (y) в некотором населенном пункте.
<table>
<tr><td>x_i</td><td>0,5</td><td>1,2</td><td>2</td><td>3,1</td><td>4</td><td>5,2</td><td>5,9</td><td>6,1</td><td>6,2</td><td>6,3</td></tr>
<tr><td>y_i</td><td>4,25</td><td>4,32</td><td>4,4</td><td>4,51</td><td>4,6</td><td>4,72</td><td>4,79</td><td>4,9</td><td>5,0</td><td>5,2</td></tr>
</table>
В предположении, что между x и y существует линейная зависимость, определить параметры линейной регрессии $y = kx + b$ методом наименьших квадратов. Спрогнозировать уровень преступности в случае, когда безработица отсутствует.</td></tr>
<tr><td>3</td><td>В таблице приведены данные о динамике темпов прироста курса акций (y, в %) за определенный период (t – одна неделя).
<table>
<tr><td>t_i</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr>
<tr><td>y_i</td><td>10,2</td><td>8,3</td><td>5,4</td><td>4,1</td><td>2,2</td><td>0</td><td>-1,6</td><td>-3,9</td><td>-5,9</td><td>-7,8</td></tr>
</table>
В предположении, что между t и y существует линейная зависимость, определить параметры линейной регрессии $y = kt + b$ методом наименьших квадратов. Сделать выводы о возможной динамике темпов прироста на 12 неделе.</td></tr>
<tr><td>4</td><td>Торговое предприятие имеет сеть, состоящую из 10 магазинов, информация о деятельности которых: годовой товарооборот (y, млн. руб.) и торговая площадь (x, тыс. м^2) представлена в таблице.
<table>
<tr><td>x_i</td><td>0,24</td><td>0,41</td><td>0,55</td><td>0,58</td><td>0,78</td><td>0,94</td><td>0,98</td><td>1,21</td><td>1,28</td><td>1,32</td></tr>
<tr><td>y_i</td><td>19,8</td><td>38,1</td><td>41,0</td><td>43,1</td><td>56,3</td><td>68,5</td><td>75,0</td><td>89,1</td><td>91,1</td><td>91,3</td></tr>
</table>
В предположении, что между x и y существует линейная зависимость, определить параметры линейной регрессии $y = kx + b$ методом наименьших квадратов. Спрогнозировать годовой товарооборот в случае, если торговая площадь составит ровно 1 тыс. м^2.</td></tr>
<tr><td>5</td><td>Показатели по объему производства (x, у.е.) и затратам (y, тыс. руб.), взятые из отчетной ведомости предприятия за 10 месяцев, приведены в таблице.
<table>
<tr><td>x_i</td><td>2,32</td><td>2,33</td><td>2,38</td><td>2,41</td><td>2,44</td><td>2,48</td><td>2,51</td><td>2,55</td><td>2,58</td><td>2,60</td></tr>
<tr><td>y_i</td><td>427</td><td>430</td><td>440</td><td>444</td><td>448</td><td>455</td><td>460</td><td>462</td><td>465</td><td>466</td></tr>
</table>
Полагая, что зависимость между x и y задается формулой $y = kx + b$, где b – постоянные затраты в тыс. руб., k – переменные затраты на 1 условную единицу</td></tr>
</table>

<table>
<tr><td></td><td>продукции, определить параметры k и b методом наименьших квадратов. Рассчитать возможные затраты на производство в случае, если объем производства достигнет 3 у.е.</td></tr>
<tr><td>6</td><td>В таблице приведена динамика валового выпуска (y, у.е.) за последние 10 лет (x – год)
<table>
<tr><td>x_i</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr>
<tr><td>y_i</td><td>178</td><td>182</td><td>190</td><td>199</td><td>200</td><td>213</td><td>220</td><td>231</td><td>235</td><td>242</td></tr>
</table>
Предполагая линейную зависимость валового выпуска от времени, определить параметры линейной регрессии $y = kx + b$, используя метод наименьших квадратов. Получить прогноз валового выпуска на следующий год.</td></tr>
<tr><td>7</td><td>Показатели стоимости основных производственных фондов (x, млн. руб.) и среднесуточной производительности (y, тонны) приведены в таблице.
<table>
<tr><td>x_i</td><td>2,1</td><td>2,3</td><td>2,4</td><td>2,9</td><td>4,1</td><td>4,7</td><td>5,5</td><td>7,2</td><td>10,2</td><td>14,3</td></tr>
<tr><td>y_i</td><td>27</td><td>29</td><td>30</td><td>35</td><td>36</td><td>44</td><td>47</td><td>55</td><td>63</td><td>73</td></tr>
</table>
Предполагая линейную зависимость y от x, определить параметры линейной регрессии $y = kx + b$, используя метод наименьших квадратов. Получить прогноз среднесуточной производительности при стоимости основных производственных фондов 16 млн. руб.</td></tr>
<tr><td>8</td><td>В таблице приведены данные о количестве пропусков занятий (x) студентом в течение учебного семестра и результатах (y, %) написания экзаменационного теста.
<table>
<tr><td>x_i</td><td>1</td><td>3</td><td>5</td><td>6</td><td>8</td><td>10</td><td>12</td><td>14</td><td>15</td><td>16</td></tr>
<tr><td>y_i</td><td>85</td><td>75</td><td>70</td><td>60</td><td>50</td><td>40</td><td>20</td><td>10</td><td>10</td><td>5</td></tr>
</table>
Предполагая наличие линейной зависимости между x и y определить параметры линейной регрессии $y = kx + b$, используя метод наименьших квадратов. Получить прогноз результатов теста при отсутствии пропусков.</td></tr>
<tr><td>9</td><td>В таблице приведены данные об объемах производства (x, у.е.) некоторой компании в течение 10 месяцев и соответствующей операционной прибылью (y,тыс. руб.).
<table>
<tr><td>x_i</td><td>500</td><td>520</td><td>523</td><td>530</td><td>550</td><td>555</td><td>560</td><td>562</td><td>565</td><td>570</td></tr>
<tr><td>y_i</td><td>61</td><td>66,8</td><td>67</td><td>69</td><td>74</td><td>76,7</td><td>78</td><td>79</td><td>79,3</td><td>81</td></tr>
</table>
В предположении, что между x и y существует линейная зависимость, определить параметры линейной регрессии $y = kx + b$ методом наименьших квадратов. Сделать выводы о возможной месячной прибыли, если объем производства достигнет 600 у.е.</td></tr>
</table>

Задание 15. Компьютерные методы моделирования

Пример 1. Рассмотрим задачу о распространении эпидемии инфекционного заболевания в рамках одной популяции. Пренебрегая неоднородностью распределения популяции по пространству, введем две функции x(t) и y(t), характеризующие число незараженных и зараженных особей в момент времени t. В начальный момент времени t=0 известны начальные значения x(0)=n и y(0)=a. Для того чтобы построить математическую модель, воспользуемся гипотезой: инфекция передается при встрече зараженных особей с незараженными.

$$x(t) = \frac{n(n+a)}{n + ae^{\beta(n+a)t}}$$

Пример 2. Проект «Численность популяций» на языке Turbo Delphi

1. Поместить на форму текстовые поля для ввода:
• значений коэффициентов а,Ь, с и /, влияющих на изменение численности жертв: EditA, EditB, EditC и EditF;
• значений коэффициентов dvig, влияющих на изменение численности хищников: EditD и EditG;
• начальной численности популяций жертв и хищников: EditX и EditY;
• количества рассматриваемых жизненных циклов (лет) EditN.
2. Поместить на форму надписи для вывода численности популяций через заданное количество лет:
• при неограниченном росте LabelNR;
• при ограниченном росте LabelOR;
• при ограниченном росте с отловом LabelORO;
• в модели «жертва-хищник» LabeIX YnLabelY X.
3. Поместить на форму графическое поле Imagel (например, 300, 500), в котором будут строиться графики зависимости численности популяций от количества жизненных циклов (лет).
4. Поместить на форму надписи для вывода обозначений и поясняющих текстов.
5. Прежде всего, необходимо объявить переменные: var
A: real; //коэффициент роста популяции
B: real; //коэффициент уменьшения популяции
C: real; //коэффициент отлова
D: real; //коэффициент уменьшения численности
//хищников в отсутствие жертв G: real; //коэффициент увеличения численности
//хищников при наличии жертв F: real; //коэффициент уменьшения численности
//жертв при наличии хищников X: real; //первоначальное количество жертв
Y: real; //первоначальное количество хищников
N: integer; //количество циклов (лет)
I: integer; //счетчик цикла
6. Поместить на форму кнопку Buttonl и начать создание событийной процедуры TForml .ButtonlClick () . Присвоить переменным значения, вводимые в текстовые поля, с использованием функций преобразования типов данных StrToFloat()и StrToInt():

```
procedure TForml.ButtonlClick{Sender: TObject) ; begin
//Ввод данных A:=StrToFloat(EditA.Text)
B:=StrToFloat(EditB.Text)
C:=StrToFloat(EditC.Text)
D:=StrToFloat(EditD.Text)
G:=StrToFloat(EditG.Text)
F:=StrToFloat(EditF.Text)
X:=StrToFloat(EditX.Text)
Y: =StrToFloat(EditY.Text)
N: =StrToInt(EditN.Text) ;
```

7. В событийной процедуре установить ширину линий рисования на холсте, равную, например, 3 пикселям:
//Установка ширины линии рисования Imagel.Canvas.Pen.Width:=3;
8. Ввести программный код модели неограниченного роста, где:

• задается начальная точка графика с использованием метода MoveTo ();
• задается цвет графика путем задания значения свойства Color;
• в цикле вычисляется численность популяции и строится график с использованием метода LineTo O ;
• конечная численность населения выводится на надпись LabelNR с использованием функции преобразования типов данных FloatToStr (X):

```
//Неограниченный рост X:=StrToFloat(EditX.Text);
Imagel.Canvas.MoveTo(0,250);
Imagel.Canvas.Pen.Color:=clDkGray;
For I:=l to N Do begin
Imagel.Canvas.LineTo(25*1-25,250-Round(25*X)+25);
X:=A*X; end;
LabelNR.Caption:=FloatToStr(X);
```

9. Ввести программный код модели ограниченного роста:

```
//Ограниченный рост X:=StrToFloat(EditX.Text) ;
Imagel.Canvas.MoveTo(0,250);
Imagel.Canvas.Pen.Color:=clDkGray;
For I:=l to N Do begin
Imagel.Canvas.LineTo(25*1-25,250-Round(25*X)+25);
X:=(A-B*X)*X; end;
LabelOR.Caption:=FloatToStr(X);
```

10. Ввести программный код модели ограниченного роста с отловом:

```
//Ограниченный рост с отловом X:=StrToFloat(EditX.Text);
Imagel.Canvas.MoveTo(0,250);
Imagel.Canvas.Pen.Color:=clBlue;
For I:=l to N Do begin
Imagel.Canvas.LineTo(25*1-25,250-Round(25*X)+25);
X:=(A-B*X)*X-C;
LabelORO.Caption:=FloatToStr(X); end;
LabelORO.Caption:=FloatToStr(X);
```

11. Ввести программный код модели «жертва-хищник* для вычисления численности жертв:

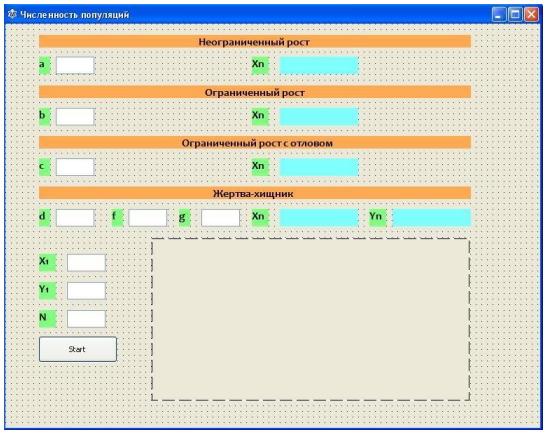
```
//Жертвы
X:=StrToFloat(EditX.Text);
Y:=StrToFloat(EditY.Text) ;
Imagel.Canvas.MoveTo(0,250) ;
Imagel.Canvas.Pen.Color:=clGreen;
For I:=l to N Do begin
Imagel.Canvas.LineTo(25*1-25,250-Round(25*X)+25); X:=(A-B*X)*X-C-F*X*Y;
Y:=D*Y+G*X*Y;
end;
LabelX_Y.Caption:=FloatToStr(X);
```

12. Ввести код модели «жертва-хищник» для вычисления численности хищников:

```
//Хищники
X:=StrToFloat(EditX.Text);
Y:=StrToFloat(EditY.Text);
Imagel.Canvas.MoveTo(0, 250) ;
Imagel.Canvas.Pen.Color:=clRed;
For I: =1 to N Do begin
Imagel.Canvas.LineTo(25*1-25,250-Round(25*Y)+25) ; X:=(A-B*X)*X-C-F*X*Y;
```

```
Y:=D*Y+G*X*Y; end;
LabelY_X.Caption:=FloatToStr(Y) ;
end;
end.
```

Запустить проект и ввести значения коэффициентов, начальное количество жертв и хищников и количество жизненных циклов (лет). (Для простоты примем начальные количества жертв и хищников за единицу.) Щелкнуть по кнопке Пуск. Графики покажут динамику развития популяций, а на надписи будут выведены конечные значения численности популяций (см. рис. ниже).



Лабораторная работа № 16. Задачи по моделированию из различных предметных областей - биология

Составить модель биоритмов для конкретного человека от указанной текущей даты (дня отсчета) на месяц вперед с целью дальнейшего анализа модели. На основе анализа индивидуальных биоритмов прогнозировать неблагоприятные дни, выбирать благоприятные дни для разного рода деятельности.

Постановка задачи

Цель моделирования — составить модель биоритмов для конкретного человека от указанной текущей даты на месяц вперед с целью ее дальнейшего анализа.

Объектом моделирования является любой человек, для которого известна дата его рождения.

В жизни человека бывают творческие и бесплодные, счастливые и несчастливые дни, дни, когда он бывает в приподнятом или в подавленном настроении. Существует теория, что жизнь человека подчиняется циклическим процессам, называемым биоритмами. Эти циклы описывают три стороны самочувствия человека: физическую, эмоциональную и интеллектуальную. Биоритмы характеризуют подъемы и спады нашего состояния. Многие полагают, что “взлетам” графика, представляющего собой синусоидальную зависимость, соответствуют более благоприятные дни. Дни, в которые график переходит через ось абсцисс, являются критическими, т.е. неблагоприятными. Если у каких-либо двух (или у всех трех) биологических ритмов совпадают критические дни, то такой день называется дважды (трижды) критическим.

За точку отсчета трех биоритмов берется день рождения человека.

Физический биоритм характеризует жизненные силы человека, т.е. его физическое состояние. Периодичность ритма 23 дня.

Эмоциональный биоритм характеризует внутренний настрой человека, его возбудимость, способность эмоционального восприятия окружающего. Продолжительность периода эмоционального цикла равна 28 дням.

Третий биоритм характеризует мыслительные способности, интеллектуальное состояние человека. Цикличность его — 33 дня.

5. Образовательные технологии, применяемые при освоении дисциплины

Предусматривается широкое использование в учебном процессе активных и интерактивных форм: организация дискуссий и обсуждений спорных вопросов, использование метода мозгового штурма и метода проектов, а также технология электронного портфолио.

При обучении лиц с ограниченными возможностями и инвалидов используются подходы, способствующие созданию безбарьерной образовательной среды: технологии дифференциации и индивидуализации обучения, сопровождение тьюторами в образовательном пространстве; увеличивается время на самостоятельное освоение материала.

Удельный вес занятий, проводимых в интерактивных формах, определяется главной целью программы, особенностью контингента обучающихся и содержанием конкретных дисциплин, и в целом в учебном процессе они должны составлять не менее 50% аудиторных занятий.

В рамках практической подготовки по данной дисциплине используются проектные задания, выполнение которых направлено на формирование таких профессиональных действий как способность использовать математический аппарат, методы программирования и современные информационно-коммуникационные технологии для решения практических задач получения, хранения, обработки и передачи информации

Примеры проектных заданий приведены в фондах оценочных средств.

6. Учебно-методическое обеспечение самостоятельной работы студентов. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины.

Фонд оценочных средств дисциплины включает в себя: задания контрольных работ, контрольные вопросы, задания для самостоятельных работ, задания для написания рефератов.

В рамках самостоятельной работы студенты изучают дополнительную литературу, интернет ресурсы по тематике курса.

Для реализации принципа индивидуального подхода на занятиях студентам предлагаются темы индивидуальных докладов и рефератов, написание которых практикуется в учебном процессе в целях приобретения студентом необходимой профессиональной подготовки, развития навыков самостоятельного научного поиска; изучения литературы по выбранной теме; анализа различных источников и точек зрения, обобщения материала, выделения главного, формулирования выводов и т.п. С помощью рефератов и докладов студент глубже постигает наиболее сложные проблемы курса; учится лаконично излагать свои мысли, правильно оформлять работу, докладывать результаты своего труда. Содержание реферата и доклада

должно соответствовать теме и ее плану. Процесс написания реферата и доклада включает в себя: 1) выбор темы; 2). подбор литературы и иных источников, их изучение; 3) составление плана; 4) введение (краткое введение, в котором обосновывается актуальность темы); 5) основной текст; 6) заключение; 7) список использованной литературы.

Студенты выполняют задания самостоятельно, пользуясь интернет-ресурсами, дополнительной литературой.

Задания для самостоятельной работы

Методические указания.

Задания студенты выполняют во внеурочное время, самостоятельно. Результаты предоставляются преподавателю в электронном виде.

Решение задач и обсуждение вопросов по темам:

1. История появления моделирования.

Основные понятия теории моделирования.

Цели и задачи моделирования.

Материальные (физические) и идеальные модели.

Когнитивные, содержательные, концептуальные, формальные модели.

Подходы и программные средства при структурно-функциональном моделировании.

2. Имитационное моделирование как специфический вид компьютерного моделирования.

Достоинства и недостатки имитационного моделирования.

Инструментарии имитационного моделирования.

3. Этапы построения моделей.

Основные модели, используемые в системном анализе.

Классификация систем по различным признакам.

Сложные системы: определения.

Факторы, действующие на функционирование сложных систем.

Задачи исследования сложных систем.

4. Моделирование в школьном курсе информатики. Решение задач.

5. Сети Петри, раскрашенные сети Петри.

Понятие систем массового обслуживания.

Классификация систем массового обслуживания.

6. Бизнес-процессы.

Анализ бизнес-процессов.

Оптимизация бизнес-процессов.

Математическое моделирование.

Индивидуальная самостоятельная работа
Тема "Изучение дополнительного материала"

Изучить дополнительный материал. Выступить с сообщением и презентацией.

Вопросы, подлежащие усвоению и для самоконтроля.

1. Понятие модели и моделирования.
2. Этапы компьютерного моделирования.
3. Классификация абстрактных (идеальных) моделей.
4. Основные виды моделирования.
5. Материальное и идеальное моделирование.
6. Вербальные, математические, информационные модели.
7. Адекватность, потенциальность модели.
8. Графическое представление объекта черным ящиком и системой.
9. Взаимосвязь моделей.
10. Что такое «информационная модель»?
11. Что такое «объект» с точки зрения информационного моделирования?
12. Приемы представления информационной модели в виде таблицы.
13. Примеры объектов, имеющих связи «один к одному», «один ко многим»
14. Приведите примеры информационных моделей типа «очередь», «цикл», «дерево».
15. Понятие системы Классификация систем по различным признакам. Уровни качества систем с управлением.
16. Методы оценивания систем Методы качественного оценивания систем. Методы количественного оценивания систем. Методы измерения компьютерных систем.
17. Сложные системы. Динамические системы. Объектно-ориентированное моделирование. Подходы к визуальному моделированию сложных динамических систем.
18. Общая постановка задачи линейного программирования.
19. Системы программирования для моделирования предметно-коммуникативных сред.
20. Компьютерная обучающая программа.
21. Экспертные системы.
22. Выбор инструментальной среды моделирования.
23. Обследование объекта моделирования.
24. Структура компьютерных обучающих программ.
25. Специальные инструментальные средства педагогических исследований.

Примеры проектных заданий:

1. Транспортная задача Задание Производственное объединение в своём составе имеет n филиалов Ai , i=1, 2,…, n, которые производят однородную продукцию в количестве ai , i=1, 2,…, n. Эту продукцию получают m потребителей Bj , j=1, 2, …, m, расположенных в разных местах. Их потребности соответственно равны bj , j=1, 2, …, m. Тарифы перевозок единицы продукции от каждого из филиалов потребителям задаются матрицей Cij (i=1, 2,…, n; j=1, 2, …, m). Составить план прикрепления

получателей продукции к ее поставщикам, при котором общая стоимость перевозок была минимальной. 1. Построить математическую модель задачи. 2. Создать на рабочем листе Excel таблицу для ввода исходных данных. 3. Заполнить таблицу исходными данными и необходимыми формулами. 4. Найти решение задачи средствами надстройки Поиск решения.

2. Постройте в эмуляторе модели задач рассмотренных на лекции:

1. Задача о сборке узлов предприятиями A, B, C.

2. Задача о работе станции скорой помощи.

3. Домашнее задание: Задача о поломанных блоках.

4. Домашнее задание: Светофор.

5. Домашнее задание:

Постройте в виде Сети Петри схему принятия решения об обеде в столовой, используя при этом примерно следующие позиции: осознание голода, выбор зала, занятие очереди, выбор блюда, проверка наличия блюда (блюдо есть), проверка наличия денег, переход к другому блюду, переход в другой зал, отказ от обеда, расчёт с кассиром, выбор столовых приборов, выбор свободного стола и занятие стола, потребление пищи, принятие решения о покупке дополнительных блюд, принятие решения о передачи грязной посуды на мойку, передача грязной посуды на мойку, выход из столовой.

2. Метод анализа иерархий (МАИ).

Для выбранного задания в соответствии с методикой системного моделирования провести исследование, используя МАИ для поддержки процесса принятия решений по следующему алгоритму:

1) определить систему с обоснованием ее структуры, функций, основных свойств и характеристик; сформулировать цель системы, а также проблемную ситуацию, как несоответствие целевого и желаемого состояния системы;

2) построить иерархическую структуру исследуемой системы, т.е. сформировать цель системы, определить пути достижения этой цели (альтернативы), выявить критерии эффективности выбора альтернатив;

3) построить матрицы парных сравнений (см. задание);

4) рассчитать векторы приоритетов;

5) проверить согласованность суждений ЛПР(лиц принмающих решение) и при необходимости пересмотреть суждения;

6) выявить наилучшую альтернативу (или проранжировать альтернативы по степени значимости);

7) добавить в построенную иерархию новую альтернативу (с собственными экспертными суждениями) и проверить изменение значимостей ранее исследуемых альтернатив.

Обработка имеющейся информации должна осуществляться с помощью программных средств (MS Excel и др.) с возможностью адаптивно реагировать на изменения исходных данных. Для этого следует запрограммировать все алгебраические операции МАИ в MS Excel.

3. Построение нейронных сетей

4. Интеллектуальные модели в Deductor Academic

Контрольная работа по теме: «Моделирование и формализация»

Задание 1. Ответьте на вопросы.

Вариант 1.

Какие пары объектов не находятся в отношении "объект - модель"?

а) компьютер – данные;

б) компьютер – его функциональная схема;

в) компьютер – программа;

г) компьютер – алгоритм.

Вариант 2.

Какая модель компьютера является формальной (полученной в результате формализации)?

а) техническое описание компьютера;

б) фотография компьютера;

в) логическая схема компьютера;

г) рисунок компьютера.

Вариант 3.

Какая модель является статической (описывающей состояние объекта)?

а) формула химического соединения;

б) формулы равноускоренного движения;

в) формула химической реакции;

г) второй закон Ньютона.

Вариант 4.

Какая модель является динамической (описывающей изменение состояния объекта)?

а) формула химического соединения;

б) формула закона Ома;

в) формула химической реакции;

г) закон Всемирного тяготения.

Вариант 5.

Формальной информационной моделью является:

а) анатомический муляж;

б) техническое описание компьютера;

в) рисунок функциональной схемы компьютера;

г) программа на языке программирования.

Вариант 6.

В информационных моделях разомкнутых систем управления отсутствует:

а) управляющий объект;

б) управляемый объект;

в) канал управления;

г) канал обратной связи.

Вариант 7.

Какие из приведенных ниже определений понятия «модель» верные?

а) модель - это некое вспомогательное средство, объект, который в определенной ситуации заменяет другой объект;

б) модель - это новый объект, который отражает некоторые стороны изучаемого объекта или явления, существенные с точки зрения цели моделирования;

в) модель - это физический или информационный аналог объекта, функционирование которого - по определенным параметрам - подобно функционированию реального объекта;

г) модель некоторого объекта - это другой объект (реальный, знаковый или воображаемый), отличный от исходного, он обладает существенными для целей моделирования свойствами и в рамках этих целей полностью заменяет исходный объект.

Вариант 8.

Если материальная модель объекта — это его физическое подобие, то информационная модель объекта - это его:

а) описание;

б) точное воспроизведение;

в) схематичное представление;

г) преобразование.

Вариант 9.

Какое из утверждений верно?

а) информационные модели одного и того же объекта, пусть даже предназначенные для разных целей, должны быть во многом сходны;

б) информационные модели одного и того же объекта, предназначенные для разных целей, могут быть совершенно разными.

Вариант 10.

Может ли передаваться информация от человека к человеку и от поколения к поколению без использования моделей?

а) нет, без моделей никогда не обойтись;

б) да, иногда, например, генетическая информация;

в) да, чаще всего знания передаются без использования каких-либо моделей;

г) модели передаются по наследству.

Вариант 11.

Верно ли, что моделирование представляет собой один из основных методов познания, способ существования знаний?

а) нет; б) да.

Вариант 12.

Какие из приведенных ниже моделей являются вероятностными? Выбрать три правильных ответа.

а) прогноз погоды;

б) отчет о деятельности предприятия;

в) схема функционирования устройства;

г) научная гипотеза;

д) оглавление книги;

е) проведение факультетского социологического опроса.

Вариант 13.

Правильно ли определен вид следующей модели: «Компьютерная модель полета мяча, брошенного вертикально вверх, - динамическая формализованная модель, имитирующая поведение данного объекта»?

а) нет; б) да.

Вариант 14.

Какие из приведенных ниже моделей являются статическими? Выбрать три правильных ответа.

а) карта местности;

б) граффити на стене;

в) программа, имитирующая движение стрелок циферблата на экране дисплея;

г) план праздничных мероприятий;

д) график изменения температуры воздуха в течение суток.

Вариант 15.

Какие из утверждений являются верными? Выбрать два правильных ответа.

а) математическая формула является информационной моделью;

б) график движения поезда - табличная статическая модель;

б) план дома - графическая детерминированная модель, описывающая структуру объекта;

г) турнирная таблица чемпионата по футболу - эталонная динамическая модель.

Задание 2. Постройте модель согласно варианту.

Вариант 1.

а.) Разработка модели идеального студента, модели парохода, модели броуновского движения.

б.) Объекты и модели: аудитория, шашки.

в.) Примеры различных видов моделей в естественных науках.

г.) Примеры различных видов моделей в технических науках.

Вариант 2.

а.) Разработать примеры древовидных структур данных из окружающей реальности.

б.) Разработать информационную модель для процесса «Экзамен».

в.) Разработать информационную модель для объекта «Преподаватель».

г.) Разработать информационную модель для объекта «Факультет».

Вариант 3.

а.) Разработать блок-схему переноса начала координат.

б.) Разработать блок-схему сжатия координат.

в.) Разработать блок-схему переноса начала координат.

Вариант 4.

а.) Моделирование блок-схемы алгоритма решения дифференциального уравнения методом Эйлера.

б.) Имитационное моделирование процесса «парадокса близнецов» из теории относительности.

Вариант 5.

а.) Разработка динамической модели популяции без внутривидовой конкуренции.

б.) Разработка динамической модели популяции с внутривидовой конкуренции.

Вариант 6.

Моделирование процесса распределения длительности ожидания в очереди и длительности простоя «продавца» и соответственно средние времена ожидания в системе с одним «прилавком» при различных комбинациях распределений промежутков времен

между приходами «покупателей» и времен обслуживания, используя следующие распределения:

а) равновероятное; б) нормальное.

Вариант 7.

Разработка качественной информационной модели специалиста, подготовленного к коммуникативной деятельности.

Вопросы для проведения промежуточной аттестации (экзамен)

1. Модели. История формирования и основные понятия.
2. Задачи, виды и методы моделирования. Моделирование как метод познания.
3. Классификация моделей. Примеры моделей для различных целей моделирования.
4. Математическое моделирование.
5. Имитационное моделирование.
6. Математические модели в гуманитарных исследованиях.
7. Компьютерная реализация моделей.
8. Примеры компьютерных моделей в гуманитарных исследованиях.
9. Реализация методов обработки графической информации в табличном процессоре.
10. Основные прикладные программы моделирования и экранного просмотра трёхмерных графических и анимационных файлов.
11. Компьютерное моделирование с приложениями к социальным, биологическим и экологическим задачам.
12. Этап формализации. Параметры модели. Классификация моделей по свойствам их параметров. Анализ результатов моделирования.
13. Различные подходы к классификации математических моделей.
14. Аналитическое моделирование в физике. Примеры использования визуализации в моделировании.
15. Классификация моделей по общематематическим свойствам: линейные и нелинейные модели. Примеры. Интегрирование дифференциальных уравнений.
16. Численное моделирование. Численный эксперимент. Его взаимосвязи с теорией и лабораторным экспериментом.
17. Математические модели в экологии. Основные понятия экологии. Особенности и направления использования математических моделей в биологии.
18. Моделирование стохастических систем, основные понятия. Метод статистических испытаний (метод Монте-Карло).
19. Моделирование в среде Lazarus

Контрольные вопросы

1. Дать общую формулировку задач оптимизации в форме модели динамического программирования.
2. Дать понятие простой и сложной динамической системы.
3. Привести инструментальные средства реализации имитационных моделей.
4. Привести классификации моделей по свойствам их параметров.
5. Привести примеры связей между хаосом и самоорганизацией.
6. Перечислить системы программирования при реализации математических моделей.
7. Примеры компьютерных моделей в гуманитарных исследованиях.
8. Основные принципы системного анализа.
9. Общие методы моделирования дискретных и непрерывных случайных величин.
10. Численный эксперимент: его взаимосвязи с теорией и лабораторным экспериментом.
11. Что такое синергетика?
12. Приведите известные методы статистических испытаний.
13. Основные критерии древовидной информационной модели.
14. Определение модели типа граф.
15. Понятие определения “генераторы случайных чисел”.
16. Особенности использования метода Монте-Карло.
17. Может ли передаваться информация от человека к человеку и от поколения к поколению без использования моделей?
18. Охарактеризовать основные пакеты прикладных программ.
19. Примеры визуализации сложных процессов в компьютерном моделировании.
20. Привести основные алгоритмы построения графиков функций, траекторий движения объектов.
23. Если различие между абстрактным и компьютерным моделированием?
24. Привести примеры виртуальных образовательных систем.
25. Привести примеры стохастических моделей в образовании.

Темы рефератов

- История появления моделирования.
- Основные понятия теории моделирования.
- Цели и задачи моделирования.
- Материальные (физические) и идеальные модели.
- Когнитивные, содержательные, концептуальные, формальные модели.
- Подходы и программные средства при структурно-функциональном моделировании.

- Имитационное моделирование как специфический вид компьютерного моделирования.
- Достоинства и недостатки имитационного моделирования.
- Инструментарии имитационного моделирования.
- Этапы построения моделей.
- Основные модели, используемые в системном анализе.
- Классификация систем по различным признакам.
- Сложные системы: определения.
- Факторы, действующие на функционирование сложных систем.
- Задачи исследования сложных систем.
- Этапы при моделировании сложных систем.
- Понятие о модельном времени.
- Сетевые методы.
- Сети Петри, раскрашенные сети Петри.
- Математическое моделирование

7. Данные для учета успеваемости студентов в БАРС

Таблица 1.1 Таблица максимальных баллов по видам учебной деятельности.

Семестр	Лекции	Лабораторные занятия	Практические занятия	Самостоятельная работа	Автоматизированное тестирование	Другие виды учебной деятельности	Промежуточная аттестация	Итого
6	10	35	0	15	0	10	30	100

Программа оценивания учебной деятельности студента
6 семестр

Лекции: посещаемость, активность; за один семестр – от 0 до 10 баллов.

Лабораторные занятия: Контроль выполнения заданий в течение одного семестра – от 0 до 35 баллов.

Практические занятия: не предусмотрены.

Самостоятельная работа: Контроль выполнения заданий для самостоятельной работы, рефератов, докладов в течение семестра – от 0 до 15 баллов.

Автоматизированное тестирование: не предусмотрено

Другие виды учебной деятельности:
Подготовка двух рефератов – от 0 до 6 баллов.
Выполнение контрольной работы – от 0 до 4 баллов.

Таким образом, за семестр студент может получить всего от 0 до 10 баллов за другие виды учебной деятельности.

Промежуточная аттестация:

При определении разброса баллов при аттестации преподаватель может воспользоваться следующим примером ранжирования:

- 25-30 баллов – ответ на «отлично»
- 19-24 баллов – ответ на «хорошо»
- 11-18 баллов – ответ на «удовлетворительно»
- 0-10 баллов – неудовлетворительный ответ.

Таким образом, максимально возможная сумма баллов за все виды учебной деятельности студента за шестой семестр по дисциплине «Компьютерное моделирование и системы программирования» составляет 100 баллов.

Таблица пересчета полученной студентом суммы баллов по дисциплине «Компьютерное моделирование и системы программирования» в оценку (экзамен):

90-100 баллов	«отлично»
76-89 баллов	«хорошо»
61-75 баллов	«удовлетворительно»
0-60 баллов	«не удовлетворительно»

8. Учебно-методическое и информационное обеспечение дисциплины «Компьютерное моделирование и системы программирования».

а) литература:

Ачкасов, В. Ю. Программирование на Lazarus [Электронный ресурс] / В. Ю. Ачкасов. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 520 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73711.html

Петлина, Е. М. Компьютерное моделирование [Электронный ресурс] : учебное пособие для СПО / Е. М. Петлина. — Электрон. текстовые данные. — Саратов : Профобразование, 2019. — 131 c. — 978-5-4488-0250-8. — Режим доступа: http://www.iprbookshop.ru/83270.html

Тупик, Н. В. Компьютерное моделирование [Электронный ресурс] : учебное пособие / Н. В. Тупик. — 2-е изд. — Электрон. текстовые данные. — Саратов : Вузовское образование, 2019. — 230 c. — 978-5-4487-0392-8. — Режим доступа: http://www.iprbookshop.ru/79639.html

б) программное обеспечение и Интернет-ресурсы

1. Библиотека научной и студенческой информации. http://www.bibliofond.ru/
2. Методическая копилка учителя информатики. http://www.metod-kopilka.ru/
3. Видео уроки в сети интернет. http://www.videouroki.net/
4. Учебный Центр «Специалист» при МГТУ имени Н. Э. Баумана – www.specialist.ru
5. Дистанционное обучение. Компьютерное моделирование. http://do.rksi.ru/library/courses/km
6. Боресков А.В. Шикин Е.В. Компьютерная графика. Динамика, реалистические изображения. Учебное пособие. http://www.biblioclub.ru/book/54731
7. Компьютерное моделирование. Лекции и задания для лабораторных занятий. http://www.fizmat.vspu.ru/books/model-m5
8. Математическое моделирование в естественных науках. Виртуальные лаборатории http://mathmod.aspu.ru
9. Мехмат МГУ - http://mech.math.msu.su/russian/welcome.htm

Лицензионное программное обеспечение:
Office Professional Plus 2007 (44107825)
Бесплатное программное обеспечение
Lazarus: https://www.lazarus-ide.org/
PascalABC.NET: http://pascalabc.net/

9. Материально-техническое обеспечение дисциплины «Компьютерное моделирование и системы программирования»

Для проведения практических занятий требуются компьютерные классы с программным обеспечением (Microsoft Office, Lazarus, PascalABC), рассчитанные на обучение группы студентов из 10–15 человек, удовлетворяющие санитарно-гигиеническим требованиям, работающие под управлением операционной системы Windows с подключением к Internet.

Для проведения групповых лекционных занятий необходим проектор, подключенный к компьютеру, и экран. Требования к программному обеспечению:

- Операционная система Windows;
- Microsoft Office Power Point.
- Lazarus
- PascalABC

Реализация практической подготовки в рамах учебных занятий запланирована на базе на кафедры информационных систем и технологий в обучении.

Программа составлена в соответствии с требованиями ФГОС ВО с учетом Примерной ООП ВО по направлению 44.03.01 – Педагогическое образование и профилю подготовки «Информатика».

Автор
к. п. н., доцент _____ В. А. Векслер

Программа одобрена на заседании кафедры информационных систем и технологий в обучении от 31 августа 2021 года, протокол № 1.