Оглавление

[Введение 2](#_Toc118671308)

[1. Аналитический обзор 5](#_Toc118671309)

[1.1 Определение популяции 5](#_Toc118671310)

[1.2 Модель Ферхюльста (рождаемость и смертность с учетом роста численности) 6](#_Toc118671311)

[1.3. Решение дифференциальных уравнений 7](#_Toc118671312)

[1.4. Метод Эйлера 8](#_Toc118671313)

[1.5. Улучшенный метод Эйлера 9](#_Toc118671314)

[2. Язык Pascal 11](#_Toc118671315)

[3. Язык С 13](#_Toc118671316)

[4. Среда быстрой разработки Delphi 15](#_Toc118671317)

[5. Алгоритм решения задачи 16](#_Toc118671318)

[6. Тестирование программ 17](#_Toc118671319)

[Выводы 18](#_Toc118671320)

[Список использованной литературы 19](#_Toc118671321)

[Предметный указатель 20](#_Toc118671322)

# Введение

Для математического моделирования природных процессов чаще используются не трансцендентные, а дифференциальные, в том числе обыкновенные дифференциальные уравнения. Наиболее известные примеры - кинетика химических (биохимических) реакций, динамика биологических популяций, движение космических объектов, задачи экономики природопользования

Целью данной курсовой работы является разработка программ на языках С, Pasсal и в среде Delphi для описания процессов развития динамики популяции. Данные процессы описываются моделью Ферхюльда, представляющей собой дифференциальное уравнение первого порядка. Решение данного уравнения будем искать с помощью метода Эйлера. В ходе курсового проектирования будет проведен обзор теоретических вопросов, разработан алгоритмы решения поставленной задачи.

Для математического моделирования природных процессов чаще используются не трансцендентные, а дифференциальные, в том числе обыкновенные дифференциальные уравнения. Наиболее известные примеры - кинетика химических (биохимических) реакций, динамика биологических популяций, движение космических объектов, задачи экономики природопользования

Целью данной курсовой работы является разработка программ на языках С, Pasсal и в среде Delphi для описания процессов развития динамики популяции. Данные процессы описываются моделью Ферхюльда, представляющей собой дифференциальное уравнение первого порядка. Решение данного уравнения будем искать с помощью метода Эйлера. В ходе курсового проектирования будет проведен обзор теоретических вопросов, разработан алгоритмы решения поставленной задачи.

Для математического моделирования природных процессов чаще используются не трансцендентные, а дифференциальные, в том числе обыкновенные дифференциальные уравнения. Наиболее известные примеры - кинетика химических (биохимических) реакций, динамика биологических популяций, движение космических объектов, задачи экономики природопользования

Целью данной курсовой работы является разработка программ на языках С, Pasсal и в среде Delphi для описания процессов развития динамики популяции. Данные процессы описываются моделью Ферхюльда, представляющей собой дифференциальное уравнение первого порядка. Решение данного уравнения будем искать с помощью метода Эйлера. В ходе курсового проектирования будет проведен обзор теоретических вопросов, разработан алгоритмы решения поставленной задачи.

Для математического моделирования природных процессов чаще используются не трансцендентные, а дифференциальные, в том числе обыкновенные дифференциальные уравнения. Наиболее известные примеры - кинетика химических (биохимических) реакций, динамика биологических популяций, движение космических объектов, задачи экономики природопользования

Целью данной курсовой работы является разработка программ на языках С, Pasсal и в среде Delphi для описания процессов развития динамики популяции. Данные процессы описываются моделью Ферхюльда, представляющей собой дифференциальное уравнение первого порядка. Решение данного уравнения будем искать с помощью метода Эйлера. В ходе курсового проектирования будет проведен обзор теоретических вопросов, разработан алгоритмы решения поставленной задачи.

Для математического моделирования природных процессов чаще используются не трансцендентные, а дифференциальные, в том числе обыкновенные дифференциальные уравнения. Наиболее известные примеры - кинетика химических (биохимических) реакций, динамика биологических популяций, движение космических объектов, задачи экономики природопользования

Целью данной курсовой работы является разработка программ на языках С, Pasсal и в среде Delphi для описания процессов развития динамики популяции. Данные процессы описываются моделью Ферхюльда, представляющей собой дифференциальное уравнение первого порядка. Решение данного уравнения будем искать с помощью метода Эйлера. В ходе курсового проектирования будет проведен обзор теоретических вопросов, разработан алгоритмы решения поставленной задачи.

Для математического моделирования природных процессов чаще используются не трансцендентные, а дифференциальные, в том числе обыкновенные дифференциальные уравнения. Наиболее известные примеры - кинетика химических (биохимических) реакций, динамика биологических популяций, движение космических объектов, задачи экономики природопользования

Целью данной курсовой работы является разработка программ на языках С, Pasсal и в среде Delphi для описания процессов развития динамики популяции. Данные процессы описываются моделью Ферхюльда, представляющей собой дифференциальное уравнение первого порядка. Решение данного уравнения будем искать с помощью метода Эйлера. В ходе курсового проектирования будет проведен обзор теоретических вопросов, разработан алгоритмы решения поставленной задачи.

Когда говорят о знании, всегда предполагается, что оно должно быть *обоснованным* (выводимым). Аристотель связывал понятие знания (если не "фронезиса" и не "техне", то, во всяком случае, "эпистемы") с указанием оснований или причин: "Мы полагаем, что знаем каждую вещь безусловно, а не софистически, привходящим образом, когда полагаем, что знаем причину...". И Витгенштейн, размышляя о различиях в употреблении слов "вера" и "знание", писал, что "в зале суда никого не убедило бы простое заверение свидетеля: “Я знаю...”. Должно быть показано, что свидетель был в состоянии знать".

# 1. Аналитический обзор

## 1.1 Определение популяции

Популяция - это совокупность особей одного вида, находящихся во взаимодействии между собой и совместно заселяющих общую территорию.

Основные характеристики популяции: численность, плотность, рождаемость, смертность, темп роста и др. Кроме того, популяции имеют определенную структуру: возрастную (соотношение особей разного возраста), сексуальную (соотношение полов), пространственную (колонии, семьи, стаи и пр.). Так возрастная структура популяции является важной характеристикой влияющей на рождаемость и смертность.

Соотношение разных возрастных групп в популяции определяет ее способность к размножению в данный момент, причем обычно в быстро растущих популяциях значительную долю составляют молодые особи. Соотношение молодых особей у промысловых птиц и пушных зверей к численности всей популяции определяет во время охотничьего сезона размер допустимых квот на отстрел или отлов. Соотношение полов также имеет практическое значение (стада домашних животных, когда без ущерба динамики численности популяции можно изъять определенное количество особей того или иного вида).

Все живые организмы теоретически способны к очень быстрому увеличению численности. При неограниченных ресурсах и отсутствии гибели от болезней, хищников и т.п. даже при низкой исходной численности популяция любого вида за сравнительно короткий срок может так вырасти, что покроет весь земной шар сплошным слоем. Способность к увеличению численности за данный промежуток времени называют биотическим потенциалом вида. У разных видов биотический потенциал разный: у крупных млекопитающих численность может возрастать в год лишь в 1,05 - 1,1 раза, а у мелких насекомых (рачков, дафний) численность в год может возрасти в 1010-1030 раз. А у бактерий и одноклеточных водорослей еще быстрее. Во всех этих случаях, при идеальных условиях численность будет расти в геометрической прогрессии и график изменения численности будет представлять собой экспоненту. Рост численности в геометрической прогрессии называется экспоненциальным ростом. В лабораторных условиях наблюдать экспоненциальный рост можно в популяциях дрожжей, водоросли хлореллы, бактерий на начальных стадиях роста.

В природе экспоненциальный рост наблюдается при вспышках саранчи, непарного шелкопряда и других насекомых. Экспоненциально может расти численность животных, заселенных в новую местность, где у них мало врагов и много пищи (классический пример - рост численности кроликов, завезенных в Австралию). Во всех этих случаях экспоненциальный рост наблюдается в течение коротких промежутков времени, после чего скорость роста численности снижается.

## 1.2 Модель Ферхюльста (рождаемость и смертность с учетом роста численности)

Постановка задачи.

Как правило, численность популяции зависит не только от рождаемости и смертности, но и от ограниченности пищевых и других ресурсов. Вскоре за созданием модели Мальтуса, бельгийский математик Ферхюльст задался вопросом: будет ли население Бельгии расти неограниченно? Ответом на этот вопрос было создание новой модели динамики численности популяции при ограниченных ресурсах, описываемая следующим уравнением:

dN/dt=r\*N-m\*N2 (1),

где r - удельная скорость роста численности,

N - численность популяции,

m - число встреч членов популяции, при котором они могут конкурировать за какой-либо ресурс.

Уравнение это отличается от уравнения экспоненциального роста (уравнения Мальтуса) выражением m\*N2, которое как раз и отражает ограниченность ресурсов. Перепишем уравнение (1) так:

dN/dt=N(r-m\*N).(2)

Выражение в скобках - это удельная скорость роста популяции. Причем чем больше численность популяции (N), тем меньше скорость роста. Если в правой части уравнения вынести за скобки выражение r

dN/dt=N\*r(1-N\*m/r)

и обозначить m/r за 1/K, то уравнение (1) можно переписать так:

dN/dt=N\*r(1-N/K) (3)

При малых N значением N/K можно пренебречь, и тогда рост численности идет по экспоненциальному закону, при возрастании N и неизменном K рост численности будет замедляться, и при N близком к К рост остановится.

Величину К называют емкостью среды. Она отражает возможности среды обитания предоставить популяции нужные для ее роста ресурсы.

Уравнение (3) графически отображается в виде S- образной кривой. Эта кривая называется логистической кривой, а рост численности, соответствующий уравнению (3) - логистическим. Исследуя кривую, можно сказать, что максимальная скорость роста достигается, когда численность равна K/2. В некоторый момент численность стабилизируется и остается постоянной величиной. Популяции, существующие в условиях ограниченных ресурсов, часто хорошо подчиняются правилам логистического роста. Например, когда овцы были завезены в Тасманию, рост их стада описывался логистической кривой. Но правила логистического роста приложимы не ко всем случаям. Например, у размножающихся половым путем видов, при слишком малой численности мала вероятность встреч особей разного пола и размножение может вообще прекратиться. Для реализации модели в среде электронных таблиц уравнение (3) следует представить в дискретном виде:

N(i+1)=N(i)\*r\*(1-N(i)/K) (4)

где N(i) - численность популяции в i-й момент времени;

r - удельная скорость роста популяции (рождаемость/смертность);

К - емкость среды.

Для этой модели нужно взять побольше временной диапазон ,т.к. она наглядна на длинном промежутке времени.

## 1.3. Решение дифференциальных уравнений

При решении научных и инженерно-технических задач часто бывает необходимо математически описать какую-либо динамическую систему. Лучше всего это делать в виде дифференциальных уравнений (ДУ) или системы дифференциальных уравнений. Приведём примеры:

1. Кинетика химических реакций.
2. Явления переноса. Наиболее часто такая задача возникает при решении проблем, связанных с моделированием кинетики химических реакций и различных явлений переноса (тепла, массы, импульса) – теплообмена, перемешивания, сушки, адсорбции, при описании движения макро- и микрочастиц.
3. Динамика популяций. Изменения численности популяций хищников и их жертв можно описать, например, двумя несложными дифференциальными уравнениями, предполагая, что известна зависимость рождаемости и гибели отдельных видов животных от их количества.
4. Распространение «нового». Распространение эпидемий, слухов, опыта и мнений в определённых условиях также можно описать дифференциальными уравнениями. Для эпидемий следует учитывать также различия между «здоровыми», «инфицированными», «резистентными», и «изолированными» популяциями. Кинетические закономерности для каждой из этих популяций в соответствии с выбранной моделью могут различаться.

При решении дифференциальных уравнений могут встретиться 3 типа задач:

1. Дифференциальные уравнения с заданными начальными условиями, т.е необходимо найти такое частное решение дифференциального уравнения, которое удовлетворяет определенным начальными условиям, заданным в одной точке. Такого рода задача встречается при решении ДУ, которые описывают, например, кинетику химических реакций. В этом случае известны концентрации веществ в начальный момент времени (t = 0), и необходимо найти концентрации веществ через некоторый промежуток времени (t). В качестве примера можно так же привести задачу о теплопереносе или массопереносе (диффузии), уравнение движения материальной точки под действием сил и т.д.
2. Краевая задача. (в этом случае известны значение функции или её производных в определённых точках, и необходимо найти решение между этими точками.) Сами дополнительные условия в этом случае называются краевыми (граничными) условиями. Краевая задача может решаться для ДУ не ниже 2-го порядка. Ниже приведен пример ДУ второго порядка с граничными условиями-заданы значения функции в двух различных точках
3. Задачи на собственное значение. Задачи этого типа очень похожи на краевую задачу. При их решении необходимо найти, при каких значениях какого-либо параметра решение ДУ удовлетворяет граничным условиям (собственные значения) и функции, которые являются решением ДУ при каждом значении параметра (собственные функции). Например, многие задачи квантовой механики являются задачами на собственные значения.

Рассмотрим два метода, которые позволяют решать дифференциальные уравнения: метод Эйлера и улучшенный метод Эйлера.

## 1.4. Метод Эйлера

Исторически первым и наиболее простым способом численного решения задачи Коши для ДУ первого порядка является метод Эйлера. В его основе лежит аппроксимация производной отношением конечных приращений зависимой (y) и независимой (x) переменных между узлами равномерной сетки.

Например, даны некоторое дифференциальное уравнение и начальное условие, необходимо найти такую функцию, которая удовлетворяла бы как дифференциальному уравнению, так и начальному условию. Рассмотрим такой численный метод на простых и наглядных примерах. Для этого заменим производную в левой части дифференциального уравнения на близкое ей по величине отношение конечных приращений зависимой и независимой переменной.

Если теперь преобразовать полученное уравнение, то получится итерационная формула, по которой можно вычислить yi+1, если известно yi в точке хi.

Сравнивая формулу Эйлера с общим выражением, полученным ранее, видно, что для приближённого значения интеграла в методе Эйлера используется простейшая формула интегрирования – формула прямоугольников по левому краю отрезка. Суть метода Эйлера заключается в замене функции y(x) на отрезке интегрирования прямой линией, касательной к графику в точке x=xi. Если искомая функция сильно отличается от линейной на отрезке интегрирования, то погрешность вычисления будет значительной. Ошибка метода Эйлера прямо пропорциональна шагу интегрирования: ошибка ~ h.

Процесс вычислений строится следующим образом: при заданных начальных условиях x0 и y0 можно вычислить

Таким образом строится таблица значений функции y(x) с определенным шагом (h) по x на отрезке [x0, xN]. Ошибка в определении значения y(xi) при этом будет тем меньше, чем меньше выбрана длина шага h (что определяется точностью формулы интегрирования).

При больших h метод Эйлера весьма неточен. Он дает все более точное приближение при уменьшении шага интегрирования. Если отрезок [xi, xi+1] слишком велик, то каждый участок [xi, xi+1] разбивается на N отрезков интегрирования и к каждому их них применяется формула Эйлера с шагом , то есть шаг интегрирования h берется меньше шага сетки, на которой определяется решение.

Если программа не предусматривает автоматического выхода из бесконечного цикла и прекращения вычислений по достижении заданной точности результата, то блок-схема алгоритма решения дифференциальных уравнений методом Эйлера состоит из трёх частей: первая часть соответствует основной программе, в ней вызывается подпрограмма, реализующая алгоритм Эйлера. Эта подпрограмма в свою очередь при каждом обращении к ней вызывает подпрограмму ДУ, в которой вычисляется правая часть дифференциального уравнения.

Если решается тип задач с заданными начальными условиями, то из исходных данных сначала вводится начальное условие, т.е надо указать начальное значение (х) и соответствующее ему начальное значение (у). Кроме того, необходимо ввести значение (х), для которого надо найти значение (у).

Если хотят рассчитать значение (у) не в одной, а в нескольких точках, то необходимо запустить программу соответствующее число раз (с тем же начальным условием). Затем полученные точки вместе с начальным условием можно нанести на диаграммную бумагу и построить график искомой функции.

Правильность решения можно легко проверить, подставив эту функцию в исходное уравнение.

## 1.5. Улучшенный метод Эйлера

Метод Эйлера для расчёта дифференциальных уравнений имеет небольшую точность расчёта. Как было показано ранее, точность расчёта у него зависит от размера шага линейно, зависимость точности от шага – первой степени. То есть, чтобы увеличить точность в десять раз, надо уменьшить шаг в десять раз. На практике интересуются более совершенными методами. Вопрос стоит так: можно ли увеличить точность на порядок, но при этом сэкономить на количестве вычислений? Да, такие методы есть. Модифицированный метод Эйлера имеет точность второго порядка. В методе Эйлера производная берётся в начале шага и по ней прогнозируется движение системы на конец шага, считая, что во время шага производная неизменна. То есть в течение всего шага производная считается той, какой она была в самом начале шага. Это и есть основной источник неточности.

Улучшение метода состоит в том, что берётся производная не в начале шага, а как промежуточное или среднее на разных участках одного шага. В разных вариантах метода вычисляют несколько производных в разных частях шага и усредняют их. Конечно, в этом случае число вычислений увеличивается, но не в десятки раз ,а вот точность возрастает на порядок, в этом и состоит выигрыш.

Пусть, требуется решить уравнение y' = f(x, y, t). Идея уточнённого метода Эйлера состоит в том, что производную вычисляют не в i-ой точке, а между двумя соседними точками: i и i + 1. Данная процедура состоит из следующих шагов:

1. в точке i вычисляют значение производной: f(xi, yi, t);
2. делают пол-шага и вычисляют значение функции на середине отрезка: yi + 1/2 = yi + f(xi, yi, t) · Δt/2;
3. в точке i + 1/2 вычисляют производную: f(xi + 1/2, yi + 1/2, t + Δt/2);
4. делается полный шаг из точки i в точку i + 1 по значению уточненной производной: yi + 1 =yi + f(xi + 1/2, yi + 1/2, t + Δt/2) · Δt;
5. значение t увеличивается: t := t + Δt. Вся процедура повторяется сначала.

На рисунке показано, какой будет ошибка, если шаг делается по значению производной, вычисленной в точке i, как это делается в методе Эйлера. Эта ошибка может быть достаточно велика!

На рисунке показано, как по значению производной, вычисленной в точке i, делается полшага до точки t + Δt/2 и в этой точке вычисляют новую производную. Касательная в этой точке будет другой.

Далее переносят линию B обратно в точку t. Это соответствует тому, что из точки t снова делается, — но уже полный, — шаг Δt до точки t + Δt по направлению, соответствующему линии C. Линия C параллельна B. То есть значение производной в точке t берется искусственно равным производной в точке t + Δt/2. Ошибка расчета во многих случаях при этом уменьшается. Точность метода Эйлера можно повысить, если воспользоваться для аппроксимации интеграла более точной формулой интегрирования - формулой трапеции.

Данная формула оказывается неявной относительно yi+1 (это значение есть и в левой и правой части выражения), т.е является уравнением относительно yi+1 , решать которое можно, например, численно, применяя какой-либо итерационный метод. Однако, можно поступить иначе и приблизительно вычислить значение функции в узле i+1 с помощью обычной формулы Эйлера, которое затем использовать при вычислении. Таким образом получается улучшенный метод Эйлера (или его ещё называют метод Эйлера с пересчётом). Для каждого узла интегрирования производится следующая цепочка вычислений.

Благодаря более точной формуле интегрирования, погрешность этого метода пропорциональна уже квадрату шага интегрирования. Правую часть дифференциального уравнения вычисляют исходя из известных уже значений. Метод решения дифференциальных уравнений такого типа работает как при положительных приращениях, так и при отрицательных. С помощью так называемого прогноза можно найти вспомогательную величину yi+1. В качестве прогноза мы принимаем рекуррентную формулу Эйлера. За прогнозом следует коррекция, которая позволяет оценить правую часть дифференциального уравнения между обеими точками. Если применить этот простой метод прогноза и коррекции, то ошибка уменьшается пропорционально h в кубе. Подход, использованный в данном методе, используется для построения так называемых методов прогноза и коррекции.

# 2. Язык Pascal

В 1970 г. возник еще один новый элегантный язык, который в последующие годы оказал влияние на программистов во всем мире.

Язык назывался Паскаль - в честь французского философа и математика XVII в. Блеза Паскаля, а его автором был Никлаус Вирт (Niklaus Wirth).

Он начал писать Паскаль в 1968 г., когда Хоару так и не удалось воспрепятствовать утверждению Алгола-68. В то время Вирт был профессором информатики в Федеральном техническом университете в Швейцарии (где в первый раз собирался комитет по Алголу-58) и нуждался в инструменте для обучения студентов навыкам программирования.

Вирта не удовлетворял не только новый Алгол, но и все «ныне используемые основные языки программирования, свойства и конструкции которых зачастую нельзя объяснить логически и убедительно и которые нередко просто оскорбляют здравый смысл». Еще подростком Вирт страстно увлекался конструированием радиоуправляемых моделей самолетов. Это привело к тому, что в 1963 г. он получил степень бакалавра по электротехнике в Калифорнийском университете. Он подошел к разработке языка, как инженер подошел бы к конструированию машины. «Если смотреть на программирование как на проектирование машины, - писал он, - то необходимость точности становится более очевидной».

Новый язык Вирта отражал эту точку зрения. В отдельном разделе в начале программы на Паскале программист должен определить все переменные и задать явно типы их значений, т. е. указать, будет ли содержимое конкретной переменной рассматриваться, например, как целое число или как строка литер.

Паскаль также поощряет использование логической структуры, которая делит программу на небольшие простые подзадачи.

Вводя такой раздел, Паскаль ограничивает свободу программиста. Но одновременно язык способствует более строгому стилю программирования, который сокращает количество ошибок.

Структура Паскаля делает программы легко читаемыми, позволяя даже людям, не писавшим ту или иную программу, обнаруживать и исправлять имеющиеся в ней ошибки и вносить изменения.

Благодаря этому Паскаль особенно удобен для изучения теории и практики программирования, но не для реальных применений. Например, Паскаль, подобно Алголу, не имеет в самом формальном языке аппарата ввода и вывода. По мнению Вирта, машинно-зависимые действия, такие, как ввод данных с клавиатуры или запись данных во внешнюю память, не соответствуют логике изучения программирования; компиляторы, написанные для реальных компьютеров, должны иметь особые средства реализации таких возможностей.

Успех Паскаля намного превысил скромные ожидания Вирта. Коллеги по обе стороны Атлантического океана приняли его как средство обучения программированию будущих специалистов по информатике.

Возможно, самое важное состояло в том, что Паскаль стал путеводной звездой зарождающегося в то время движения за структурное программирование, которое обрело силу в конце 70-х годов. Оно ставило своей целью пересмотр способа разработки программ, и начало ему было положено публикацией в 1972 г. книги «Структурное программирование» англичанина К.А.Р. Хоара, норвежца О.Дж. Дала и выдающегося голландского специалиста по информатике Эдсгера Дейкстры.

Упор на логику. Хотя эту фразу разные программисты могут понимать совершенно по-разному, однако по существу она характеризует систематический математический подход к созданию программного обеспечения, в частности требует разделения программы на небольшие логически увязанные задачи, как это делается в Паскале.

Одна из специфических целей структурного программирования - уменьшение использования так называемых безусловных переходов, или операторов GOTO.

В большинстве основных языков программирования этот оператор используется для передачи управления из одного места программы в другое, которое может быть расположено достаточно далеко по тексту. Хотя оператор GOTO и удобен при написании программ, но обычно усложняет их чтение, а значит, увеличивает вероятность пропуска возможной ошибки.

Хотя Pascal в целом очень удобный и полезный язык, у него есть свои недостатки, перечень которых приведен ниже.

1. В определении этого языка имеется некоторое противоречие между идеологией самого языка и его реализацией. Например, конструкция forward нужна только для того, чтобы компиляция могла выполняться в один проход, - это следствие представлений о том, что таким образом достигается максимальная эффективность компиляции. Но это не всегда верно. Например, компилятор PL/C для языка PL/I совершал три прохода и вместе с тем являлся одним из самых эффективных среди наиболее распространенных компиляторов своего времени . Кроме того, в настоящее время при использовании недорогих быстродействующих компьютеров скорость компиляции не имеет большого значения.
2. Возможно, самой главной слабостью языка Pascal является то, что массивы рассматриваются как отдельные типы, а не как агрегация различных объектов одного типа. Это приводит к тому, что, например, array [1. .10] of Integer и аггау[1. .20] of integer представляют собой/разные типы данных. В результате алгоритмы обработки массивов усложняются, поскольку массивы различных размеров невозможно передать общей подпрограмме (например, подпрограмме перемножения матриц). Строки реализованы как массивы символов, что также затрудняет их обработку в случае строк различной длины.
3. Синтаксис определения процедуры в Pascal выглядит следующим образом: заголовок процедуры локальные переменные локальные параметры begin тело\_процедуры end Поскольку в программе может содержаться большое количество вложенных локальных процедур, то определение локальной переменной, которая используется в какой-либо процедуре, оказывается (синтаксически) сильно отдаленным от места ее использования в теле подпрограммы. Это приводит к затруднениям при создании документации и чтении больших программ на Pascal.
4. Возможности, предоставляемые языком, должны выполняться не с помощью пропуска некоторой информации, а явным указанием этой информации. В Pascal передача параметров нарушает это правило. Все параметры в Pascal передаются по значению, если только в списке параметров не указан явным образом атрибут var, который означает, что соответствующий параметр должен передаваться по ссылке. Многие начинающие программисты (в том числе один из авторов этой книги) часами рассматривали листинги программ, стараясь обнаружить ошибку, связанную с пропуском ключевого слова var.
5. Pascal был реализован таким образом, что компиляция программы представляла собой единый процесс, то есть не была предусмотрена возможность компилировать отдельные программные модули. В большинстве реализаций, однако, эту проблему удалось решить: было принято соглашение, что допускаются дополнительные внешние процедуры, аналогичные заголовочным файлам с расширением .h в языке С. Но такая нестандартная реализация ограничивает возможность перенесения программ на Pascal на другие машины.
6. Хотя в Pascal допускается определение новых типов данных для поддержки абстракций, в нем фактически не предусмотрена возможность инкапсуляции и сокрытия информации. (Это замечание является скорее не критикой данного языка, а комментарием, касающимся общего уровня развития программирования в 1970 г., когда создавался Pascal.)

# 3. Язык С

Язык - Си (C) появился в 1969г. Он родился в «Белл телефон лабораторис» (Bell Telefon Laboratoris) (научно-исследовательской фирме, принадлежащей корпорации «Американ телефон энд телеграф» (American Telephone and Telegraph), AT&T, и расположенной в Мюррей-хилл, шт. Нью-Джерси) в атмосфере, которую один из участников разработки определил как «благотворное пренебрежение».

Никто из руководителей компании не заказывал нового языка, на него не было особого спроса. Язык Си возник лишь как результат дружеского соревнования внутри небольшой группы программистов лаборатории, искавших язык, пригодный для экспериментов с новым программным обеспечением. «Не было ни проектов, ни спецификаций, ни требований, - вспоминал член группы, - почти все родилось просто из обсуждений».

Название «Си» (C) появилось так же случайно, как и сам язык. Он оказался преемником ранее созданного языка для внутреннего использования, получившего название Би (B) (B, C - вторая и третья буквы латинского алфавита).

Отчасти язык Би основывался на созданном в Кембриджском университете языке БКПЛ (BCPL, от Basic Combined Programming Language - базовый комбинированный язык программирования), который в свою очередь был потомком Алгола-60.

Язык Си разработал в 1972 г. Денис Ритчи (Dennis Ritchie), 31-летний специалист по системному программированию, который, получив степень бакалавра по прикладной математике в Гарвардском университете, поступил в 1968 г. на службу в «Белл телефон лабораторис».

Ритчи надеялся, что его новый язык пригодится для программирования новой операционной системы «Юникс» (UNIX), работа над которой тоже не входила в официальные планы компании.

Автор «Юникса» Кен Томпсон (Kenneth Thompson) считался среди коллег лучшим программистом в мире. Операционная система - это совокупность программ, выполняющих такие важные задачи, как прием, сохранение и выдача информации, и обеспечивающих взаимодействие аппаратуры компьютера с прикладными программами.

Ради повышения скорости работы операционных систем они традиционно писались на языке низкого уровня - ассемблере, но язык Си настолько хорошо зарекомендовал себя, что в конечном счете на нем было написано более 90% всего кода центральной программы (ядра) системы «Юникс».

Как только система «Юникс» получила признание, язык Си обрел популярность как так называемый язык среднего уровня, в котором удобство, краткость и мобильность языков высокого уровня сочетаются с возможностью непосредственного доступа к аппаратуре, что традиционно обеспечивалось ассемблером.

Язык C включает основные конструкции потока управления, требуемые для хорошо структурированных программ: группирование операторов, принятие решений (if), циклы с проверкой завершения в начале (while, for) или в конце (do) и выбор одного из множества возможных вариантов (switch).

В языке C существуют указатели и реализована возможность адресной арифметики. Аргументы передаются функциям через копирование значений аргумента. Имена массивов передаются с помощью указания начала массива, таким образом аргументы типа массив вызываются по ссылке.

К любой функции можно обратиться рекурсивно, а ее локальные переменные обычно автоматические, т.е. создаются заново при каждом обращении. Описание одной функции не может содержаться внутри другой, но переменные могут описываться в соответствии с обычной блочной структурой. Переменные в функции могут быть внутренними, внешними, но видимыми только в пределах исходного файла, или глобальными. Внутренние переменные могут быть автоматическими или статическими.

Язык C зарекомендовал себя как исключительно эффективный и выразительный язык для широкого разнообразия применений программирования.

# 4. Среда быстрой разработки Delphi

Delphi - среда разработки, которая использует язык программирования Object Pascal, разработанный фирмой Borland и изначально реализованный в её пакете Borland Delphi, от которого и получил в 2003 году своё нынешнее название. Object Pascal - по сути является наследником языка Pascal с объектно-ориентированными расширениями.

Изначально среда разработки была предназначена исключительно для разработки приложений Microsoft Windows, затем был реализован также для платформ GNU/Linux (как Kylix), однако после выпуска в 2002 году Kylix 3 его разработка была прекращена, и, вскоре после этого, было объявлено о поддержке Microsoft .NET. При этом высказывались предположения, что эти два факта взаимосвязаны.

Реализация среды разработки Delphi проектом Free Pascal позволяет использовать его для создания приложений для таких платформ, как Mac OS X, Windows CE и Linux.

Delphi — результат развития языка Турбо Паскаль, который, в свою очередь, развился из языка Паскаль. Паскаль был полностью процедурным языком, Турбо Паскаль начиная с версии 5.5 добавил в Паскаль объектно-ориентированные свойства, а Delphi — объектно-ориентированный язык программирования с возможностью доступа к метаданным классов (то есть к описанию классов и их членов) в компилируемом коде, также называемом интроспекцией. Так как все классы наследуют функции базового класса TObject, то любой указатель на объект можно преобразовать к нему, после чего воспользоваться методом ClassType и функцией TypeInfo, которые и обеспечат интроспекцию. Также отличительным свойством Дельфи от С++ является отсутствие возможности располагать объекты в стеке (объекты, унаследованные из Турбо Паскаля, располагаться в стеке могут) — все объекты попадают в динамически выделяемую область (кучу).

Де-факто Object Pascal, а затем и язык Delphi являются функциональными наращиваниями Turbo Pascal. Об этом говорят обозначения версий компилятора. Так, в Delphi 7 компилятор имеет номер версии 15.0 (Последняя версия Borland Pascal / Turbo Pascal обозначалась 7.0, в Delphi 1 компилятор имеет версию 8.0, в Delphi 2 — 9.0, и т. д. Номер версии 11.0 носит компилятор Pascal, входивший в состав среды C++Builder).

Delphi – это RAD (Rapid Application Development), среда быстрой разработки приложений. Delphi включает в себя помимо средств для работы с языком Object Pascal ряд дополнительных средств, помогающих максимально ускорить и упростить создание программ. К таким средствам относятся: визуальный редактор форм, при помощи которого быстро и без лишних усилий создается полноценно выглядящая программа, а так же другие составные части визуальной среды разработки программ.

Delphi является средой разработки приложений и включает развитые средства для написания и отладки кода: специализированный текстовый редактор, оптимизирующий компилятор и отладчик. Основой Delphi является IDE (Integrated Development Environment), интегрированная среда разработки, объединяющая в себе редактор кода и средства визуальной разработки, а также связывающая с компилятором, средствами разработки баз данных и другими составными частями Delphi.

Разумеется, Delphi – не единственная среда быстрой разработки приложений. Существуют и другие RAD, столь же удобные для визуальной разработки программ, например, Visual Basic. Но BASIC известен как не самый мощный и удобный язык программирования, кроме того, программы на нем отличаются сравнительно невысоким быстродействием.

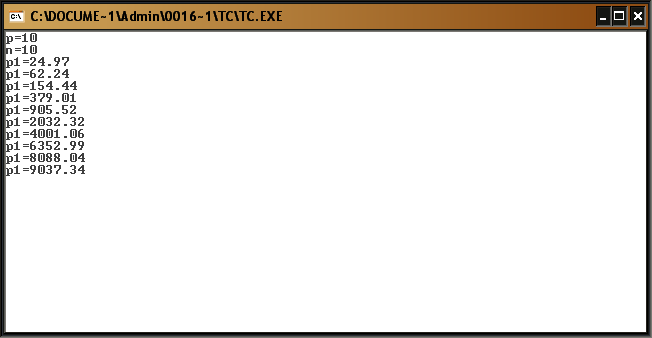
В результате мы получаем практически идеальный продукт для быстрой разработки программ: с одной стороны простота и удобство, сравнимые с Visual Basic, а с другой – мощь, скоростьи гибкость, характерные для C++.

# 5. Алгоритм решения задачи

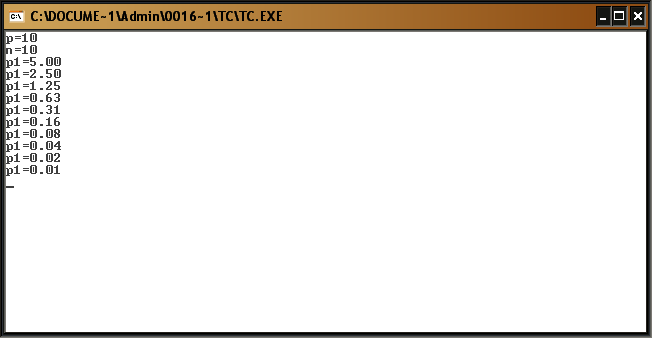


# 6. Тестирование программ

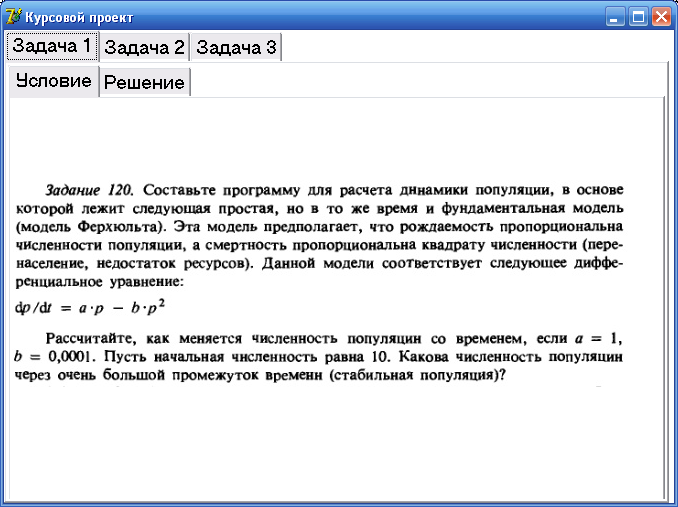
6.1. Тестирование на языке С

****

* 1. Тестирование на языке Pascal

****

* 1. Тестирование приложения в Delphi



# Выводы

В данной курсовой работе был разработан алгоритм решения задач развития популяций по модели Фюрхельста. Для решения дифференциальных уравнений использовался численный метод Эйлера.

В ходе курсового проектирования были изучены теоретические основы метода Эйлера и его модификации для решения дифференциальных уравнений. Рассмотрены основные характеристики используемых языков программирования – С и Pascal, а также проанализирована среда разработки приложений Delphi. На основе разработанного алгоритма решения задачи были разработаны 3 программы: на языке С, Pascal и в среде Delphi. Все программы протестированы на различных наборах данных и ошибок выявлено не было.

# Список использованной литературы

1) Каханер Д., Моулер К., Неш С. Численные методы и программное обеспечение. – М.: Мир, 1998 – 356с.

2) Культин Н. Программирование в Turbo Pascal 7.0 и Delphi. – 2-е изд.,перераб. И доп. – СПб.: БХВ – Санкт-Петербург, 2002. – 416 с.

3) Марченко А.И., Марченко Л.А. – Программирование в среде Turbo Pascal 7.0 – К.: ВЕК+, М.: Бином Универсал, 1998. – 496 с.

4) Фаронов В.В. Турбо Паскаль 7.0. Практика программирования. Учебное пособие. Издание 7-е, переработанное. –М.: «Нолидж», издатель Молгачева С.В., 2001. -416 с.

# Предметный указатель

C, 10, 12, 13, 14, 15, 16

Delphi, 2, 3, 15, 16, 17, 18, 19

метод Эйлера, 8, 9, 10, 18

Модель Ферхюльста, 6

Паскаль, 11, 12, 15, 19

Популяция, 5