ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Главными задачами изучения предмета «Информатика и ИКТ» в общеобразовательных учреждениях являются формирование у обучаемых системного информационного подхода к анализу окружающего мира; формирование знаний, умений и навыков использования информационно-коммуникационных технологий для решения жизненных задач; проведения вычислительного эксперимента для открытия новых свойств исследуемого объекта. Объектом изучения содержательной линии «Моделирование и формализация» в предмете «Информатика и ИКТ» являются информационные и компьютерные модели. Содержание этой линии определяется через следующие понятия**:** моделирование как метод познания, цели моделирования, жизненная задача, объект, система, структура системы, системный анализ, формализация, модель, информационная модель, компьютерная модель, адекватность модели изучаемому объекту и др.

Моделирование – это метод познания, состоящий в создании и исследовании моделей. Это одна из важнейших форм познания мира. *Формализация* – это замена реального объекта или процесса его формальным описанием с помощью формальных языков, то есть его информационной моделью. *Вычислительный эксперимент* – это проведение расчетов на компьютере и анализ результатов.

При изучении учебного материала содержательной линии «Моделирование и формализация» перед учителем можно поставить следующие *задачи:*

1. Сформировать у учащихся представление о моделировании как о методе научного познания. Научить использовать информационные и компьютерные модели для познания окружающей действительности, исследования объектов, явлений, процессов.
2. Развить у учащихся системное мышление.
3. Познакомить с этапами составления компьютерных моделей. Раскрыть содержание формализации как основного этапа моделирования.

4. Научить учащихся понимать смысл отношения объект-модель.
5. Выработать у учащихся практические навыки:

– структурирования изучаемого объекта (процесса, явления);
– построения информационных и компьютерных моделей;
– исследования моделей с использованием компьютера.

Учащиеся должны понимать, что не существует других способов мыслительной обработки воспринимаемой человеком окружающей действительности, кроме ее модельного представления, и что при решении задач с использованием компьютера рутинная работа по её решению передаётся компьютеру.

Ведущие идеи, которые должны быть реализованы при изучении учебного материала данной содержательной линии:

- моделирование – основной метод познания окружающего мира;
- формализация информации об исследуемом объекте – основной этап моделирования;
- компьютер – могучий помощник человека, возможности его огромны, но работает он только по программам, составленным человеком;
- построение разных информационных моделей – это реализация различных методов решения одной задачи;
- самостоятельное построение учащимися логических схем типовых логических устройств компьютера – это путь к пониманию принципов функционирования компьютера, осознанному пониманию обработки и продвижению информации в компьютере, правильному представлению архитектуры компьютера.

Виды деятельности учащихся, формируемые в процессе работы с учебным материалом данной содержательной линии

Виды аналитической деятельности:

- выделение в исследуемой ситуации объекта моделирования;
- анализ свойств объекта и выделение среди них существенных с точки зрения целей моделирования;

- выбор метода решения задачи, разбиение процесса решения задачи на этапы;
- исследование учебных моделей.

Виды практической деятельности:

- определение структуры исходных данных и установление их связи с ожидаемым результатом;
- составление информационных моделей различных типов: логических, табличных, графов, функциональных схем и др.;
- преобразование одного вида информационной модели исследуемого объекта в другой;
- разработка алгоритмов для представления информационной модели на компьютере;
- реализация представленного алгоритма на компьютере;
- проведение вычислительного эксперимента;
- построение логических схем типовых логических устройств компьютера.

Требования к уровню подготовки учащихся

Учащиеся должны понимать:

- моделирование – это метод познания окружающей действительности, практической деятельности;
- сущность технологического подхода к решению задачи;
- смысл модели как заменителя объекта в процессе решения практической задачи;
- суть формализации как основного этапа моделирования;
- суть адекватности как соответствия модели моделируемому объекту и целям моделирования;
- компьютерный эксперимент имеет и существенные преимущества, и определенные ограничения перед натурным экспериментом;
- в процессе формализации решения данной задачи возникает три вида объектов: исходный информационный объект, итоговый ин-

формационный объект, информационный процесс преобразования исходного информационного объекта в итоговый;

- управляя исполнителями, учащиеся на самом деле работают с их компьютерными моделями;
- отладка и тестирование программы, реализующей алгоритм решения задачи, – это проведение компьютерного эксперимента.

Учащиеся должны знать:

- общую схему информационной технологии решения задачи;
- виды информационных моделей;
- цели и этапы проведения компьютерного эксперимента.

Учащиеся должны уметь:

- выделять среди свойств данного объекта существенные свойства с точки зрения целей моделирования;
- строить модели *задачи* (выделять исходные данные, результаты, устанавливать соотношения между ними, отражать эти отношения с помощью формул, таблиц, графов, функциональных схем);
- раскрывать содержание понятия «информационная модель» на примерах;
- анализировать задачу и определять, какая часть может быть поручена компьютеру, а какая требует человеческой интуиции и творческих способностей к принятию решения;
- по постановке задачи определять возможность использования компьютерных программ для ее решения;
- интерпретировать таблицы, диаграммы, графы, схемы;
- строить функциональные схемы типовых логических устройств;
- проводить компьютерный эксперимент для построенных моделей;
- оценивать адекватность построенной модели объекту-оригиналу и целям моделирования;
- на каждом шаге решения задачи критически осмысливать работу компьютера и определять правильность выбранных методов решения.

Для работы с компьютерными моделями можно выбрать следующие *инструменты*:

- системы программирования: Ершол, QBasic, Turbo Pascal, Visual Basic. Net, Turbo Delphi, Visual C#, Visual J# и др. ;
- электронные таблицы (Microsoft Office Excel);
- системы управления базами данных (Microsoft Office Access);
- специализированные математические пакеты (Mathcad);
- другие средства.

Выбор инструмента моделирования определяется целью моделирования и формой представления информации об изучаемом объекте при его исследовании. В данной работе в качестве основного средства реализации информационных моделей на компьютере выступают системы программирования. Перед теми, кто научился программировать, должна открыться масса интересных и полезных задач. Программирование – это только инструмент для решения жизненных задач.

ГЛАВА 1. ОСНОВНЫЕ ПОНЯТИЯ СОДЕРЖАТЕЛЬНОЙ ЛИНИИ «МОДЕЛИРОВАНИЕ И ФОРМАЛИЗАЦИЯ»

«Нет, и не может быть, других способов мыслительной обработки воспринимаемой человеком окружающей действительности, нежели её модельное представление»

А. Г. Гейн

1. Понятие модели

Построение и использование моделей – мощное орудие познания. Реальные объекты и процессы бывают столь многогранны и сложны, что лучшим способом их изучения является построение моделей, отражающих лишь интересующие свойства этих объектов и поэтому более простых, чем эти объекты. Модели позволяют в наглядной форме представить объекты и процессы, недоступные для непосредственного восприятия (очень большие или очень маленькие объекты, очень быстрые или очень медленные процессы; такие, что исследование объекта или процесса опасно для окружающих или для исследуемого объекта). Поэтому для получения нужной информации об объектах-оригиналах проводят исследования на созданных моделях. При создании моделей нельзя учесть все свойства оригинала, какие-то свойства оказываются наиболее важными, а какими-то надо пренебречь. При этом появляется модель объекта (modulus – образ, способ, мера, от латинского modus).

Модель – это мысленно представимая или материально реализованная система, некоторый объект-заменитель, который в определенных условиях может заменять объект-оригинал, воспроизводя интересующие нас свойства и характеристики оригинала.

Примеры моделей: манекен – модель человеческого тела; модели самолетов, кораблей, автомобилей; таблица Менделеева; расписание занятий; уравнение зависимости перемещения тела при равномерном прямолинейном движении от скорости и времени движения.

У модели имеются существенные преимущества – наглядность, обозримость, доступность при исследовании. Модель замещает в процессе изучения, принятия решения оригинал, сохраняя те его черты, которые существенны для целей исследования. Изучение модели, объекта-заменителя, дает новую информацию об объекте-оригинале. Знания об окружающем мире меняют модели. Примером смены одних моделей другими, более адекватными, является история моделей Солнечной системы.

2. Классификация моделей

Рассматривая примеры моделей, ответим на следующие вопросы:

1. В какой форме представлена модель?
2. Что является оригиналом для данной модели?
3. Какова цель создания данной модели?
4. Какие свойства оригинала сохранены в модели?

Замечаем, что все приведенные модели можно разделить на два класса: *натурные и информационные модели*.

Примеры натурных моделей: глобус – модель планеты Земля; модели атомного строения вещества в химии; муляжи в анатомии; модели самолетов, кораблей, автомобилей. Общая черта, присущая приведенным моделям, состоит в том, что они копируют исходный объект, сохраняя внешний вид, структуру или поведение оригинала. Такие модели называют *натурными*.

В предмете «Информатика и ИКТ» рассматриваются другие модели – информационные.

Информационная модель – это модель, представляющая исследуемый объект или процесс набором данных и связей между ними, отражающими необходимую информацию об оригинале. В

информационной модели отражаются знания человека об объекте моделирования. Пример информационной модели: представим себе, что нам подробно описали внешность человека, которого мы не видели, а затем по описанию мы узнали этого человека. Стало быть, в нашем сознании сложился некоторый образ человека, то есть создалась информационная модель. Большинство знаний, которые получает человек, носят характер информационных моделей. Решая жизненные задачи по физике, химии, экономике мы составляем уравнения, неравенства, системы уравнений и неравенств, в которых выражены интересующие нас связи между выделенными данными – мы составляем информационную модель. В дальнейшем будет приведено множество примеров информационных моделей.

Разные предметные области предлагают свою классификацию информационных моделей. Мы рассмотрим классификацию моделей, представленную на схеме (рис. 1).

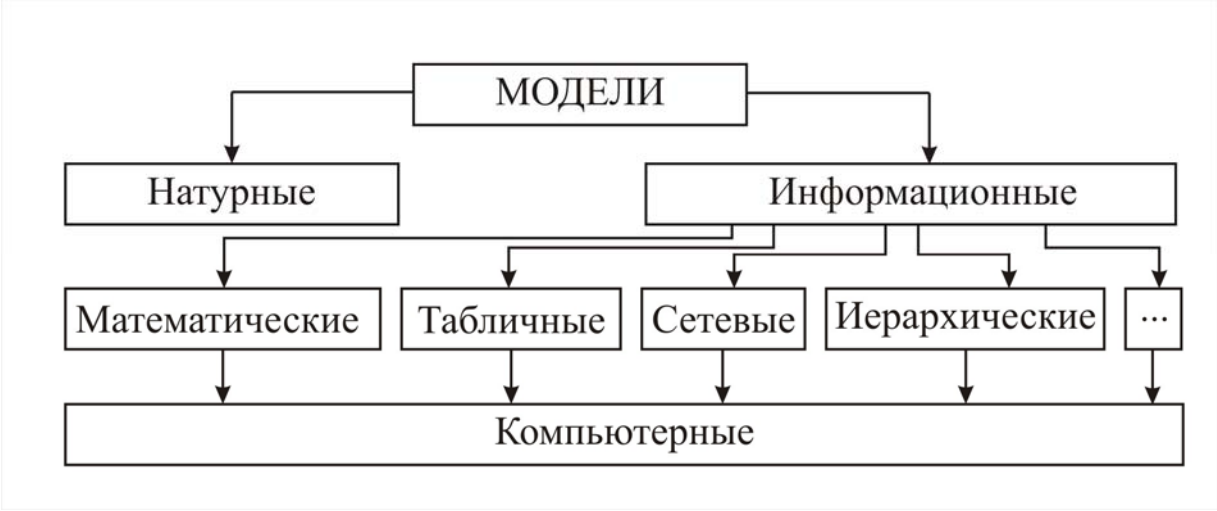


Рис. 1

По организации структуры данных и по форме представления этой структуры информационные модели можно классифицировать так:

1. *Табличные* информационные модели [15, 16].
2. *Иерархические* информационные модели [15, 16].
3. *Сетевые* информационные модели [15, 16].

4. *Математические модели* – это информационные модели, в которых параметры и зависимость между ними выражены в математической форме.

5. *Компьютерные модели* – это модели, составленные в расчете на исполнителя, имитированного на компьютере. Исходные данные, результаты и связи между ними в компьютерной модели представлены в виде, «понятном» компьютерному исполнителю. Компьютерная модель – это информационная модель плюс алгоритм для реализации этой модели. Примеры исполнителей, имитированных на компьютере: Ершол-система, Delphi-система и др.

6. *Аналоговые модели* – это модели, в которых исследуемое явление изучается по аналогичному процессу, имеющему иную физическую природу, но которое описывается такими же математическими соотношениями. Например, колебания груза на пружине аналогично колебаниям тока в электрическом контуре, поэтому для изучения колебаний груза можно изучать колебание тока.

Математические модели можно разделить на следующие виды:

- *Детерминированная модель* (причинно-следственная модель) – это модель, параметры в которой, по крайней мере теоретически, поддаются точному измерению и управлению. Это аналитически представимая закономерность в объекте или процессе, при которой для любой комбинации значений параметров на входе будет получен единственный результат на выходе.

- *Стохастическая модель* – это математическая модель задачи, в которой параметры и связи между ними представляют собой случайные величины или случайные функции, законы распределения которых и другие статистические характеристики известны, а неопределенные факторы отсутствуют. Величина на выходе такой модели может быть представлена только в виде математического ожидания. Например, модель задачи массового обслуживания.

- *Математическая модель с неопределенными условиями* – модель, получаемая, когда параметры и связи не могут быть описаны

статистическими методами. Методами обоснования решения в условиях неопределенности занимаются математическая теория игр и теория статистических решений.

3. Цели моделирования

1. *Понимание* объекта-оригинала. Модель нужна для того, чтобы узнать, как устроен конкретный объект, какова его структура, основные свойства объекта, законы развития и его взаимодействия с окружающим миром.

2. *Управление* объектом-оригиналом. Модель нужна для того, чтобы научиться управлять объектом, процессом, явлением и определить наилучшие способы управления при заданных целях и критериях.

3. *Прогнозирование* поведения объекта-оригинала. Модель нужна для того, чтобы прогнозировать прямые и косвенные последствия реализации заданных способов и форм воздействия на объект. Найти ответы на вопросы: «Что будет через какое-то время?» или «Что будет, если...?».

4. Границы адекватности модели

Если построенная модель дает удовлетворительные результаты при решении данной задачи, то говорят, что модель адекватна рассматриваемому объекту, процессу.

Модель не эквивалентна исходному объекту или процессу, а поэтому модель имеет ограниченную область адекватности. За пределами этой области она перестает удовлетворительно отражать свойства моделируемого объекта. Поэтому применять модель для решения той или иной жизненной задачи допустимо только тогда, когда мы убедились, что не вышли за границу области адекватности. Адекватность модели определяется ее согласованностью с практикой и общетеоретическими положениями.

В примерах данного пособия адекватность модели определяется либо сравнением результатов, полученных при исследовании модели,

с результатами экспериментов с использованием других теоретически точных методов решения данной задачи, либо анализом частных решений поставленной задачи.

5. Основные понятия системологии

Понятие «система» часто употребляется как в науке, так и в повседневной жизни. Примеры: Солнечная система, периодическая система химических элементов, система образования, система транспорта, файловая система, операционная система и многие другие.

Объект – материальный объект, явление природы, процесс – это некоторая часть окружающей нас действительности, рассматриваемая как единое целое.

Под *системой* будем понимать сложный объект, состоящий из множества взаимосвязанных частей и существующий как единое целое. Всякая система имеет определенное назначение, функции и цель. В информатике понятие «система» употребляется достаточно часто. Например, совокупность взаимосвязанных данных, предназначенных для обработки на компьютере – система данных.

Важной характеристикой всякой системы является ее структура. *Структура* – это внутренняя организация системы, определенный порядок объединения элементов, составляющих систему.

Основным методическим принципом информационного моделирования считается *системный подход,* согласно которому всякий объект моделирования рассматривается как система. Из всего множества элементов, свойств и связей выделяются лишь те, которые будут существенными для целей моделирования.

Задача системного анализа, который проводит исследователь, – учесть системные связи всякого объекта изучения для того, чтобы в дальнейшем отразить их в информационной модели. Сама *информационная модель* представляет собой некоторую *систему данных и отношений между ними*. Эти данные и отношения могут быть представлены в разной форме: графической, математической, табличной и др. Составленная информационная модель далее

реализуется на компьютере: составляются алгоритм и программа для проведения вычислительного эксперимента.

Схема перехода от реального объекта к компьютерной модели представлена на рис. 2.

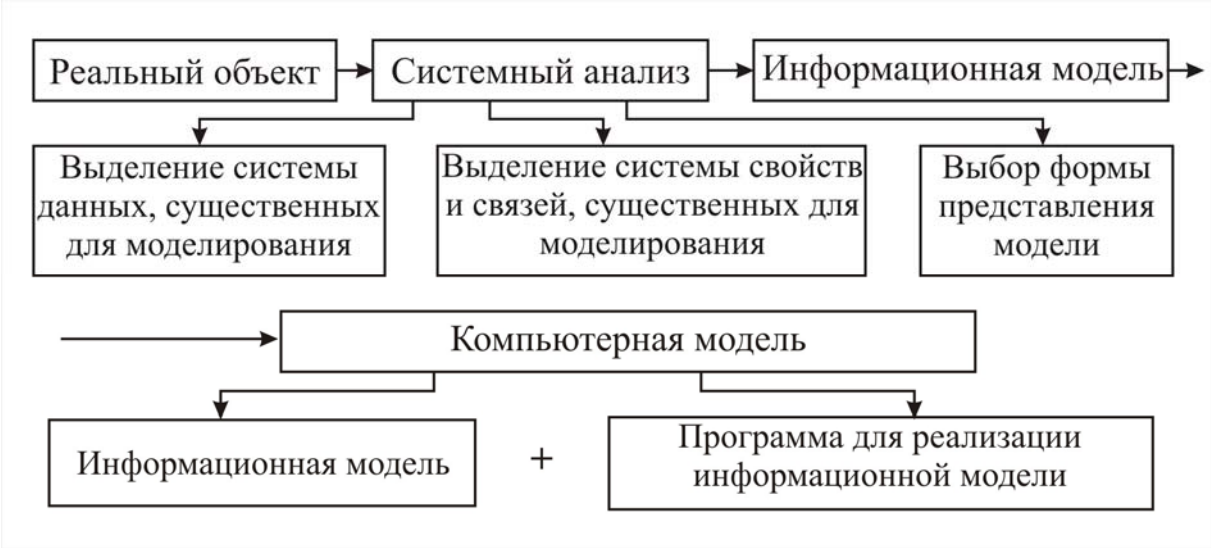
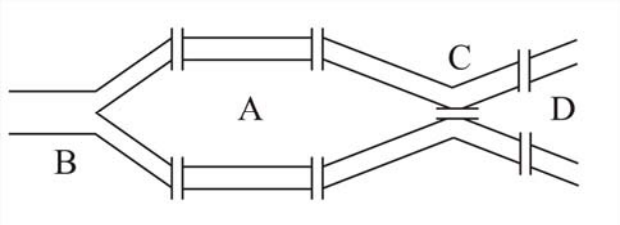


Рис. 2

Приведем классический *пример системного подхода* к составлению информационных моделей. Гуляя по городу Кенигсбергу (Калининграду), великий математик Леонард Эйлер (1707 – 1783 гг.) решал следующую задачу: как выбрать маршрут, чтобы побывать на всех берегах и островах реки Преголь, пройдя через каждый мост ровно один раз. План мостов в городе Кенигсберге:



Решение

Для решения задачи Эйлер использовал системно-информационный подход к анализу окружающей действительности. Объектом исследования являлась часть местности. Проведя системный анализ объекта моделирования, Эйлер выделил интересующие его элементы

системы, установил между ними связь. Элементы системы – берега и острова, связь между ними – мосты, соединяющие данные элементы. Для простоты решения задачи он изобразил острова и оба берега точками, а мосты – отрезками линий (рис. 3). Эйлер построил информационную модель в виде графа.

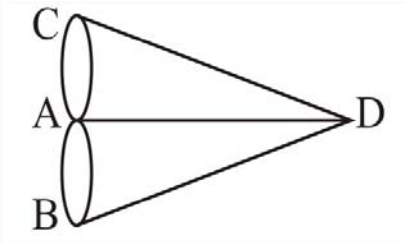


Рис. 3

Решение задачи стало теперь почти очевидным. Чтобы удовлетворить требования задачи, нужно иметь возможность прийти в какую-либо точку по одному отрезку, а выйти из нее – по другому. Это не относится к начальной и конечной точкам, следовательно, из каждой точки, кроме, может быть, двух, должно выходить четное количество отрезков-рёбер. Для полученного рисунка сформулированное требование не выполняется. Значит, по мостам нельзя пройти, соблюдая условие задачи.

6. Этапы решения задач с использованием компьютера

Под процессом решения задачи с использованием компьютера надо понимать совместную деятельность человека и компьютера. На долю человека приходятся этапы, связанные с творческой деятельностью: постановка задачи, составление информационной модели, разработка алгоритма решения поставленной задачи, составление компьютерной модели, анализ результатов, а на долю компьютера – этапы обработки информации в соответствии с разработанным алгоритмом (рис. 4).

Этапы решения задач

1. *Постановка задачи*. Ставит задачу человек, который хорошо разбирается в предметной области задачи. На этом этапе происходит выбор объекта моделирования, определяется *цель* решения задачи, выполняются сбор информации об объекте моделирования, формулировка условия задачи, описание каждого исходного данного, определение формы выдачи результатов и предлагается общий подход к решению поставленной задачи.

Рис. 4

2. *Системный анализ объекта моделирования и построение информационной модели*. Здесь происходят формализация задачи, анализ существующих аналогов изучаемого процесса, определение структур данных и типов связей между ними, выбор технических и программных средств. И в соответствии с выбранными средствами разрабатывается *информационная модель*. Изучаемый объект или процесс заменяется информацией о нём в выбранной форме.

3. *Алгоритмизация решения задачи* – это выбор компьютерного исполнителя, метода проектирования алгоритма, формы записи алгоритма и проектирование алгоритма. Чаще всего алгоритм решения

поставленной задачи на этом этапе описывается словесно или с использованием блок-схем.

4. *Создание компьютерной модели* – это описание решения задачи в виде программы для выбранного исполнителя, реализация алгоритма на компьютере.

5. *Тестирование и отладка* – синтаксическая отладка, отладка логической структуры, тестовые расчеты и анализ результатов тестирования, совершенствование программы.

6. *Компьютерный эксперимент*. Составляется план эксперимента и проводится сам эксперимент.

7. *Анализ результатов моделирования*. Проверяется адекватность модели и, если возможно, выделяются границы адекватности. Результаты проведённого эксперимента могут соответствовать или не соответствовать цели моделирования. Если результат эксперимента соответствует цели моделирования, то нужно перейти на шаг 8, иначе необходимо уточнить компьютерную модель с повторным выполнением шагов 2 – 7.

8. *Доработка программы для решения конкретных задач, составление документации к компьютерной модели*.

Следует заметить, что приведенная последовательность действий ориентирована на решение задач любой сложности. Для простейших задач некоторые этапы, возможно, не понадобятся. Для более сложных задач некоторые этапы могут существенно усложниться.

7. Пример решения задачи, в которой происходит развитие информационной модели

Рассмотрим пример решения задачи, в которой происходит развитие информационной модели вследствие её уточнения, так как результат эксперимента с первоначальной компьютерной моделью не соответствует цели моделирования, противоречит практике.

Задача: Два завода железобетонных изделий снабжаются цементом из двух складов. В сутки первому заводу необходимо 50 т цемента, второму – 90 т. С первого склада можно вывезти в сутки

60 т, со второго – 80 т цемента. Стоимость доставки тонны цемента из первого склада на первый завод составляет 1,4 тыс. руб., на второй – 2 тыс. руб. Стоимость доставки тонны цемента со второго склада на первый завод – 1,2 тыс. руб., на второй завод – 1,6 тыс. руб. Как распределить поставки, чтобы общая сумма расходов на перевозку была минимальной (табл. 1).

Таблица 1

Склад	1-й завод	2-й завод	Доставка цемента
Первый	$x1$ 1400 руб.	$x2$ 2000 руб.	60 т
Второй	$x3$ 1200 руб.	$x4$ 1600 руб.	80 т
	50 т	90 т	140 т

1-й этап. Постановка задачи. Поставим вопрос, какие факторы могут повлиять на организацию перевозок цемента со складов на заводы? Их много: болезнь водителей, выход из строя автомобилей, трудности с бензином и т.д. Эти факторы мы учитывать не будем. Предположим, что на автобазе есть резервы автомобилей, хорошо поставлено медицинское обслуживание, нет трудностей с горюче-смазочными материалами и т.д. Нас интересует только, как распределить цемент между заводами.

2-й этап. Системный анализ объекта моделирования и построение информационной модели. Выделим из условия задачи необходимую информацию и формализуем её. Введем обозначения: $x1$ т – количество цемента, вывозимого с первого склада на первый завод; $x2$ т – количество цемента, вывозимого с первого склада на второй завод; $x3$ т – количество цемента, вывозимого со второго склада на первый завод; $x4$ т – количество цемента, вывозимого со второго склада на второй завод. Так как в сутки потребность первого завода в цементе составляет 50 т, то должно выполнятся равенство $x1 + x3 = 50$.

По аналогии: $\begin{cases} x1 + x3 = 50, \\ x2 + x4 = 90, \\ x1 + x2 = 60, \\ x3 + x4 = 80. \end{cases}$ (1)

Эти равенства должны выполняться одновременно. По условию стоимость перевозки цемента в сутки на первый и второй заводы равна

$$f = 1{,}4 \cdot x1 + 2 \cdot x2 + 1{,}2 \cdot x3 + 1{,}6 \cdot x4 .$$

Из бесконечного множества решений системы (1) необходимо найти значения $x1$, $x2$, $x3$, $x4$, при которых линейная функция f принимает наименьшее неотрицательное значение.

3-й этап. Алгоритмизация решения задачи. Выберем исполнителя – Basic-систему. В системе ограничений все переменные выразим через $x1$: $x2 = 60 - x1$; $x3 = 50 - x1$; $x4 = 30 + x1$, подставим в функцию f выражения переменных и получим более простой вид функции f

$$f = 1{,}4 \cdot x1 + 2(60 - x1) + 1{,}2(50 - x1) + 1{,}6(30 + x1) = -0{,}2 \cdot x1 + 228 .$$

Мы уже говорили, что расходы на перевозку измеряются неотрицательной величиной, т. е. $f \geq 0$. Следовательно, минимальное значение f равно нулю.

Алгоритм решения задачи:

1. Найти $x1$ из уравнения: $-0{,}2 \cdot x1 + 228 = 0$;
2. Найти $x3$ из уравнения: $x3 = 50 - x1$;
3. Найти $x2$ из уравнения: $x2 = 60 - x1$;
4. Найти $x4$ из уравнения: $x4 = 30 + x1$.

4-й этап. Создание компьютерной модели.

```
rem программа перевозки
dim x1 as single, x2 as single, x3 as single, x4 as single
let x1 = 228 / 0.2: let x2 = 60 – x1
let x3 = 50 – x1: let x4 = 30 + x1
print “X1 =”; x1; “X2 =”; x2; “X3 =”; x3; “X4 =”; x4
end
```

6-7-й этапы. Вычислительный эксперимент и анализ результатов эксперимента. В результате работы программы получили

$$x1 = 1140 \text{ т}; \quad x2 = -1080 \text{ т}; \quad x3 = -1090 \text{ т}; \quad x4 = 1170 \text{ т}; \quad f = 0.$$

Можно сделать вывод, что результаты неправильные, они противоречат требованиям, поставленным в условии задачи (получили, что для обеспечения нормальной работы заводов, цемент с заводов надо вывозить). Модель неадекватна изучаемому процессу. Следовательно, при составлении информационной модели не были учтены важные факторы для решения поставленной задачи. Вернемся на шаг 2.

2-й этап. Уточним информационную модель, введя добавочные ограничения $x1 \ge 0, \ldots, x4 \ge 0$.

$$\begin{cases} x1 + x3 = 50 \\ x2 + x4 = 90 \\ x1 + x2 = 60 \\ x3 + x4 = 80 \\ x1 \ge 0, x2 \ge 0, x3 \ge 0, x4 \ge 0 \end{cases} \qquad f = 1{,}4 \cdot x1 + 2 \cdot x2 + 1{,}2 \cdot x3 + 1{,}6 \cdot x4$$

3-й этап. Алгоритмизация решения задачи. Для составления алгоритма исследуем математическую модель.

$$\begin{cases} x3 = 50 - x1 \\ x2 = 60 - x1 \\ x4 = 30 + x1 \\ x1 \ge 0, x2 \ge 0, x3 \ge 0, x4 \ge 0 \end{cases} \Rightarrow \begin{cases} 50 - x1 \ge 0 \\ 60 - x1 \ge 0 \\ 30 + x1 \ge 0 \\ x1 \ge 0, x2 \ge 0, x3 \ge 0, x4 \ge 0 \end{cases} \Rightarrow \begin{cases} x1 \le 50 \\ x1 \le 60 \\ x1 \ge -30 \\ x1 \ge 0 \end{cases} \Rightarrow$$

$$\Rightarrow 0 \le x1 \le 50.$$

Легко видеть, что, если $x1$ возрастает к 50, то функция $f = -0{,}2 \cdot x1 + 228$ убывает, следовательно, наименьшее значение эта функция принимает при $x1 = 50$.

6-7-й этапы. Составление компьютерной модели, вычислительный эксперимент.

```
rem программа перевозки
dim x1 as single, x2 as single, x3 as single, x4 as single
```

```
let x1 = 50:  let x2 = 60 – x1
let x3 = 50 – x1:  let x4 = 30 + x1
print “X1 =”; x1; “X2 =”; x2; “X3 =”; x3; “X4 =”; x4; “f = ”;-0.2*x1+228
end
```

Результаты эксперимента: $x1 = 50;\ x2 = 10;\ x3 = 0;\ x4 = 80$.

Наименьшее значение функции $f = 218$. Решение удовлетворяет поставленным целям. Получили, что минимальные затраты на перевозку будут составлять 218 тыс. руб., если с первого склада ежедневно вывозить на первый завод 50 т цемента, на второй – 10 т, а со второго склада вывозить цемент только на второй завод 80 т ежедневно.

8. Компьютерная модель «Электронный кассир»

Рассмотрим компьютерную модель «Электронный кассир».

1-й этап. Постановка задачи. Пусть имеется кинозал с r рядами по m мест в ряду. Предположим, что цена билета на сеанс зависит только от номера ряда, например, места в первых двух рядах и последних двух рядах кинозала дешевле, чем места в других рядах. Составьте компьютерную модель продажи n билетов на соседние свободные места на один сеанс ($n = 1, 2, 3$).

2-й этап. Создание информационной модели. Как представить информацию о зале? Для реализации поставленной в задаче цели представим кинозал как последовательность мест, каждое из которых определено номером ряда и позицией расположения этого места в ряду. Иными словами, кинозал может быть представлен как двумерный массив из r строк и m столбцов. Тип элементов этого массива бинарен (билет продан на рассматриваемое место на данный сеанс в данный момент и билет не продан). Элементы этого массива могут быть либо логическими, принимающими значения *true, false*, либо целыми, принимающими значения 1, 0. Билеты на каждое место имеют определенную цену. Следовательно, для создания информационной модели нужен еще одномерный массив, состоящий из r элементов, где значе-

ние каждого элемента массива – это цена билета в соответствующем ряду.

Имеем два массива: ***цел таб card*** [1 : r, 1 : m], где card [i , j]=1, если билет на *j*-е место в *i*-м ряду продан, и card [i , j]=0, если билет на *j*-е место в *i*-м ряду не продан, $1 \le i \le r$, $1 \le j \le m$; ***цел таб cost*** [1 : r], где cost [j] – цена билета в *j*-м ряду, где $1 \le j \le r$.

Информационная модель данной задачи – это структурированный набор данных (два массива: card и cost).

Примеры массивов card и cost:

	1	2	3	4	5	6	7
1	0	0	1	1	0	1	1
2	1	0	0	0	0	1	1
3	1	1	0	1	0	1	1
4	0	0	0	1	0	0	1
5	0	1	1	1	1	0	1

Цел таб card[1:5,1:7]

1	200
2	200
3	300
4	200
5	200

Цел таб cost[1:5]

Билет на третье место в первом ряду продан, так как card [1 , 3]=1; билет на пятое место в четвертом ряду не продан, так как card [4 , 5]=0. Цена билетов на втором ряду – 200 рублей, так как cost[2]=200; цена билетов на третьем ряду – 300 рублей, так как cost[3] = 300.

3-4-й этапы. Алгоритмизация решения задачи и создание компьютерной модели. Используя информационную модель, созданную на предыдущем этапе, легко построить компьютерную модель различной сложности и реализации. Цель данных этапов – составить алгоритм и программу для выбранного исполнителя, моделирующие работу кассира по продаже *n* билетов на соседние свободные места на один киносеанс.

В самом общем виде алгоритм может быть описан так:

алг cashier

нач

1. Подготовка исходного состояния продажи билетов на данный сеанс в текущее время, т.е. формирование начального состояния массивов card [1 : r, 1 : m] и cost [1 : r].

2. Реализация диалога кассира и покупателя: определение необходимых покупателю *n* соседних свободных мест на данный сеанс, фиксирование ответов покупателя на задаваемые электронным кассиром вопросы, продажа билетов, фиксирование результатов продажи.

3. Анализ диалога между покупателем и электронным кассиром и анализ результатов продажи билетов.

кон

Детализируем выделенные блоки, используя исполнителя QBasic-систему.

1-й блок. Заполним массив card, используя последовательность случайных чисел. Сгенерируем последовательность случайных чисел процедурой randomize timer.

```
CLS : sum = 0    'sum-количество проданных билетов
'Формирование массива card
randomize timer
FOR i = 1 TO r
    FOR j = 1 TO m
        card (i, j) = INT(RND*2)
        IF card (i, j) = 1 THEN sum = sum + 1
    NEXT j
card (i, m + 1) = 1   'формирование «барьера»[1] для выбора n1 рядом
```

[1] Дополним массив card (*m*+1)-м столбцом, присвоим всем элементам этого столбца значение 1, *место занято*. При поиске *n* рядом расположенных нулевых элементов массива в исследуемой строке, *рядом расположенных мест в исследуемом ряду кинозала,* можно поступить следующим способом. Суммировать количество нулевых элементов в строке массива, если найден один нулевой элемент в исследуемой строке массива, можно при выполнении условия «пока не встретится в строке элемент, равный 1», что интерпретируется так – «пока не встретится занятое место в исследуемом ряду». Тогда для любого расположения нулевых элементов в строке массива цикл закончится. Иначе необходимо проверять составное условие «пока элемент массива равен нулю и не все элементы исследуемой строки массива просмотрены». Такой метод Н. Вирт назвал методом «барьера».

```
NEXT i                    'расположенных билетов
'Заполним массив cost, используя последовательность случайных чисел.
' Цена билетов по рядам
t = INT(RND * 100)
cost(1) = t: cost(2) = t: cost(r - 1) = t
cost(r) = t: t = INT(RND * 200)
FOR i = 3 TO r - 2
    cost(i) = t + i * 10
NEXT i
```

2-й блок. Определим время диалога электронного кассира с покупателями билетов. Подсчет времени в секундах организуем с использованием функции TIMER.

```
time2 = TIMER: time1 = 0;
time3 = TIMER: time1 = time3 – time2,
```

где *time*2 – время начала диалога, *time*3 – текущее время диалога, *time*1 – время, прошедшее с начала диалога до текущего момента, и условие проверки окончания продажи билетов на данный сеанс. Определим условие продолжения диалога электронного кассира и покупателей. Диалог продолжается пока не будут проданы все билеты на данный сеанс или не пройдет время, отведенное для продажи билетов.

```
time2 = TIMER: time1 = 0
WHILE (sum < r * m) AND (time1 <= 300)
    O = 0 : LOCATE 11, 10: PRINT "Диалог с "; f; " покупателем:"
    ' f – номер обслуживаемого покупателя
    LOCATE 12, 10: INPUT "Хотите купить билеты?(1-да,2-нет)", o1
    IF o1 = 1 THEN
        LOCATE 13, 10 : INPUT "Сколько билетов Вам нужно?"; n
        If n > 0 THEN
            Sale
        END IF
    END IF
      …
```

Для осуществления диалога вызывается процедура *Sale* (продажа).

3-й блок. Анализ диалога между покупателем и электронным кассиром. Если покупатель купил билеты на данный сеанс (*O* = 1), то обновляем массив card и организуем диалог с другим покупателем. Если *O* = 0 и *fl*1 = 1, то предлагаемые билеты не удовлетворили покупателя, следовательно, организуем новый диалог с покупателем. Если *fl*1 = 0, то нет *n* требуемых покупателю и непроданных билетов на рядом расположенные места, выводим соответствующее сообщение и организуем диалог с другим покупателем. Если после покупки билетов не осталось свободных мест, то программируем вывод соответствующего сообщения и окончание диалога. Если время диалога закончилось, то есть сеанс уже начался, то тоже программируем вывод соответствующего сообщения и окончание диалога.

```
      …
'Анализ диалога с покупателем
f = f + 1
Delay: Clrscr2
IF sum = r * m THEN
  LOCATE 20, 8
   PRINT "Извините, все билеты проданы! До свидания!"
END IF
time3 = TIMER: time1 = time3 - time2
IF time1 > 300 THEN
  LOCATE 21, 8: PRINT "Извините, сеанс уже начался. До свидания!"
END IF
Delay: Clrscr2
IF O = 1 THEN Write2
IF (O = 0) AND (fl1 = 1) THEN
    LOCATE 22, 3
    PRINT "Извините, ничего больше предложить не могу!";
    PRINT "До свидания!"
END IF
IF (o1 = 2) or (n<=0) THEN LOCATE 13, 10: PRINT "До свидания!"
Delay: Clrscr2
WEND
```

Детализация процедуры Sale. В процедуре организуем последовательный просмотр мест кинозала (элементов массива card) и выбор $n1$ непроданных билетов на рядом расположенные места. Если $n1 \geq n$, где n – количество билетов, требуемых покупателю, то организуем обращение к процедуре Selection, где программируем диалог с покупателем для выбора нужных билетов. Если нет непроданных билетов на n рядом расположенных мест ($fl1 = 0$), то программируем вывод соответствующего сообщения и организацию диалога с новым покупателем.

```
SUB Sale
DIM fl AS INTEGER
DIM kol AS INTEGER
'Отыскание n1 подряд расположенных, непроданных билетов
PRINT
fl1 = 0
kol = 0
FOR I = 1 TO r
 FOR j = 1 TO m
  IF card(I, j) = 0 THEN
    fl = 0
    n1 = 0
    WHILE card(I, j) = 0
      n1 = n1 + 1
      z = j
      j = j + 1
      fl = 1
    WEND
    IF fl = 1 THEN j = j - 1
    IF n1 >= n THEN
      kol = kol + 1
   IF (kol > 0) AND (kol <= r * m) THEN
    IF o = 2 THEN
     LOCATE 20, 8: PRINT "Смотрим следующие свободные места:"
    END IF
   END IF
   Selection
  END IF
 END IF
 IF o = 1 THEN GOTO 5
 NEXT j
NEXT I
```

```
5 : delay1
END SUB
```

Детализация процедуры Selection. Данная процедура вызывается, если $n1 \geq n$. Если покупатель выбирает предложенные билеты ($O = 1$), то программируем возврат в процедуру *Sale* и затем в основную программу для диалога с другим покупателем. Если предложенные билеты не удовлетворяют покупателя ($O = 0$), то программируем возврат в подпрограмму *Sale* для поиска других $n1$ непроданных билетов на рядом расположенные места.

```
SUB Selection
'Диалог с покупателем
DIM k AS INTEGER, p AS INTEGER
fl1 = 1
LOCATE 15, 8
PRINT "Свободные места с "; z - n1 + 1; " по "; z; " "; I; " ряда."
LOCATE 16, 8
PRINT "Стоимость билетов в этом ряду "; cost(I); "р. за билет."
LOCATE 17, 8
PRINT "Вам подходит данный ряд?(1-да,2-нет)";:  INPUT o
IF o = 1 THEN
 LOCATE 20, 8: PRINT SPC(70); " "
 IF n1 = n THEN
  FOR k = z - n + 1 TO z
   card(I, k) = 1
  NEXT k
  sum = sum + n
    LOCATE 18, 8: PRINT " Продано! До свидания!" :    delay
 ELSE
  11 : LOCATE 18, 8: PRINT "С какого места Вы хотели бы взять билеты?";
  INPUT p
  IF p > z - n + 1 THEN
     LOCATE 19, 8: PRINT "С этого места нет "; n;
    PRINT " рядом расположенных свободных мест"
    delay
    LOCATE 18, 1: PRINT SPC(70); " "  :    LOCATE 19, 1: PRINT SPC(70); " "
    GOTO 11
  END IF
  FOR k = p TO p + n - 1
    card(I, k) = 1
  NEXT k
  sum = sum + n
```

```
 LOCATE 19, 8: PRINT "Продано!До свидания!":   delay
 END IF
END IF
clrscr2
END SUB
```

Другие процедуры. В программе описаны еще четыре процедуры: Delay, Delay1, Clrscr1, Clrscr2, Write2.

Замечание. Примерные компьютерные модели «Электронный кассир» в системе QBasic и электронных таблицах MS Excel с использованием языка VBA даны в приложении 2, с.132.

5-й этап. Отладка и тестирование. При отладке происходит локализация и устранение синтаксических ошибок и явных ошибок кодирования. В процессе тестирования проверяется работоспособность программы, не содержащей явных ошибок. Тестовые данные должны проверять работу программы

- при реальных условиях функционирования программы;
- при граничных значениях области изменения входных переменных, которые программой должны восприниматься как правильные данные;
- при значениях, которые лежат за пределами допустимой области изменения входных данных.

При тестировании программы можно составить табл. 2.

Таблица 2

Номер теста	Проверяемый случай	Результат работы программы
1	*n*>*n*1	К сожалению, вместе расположенных *n* мест нет
2	*O* = 0 *u fl*1 = 1	Извините, ничего больше предложить не могу
…	…	…

9. Некоторые методы составления информационных моделей

9.1. Составление информационных моделей с использованием метода дискретизации

Пусть на отрезке [*a, b*] задана непрерывная функция $y = f(x)$, и отрезок [*a, b*] разбит на *n* элементарных отрезков точками $x_1, \ldots, x_{n+1}$.

$$a = x_1 < x_2 < x_3 < \ldots < x_{n+1} = b.$$

На каждом из полученных отрезков $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, n$ заменим непрерывную функцию $f(x)$ одним значением этой функции, получим последовательность $y_1, y_2, \ldots, y_n$ (рис. 5). Либо на каждом из этих числовых интервалов непрерывную функцию $f(x)$ заменим одним значением функции, приближающей данную с определённой точностью, например, функции *z*. Получим последовательность $z_1, z_2, \ldots, z_n$.

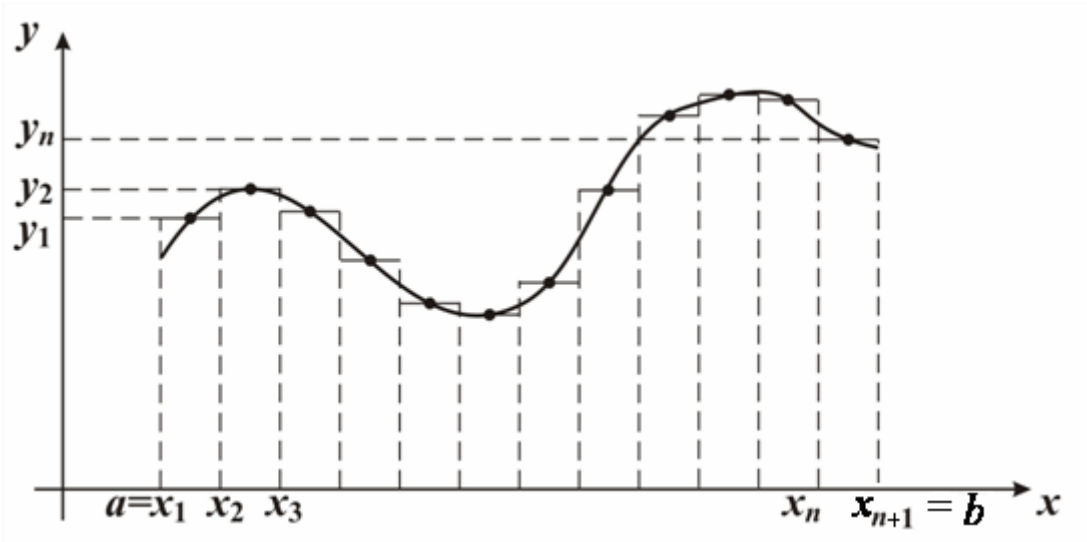


Рис. 5

Процесс замены непрерывной функции $f(x)$ на отрезке [*a*, *b*] области определения функции конечной последовательностью дискретных значений этой функции или функции, приближающей данную функцию с определенной точностью, называется *дискретизацией.*

Рассмотрим некоторые примеры применения метода дискретизации при составлении информационных моделей для решения жизненных задач.

***Пример* 1**. *1-й этап. Постановка задачи.* Тело движется прямолинейно с ускорением a м/с^2 и скоростью v м/с. Определите, какой путь пройдет тело за ts секунд [14].

Поставим цель – получить информационную модель.

2-й этап. Анализ объекта моделирования и построение информационной модели. Проведём системный анализ задачи, определим исходные данные и результаты. Аргументы: начальная скорость v, ускорение a, время движения ts. Результат – путь S.

Построим *приближённую информационную* (математическую) модель равноускоренного прямолинейного движения. Применим метод дискретизации для нахождения значения пути, пройденного телом за время ts при равноускоренном прямолинейном движении. Интервал времени ts разобьём на очень большое количество равных малых промежутков. Отрезок [0, ts] разделим на n равных отрезков длины $r = ts/n$.

Сделаем некоторые предположения:

1-е предположение. Если интервал времени разбить на очень большое количество равных малых промежутков, то мы не сильно ошибёмся, предполагая, что скорость тела на каждом из этих промежутков времени постоянна (то есть на каждом из этих промежутков движение прямолинейное равномерное) и меняется мгновенно в конце каждого промежутка.

2-е предположение. При неограниченном увеличении числа отрезков разбиения мы получим величину перемещения с любой точностью.

Замечание. Правильность и полнота предположений, на которых основана информационная модель, должны быть проверены на практике или строго доказаны.

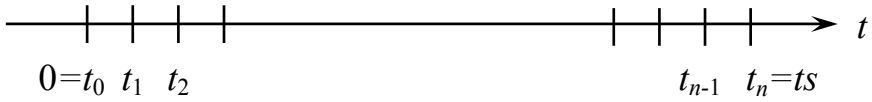
Будем находить путь на каждом промежутке, считая, что скорость тела на этих промежутках постоянна. Рассмотрим один из этих

отрезков. Выберем на нём какой-нибудь момент времени $\tau_k \in [t_{k-1}, t_k]$, $k = 1,\ldots,n$, найдём соответствующее ему значение скорости $v = v(\tau_k)$. Будем считать движение на этом отрезке равномерным со скоростью $v = v(\tau_k)$, $\tau_k \in [t_{k-1}, t_k]$. Аналогичные действия проделаем для каждого временного отрезка. Заменим исследуемую функцию (путь при равноускоренном прямолинейном движении) на каждом из интервалов деления на приближающую функцию (путь при равномерном прямолинейном движении). Разным разбиениям отрезка $[0, ts]$ на *n* равных отрезков и разному выбору моментов τ_k будут соответствовать разные значения пути, пройденного телом за *ts* секунд.

∇ Пусть τ_k совпадает с левым концом выбранного отрезка, то есть считаем, что на протяжении одного временного промежутка скорость тела не изменяется и равна скорости в начале этого временного отрезка, а меняется мгновенно по истечении этого интервала времени (скорость увеличивается). Заменим плавно увеличивающийся путь, пройденный телом за всё время *ts*, на последовательность значений пути, пройденного со скоростью $v = v(\tau_k)$, $k = 0,1,\ldots,n-1$, на каждом из отдельных временных отрезков.

Составим математическое соотношение (*информационную модель*), связывающее аргументы и результат.

Отрезок $[0, ts]$ разделили на *n* равных отрезков длины $r = ts/n$.



Найдем путь за время движения *ts* $S = S_1 + S_2 + \ldots + S_n$, где S_i – длина пути на *i*-м промежутке времени, $i = 1,2,\ldots,n$.

В силу первого предположения имеем: $S_1 \approx v_0 \cdot r$, $S_2 \approx v_1 \cdot r = (v_0 + a \cdot r)r$, $S_3 \approx v_2 \cdot r = (v_1 + a \cdot r)r = (v_0 + 2 \cdot a \cdot r)r$, …, $S_n \approx (v_0 + (n-1)a \cdot r)r$.

Тогда, $S \approx v_0 \cdot r + (v_0 + a \cdot r)r + \ldots + (v_0 + (n-1)a \cdot r)r$.

Вынесем r за скобки, для вычисления выражения в скобках воспользуемся формулой для вычисления суммы n членов арифметической прогрессии.

$$S \approx \frac{(v_0 + v_0 + (n-1)a \cdot r)n}{2} \cdot r = \frac{2 \cdot v_0 \cdot n \cdot r}{2} + \frac{(n-1)a \cdot r \cdot n \cdot r}{2} =$$

$$= v_0 \cdot ts + \frac{a \cdot ts^2}{2} - \frac{a \cdot ts^2}{2 \cdot n} \text{ (используем равенство } r = ts/n).$$

Получим

$$S \approx v_0 \cdot ts + a \cdot ts^2/2 - a \cdot ts^2/(2 \cdot n).$$

Обобщим эту формулу для произвольного t – времени движения тела

$$S \approx v_0 \cdot t + a \cdot t^2/2 - a \cdot t^2/(2 \cdot n). \qquad (2)$$

При неограниченном увеличении числа отрезков разбиения мы получим величину перемещения с любой точностью.

Формула (2) и является математическим соотношением для нахождения приближённого значения пути, которое пройдёт тело за время t (ts). Построение информационной (математической) модели закончено.

Замечание. Предел выражения, стоящего в правой части равенства (2) (при $n \to \infty$), если он существует, есть определённый интеграл от функции $v(t)$ на отрезке $[0, ts]$ и обозначается:

$$S = \int_0^{ts} v(t)dt. \qquad (3)$$

Имеем, $S = \int_0^{ts} (v_0 + a \cdot t)dt = \left(v_0 \cdot t + \frac{a \cdot t^2}{2} \right)\Bigg|_0^{ts} = v_0 \cdot ts + \frac{a \cdot ts^2}{2}$ или

$$S = v_0 \cdot t + \frac{a \cdot t^2}{2}.$$

3-4-й этапы. Алгоритмизация решения задачи и создание компьютерной модели. Выберем исполнителя – Ершол-систему. Компьютерная модель имеет вид:

```
алг путь (арг вещ a, v0, t, арг цел n, рез вещ s)
   дано | a –ускорение, v0-начальная скорость
        | t – время в пути, n- число разбиений
   надо | напечатано S –значение пройденного пути за время t
нач
   s: = v0 * t + a * t * t / 2 – a * t * t / (2 * n)
кон
```

Проведите вычисления, изменяя исходные данные, оформите результаты в виде таблицы.

*v*0	*a*	*t*	*n*	*S*

6-й этап. Вычислительный эксперимент, проверка адекватности модели изучаемому процессу.

Мы знаем другую формулу нахождения пути, который пройдёт тело за время *t*, при прямолинейном равноускоренном движении:

$$S = v_0 \cdot t + a \cdot t^2 / 2. \qquad (4)$$

Выведенная формула (2) отличается от (4) последним слагаемым, которое показывает, с какой степенью точности построенная модель описывает равноускоренное движение.

6.1. Определим адекватность модели, проведя вычисления по построенной формуле (2) и по формуле (4), используя программу:

```
алг путь (арг вещ a, v0, t, арг цел n, рез вещ s, s1, ds)
   дано | a –ускорение, v0-начальная скорость
        | t – время движения тела, n- число разбиений
   надо | напечатано S –значение пути за время t по формуле (2)
        | напечатано s1- значение пути за время t по формуле (4)
        | напечатано ds – абсолютная величина разности s и s1

нач
   s: = v0*t + a*t*t/2 – a*t*t/(2*n)
   s1: = v0*t + a*t*t/2
   ds: = abs (s1–s)
кон
```

С теоретической точки зрения, при увеличении *n* значение *ds* должно стремиться к 0. Но при вычислении с использованием компьютера происходит накопление ошибок округления, поэтому для каждого метода существует пороговое значение $n = n1$ такое, что при $n > n1$, *ds* перестает уменьшаться.

Проведите вычисления, изменяя исходные данные, оформите результаты в виде таблицы.

*v*0	*a*	*t*	*n*	*S*	*S1*	*dS*

6.2. Вычислим путь, пройденный телом, с заданной точностью. Обычно точный результат решения задачи неизвестен. Как оценить точность полученного решения? Рассмотрим простейший прием, часто применяемый на практике. Проведем вычисление пути *S* по формуле (2) с заданным *n* – числом разбиений отрезка $[0, ts]$, затем вычислим путь *S*1 с числом разбиений, равным $2 \cdot n$, и так далее. Вычислительный эксперимент показывает, что с увеличением числа разбиений соседние результаты S_k и S_{k+1} мало отличаются друг от друга. Будем считать, что, если $|S_k - S_{k+1}| \le e$, то S_{k+1} вычислено с точностью до *e,* где *e* – точность вычисления.

Опишем алгоритм и программу вычисления пути *S* по формуле (2) с заданной точностью *e,* используя рассмотренный прием.

```
алг путь1 (арг вещ a, v0, t, e, арг цел n, рез вещ  S1)
   дано | a –ускорение, v0-начальная скорость
          | e – точность вычислений
          | t – время движения, n- число разбиений
   надо | напечатано S1 –значение пути за время t,
          | вычисленное по формуле (2) с точностью e
нач
   S : = v0 * t + a * t * t / 2 – a * t * t / (2 * n)
   n:=n * 2; S1:= v0 * t + a * t * t / 2 – a * t * t / (2 * n)
 нц пока abs(S1–S) > e
   S:= S1;    n:=n * 2
```

```
  S1:= v0 * t + a * t * t / 2 – a * t * t / (2 * n)
кц
кон
```

Проведя вычисления, получаем значения пути с различной точностью *e*.

Проведите вычисления, изменяя исходные данные, оформите результаты в виде таблицы.

*v*0	*a*	*t*	*n*	*e*	*S*	*S*1
…	…	…	…		…	…

Замечание.

Рассмотрим примеры разработки других информационных и компьютерных моделей для решения поставленной задачи. Покажем, как можно использовать вновь полученные решения данной задачи для знакомства с формулами приближённого вычисления площадей криволинейных трапеций[2].

Проведём рассуждения, предложенные выше, до абзаца, начинающего со знака ∇ на 2-м этапе решения задачи, и далее рассмотрим три случая выбора τ_k.

1-й случай. Пусть τ_k совпадает с левым концом выбранного отрезка. Считаем, что на протяжении одного временного промежутка скорость тела не изменяется и равна скорости в начале этого временного отрезка, а меняется мгновенно по истечении этого интервала времени (скорость увеличивается). Заменим плавно увеличивающийся путь, на последовательность значений пути, пройденного со скоростью $v = v(\tau_k)$, $k = 1,\ldots,n$, на каждом из отдельных временных отрезков.

Составим математическое соотношение (информационную модель), связывающее аргументы и результат.

Отрезок $[0, ts]$ разделили на *n* равных отрезков длины $r = ts/n$.

[2] Фигура, ограниченная графиком непрерывной функции $y = f(x)$, прямыми $x = a$, $x = b$, $y = 0$, называется *криволинейной трапецией*.

Найдем путь за время движения *ts*:

$S = S_1 + S_2 + \ldots + S_n$, где S_i – длина пути на *i*-м промежутке времени, $i = 1, \ldots, n$.

$S_1 \approx r \cdot v(t_0)$, $S_2 \approx r \cdot v(t_1)$, …, $S_n \approx r \cdot v(t_{n-1})$.

Имеем, $S \approx r \cdot v(t_0) + r \cdot v(t_1) + r \cdot v(t_{n-1})$. (5)

Поставим теперь задачу – приближённо вычислить площадь фигуры, ограниченной графиком функции $y = v(t)$, прямыми $t = 0$, $t = ts$, $y = 0$, то есть площадь криволинейной трапеции, отмеченной на рис. 6 как *ACDB*.

Определим геометрический смысл выражения, стоящего в правой части равенства (5). Функция $y = v(t)$ неотрицательна на отрезке $[0, ts]$. Значения слагаемых выражения, стоящего в правой части равенства (5), численно равны соответственно площадям прямоугольников со сторонами длины *r* и длины $v(t_k)$, $k = 0, \ldots, n-1$. Значение выражения, стоящего в правой части равенства (5), равно площади ступенчатой фигуры, представленной на рис. 6. При увеличении числа разбиений отрезка $[0, ts]$ площадь ступенчатой фигуры будет приближаться к площади криволинейной трапеции. За искомую площадь криволинейной трапеции естественно принять предел площадей ступенчатых многоугольников при неограниченном увеличении числа разбиения отрезка $[0, ts]$. Получили, что путь, пройденный телом за время движения *ts*, численно равен площади криволинейной трапеции *ACDB* и может быть вычислен по приближённой формуле (5). В данном случае трапеция *ACDB* является

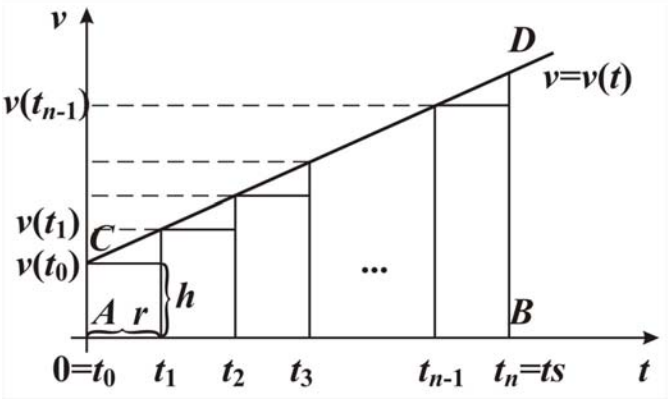


Рис. 6

элементарной линейной трапецией и её площадь можно вычислить по формуле $S = h \cdot \frac{n+m}{2}$, где h – высота, n, m – основания трапеции.

Имеем $S_{ACDB} = ts\left(v_0 + v_0 + a \cdot ts\right)/2 = v_0 \cdot ts + \left(a \cdot ts^2\right)/2$ или $S = v_0 \cdot t + \frac{a \cdot t^2}{2}$.

Проведя данные рассуждения, мы получили формулу для приближённого вычисления площади криволинейной трапеции, ограниченной графиком непрерывной функции $y = f(x)$, $f(x) > 0$, прямыми $x = a$, $x = b$, $y = 0$.

Пусть на отрезке $[a, b]$ задана непрерывная функция $y = f(x)$, $f(x) > 0$, и отрезок $[a, b]$ разбит на n элементарных отрезков точками $x_0, x_1, \ldots, x_n$, $a = x_0 < x_1 < x_2 < \ldots < x_n = b$, $r = (b-a)/n$. На каждом из полученных отрезков $[x_{i-1}, x_i]$ построим прямоугольник, одной стороной которого будет отрезок $[x_{i-1}, x_i]$, а другой – отрезок, длина которого равна $f(x_{i-1})$, $i = 1, \ldots, n$. Площадь PS криволинейной трапеции $ACDB$ можно приближённо считать равной сумме площадей построенных прямоугольников (рис. 7).

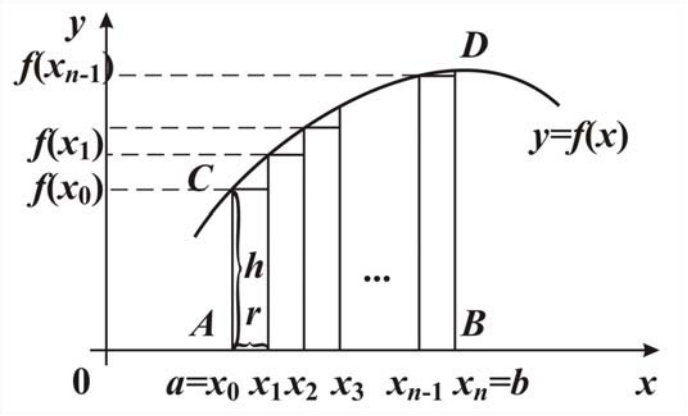


Рис. 7

$$PS \approx r\left(f(x_0) + f(x_1) + \ldots + f(x_{n-1})\right). \qquad (6)$$

При неограниченном увеличении числа отрезков разбиения мы получим величину площади криволинейной трапеции с любой точностью.

Формула (6) называется формулой *левых прямоугольников* для приближённого вычисления площади криволинейной трапеции. Геометрическая интерпретация модели 6 дана на рис. 7.

2-й случай. Пусть τ_k совпадает с правым концом выбранного отрезка. Считаем, что на протяжении одного временного промежутка скорость тела не изменяется и равна скорости в конце этого временного отрезка, а меняется мгновенно по истечении этого интервала времени (скорость увеличивается) и принимает значение скорости в конце следующего временного отрезка. Заменим плавно увеличивающийся путь на последовательность значений пути, пройденного со скоростью $v = v(\tau_k)$, $k = 1,\ldots,n$, на каждом из отдельных временных отрезков.

Составим математическое соотношение (информационную модель), связывающее аргументы и результат.

Отрезок $[0, ts]$ разделили на n равных отрезков длины $r = ts/n$.

Найдем путь за время движения *ts*:

$S = S_1 + S_2 + \ldots + S_n$, где S_i – длина пути на *i*-м промежутке времени, $i = 1,\ldots,n$.

$S_1 \approx r \cdot v(t_1)$, $S_2 \approx r \cdot v(t_2)$, …, $S_n \approx r \cdot v(t_n)$.

Имеем, $S \approx r \cdot v(t_1) + r \cdot v(t_2) + \ldots + r \cdot v(t_n)$. (7)

Поставим теперь задачу – приближённо вычислить площадь фигуры, ограниченной графиком функции $y = v(t)$, прямыми $t = 0$, $t = ts$, $y = 0$, то есть площадь криволинейной трапеции, отмеченной на рис. 8 как *ACDB*.

Определим геометрический смысл выражения, стоящего в правой части равенства (7). Функция $y = v(t)$ неотрицательна на отрезке $[0, ts]$. Значения слагаемых выражения, стоящего в правой части равенства (7), численно равны соответственно площадям прямоугольников со сторонами длины *r* и длины $v = v(t_k)$, $k = 1,\ldots,n$. Значение выражения, стоящего в правой части равенства (7), равно площади ступенчатой фигуры, представленной на рис. 8. При увеличении числа разбиений отрезка $[0, ts]$ площадь ступенчатой фигуры будет приближаться к площади криволинейной трапеции. За искомую площадь криволинейной трапеции естественно принять предел площадей сту-

пенчатых многоугольников при неограниченном увеличении числа разбиения отрезка $[0, ts]$. Получили, что путь, пройденный телом за время движения *ts*, численно равен площади криволинейной трапеции *ACDB* и может быть вычислен по приближённой формуле (7).

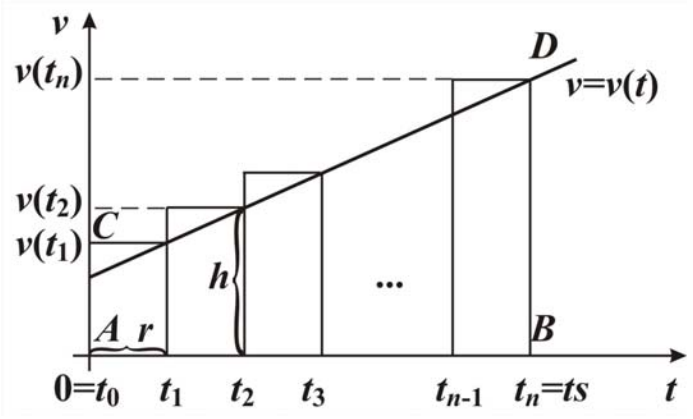


Рис. 8

Проведя данные рассуждения, мы получили ещё одну формулу для приближённого вычисления площади криволинейной трапеции, ограниченной графиком непрерывной функции $y = f(x)$, $f(x) > 0$, прямыми $x = a$, $x = b$, $y = 0$.

Пусть на отрезке $[a, b]$ задана непрерывная функция $y = f(x)$, $f(x) > 0$, и отрезок $[a, b]$ разбит на *n* элементарных отрезков точками $x_0, x_1, \ldots, x_n$, $a = x_0 < x_1 < x_2 < \ldots < x_n = b$, $r = (b - a)/n$. На каждом из полученных отрезков $[x_{i-1}, x_i]$ построим прямоугольник, одной стороной которого будет отрезок $[x_{i-1}, x_i]$, а другой – отрезок, длина которого равна $f(x_i)$, $i = 1, \ldots, n$. Площадь *PS* криволинейной трапеции *ACDB* можно приближённо считать равной сумме площадей построенных прямоугольников.

$$PS \approx r\left(f(x_1) + f(x_2) + \ldots + f(x_n)\right). \qquad (8)$$

При неограниченном увеличении числа отрезков разбиения мы получим величину площади криволинейной трапеции с любой точностью.

Это формула *правых прямоугольников* для приближённого вычисления площади криволинейной трапеции. Геометрическая интерпретация модели 8 дана на рис. 9.

3-й случай. Пусть τ_k совпадает с серединой выбранного отрезка. Считаем, что на протяжении одного временного промежутка скорость

тела не изменяется и равна скорости в середине этого временного отрезка, а меняется мгновенно по истечении этого интервала времени (скорость увеличивается). Заменим плавно увеличивающийся путь на последовательность значений пути, пройденного со скоростью $v = v(\tau_k)$, $k = 1,\ldots,n$, на каждом из отдельных временных отрезков.

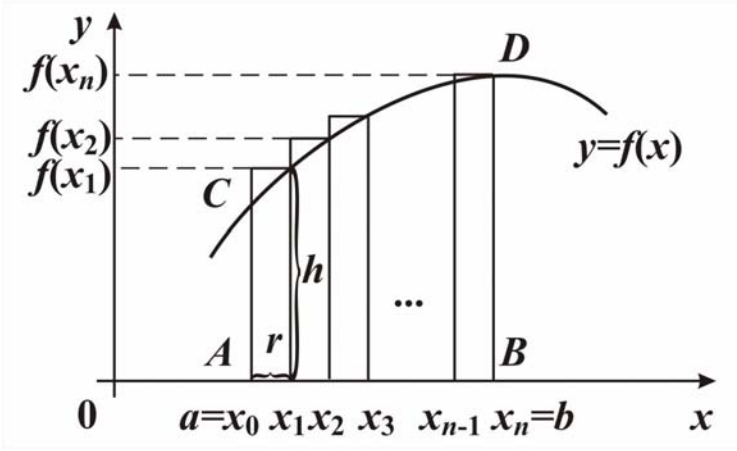


Рис. 9

Составим математическое соотношение (информационную модель), связывающее аргументы и результат.

Отрезок $[0, ts]$ разделили на n равных отрезков длины $r = ts/n$.

Найдем путь за время движения *ts*:

$S = S_1 + S_2 + \ldots + S_n$, где S_i – длина пути на *i*-м промежутке времени, $i = 1, 2, \ldots, n$.

$S_1 \approx r \cdot v(t_0 + r/2)$, $S_2 \approx r \cdot v(t_1 + r/2)$, …, $S_n \approx r \cdot v(t_{n-1} + r/2)$.

Имеем $S = r \cdot v(t_0 + r/2) + r \cdot v(t_1 + r/2) + \ldots + r \cdot v(t_{n-1} + r/2)$. (9)

Определим геометрический смысл выражения, стоящего в правой части равенства (9). Функция $y = v(t)$ неотрицательна на отрезке $[0, ts]$. Значения слагаемых выражения, стоящего в правой части равенства (9), численно равны соответственно площадям прямоугольников со сторонами длины r и длины $v = v(t_k)$, $t_k = t_{k-1} + r/2$, $k = 1, 2, \ldots, n$. Значение выражения, стоящего в правой части равенства (9), равно площади ступенчатой фигуры, представленной на рис. 10.

При увеличении числа разбиений отрезка $[0, ts]$ площадь ступенчатой фигуры будет приближаться к площади криволинейной трапеции. За искомую площадь криволинейной трапеции естественно принять предел площадей ступенчатых многоугольников при неогра-

ниченном увеличении числа разбиения отрезка $[0, ts]$. Получили, что путь, пройденный телом за время движения *ts*, численно равен площади криволинейной трапеции *ACDB* и может быть вычислен по приближённой формуле (9).

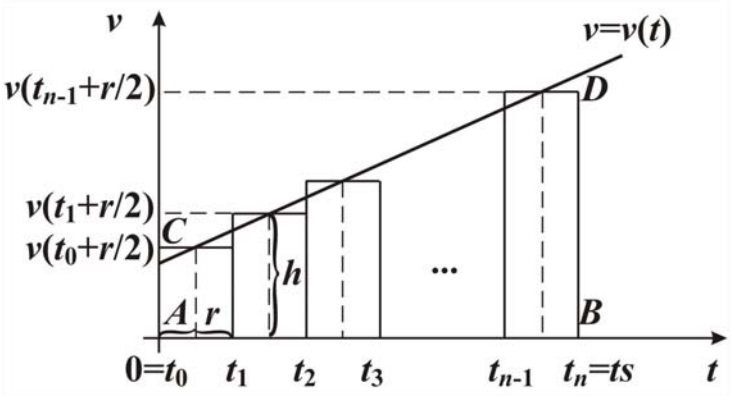


Рис. 10

Проведя данные рассуждения, мы получили ещё одну формулу для нахождения площади криволинейной трапеции, ограниченной графиком непрерывной функции $y = f(x)$, $f(x) > 0$, прямыми $x = a$, $x = b$, $y = 0$.

Пусть на отрезке $[a, b]$ задана непрерывная функция $y = f(x)$, $f(x) > 0$, и отрезок $[a, b]$ разбит на *n* элементарных отрезков точками $x_0, x_1, \ldots, x_n$, $a = x_0 < x_1 < x_2 < \ldots < x_n = b$, $r = (b - a)/n$. На каждом из полученных отрезков $[x_{i-1}, x_i]$ построим прямоугольник, одной стороной которого будет отрезок $[x_{i-1}, x_i]$, а другой – отрезок, длина которого равна $f(x_{i-1} + r/2)$, $i = 1, 2, \ldots, n$. Площадь *PS* криволинейной трапеции ACDB можно приближённо считать равной сумме площадей построенных прямоугольников.

$$PS \approx r\left(f(x_0 + r/2) + f(x_1 + r/2) + \ldots + f(x_{n-1} + r/2)\right). \qquad (10)$$

При неограниченном увеличении числа отрезков разбиения мы получим величину площади криволинейной трапеции с любой точностью.

Это формула *серединных прямоугольников* для приближённого вычисления площади криволинейной трапеции. Геометрическая интерпретация модели 10 дана на рис. 11.

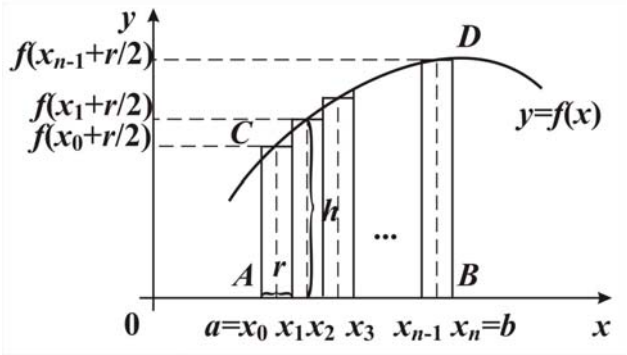


Рис. 11

Компьютерная модель, соответствующая информационной модели (9), на языках Ершол и Turbo Pascal.

Язык программирования Ершол	*Язык программирования Turbo Pascal*

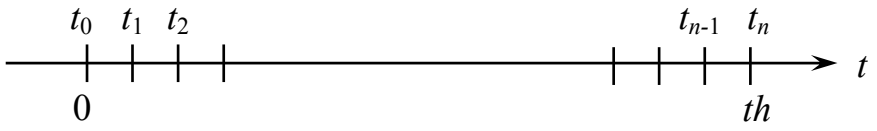
алг модель9 (арг вещ ts, a,v0, арг цел n,
рез вещ S2)
дано| a –ускорение
|v0 –скорость
| ts–время движения
|n–число разбиений ts
надо|S2 – путь, пройденный
|телом за время ts
нач вещ t, r, цел i
| r:=ts/n
| S2:=0
| t:=–r/2
| i:=1
| нц пока i<=n
| | t:=t+r
| | S2:=S2+f(t,v0,a)*r
| | i:=i+1
| кц
кон
алг вещ f (вещ t,v0,a)
нач
| знач:=v0+a*t
кон
	Program model9;
	Uses Crt;
	Var v0, a,ts,s2,t,r:real;
	n,i:integer;
	function f (t1:real):real;
	begin f:=v0+a*t1
	end;
	Begin
	write (‘введите a –ускорение’);
	readln (a);
	write (‘введите v0 –скорость’);
	readln (v0);
	write (‘введите ts–время движения’);
	readln (ts);
	write (‘введите n–число разбиений ts’);
	readln (n);
	r:=ts/n; S2:=0;
	t:=–r/2; i:=1;
	while i<=n do
	begin
	t:=t+r; S2:=S2+f(t)*r;
	i:=i+1;
	end;
	writeln(‘тело прошло путь s2 = ‘,
	s2:7:2);
	end.

Замечание. Программу вычисления площади криволинейной трапеции с заданной точностью методом серединных прямоугольников смотрите на с. 62.

***Пример* 2.** *1-й этап. Постановка задачи.* Тело падает с высоты *h* м, определите высоту тела над землёй через *th* секунд после начала падения. При решении задачи учтите сопротивление воздуха. Для выбранных начальных значений аргументов, проводя эксперимент, определите возможные границы *th*, задавайте при вычислительном эксперименте реальные значения *th* [10].

2-й этап. Анализ объекта моделирования и построение информационной модели. Определим исходные данные и результаты задачи. Аргументами являются: g – ускорение свободного падения, $h0$ – начальная высота падения, $v0$ – начальная скорость, th – время падения тела. Результат – высота тела над землёй через th секунд после начала падения. Мы не знаем точных формул, выражающих высоту тела над землёй с учётом сопротивления воздуха через th секунд после начала падения.

Экспериментально установлено, что сила сопротивления воздуха пропорциональна квадрату скорости, коэффициент пропорциональности зависит от формы тела. Пусть для некоторого момента времени t нам известны высота тела h_t и скорость тела v_t. Тогда ускорение можно вычислить по формуле $a_t = g - k \cdot v_t^2$, где $k \approx 0{,}004$ (для усредненного тела). Для составления информационной модели применим метод дискретизации непрерывных процессов. Разобьем время падения th на небольшие интервалы dt.



Будем считать, что на протяжении одного интервала параметры задачи не изменяются, а меняются мгновенно по истечении этого интервала (уменьшается ускорение, увеличивается скорость падения тела, уменьшается высота расположения тела над землей). Заменим плавно уменьшающуюся высоту тела над землёй на последовательность значений высот в моменты времени $t_0 = 0$, $t_1 = t_0 + dt$, $t_2 = t_0 + 2 \cdot dt, \ldots$, $t_n = t_0 + n \cdot dt$. Подсчитаем высоту тела над землёй и скорость тела через промежуток времени dt: $h_{t+dt} = h_t - v_t \cdot dt$ – считаем, что движение на этом промежутке времени равномерное; $v_{t+dt} = v_t + a_t \cdot dt$ – считаем, что ускорение a на этом промежутке не меняется.

Информационная модель данного процесса имеет вид:

$$\left.\begin{aligned} a &= g - k \cdot v \cdot v, \\ t &= t + dt, \\ h &= h - v \cdot dt, \\ v &= v + a \cdot dt, \end{aligned}\right\} \qquad (11)$$

где t – время, прошедшее с начала падения, h – расстояние тела до поверхности земли в момент времени t, v – скорость тела в момент времени t, a – ускорение падения в момент времени t, dt – интервал времени, в течение которого не изменяется ускорение и скорость.

3-4-й этапы. Алгоритмизация решения задачи, создание компьютерной модели. Выберем исполнителя Pascal-систему. Зададим начальные значения аргументов: $t = 0$, $k = 0{,}004$, $g = 10$. Введем начальные значения $v0$ и $h0$. Значения ускорения, высоты, скорости в следующий момент времени $t = t + dt$ будем вычислять по рекуррентным соотношениям (11). Определим, на какой высоте над землёй будет находиться тело через th секунд после начала падения. Вычисления проводим пока $t < th$. Программа на языке Pascal имеет вид:

```
Program height;
 var v0, h0,v, h, t, dt, a, th: real;   const g = 10; k = 0.004;
begin
   t :=0; write(‘ введите v0’); readln(v0); v :=v0; write(‘ введите th’); readln(th);
  write (‘введите h0, dt’);  readln (h0, dt); h:=h0;
  while t<th do
    begin
      t: = t + dt;  a: = g – k * v * v; h: = h – v * dt; v: = v + a * dt;
        write (‘t = ‘, t, ‘h = ‘, h, ’v= ‘, v);
     end;
end.
```

Проведите вычисления, изменяя исходные данные, оформите результаты в виде таблицы.

th	*dt*	*v0*	*h0*	*t*	*v*	*h*

6-й этап. Вычислительный эксперимент, проверка адекватности модели изучаемому процессу.

6.1. Проверим адекватность модели исследуемому процессу. Если $k = 0$, то высота тела над землёй через *th* секунд после начала падения с учетом сопротивления будет соответствовать высоте тела над землёй через *th* секунд после начала падения в вакууме. Высоту *h* и скорость тела *v* в любой момент времени падения без учета сопротивления воздуха можно вычислить точно по формулам

$$v = v_0 + g \cdot t\,;\; h = h_0 - v_0 \cdot t - g \cdot t^2 / 2. \qquad (12)$$

Определим высоту тела над землёй, используя точные формулы (12) и формулы (11), где $k = 0$, и сравним полученные результаты. Программа для эксперимента имеет вид:

```
program height1;
  var h1, v0, h0, v1, h,v, t, dt, a, th: real;
  const g = 10; k=0;
begin
t :=0; write(‘ введите v0’); readln (v0); v :=v0; write(‘ введите th’); readln(th);   write
(‘введите h0, dt’);  readln (h0, dt); h:=h0; v1:=v0;h1:=h0;
   writeln (‘ t = ‘, t, ‘ h = ‘, h,  ‘ v = ‘, v, ‘ v1 = ‘, v1, ‘ h1 = ‘, h1);
   while t<th  do
   begin
      a:= g – k*v*v;  t:= t + dt;  h:= h – v*dt;   v:= v + a*dt;
      v1:= v0 + g*t;   h1:= h0 – v0*t – g*t*t/2;
     writeln (‘ t = ‘, t, ‘ h = ‘, h,  ‘ v = ‘, v, ‘ v1 = ‘, v1, ‘ h1 = ‘, h1);
   end;
end.
```

Проведя вычислительный эксперимент с фиксированной начальной скоростью и начальной высотой и различными значениями *dt*, можно сделать вывод, что с уменьшением *dt* приближенное решение приближается к точному решению.

Проведите вычисления, изменяя исходные данные, оформите результаты в виде таблицы.

th	*dt*	*v0*	*h0*	*t*	*v*	*v1*	*h*	*h1*

6.2. Вычислим с заданной точностью высоту тела над землёй через th секунд после начала падения.

Для получения результата с заданной точностью *e* применим приём, рассмотренный в примере 1. Используя модель, найдем высоту h_1 над землей через *th* секунд после падения при выбранном *dt*; возьмем $dt = dt/2$ и вычислим h_2, высоту тела над землей через *th* секунд после падения. Затем снова уменьшим *dt* в два раза, вычислим h_3 и так далее. Вычислительный эксперимент показывает, что с увеличением числа разбиений временного отрезка *th* соседние результаты h_k и h_{k+1} мало отличаются друг от друга. Будем считать, что если $|h_k - h_{k+1}| \le e$, то h_{k+1} есть искомая высота тела над землей через *th* секунд после начала падения, вычисленная с заданной точностью *e*.

```
program fall;
 const k = 0.004;
        g = 10;
 var h1, h2, h0, h, v0, v, t, dt, a, e, th: real;
 procedure height (dt: real; var h: real);
  begin
     v:= v0; h:= h0; t:=0;
      while t < th do
           begin
           a:= g – k*v*v;  t:= t + dt;  h:= h – v*dt;  v:= v + a*dt;
           end;
  end;
begin
   write (‘Введите h0, v0, dt, e, th’);  readln (h0, v0, dt, e, th);
   height (dt, h);  h1:= h; dt:= dt/2;   height (dt, h); h2:= h;
   while abs (h2 – h1) > e do
             begin
                h1:= h2;  dt:= dt/2; height (dt, h);  h2:= h;
              end;
    write (‘ h = ‘, h2, ‘ h1 =  ‘, h1, ‘ e =’, e , ‘ th = ‘,  th);
 end.
```

Проведя вычисления, получаем значения высоты тела над землёй с различной точностью *e*.

Проведите вычисления, изменяя исходные данные, оформите результаты в виде таблицы.

$h0$	$v0$	th	dt	e	$h1$	$h2$

9.2. Метод Монте-Карло

9.2.1. Введение

Метод Монте-Карло – метод решения задач при помощи моделирования случайных величин. Датой рождения метода Монте-Карло принято считать 1949 год, когда появилась статья под названием «The Monte Carlo method». Создателями этого метода считают американских математиков Дж. Неймана и С. Улама. В Советском Союзе первые статьи о методе Монте-Карло были опубликованы в 1955-1956 годах. Теоретическая основа метода была известна уже давно, некоторые задачи статистики рассчитывались иногда с помощью специальных выборок, то есть фактически методом Монте-Карло. Однако до появления электронно-вычислительных машин этот метод не мог найти сколько-нибудь широкого применения, ибо моделировать случайные величины вручную – очень трудоёмкая работа. Возникновение метода Монте-Карло как универсального метода моделирования стало возможным только благодаря появлению электронно-вычислительных машин. Само название «Монте-Карло» происходит от города Монте-Карло в княжестве Монако, знаменитого своими игорными домами.

Как можно использователь метод Монте-Карло при составлении компьютерных моделей? Рассмотрим использование метода Монте-Карло при составлении компьютерных моделей на примере вычисления площадей плоских фигур. Предположим, что нам нужно вычислить площадь S плоской фигуры. Это может быть совсем произвольная фигура с криволинейной границей, заданная графически или аналитически, связная или состоящая из нескольких кусков.

Пусть это будет фигура *F,* изображенная на рис. 12, и предположим, что она вся расположена внутри единичного квадрата, площадь которого $S1 = 1$.

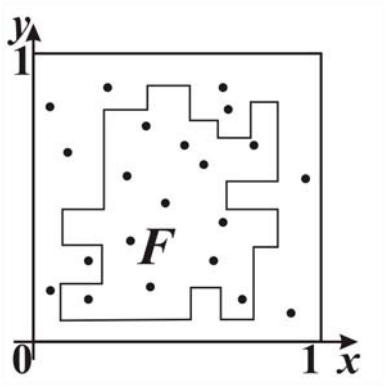


Рис. 12

Будем *многократно* бросать в квадрат случайные точки (песчинки). Обозначим через *n* количество точек, брошенных в квадрат, через *m* – количество точек, попавших при этом внутрь фигуры *F*. Геометрически очевидно, что отношение площадей $S/S1$ приближенно равно отношению m/n. Отсюда $S \approx S1 \cdot m/n$. Чем больше будет *n*, тем точнее вычисленная площадь.

На практике для вычисления площадей плоских фигур метод Монте-Карло не используют, для этого есть другие методы, хотя и более сложные, но зато обеспечивающие большую точность. Однако указанный в нашем примере подход к составлению информационных моделей, метод Монте-Карло, позволяет просто вычислять «многомерный объем» тела в многомерном пространстве. И в этом случае метод Монте-Карло часто оказывается единственным методом, дающим возможность решить задачу. Включение задач на вычисление площадей плоских фигур с использованием метода Монте-Карло в школьный предмет «Информатика и ИКТ» позволяет понять суть этого метода при моделировании. Метод Монте-Карло вычисления площадей будет справедлив только тогда, когда случайные точки будут не «просто случайными», а еще и «равномерно разбросанными» по всей плоскости.

Выделим задачи, которые решаются методом Монте-Карло. Во-первых, метод Монте-Карло позволяет моделировать любой процесс, на протекание которого влияют случайные факторы. Во-вторых, для многих математических задач, не связанных с какими-либо случайностями, можно искусственно придумать вероятностную модель, позволяющую решать эти задачи.

Две особенности метода Монте-Карло.

1. *Простая структура вычислительного алгоритма*. Как правило, составляется программа для осуществления одного случайного испытания (в примере надо выбрать случайную точку в квадрате и проверить, принадлежит ли она F), затем это испытание *многократно* повторяется, причем каждый опыт не зависит от всех остальных. Иногда метод Монте-Карло называют методом статистических испытаний.

2. *Ошибки вычислений, как правило, пропорциональны* $\sqrt{D/n}$, где D – некоторая постоянная, а n – число испытаний. Из этой формулы видно, что для того чтобы уменьшить ошибку в 10 раз (иначе говоря, чтобы получить в ответе еще один верный десятичный знак), нужно увеличить n (то есть объем работы) в 100 раз. Обычно говорят, что метод Монте-Карло особенно эффективен при решении тех задач, в которых результат нужен с небольшой точностью (5-10 %). Во многих задачах удается значительно увеличить точность, выбрав способ расчета, которому соответствует значительно меньшее значение D.

9.2.2. Примеры решения задач с использованием метода Монте-Карло

***Пример* 3.** Рассмотрим в качестве примера информационную (математическую) и компьютерную модели для приближенного нахождения площади круга радиусом r, рис. 13.

1-й этап. Постановка задачи. Дан круг радиусом r, вычислите площадь круга. Результатом решения задачи является площадь S данного круга. Будем считать, что аналитическую формулу для вычисления площади круга мы не знаем.

2-й этап. Анализ объекта моделирования и построение информационной модели. Можно предложить разные модели для решения этой задачи. Например, можно использовать палетку: на фигуру накладывается клетчатая прозрачная бумага (палетка) и подсчитывается количество квадратиков, попавших в фигуру. В этой модели *предполагается,* что чем мельче клетки, тем точнее будет результат вычис-

лений независимо от того, каким образом наложить палетку на фигуру. Можно придумать и «физическую» модель: скопировать фигуру из картона, аккуратно вырезать ее, взвесить и поделить на вес единичного квадрата из этого же картона. Однако по этим моделям трудно составить алгоритм и программу для расчетов на компьютере.

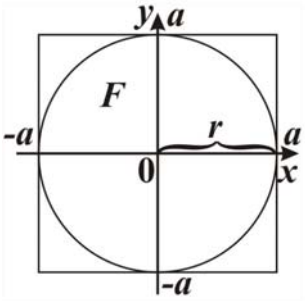


Рис. 13

Построим информационную (математическую) модель. Для построения информационной модели используем подход, рассмотренный выше для нахождения площади фигуры *F*. Поместим фигуру *F* в квадрат наименьшего размера.

Будем многократно бросать в квадрат случайные точки. Введем следующее допущение. Пусть исходы испытания распределены равномерно. Это означает, что если разделить некоторую область *S* на конечное число равновеликих частей S_i, $i = 1, 2, \ldots, n$, то можно считать, что вероятность попадания наудачу выбранной точки из области *S* в какую-либо часть S_i этой области пропорциональна мере этой части и не зависит от ее расположения и формы. Следовательно, $P(E) = m(S_i)/m(S)$, где $P(E)$ – вероятность того, что наудачу выбранная точка из области S окажется в области S_i, а $m(S_i)$ и $m(S)$ есть меры соответствующих областей, выраженных в единицах длины, площади или объема.

Исходными данными предложенной задачи являются *r* – радиус круга, количество точек *n,* которые случайным образом выбираются внутри квадрата. Число *m* – это количество точек, попавших внутрь круга, *S* – площадь круга. Результат и исходные данные связаны между собой соотношением $S = m/n \cdot S1$, где $S1$ – площадь квадрата. К связям между исходными данными и результатом следует отнести и математические соотношения, позволяющие определить, попала ли выбранная точка в фигуру *F*.

Примем, *a*=*r*, где *a* – половина длины стороны квадрата, площадь $S1$ квадрата вычисляется по формуле $S1 = 2 \cdot a \cdot 2 \cdot a$.

Случайным образом выбираем точку, принадлежащую квадрату. Выбрать точку – значит задать ее координаты, то есть задать значения *x* и *y*.

Информационная модель

$$S = m/n \cdot S1, \qquad S1 = 2 \cdot r \cdot 2 \cdot r.$$

Точка принадлежит квадрату, если выполняются соотношения $-a \le x \le a$, $-a \le y \le a$ или $-r \le x \le r$, $-r \le y \le r$. Точка принадлежит кругу, если справедливо неравенство $x^2 + y^2 \le r^2$ или $x^2 + y^2 \le a^2$.

3-4-й этапы. Алгоритмизация решения задачи и создание компьютерной модели. Выберем исполнителя программы – Basic-систему. Запрограммируем создание последовательности случайных чисел и выбор случайных чисел из этой последовательности. Для создания последовательности случайных чисел используем процедуру *randomize timer*, для выбора «следующего» случайного числа из созданной последовательности случайных чисел используем функцию *rnd,* которая возвращает случайное число в диапазоне от 0 до 1.

Разработаем программу вычисления площади круга радиуса *r=a.*

```
randomize timer ′создание последовательности случайных чисел.
Let x = (a - (-a)) * rnd + (-a)  ′выбор следующего случайного числа
'из диапазона от –a до a и присвоение этого значения переменной x.
Let y = (a - (-a)) * rnd + (-a)   'выбор следующего случайного числа
'из диапазона от –a  до a и присвоение этого значения переменной y.
```

Пару случайных чисел *x* и *y* будем рассматривать как координаты точки, принадлежащей квадрату со стороной $2a$.

Совокупность команд, определяющих, принадлежит ли точка *M(x,y)* фигуре, площадь которой нужно найти, оформим в виде подпрограммы-функции Belong%.

Программа для вычисления площади круга радиуса *r* имеет вид:

```
DECLARE FUNCTION belong% (x AS DOUBLE, y AS DOUBLE, a AS DOUBLE)
' Метод Монте-Карло для вычисления площади фигуры F
DIM a AS DOUBLE, x AS DOUBLE, y AS DOUBLE, S AS DOUBLE
DIM S1 AS DOUBLE, n AS LONG, i AS LONG, m AS LONG, r AS DOUBLE
PRINT "Введите n - количество бросаемых точек";
```

```
INPUT n
PRINT "Введите r";
INPUT r :  Let a=r:   LET m= 0:   RANDOMIZE TIMER
FOR i    = 1 TO n
      LET x = 2 * a * RND - a: LET y = 2 * a * RND - a
      IF Belong%((x), (y), (a)) = 1 THEN
          LET m = m + 1
      END IF
NEXT i
LET S1 = 2 * a * 2 * a: LET S = (m / n) * S1
PRINT "n= "; n; " s = "; S
END
FUNCTION Belong% (x AS DOUBLE, y AS DOUBLE, a AS DOUBLE)
 LET belong% = 0
 IF x * x + y * y <= a * a THEN
   LET Belong% = 1
 END IF
END FUNCTION
```

Программа на языке Turbo Pascal	Программа на языке Ершол

program area;
  uses crt;
  var a,r,S,S1,x,y:real;
    m,n,i:longint;
  Function belong:boolean;
   begin
    belong := false;
    IF x * x + y * y <= a * a THEN
     belong:= true;
   end;
begin
  write ('Введите радиус круга r ');
  readln (r);
  write ('Введите количество бросаемых
  точек n ');
  readln (n); a:=r; m:=0; randomize;
  for i:=1 to n do
   begin
	алг площадь круга (арг вещ r,арг цел
	  n,рез вещ S)
	нач вещ S1,a,x,y, цел i,m
	  a:=r; m:=0
	  нц для i от 1 до n
	  x := 2 * a * rnd(1) - a
	  y := 2 * a * rnd(1) - a;
	  если Belong (x,y,a)
	  то m := m + 1
	  все
	  кц
	  S1 := 2 * a * 2 * a; S := (m / n) * S1
	  вывод нс,S,нс
	кон
	алг лог Belong (арг вещ x,y,a)
	нач
	  знач := нет
	  если x * x + y * y <= a * a
	  то знач:= да
	  все

x := 2 * a * random - a;
y := 2 * a * random - a;
if belong then
m := m + 1;
end;
S1 := 2 * a * 2 * a; S := (m / n) * S1;
writeln('n= ', n, ' s = ', S);
end.
	кон

Замечание. Предложенные компьютерные модели можно использовать для вычисления числа π, если при выполнении программ задавать радиус круга равный единице. Программы можно дополнить имитацией «бросания» точек в квадрат и вычислением времени счета.

Компьютерная модель на языке Turbo Delphi

Предлагается программа вычисления числа π на Turbo Delphi с имитацией «бросания» точек в квадрат (рис. 14). Проанализируйте решение задачи с целью определения логической структуры компьютерной модели, понимание выбора компонентов для описания отдельных блоков компьютерной модели, механизмов реализации этих блоков.

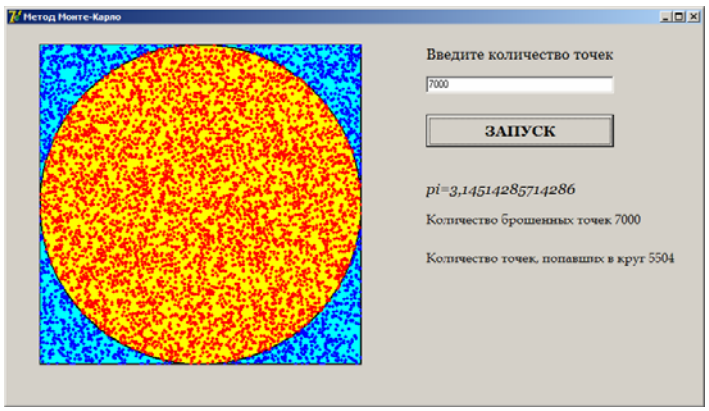


Рис. 14

```
procedure
TForm1.Edit1KeyPress(Sende
r: TObject; var Key: Char);
begin
//ограничение, чтобы в
текстовое поле могли быть введены только числа,
//также могут быть нажаты клавиши Enter и BackSpace
if (not (key  in ['0','1','2','3','4','5','6','7','8','9'])) and (key<>#13) and(key<>#8)  then
        begin
        showmessage('Могут быть введены только цифры');  key:=#0;
        end;
```

```
end;
procedure TForm1.Button1Click(Sender: TObject);
var n,k,i:longint; x,y:integer; pi:real;
begin
randomize;
image1.Canvas.pen.Color:=clBlack; image1.Canvas.Pen.Width:=2;
image1.Canvas.Pen.Style:=psSolid; image1.Canvas.Brush.Color:=claqua;
image1.Canvas.Brush.Style:=bsSolid; image1.canvas.Rectangle(0,0,400,400);
image1.Canvas.Brush.Color:=clyellow; image1.Canvas.Ellipse(0,0,400,400);
Application.ProcessMessages; image1.Canvas.Pen.Width:=1;
if edit1.text<>'' then
 begin
 n:=strtoint(edit1.Text); //количество точек, бросаемых в квадрат
 k:=0; //количество точек, попавших в круг
 for i:=1 to n do
  begin
  //задание координат точек случайным образом
  x:=random(400);    y:=random(400);
  if (sqr(x-x0)+sqr(y-y0)<=sqr(r)) then
   begin
   k:=k+1;
   image1.Canvas.Pen.color:=clRed;   image1.canvas.Brush.Color:=clred;
   image1.Canvas.Ellipse(x-2,y-2,x+2,y+2);
   image1.Canvas.FloodFill(x,y,clRed,fsborder);
   //для постепенного заполнения точками области квадрата
   if i mod 50 =0 then    Application.ProcessMessages;
   end
  else
   begin
   image1.Canvas.Pen.Color:=clBlue;   image1.canvas.Brush.Color:=clblue;
   image1.Canvas.Ellipse(x-2,y-2,x+2,y+2);
   image1.Canvas.FloodFill(x,y,clBlue,fsborder);
   if i mod 50 =0 then   Application.ProcessMessages;  end;   //end if
pi:=4*k/i;
  //вывод полученного значения pi
  label2.Caption:='pi='+floattostr(pi);
  label3.Caption:= 'Количество брошенных точек '+inttostr(i);
  label4.Caption:='Количество точек, попавших в круг '+inttostr(k);
```

```
    end;  //end for
  end
else
  ShowMessage('введите количество точек');
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
//задание размеров области рисования
image1.Height:=400; image1.Width:=400;
x0:=image1.width div 2; y0:=image1.height div 2; r:=image1.Height div 2;
//прорисовка  начального состояния области
image1.Canvas.pen.Color:=clBlack; image1.Canvas.Pen.Width:=2;
image1.Canvas.Pen.Style:=psSolid; image1.Canvas.Brush.Color:=claqua;
image1.Canvas.Brush.Style:=bsSolid; image1.canvas.Rectangle(0,0,400,400);
image1.Canvas.Brush.Color:=clyellow; image1.Canvas.Ellipse(0,0,400,400);
end;
```

6-7-й этапы. Вычислительный эксперимент, анализ результатов эксперимента. Для проведения вычислительного эксперимента выберите радиус круга, равным единице. Площадь круга в этом случае будет равна π, результат работы программы – число π.

Результаты эксперимента оформите в виде таблицы:

n			…		…
π			…		…

Замечание. Рассмотренные в пункте 9.2.2 математические модели являются вероятностными. Результаты, полученные с их помощью, подчиняются закону больших чисел, то есть в данном случае точность в определении площади зависит от количества *n* брошенных случайно точек. При большом количестве брошенных случайно точек площадь круга единичного радиуса приближается к значению π.

***Пример* 4.** *1-й этап. Постановка задачи*. Составьте компьютерную модель для вычисления площади фигуры *F*, изображенной на рис. 15. Проведите анализ предложенных решений задачи: вспомните схему этапов составления компьютерной модели объекта, выделите цели

реализации этих этапов, определите, достигнуты ли эти цели в предложенных решениях.

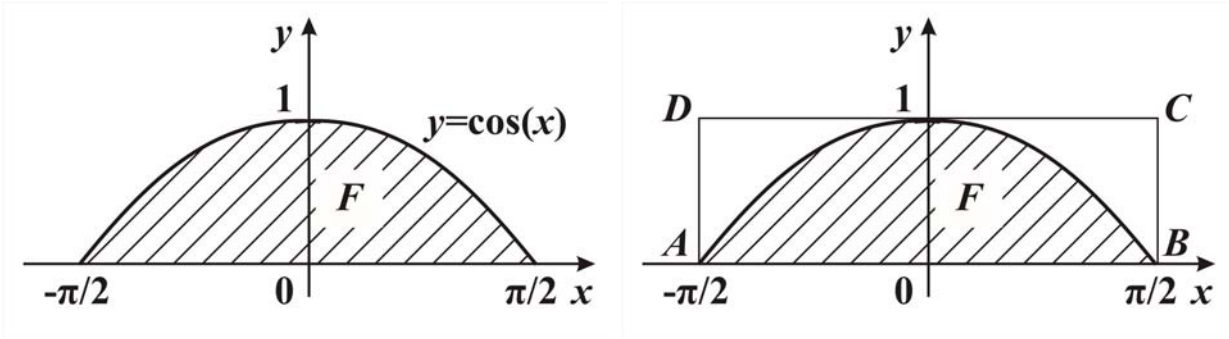


Рис. 15

2-й этап. Анализ объекта моделирования и построение информационной модели. Поместим фигуру F в прямоугольник $ADCB$ наименьшего размера, площадь которого легко вычислить:

$$S_{ADCB} = \pi \cdot 1 = \pi .$$

Будем бросать в прямоугольник n случайных точек и определять число m – число точек, попавших внутрь фигуры F. Составим информационную модель процесса нахождения площади фигуры F с использованием метода Монте-Карло.

Результат и исходные данные связаны между собой соотношением $S = m/n \cdot S1$, где S – площадь искомой фигуры F, а $S1$– площадь прямоугольника. К связям между исходными данными и результатом следует отнести и математические соотношения, позволяющие определить, попала ли выбранная точка в фигуру F.

Точка принадлежит прямоугольнику $ADCB$, если выполняются соотношения

$$\begin{cases} -\pi/2 \le x \le \pi/2, \\ 0 \le y \le 1. \end{cases}$$

Точка принадлежит фигуре F, если выполняются соотношения

$$\begin{cases} -\pi/2 \le x \le \pi/2, \\ y \le \cos(x). \end{cases}$$

3-4-й этапы. Алгоритмизация решения задачи и создание компьютерной модели. Выберем исполнителя Ершол–систему. Получить случайное вещественное число x в диапазоне $[A, B)$ можно по формуле x:= (B–A)*rnd(1)+A. Получить случайное вещественное число x в диапазоне $-\pi/2 \le x \le \pi/2$ можно операторами присваивания pi:=4*arctg(1); x:=pi*rnd(1)-pi/2. Получить случайное вещественное число y в диапазоне $0 \le y \le 1$ можно оператором присваивания y:=rnd(1).

Программа на языке Ершол.

```
алг площадь
нач цел n,i,m, вещ s,x,y,pi
   вывод "Введите количество бросаемых точек "; ввод n
   pi:=4*arctg(1); m:=0
   нц для i от 1 до n
      x:=pi*rnd(1.000001)-pi/2; y:=rnd(1.000001)
      если y<=cos(x)
         то m:=m+1
      все
   кц
   s:=(m/n)*pi
   вывод "Площадь фигуры F равна ",s
   вывод нс, "Точное значение площади фигуры F равно",2
кон
```

6-7-й этапы. Вычислительный эксперимент, анализ результатов эксперимента. Вычислим площадь фигуры *F,* используя формулу Ньютона – Лейбница: $\int_{-\pi/2}^{\pi/2} \cos(x)dx = \sin(x)\Big|_{-\pi/2}^{\pi/2} = \sin(\pi/2) - \sin(-\pi/2) = 2.$

Результаты эксперимента оформите в виде таблицы.

n			…		…
S			…		…

Сравните результаты вычислений по формуле $S = m/n \cdot S1$ с точным результатом $S = 2$, полученным при решении задачи с использованием формулы Ньютона – Лейбница.

Выполнение этапов 2 – 7 для решения поставленной задачи с использованием приближённого нахождения площади криволинейной трапеции методом серединных прямоугольников[3].

2-й этап. Анализ объекта моделирования и построение информационной модели. Требуется приближённо вычислить площадь фигуры, ограниченной графиком непрерывной функции $y = f(x)$, где $f(x) = \cos(x)$ и $f(x) \geq 0$, прямыми $x = -\pi/2$, $x = \pi/2$, $y = 0$, рис. 16. Фигура, изображенная на рис. 16, является *криволинейной трапецией*.

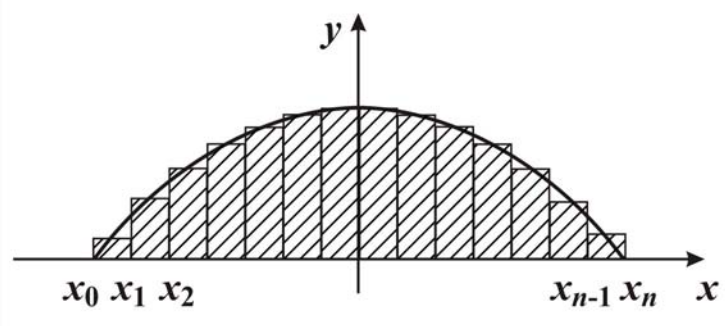


Рис. 16

Отрезок $[-\pi/2; \pi/2]$ разобьём на n элементарных отрезков длины h точками $x_0, x_1, \ldots, x_n$, $-\pi/2 = x_0 < x_1 < x_2 < \ldots < x_n = \pi/2$.

Введём обозначение $h = (\pi/2 - (-\pi/2))/n$ или $h = \pi/n$. На каждом из полученных отрезков $[x_{i-1}, x_i]$ построим прямоугольник, одной стороной которого будет отрезок $[x_{i-1}, x_i]$, а другой – отрезок, длина которого равна $f(x_{i-1} + h/2)$, $i = 1, 2, \ldots, n$.

Площадь криволинейной трапеции можно приближённо считать равной сумме площадей заштрихованных прямоугольников, построенных на каждом из отрезков деления. Для заданной функции площадь вычисляется по формуле серединных прямоугольников так:

$$S = h(\sin(x_0 + h/2) + \sin(x_1 + h/2) + \ldots + \sin(x_{n-1} + h/2)),$$

[3] В содержание предмета «Информатика и ИКТ», мы считаем, необходимо включить раздел «Численные методы и компьютерное моделирование», обоснование введения и примерное содержание этого раздела предложено в разработке [14].

где $h = \pi/n$, $x_0 = -\pi/2$.

3-4-й этапы. Алгоритмизация решения задачи и создание компьютерной модели. Ершол–систему выберем исполнителем программы. Составим для исполнителя программу вычисления площади фигуры *F* для фиксированного значения *n*.

```
алг площадь
нач цел n, вещ s, x, pi, h
  вывод "Введите количество разбиений отрезка [-pi / 2, pi / 2] "
  ввод n
  pi := 4 * arctg (1);   h := pi / n ;   x := -pi / 2 – h / 2 ;   s := 0
  нц пока x <= pi / 2 - h / 2
    x := x + h; s := s + h * f(x)
  кц
  вывод "Площадь фигуры F равна ", s
  вывод нс, "Точное значение площади фигуры F равно",2
кон
алг вещ f (арг вещ x)
нач
  знач := cos(x)
кон
```

Результаты эксперимента оформите в виде таблицы.

n			…		…
S_n			…		…
Sт.	2	2	2	2	2

Сравните результаты вычислений площади фигуры *F* с использованием формулы серединных прямоугольников для фиксированного n с точным результатом $S = 2$, полученным при решении задачи с использованием формулы Ньютона – Лейбница. Вычислительный эксперимент показывает, что с увеличением *n*, числа разбиений отрезка $[-\pi/2; \pi/2]$, площади фигуры *F*, вычисленные по приближенной формуле, мало отличаются друг от друга.

Составим программу для вычисления площади фигуры F с точностью e методом серединных прямоугольников. Для вычисления площади фигуры F с заданной точностью воспользуемся способом, предложенным в примерах 1, 2. Вычисление площади фигуры F выполняется два раза с заданным числом разбиений n и числом разбиений $2n$. Проверяется условие $|S_{2n} - S_n| \le e$[4], если оно истинно, то за значение площади фигуры F берётся значение S_{2n}, и вычисления заканчиваются. Если условие ложно, то число разбиений снова удваивается, переменная S_n принимает значение S_{2n}, вычисляется значение площади фигуры F при вновь удвоенном n, и это значение присваивается переменной S_{2n}, проверяется выполнение условия $|S_{2n} - S_n| \le e$ и т. д. до тех пор, пока проверяемое на каждом шаге условие не будет истинным.

Программа для приближённого вычисления площади с заданной точностью:

```
алг площадь1
нач цел n, вещ s1, s, e, s2
   вывод "Введите количество разбиений отрезка [-pi / 2,pi / 2] "; ввод n
   вывод нс,"Введите точность вычисления e "; ввод e
   площадь(n, s) ; s1:= s; n := n * 2;площадь(n, s); s2 := s
   нц пока abs(s1- s2) > e
      s1:= s2; n := 2*n; площадь(n, s); s2 := s
   кц
   вывод "Площадь фигуры F равна ", s1
   вывод нс, "Точное значение площади фигуры F равно", 2
кон
алг площадь (арг цел n, рез вещ s)
нач вещ x, pi, h
   pi:=4 * arctg(1); h := pi / n ; x := -pi / 2 – h / 2; s := 0
   нц пока x <= pi / 2 – h / 2
```

[4]Если учащиеся рассматривали учебный материал раздела «Численное интегрирование», то для вычисления площади криволинейной трапеции по методу серединных прямоугольников с заданной точностью можно воспользоваться формулой $|S_{2n} - S_n| \le 3 \cdot e$.

```
        x := x + h; s := s + h * f(x)
    кц
кон
алг вещ f (арг вещ x)
нач
    знач := cos(x)
кон
```

Результаты эксперимента оформите в виде таблицы.

e			…		…
S_{2n}			…		…
Sт.	2	2	…	2	2

***Пример* 5.** *Задача о встрече. 1-й этап. Постановка задачи.* Каждый день резидент приходит на встречу со своим агентом в случайный момент времени с 13 до 14 часов и ждет его 20 минут, после чего уходит. В свою очередь агент приходит на встречу с резидентом каждый день также в случайный момент времени с 13 до 14 часов и тоже ждет 20 минут, после чего уходит. Какова вероятность встречи этих лиц, если каждый в течение часа приходит к этому месту наудачу и моменты прихода независимы друг от друга?

2-й этап. Анализ объекта моделирования и построение информационной модели. Изобразим исходы испытаний на координатной плоскости. На прямой Ox от точки $O(0,0)$ отложим отрезок, равный единице длины, принимая за единицу масштаба один час, – это время прихода резидента. На прямой Oy от точки $O(0,0)$ тоже отложим отрезок, равный единице длины – это время прихода агента. Тогда множество всех возможных исходов испытания можно изобразить точками квадрата, сторона которого равна единице. Встреча произойдет лишь в том случае, если разность моментов прихода резидента и агента по абсолютной величине не будет превосходить $\frac{1}{3}$ часа (два-

дцати минут), т. е. если произойдет событие, удовлетворяющее неравенству $|y-x| \le \frac{1}{3}$.

Запишем это неравенство в виде системы неравенств:

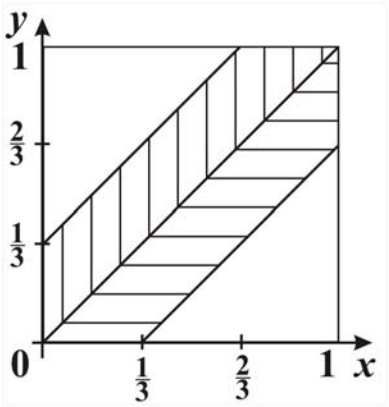


Рис. 17

$$\begin{cases} y \ge x - \frac{1}{3}, \\ y \le x + \frac{1}{3}, \\ 0 \le x \le 1. \end{cases}$$

Эта система неравенств задает заштрихованную область на рис. 17. Искомая вероятность равна отношению площади заштрихованной области к площади квадрата.

3-4-й этапы. Алгоритмизация решения задачи и создание компьютерной модели.

1-й способ нахождения вероятности встречи резидента и агента. Для получения результата воспользуемся точными формулами вычисления площадей:

$$P\left(|x-y| \le \frac{1}{3}\right) = \frac{1^2 - \left(\frac{2}{3}\right)^2}{1^2} \approx 0{,}56.$$

2-й способ нахождения вероятности встречи резидента и агента. Воспользуемся для решения задачи методом Монте-Карло. Будем *многократно* бросать в квадрат случайные точки (песчинки). Обозначим через n количество точек, брошенных в квадрат, через m – количество точек, попавших при этом внутрь заштрихованной фигуры. Геометрически очевидно, что отношение площадей $S/S1$, где S – площадь заштрихованной фигуры, приближенно равно отношению m/n. Отношение площадей фигур заменим отношением m/n. При большом количестве брошенных случайно точек вероятность встречи агента и резидента будет вычислена точнее.

Составим компьютерную модель для исполнителя Turbo Pascal.

```
Program meeting;
 uses crt;
 Var I,n,m: longint; x,y,k: real;
Begin
  write ('Введите количество бросаемых
              точек n ');
  readln (n);
  i:=1;m:=0;
  Randomize;
While i<=n do
  begin
    x:=random;
    y:=random;
    If abs(x-y)<=1/3 then m:=m+1;
    i:=i+1;
  end;
  k:=m/n*1;
  writeln(' вероятность встречи =  ',k);
  writeln(1-4/9);
end.
```

6-7-й этапы. Вычислительный эксперимент, анализ результатов эксперимента.

Результаты эксперимента оформите в виде таблицы.

n			…		…
P			…		…

Сравните результаты вычислений с результатом, полученным при решении задачи первым способом $P \approx 0,56$.

Изложенные выше теоретический материал и методы решения жизненных задач можно использовать при подготовке к семинарским и лабораторно-практическим занятиям по данной теме.

10. Вопросы и задания к семинарским занятиям

Тема «Моделирование и формализация. Вычислительный эксперимент»

Выполните задания 1, 2, 3, 4, 5, 6, ответьте на вопросы, поставленные в этих заданиях. Доложите результаты выполнения данных заданий в студенческой аудитории в форме презентации.

1. Определите место и роль темы «Моделирование и формализация. Вычислительный эксперимент» в решении общеобразовательных задач предмета «Информатика и ИКТ». Какое место и роль отводят авторы учебных пособий данной теме и почему? Какое место и

роль отвели бы Вы? Выбор обоснуйте. Как менялось со временем место, роль и содержание данной темы?

2. Проведите сравнительный дидактический анализ содержания учебного материала по данной теме в различных учебниках и учебных пособиях по предмету «Информатика и ИКТ» для общеобразовательных учреждений на каждой ступени непрерывного курса изучения информатики: начальный курс (II – VI классы), базовый курс (VII – IX классы), профильный курс обучения (X – XI классы). Соотнесите содержание учебного материала с требованиями государственного стандарта образования по информатике.

3. Сформулируйте цели и задачи, стоящие перед учителем в процессе организации изучения школьниками данной темы на каждой ступени непрерывного курса изучения информатики. Какие учебные цели соответственно должны стоять перед учащимися?

4. Проанализируйте программное обеспечение к теме «Моделирование и формализация. Вычислительный эксперимент» как средство получения учащимися предполагаемых знаний, умений и навыков. Определите дидактические цели использования каждого выбранного программного обеспечения при изучении темы. Выделите основные методы и способы составления компьютерных моделей и проведения вычислительного эксперимента при решении задач с использованием выделенного программного обеспечения.

5. Выявите базовые понятия темы «Моделирование и формализация. Вычислительный эксперимент», определите этапы, формы и методы их формирования, установите отношения между выделенными понятиями. Составьте терминологический словарь по базовым понятиям темы. Определите общеобразовательный и мировоззренческий аспекты базовых понятий темы.

6. Отберите содержание учебного материала по теме «Моделирование и формализация. Вычислительный эксперимент» в соответствии с особенностями развития школьника на выбранном конкретном возрастном этапе. Составьте логико-структурную модель учебного материала по теме на выбранной ступени непрерывного курса изучения информатики.

7. Отберите содержание учебного материала и разработайте конспект одного урока по предложенным ниже темам на выбранной ступени непрерывного курса изучения информатики. Апробируйте в студенческой аудитории разработанный конспект урока.

7.1. «Компьютерное моделирование»; «Формализация, как процесс построения информационных моделей»; «Вычислительный эксперимент с использованием компьютера».

7.2. «Понятие модели, цели создания моделей, классификация моделей по различным признакам».

7.3 «Доказательство адекватности модели изучаемому объекту, процессу».

8. Подберите практическую задачу на выбранной ступени непрерывного курса изучения информатики и на её основе разработайте фрагмент урока по теме «Этапы решения задач с использованием компьютера». Выделите цели каждого этапа. Защитите свои разработки в студенческой аудитории.

9. Сконструируйте серию уроков по изучению содержательной линии «Моделирование и формализация», в которых для составления информационных моделей используются:

9.1. метод дискретизации непрерывных процессов;

9.2. метод Монте-Карло – моделирование с использованием случайных величин.

Апробируйте в студенческой аудитории фрагмент одного из разработанных уроков по указанным темам.

10. На основе классификации компьютерных моделей предложите типологию задач, используемых в процессе изучения данной содержательной линии на выбранной ступени непрерывного курса изучения информатики. Защитите свои разработки по классификации задач в студенческой аудитории.

11. Проанализируйте дидактические возможности учебного материала содержательной линии «Моделирование и формализация» на каждой ступени непрерывного курса изучения информатики для реализации задач:

11.1. формирования системно-информационных представлений и информационной культуры учащихся, в процессе изучения информатики (понимания, что моделирование – метод научного познания, понимания сущности метода информационного моделирования, умения строить информационные модели с помощью формальных языков – навыки формализации, умения структурировать изучаемый объект, понимания смысла отношения объект – модель);

11.2. развивающего обучения.

Доложите результаты своего дидактического анализа в студенческой аудитории.

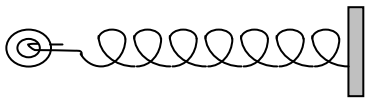
11. Лабораторно-практическая работа

Тема «Моделирование и формализация. Вычислительный эксперимент»

Используйте для решения задач исполнителей: Ершол, QBasic, Turbo Pascal, Visual Basic.Net, Turbo Delphi, Microsoft Excel и др.

Для решения задач 1, 2, 3 составьте компьютерные модели, рассмотрев все этапы реализации компьютерных моделей, для составления информационных моделей используйте метод дискретизации непрерывных процессов. Смотрите примеры 1, 2 теоретической части. Проведите презентацию своей работы в студенческой аудитории.

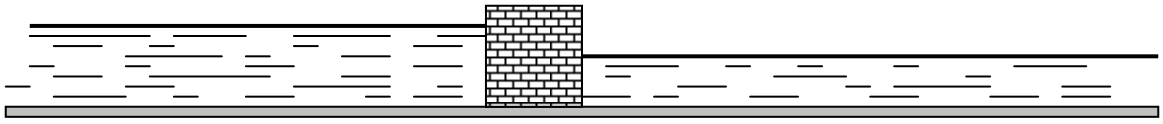
***Задача* 1.** Выполняя утреннюю зарядку, школьник подошел к стене, на которой был закреплен пружинный эспандер, и оттянул его на некоторое расстояние. Какую работу совершила при этом сила натяжения пружины [14]?



Указание к решению. При построении математической модели для этой задачи сделайте следующие предположения: сила натяжения пружины подчинена закону Гука; если весь промежуток движения тела разбить на большое число равных маленьких промежутков, то

можно считать, что сила на каждом из них постоянна и меняется «мгновенно» в конце каждого промежутка; при неограниченном увеличении числа этих промежутков величина работы получается с любой точностью.

***Задача* 2.** Плотина прямоугольной формы перегораживает реку. Определить силу давления воды на плотину [14].

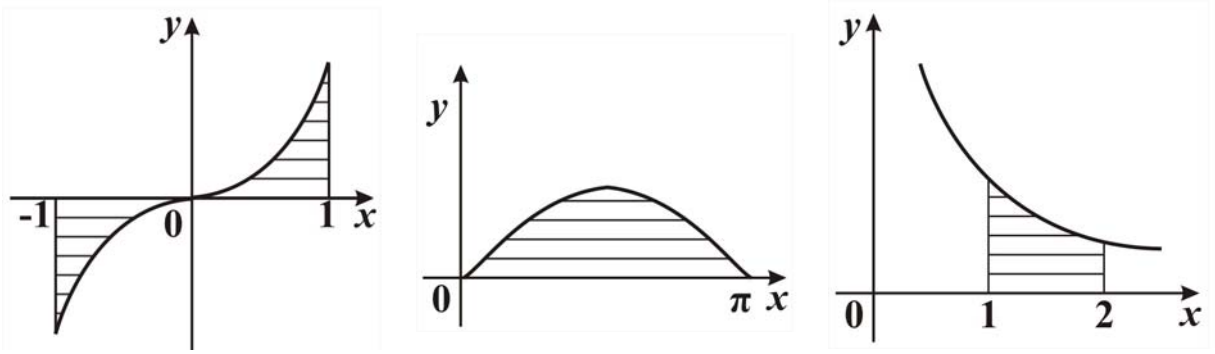


***Задача* 3.** Тело падает с высоты h м, необходимо определить время падения тела с учетом сопротивления. Экспериментально установлено, что сила сопротивления воздуха пропорциональна квадрату скорости, а коэффициент зависит от формы тела. Поэтому ускорение падающего тела вычисляется по формуле $a = g - k \cdot v^2$, где k – коэффициент, зависящий от массы и формы тела (для человека среднего роста и веса $k = 0{,}004$). Для простоты будем считать, что падение происходит на планете с ускорением свободного падения $g = 10$ м/с^2. Смотрите пример 2 теоретической части пособия.

Для решения задач 4 – 9 составьте компьютерные модели, рассмотрев все этапы решения задач с использованием компьютера, используйте для составления информационных и компьютерных моделей метод Монте-Карло. Проведите вычислительный эксперимент, результаты работы программ соотнесите с теоретическими результатами. Смотрите примеры 3 – 5 теоретической части. Проведите презентацию своей работы в студенческой аудитории.

***Задача* 4.** Составьте компьютерные модели вычисления площадей заштрихованных областей фигур, изображенных на рис. 18.

***Задача* 5.** Дано линейное уравнение $ax = b$. Если a выбирается наудачу на отрезке $0 \le a \le 8$, b – на отрезке $0 \le b \le 10$, то какова вероятность того, что корень данного уравнения будет больше единицы? Предполагается, что вероятность попадания точки в плоскую фигуру пропорционально площади фигуры и не зависит от её расположения.



$y = x^3$ $y = \sin(x)$ $y = \frac{1}{x}$

Рис. 18

***Задача* 6.** Два действительных числа x и y выбираются наугад так, что сумма их квадратов меньше 100. Какова вероятность того, что сумма квадратов x и y окажется больше 64? Предполагается, что вероятность попадания точки в плоскую фигуру пропорциональна площади фигуры и не зависит от её расположения.

***Задача* 7.** На отрезке OA длины L числовой оси Ox наудачу поставлены две точки $B(x)$ и $C(y)$, причём $y \geq x$. Найдите вероятность того, что длина отрезка BC будет меньше длины отрезка OB. Предполагается, что вероятность попадания точки на отрезок пропорциональна длине этого отрезка и не зависит от его расположения на числовой оси.

***Задача* 8.** На отрезке OA длины L числовой оси Ox наудачу поставлены две точки $B(x)$ и $C(y)$. Найдите вероятность того, что из трёх получившихся отрезков можно построить треугольник.

***Задача* 9.** Два студента условились встретиться на площади у фонтана между 15 и 17 часами дня. Студент, пришедший первым, ждёт второго в течение 1/2 часа, после чего уходит. Найдите вероятность того, что встреча состоится, если каждый студент наудачу выбирает время своего прихода (в промежутке от 15 до 17 часов).

12. Самостоятельная работа

Тема «Моделирование и формализация. Вычислительный эксперимент»

Используйте для решения задач исполнителей: Ершол, QBasic, Turbo Pascal, Visual Basic.Net, Turbo Delphi, Microsoft Excel и др.

Для выполнения заданий 1, 2, 3 разработайте и реализуйте с использованием выбранного исполнителя учебные проекты в виде компьютерных моделей, рассмотрев все этапы составления компьютерных моделей. Подготовьте проекты к защите.

***Задание* 1.** *Компьютерная модель «Электронный кассир»*

Пусть в кинозале 24 ряда по 35 мест, а цена билета зависит только от номера ряда. Составьте программу, которая могла бы имитировать продажу билетов в кинозал:

– высвечивать информацию о проданных и свободных местах на данный сеанс;

– осуществлять поиск одного, двух, трех свободных мест;

– производить расчёт с посетителем при покупке билетов;

– подсчитывать выручку;

– обновлять табло с информацией о свободных местах после продажи билетов посетителю.

***Задание* 2.** *Компьютерная модель «Гараж»*

Гараж содержит одну полосу, на которой может быть размещено до *n* автомашин. Автомобили въезжают в гараж с запада, а покидают его с восточного конца. Если автомобиль владельца, пришедшего забрать автомобиль, не расположен восточнее всех остальных, то все автомобили, стоящие восточнее этой машины, удаляются из гаража на запасную полосу, затем выезжает машина владельца, пришедшего забрать автомобиль (рис. 19). Машины с запасной полосы помещаются назад на свои места. Все машины, расположенные западнее выехавшей машины, сдвигаются на восток столько раз, сколько имеется свободных позиций в восточной части.

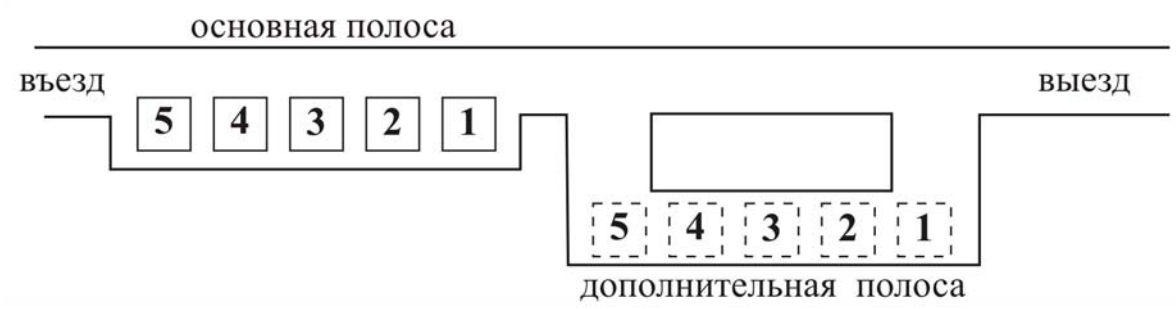


Рис. 19

Указания к выполнению задания (предполагаемые технические требования).

- Программа начинается с заставки, где указываются некоторые сведения о программе и авторе программы.
- В диалоговом режиме программа запрашивает число автомобилей в гараже и их номера.
- На экране должна отражаться динамика заполнения гаража машинами.
- Программа должна запрашивать номер машины, которая покидает гараж.
- На экране отображается динамика всех перемещений автомобилей.

***Задание* 3.** *Выбор места строительства железнодорожной станции.*

В одном районе расположены три населенных пункта. По территории района проходит железная дорога. По просьбе жителей района планируется построить железнодорожную станцию и проложить дороги от нее до каждого населенного пункта. Требуется определить наиболее удобное расположение железнодорожной станции. Место для станции надо выбрать так, чтобы наибольшее из расстояний от неё до населённых пунктов было как можно меньше.

Указание к выполнению задания. Выберите систему координат так, чтобы ось абсцисс проходила по интересующему нас участку железной дороги, а начало координат совпало с левым концом дороги. Для составления информационной модели примените метод

дискретизации непрерывных процессов, разбейте отрезок [0; S] на n равных частей, S – длина отрезка, изображающего на карте участок железной дороги, пролегающий по рассматриваемой местности.

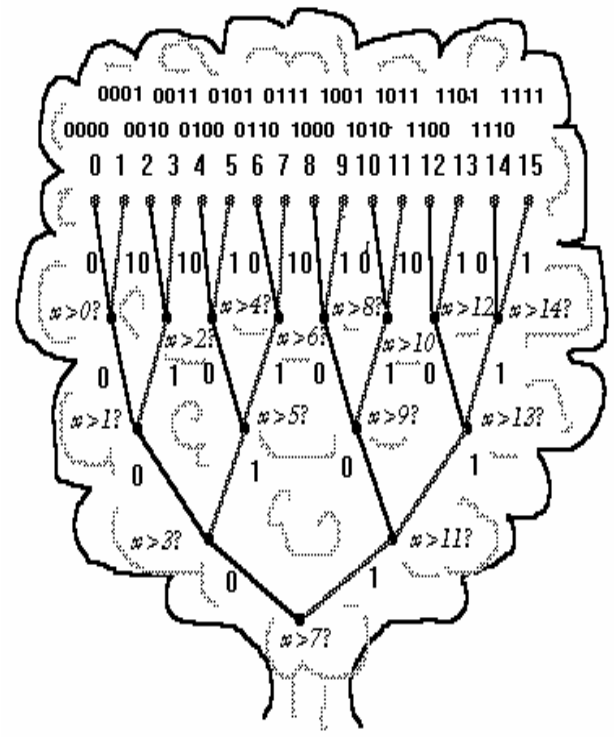
Выполните задания 4, 5, 6. Проделанную и оформленную работу подготовьте для защиты.

***Задание* 4.** Составьте поурочный план изучения данной темы по предмету «Информатика и ИКТ» на одной из ступеней непрерывного курса изучения информатики.

Задание 5. Разработайте тематику, содержание, методы и приёмы проведения лабораторных работ по теме «Моделирование и формализация. Вычислительный эксперимент» на каждой ступени непрерывного курса изучения информатики.

Задание 6. Разработайте сценарии программ (демонстрационных, обучающих, тренажеров, контролирующих) по данной теме и реализуйте один проект на выбранном программном обеспечении.

ГЛАВА 2. ГРАФЫ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ



«Теория графов предоставляет очень удобный язык для описания программных (да и многих других) моделей. Стройная система специальных терминов и обозначений теории графов позволяет просто и доступно описывать сложные и тонкие вещи. Особенно важно наличие наглядной графической интерпретации понятия графа. Картинки позволяют сразу «усмотреть» суть дела на интуитивном уровне, дополняя и украшая утомительные рациональные текстовые доказательства и сложные формулы»

Ф. А. Новиков

Методы решения задач с использованием теории графов позволяют учащимся получить опыт построения и исследования моделей *реальных* объектов на компьютере. Для успешного применения теории графов при решении задач необходимо научить школьников видеть в условии задачи структуру графа и грамотно переводить это условие на язык теории графов.

1. Мотивационные задачи к введению понятия «граф»

***Задача* 1.** Между девятью планетами Солнечной системы введено космическое сообщение, ракета летает по следующим маршрутам: Земля – Меркурий, Плутон – Венера, Земля – Плутон, Плутон – Мер-

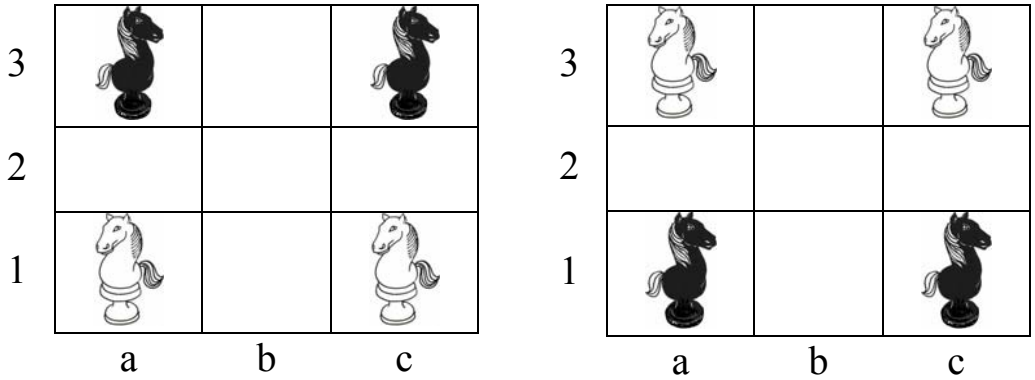
курий, Меркурий – Венера, Уран – Нептун, Нептун – Сатурн, Сатурн – Юпитер, Юпитер – Марс, Марс – Уран. Можно ли долететь от Земли до Марса?

Решение. Объектом исследования является часть Солнечной системы. Проведём системный анализ объекта. Выделим элементы системы, это планеты. Установим связь между элементами: две планеты взаимосвязаны, если между ними установлено космическое сообщение. Нарисуем схему: планеты обозначим точками, а соединяющие их маршруты линиями (рис. 20).

Рис. 20

Так как нет ломаной линии, соединяющей Землю и Марс, то можно сделать вывод, что долететь от Земли до Марса при данном сообщении нельзя.

***Задача* 2.** Задача о перестановке четырёх коней (рис. 21), исполнитель – Конюх. Напишите алгоритм перестановки коней из начального положения в конечное.



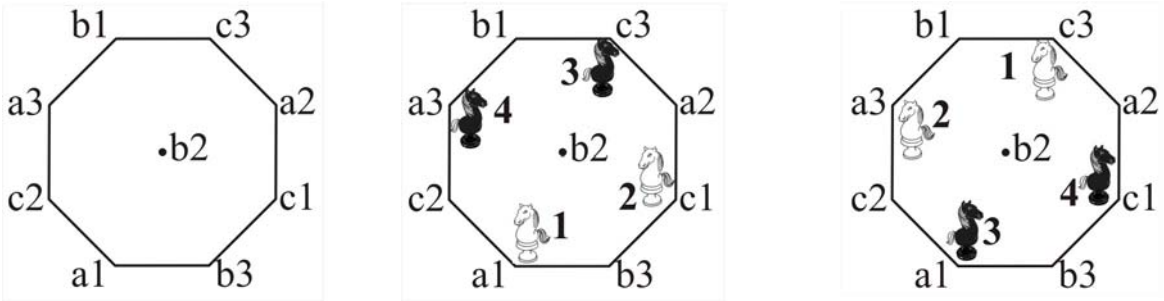
Начальное положение Конечное положение

Рис. 21

Конь перемещается за один ход либо на два поля по вертикали и одно поле по горизонтали, либо на два поля по горизонтали и одно по

вертикали (буквой «Г»). Конь может перепрыгивать через стоящие на его пути другие фигуры, но может ходить в нашей задаче только на свободные поля. Составьте для решения задачи графическую информационную модель.

Решение. Объектом исследования является клетчатое поле 3×3. Проведем системный анализ объекта. Выделим элементы системы – клетки поля. Установим связь между элементами: две клетки взаимосвязаны, если из одной клетки в другую можно попасть ходом шахматного коня. Изобразим клетки поля точками, если из одной клетки поля можно попасть в другую ходом шахматного коня, то соединим эти клетки отрезком прямой.



Начальное положение Конечное положение

Рис. 22

Начальная и конечная расстановки коней изображены на рис. 22. Написание алгоритма перестановки коней для исполнителя Конюха становится очевидным.

***Задача* 3.** Задача о Кёнигсбергских мостах (см. с. 16).

***Задача* 4.** На встрече группы хорошо и не очень хорошо знакомых делегатов конгресса по информатике состоялось много приветственных рукопожатий. Некоторые делегаты пожали четное число рук, другие – нечетное. Например, Сергей обменялся тремя рукопожатиями, а его друг, Николай, – девятью. Когда Сергей сказал своему другу, что обменявшихся нечетным числом рукопожатий, кроме Сергея и Николая, было еще 5 человек, он ответил: ошибаешься. Число людей,

пожавших нечетное число рук, непременно должно быть чётным. Прав ли он? Объясните свои выводы.

***Задача* 5.** В некоторой стране 110 городов, между некоторыми из них летают самолеты. Авиатрассы расположены так, что из любого города можно перелететь в другой (возможно с пересадками). Определите, можно ли из города *A* добраться в город *B*, делая не более 1, 2 или 3 пересадок. Составьте информационную и компьютерную модели для решения поставленной задачи.

***Задача* 6.** *Уникурсальной* называется линия, которую можно построить, не отрывая карандаша от бумаги и не проходя дважды по одному и тому же звену. Дана линия. Разработайте компьютерную модель, которая определяет, является ли данная линия уникурсальной, и выводит на экран один из маршрутов обхода линии, удовлетворяющий условию задачи (рис. 23).

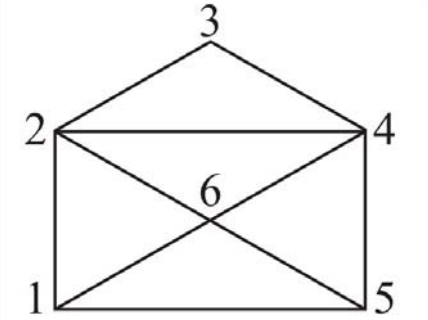


Рис. 23

2. Некоторые понятия теории графов

Графом G называют *пару множеств* (*W,E*), где *W* – это непустое конечное множество элементов, которое мы назовём вершинами графа, а *E* – это конечное множество неупорядоченных или упорядоченных пар элементов множества *W*. Элементы множества *E* –рёбра или дуги. Неупорядоченную пару вершин {*V*, *U*} будем называть ребром, а упорядоченную пару назовем дугой <*V*, *U*> (*V* – начало дуги, *U* – конец дуги). При изображении графов на рисунках рёбра и дуги будем изображать линиями, у дуг на концах будем изображать стрелки, указывающие направление связи вершин, вершины – точками, окружностями или квадратами. Такое представление графа называется *диаграммой.* Диаграмма графа – это *информационная модель* объекта, это информация о составе и структуре системы, представленная в графической форме.

Граф, содержащий только рёбра, называется *неориентированным* графом. Граф, содержащий только дуги, называется *ориентированным* графом, или орграфом. Мощности множеств W и E будем обозначать соответственно n и m (n – количество вершин, m – количество рёбер). Обычно вершины обозначаются заглавными буквами русского или латинского алфавита или числами, рёбра обозначаются парами вершин или строчными буквами русского и латинского алфавита. Когда берем пару вершин $\{V, U\}$, то будем говорить, что ребро соединяет вершины V и U. Две вершины U и V графа G называются *смежными,* если существует соединяющее их ребро $\{U, V\}$. При этом вершины U и V называются *инцидентными* этому ребру, а ребро *инцидентным* этим вершинам. Два ребра называются *смежными,* если они имеют, по крайней мере, одну общую вершину. В орграфе вершина U называется смежной вершине V, если существует дуга $<U, V>$. При этом вершины U и V называются *инцидентными дуге* $<U, V>$, а дуга – *инцидентной* соответствующим *вершинам*.

Примеры

Неориентированный граф (рис. 24)

$W = \{1, 4, 3, 2\}$

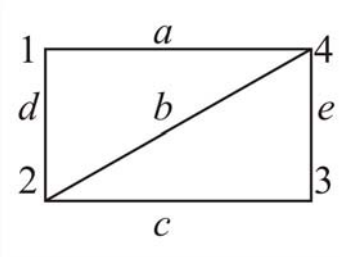


Рис. 24

$E = \{a, b, c, d, e\}$

$E = \{\{1, 4\}, \{2, 4\}, \{2, 3\}, \{1, 2\}, \{3, 4\}\}$

Мощности: $n = 4$, $m = 5$.

Ориентированный граф (рис. 25)

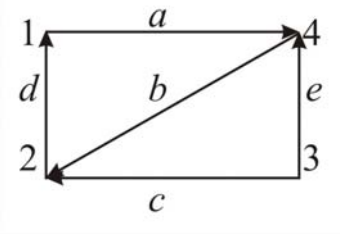


Рис. 25

$W = \{1, 4, 3, 2\}$

$E = \{<2, 1>, <3, 2>, <3, 4>, <1, 4>, <4, 2>\}$

$n = 4$, $m = 5$.

Степенью вершины называется число рёбер графа, которым принадлежит эта вершина. Вершина графа, имеющая нечётную степень, называется нечётной вершиной. Вершина графа, имеющая чётную степень – чётной верши-

ной. Вершина графа, имеющая степень 0, называется изолированной. Вершина графа, имеющая степень 1, называется висячей.

Теорема. Число нечётных вершин любого графа – число чётное.

***Задача* 7.** В классе 25 человек. Может ли быть так, что 7 из них имеют по три друга в этом классе, 11– по 4 друга, а 7 – по 5 друзей.

Решение. Граф имеет 25 вершин, 7 из них имеют по три ребра – степень 3, 10 – степень 4 и 8 – степень 5. Нечетных вершин – 15, такого быть не может!

Теорема. В графе G сумма степеней всех его вершин – число чётное, равное удвоенному числу рёбер графа.

***Задача* 8.** В посёлке Бессвязном всего 7 телефонов. Можно ли их соединить проводами так, чтобы каждый телефон был соединен ровно с тремя другими?

Решение. (3·7)/2 – нецелое число, (нельзя).

Путём от вершины U к вершине V в графе называется такая последовательность рёбер, ведущая от U к V, в которой каждые два соседних ребра имеют общую вершину, и никакое ребро не встречается более одного раза. Вершина U – начало пути, а вершина V – конец пути. *Путь* от U к V называется *простым*, если он не проходит ни через одну из вершин более одного раза.

Две *вершины* U и V называются *связными*, если в графе существует путь с концами U и V. Две вершины графа называются несвязными, если в графе не существует пути, связывающего их. *Граф* называется *связным*, если каждые две вершины его связные, в противном случае граф называется несвязным.

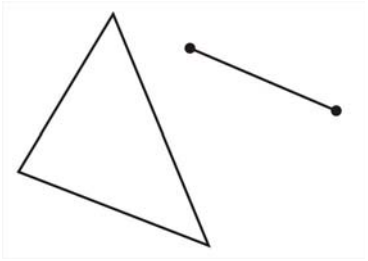


Рис. 26

На рис. 24 граф связный, а на рис. 26 – несвязный, состоит из трех связных компонентов.

Эйлеров путь

Эйлеров путь – это произвольный путь в графе, проходящий через каждое ребро графа в точности один раз, т. е. путь $v_1, v_2, \ldots, v_{m+1}$, такой, что каждое ребро $e \in E$ появляется в этом пути ровно один раз

как ребро $\{v_i, v_{i+1}\}$, где $1 \le i \le m$. *Эйлеров цикл* – это эйлеров путь, где начало пути совпадает с концом. В 1736 году Эйлер доказал первую теорему теории графов, теорему о необходимом и достаточном условии существования эйлерова пути.

Программы нахождения эйлерова цикла приведены на с. 125.

Теорема. Эйлеров путь в связном неориентированном графе существует тогда и только тогда, когда граф содержит не более чем две вершины нечетной степени.

Если граф содержит эйлеров путь, или эйлеров цикл, то диаграмма графа – уникурсальная линия.

***Задача* 9.** Определите, являются ли диаграммы графов на рис. 27, *а* и 27, *б* уникурсальными линиями.

а) *б)*

Рис. 27

3. Машинное представление графов

В настоящее время почти в каждой отрасли науки и техники встречаются задачи, для решения которых используется теория графов. Например, в электротехнике – при построении электрических схем, в биологии – при исследовании биологических систем, в экономике – при решении задач о выборе оптимального пути для потока грузового транспорта. С теорией графов связаны такие серьезные науки, как теория относительности, теория групп. Важным вопросом, особенно для приложений теории графов, является определение возможных способов представления графов. Граф можно представить в виде диаграммы, отражающей смежность вершин, и визуально определить некоторые свойства и характеристики заданного графа. Однако при наличии в графе большого числа рёбер и вершин этот способ

мало пригоден. У нас есть верный помощник, обладающий огромным быстродействием, это компьютер. Поэтому необходимо научиться представлять структуру графа в компьютере. Каким же образом предпочтительнее представлять информацию об исследуемом объекте в виде графа в компьютере? Современные технические средства допускают ввод информации в компьютер в различных формах. Выбор подходящей структуры данных для представления графа в компьютере имеет принципиальное значение при разработке эффективных алгоритмов. Способов представления графа в памяти компьютера множество, но все они основаны на базовых представлениях. Мы рассмотрим три способа представления графов в памяти компьютера в числовом виде.

Способы представления графов в памяти компьютера: матрица инцидентности, матрица смежности, перечень рёбер.

3.1. Матрица смежности

На рис. 28 представлена диаграмма графа. Диаграмма отражает смежность вершин в графе. Распространенным способом структурирования данных является также прямоугольная таблица. Преобразуем структуру данных и связей объекта, представленных в виде диаграммы, в табличную форму. Заменим диаграмму двоичной таблицей типа «объект-объект», отражающей качественную связь между объектами – смежность вершин графа (есть связь или нет связи) (рис. 29).

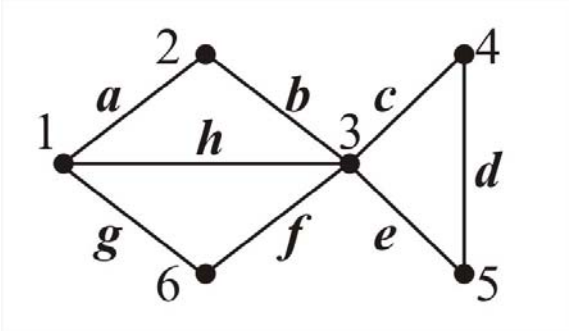


Рис. 28

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	1	0	0	1
2	1	0	1	0	0	0
3	1	1	0	1	1	1
4	0	0	1	0	1	0
5	0	0	1	1	0	0
6	1	0	1	0	0	0

Рис. 29

В полученной таблице вырежем ядро таблицы и назовём вырезанную таблицу матрицей смежности $A=[a_{ij}]$, где $i=1, 2, \ldots, 6$, $j=1, 2, \ldots, 6$, для графа на рис. 28.

В математике *прямоугольной матрицей* называется совокупность чисел расположенных в виде прямоугольной таблицы**,** содержащей n строк и m столбцов. Такая матрица записывается в виде, показанном на рис. 30, или сокращенно в виде $A=[a_{ij}]$, $i=1, 2, \ldots, n$, $j=1, 2, \ldots, m$. Если число строк матрицы равно числу столбцов, то такая матрица называется *квадратной.* Квадратная матрица называется *симметричной,* если $a_{ij}=a_{ji}$, $i=1, 2, \ldots, n$, $j=1, 2, \ldots, n$[5].

$$A=\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{11} & a_{11} & \ldots & a_{2m} \\ . & . & . & . \\ a_{n1} & a_{n2} & \ldots & a_{nm} \end{bmatrix}$$

Рис. 30

Граф, имеющий n вершин, ориентированный и неориентированный, можно представить массивом A, где *A: array* [1..n, 1..n] *of* 0..1, и

$$A(i,j)=\begin{cases} 1, \text{ если вершина } i \text{ и вершина } j \text{ смежные;} \\ 0, \text{ если вершина } i \text{ и вершина } j \text{ не являются смежными.} \end{cases}$$

Для неориентированного графа (см. рис. 28), матрица смежности представлена на рис. 32. Матрица смежности для неориентированного графа является симметричной.

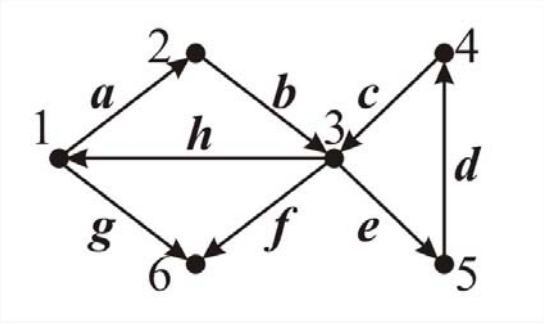


Рис. 31

Представим матрицу смежности $A=[a_{ij}]$, где $i=1, 2, \ldots, 6$, $j=1, 2, \ldots, 6$, для ориентированного графа (рис. 31). Элементы матрицы смежности выделены на рис. 33.

Основным преимуществом представления графа матрицей смежности является тот факт, что за один шаг можно получить ответ

[5] В языках программирования графы, представляются структурированными типами данных – массивами, двумерные массивы называются матрицами.

на вопрос, существует ли ребро из вершины U в V. Недостатком же является тот факт, что независимо от числа рёбер объём памяти для представления графа пропорционален n^2.

$$A=\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Рис. 32

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	0	0	0	1
2	0	0	1	0	0	0
3	1	0	0	0	1	1
4	0	0	1	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	0

Рис. 33

3.2. Матрица инцидентности

Рассмотрим представление графа с помощью матрицы I, отражающей инцидентность вершин и ребер (дуг). Опишем матрицы с использованием двумерного массива.

I: array [1..n, 1..m] of 0..1; (для неориентированного графа);

I: array [1..n, 1..m] of –1..1; (для ориентированного графа).

Неориентированный граф:

$$I(i,j)=\begin{cases} 1, \text{ если вершина } i \text{ и ребро } j \text{ инцидентны;} \\ 0, \text{ если вершина } i \text{ и ребро } j \text{ неинцидентны.} \end{cases}$$

Если в графе нет петель, то в каждом столбце матрицы инцидентности будет ровно 2 единицы. Наличие столбца с 1 единицей означает, что в графе имеется петля.

Ориентированный граф:

$$I(i,j)=\begin{cases} 1, \text{ если вершина } i \text{ и дуга } j \text{ инцидентны,} \\ \qquad\qquad \text{и вершина } i \text{ является концом дуги } j, \\ -1, \text{ если вершина } i \text{ и дуга } j \text{ инцидентны,} \\ \qquad\qquad \text{и вершина } i \text{ является началом дуги } j, \\ 0, \text{ если вершина } i \text{ и дуга } j \text{ неинцидентны.} \end{cases}$$

Для ориентированного графа петлю, то есть дугу вида $< v, v >$ удобно представить в строке v значением, например 2.

Примеры представления графов матрицей инцидентности

Матрица инцидентности для неориентированного графа (см. рис. 28)

	a	***b***	***c***	***d***	***e***	***f***	***g***	***h***
1	1	0	0	0	0	0	1	1
2	1	1	0	0	0	0	0	0
3	0	1	1	0	1	1	0	1
4	0	0	1	1	0	0	0	0
5	0	0	0	1	1	0	0	0
6	0	0	0	0	0	1	1	0

Матрица инцидентности для ориентированного графа (см. рис. 31)

	a	***b***	***c***	***d***	***e***	***f***	***g***	***h***
1	-1	0	0	0	0	0	-1	1
2	1	-1	0	0	0	0	0	0
3	0	1	1	0	-1	-1	0	-1
4	0	0	-1	1	0	0	0	0
5	0	0	0	-1	1	0	0	0
6	0	0	0	0	0	1	1	0

С алгоритмической точки зрения матрица инцидентности является плохим представлением графа. Оно требует размер памяти, пропорциональный nm, и имеет неудобный доступ к информации. Ответы на вопросы, существует ли дуга $<U,V>$, к каким вершинам ведут рёбра из U , требуют в худшем случае m шагов**.**

3.3. Список рёбер графа

Более экономным в отношении памяти, когда m гораздо меньше n^2 , является метод представления графа перечислением рёбер графа в виде пар соответствующих вершин. В языках программирования та-

кому представлению графа соответствует представление графа в виде массива $R = [r_{ij}]$ размера $2 \times m$.

R: array [1..2, 1..m] of 1..n;
R [1, i] – первая вершина (начало) ребра (дуги) *a*;
R [2, i] – вторая вершина (конец) ребра (дуги) *a*..

Примеры представления графов списком рёбер

Представление неориентированного графа списком рёбер (см. рис. 28)

	a	***h***	***g***	***b***	***c***	***e***	***f***	***d***
1	1	1	1	2	3	3	3	4
2	2	3	6	3	4	5	6	5

Представление ориентированного графа списком рёбер (см. рис. 31)

	a	***g***	***b***	***h***	***e***	***f***	***c***	***d***
1	1	1	2	3	3	3	4	5
2	2	6	3	1	5	6	3	4

Очевидно, что объём памяти для хранения массива пропорционален 2*m*. Неудобством при представлении графа в виде списка рёбер является большое число шагов – порядка *m* в худшем случае, необходимых для получения множества вершин, к которым ведут рёбра из рассматриваемой вершины. Для решения такой задачи нужно упорядочить множество пар лексикографически и применить двоичный поиск.

***Задача* 10.** Разработайте и реализуйте программы ввода описаний графа, заданного диаграммой, в память компьютера, используя все рассмотренные способы[6]. Для контроля корректности ввода предусмотрите в программе вывод каждого описания на экран.

[6] Лучшим способом представления графа считается список инцидентности. Для такого представления в языках программирования используются указатели и динамические структуры данных.

***Задача* 11.** Разработайте и реализуйте программу моделирования процессов перехода между каждыми двумя способами представления графа, рассмотренными выше, оцените вычислительную сложность каждой процедуры.

4. Поиск пути в графе

***Задача* 12.** *Поиск пути в лабиринте.* Лабиринт представлен как система комнат, некоторые из которых связаны между собой коридорами. Найдите маршрут выхода из лабиринта, представленного на рис. 34, путь от начальной комнаты (1) до конечной комнаты (11). Разработайте информационную и компьютерную модели рассматриваемого процесса отыскания маршрута выхода из лабиринта.

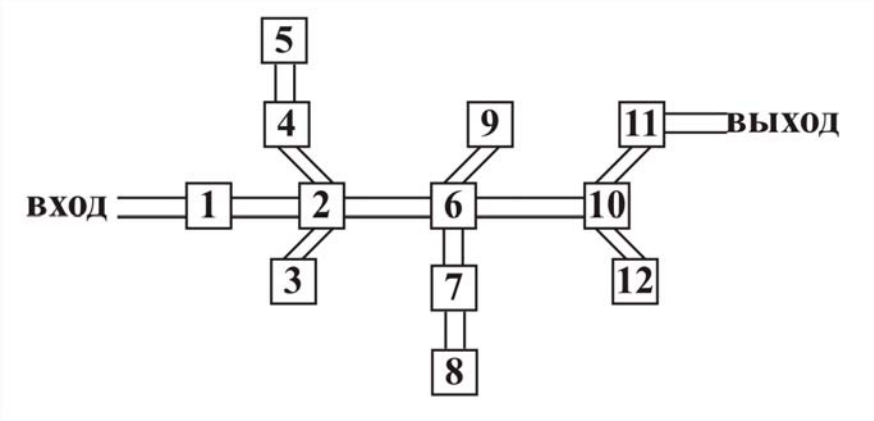


Рис. 34

Указания к решению. Определим средства, которые имеются в распоряжении путешественника в лабиринте. Будем предполагать, что он знает номер той комнаты, в которую вошел, номер комнаты, в которую ведет обозреваемый им коридор. Путешественник знает те комнаты, в которых уже побывал, а в случае необходимости может проделать часть пройденного пути в обратном порядке. При поиске интересующего пути необходимо руководствоваться следующими правилами:

а) при движении вперед из той комнаты, в которой находишься, всегда идти в комнату, в которой ещё не побывал (если таких комнат несколько, то выбрать комнату с наименьшим номером);

б) если из данной комнаты нельзя пройти в комнату, в которой еще не бывал, то нужно вернуться в предыдущую комнату.

***Задача* 13.** *Маршрут экскурсовода в зоопарке.* На рис. 35 представлена схема зоопарка. Вершины схемы – вольеры для зверей, рёбра – дорожки между вольерами. Разработайте компьютерную модель процесса отыскания маршрута, по которому экскурсовод мог бы провести посетителей, показав им всех зверей и не проходя более одного раза ни одного участка пути.

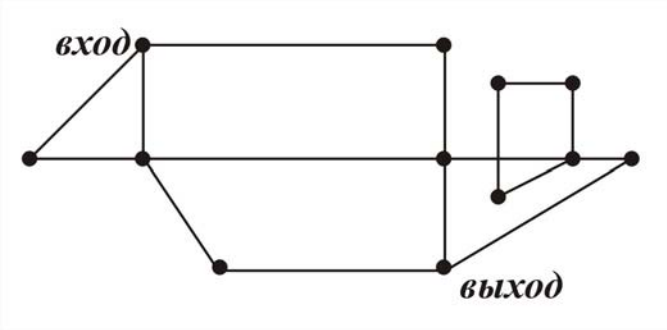


Рис. 35

Для того чтобы найти решение поставленных задач, надо уметь находить пути в графе от одной вершины к другой, то есть научиться обходить граф. Существует много алгоритмов на графах, в основе которых лежит систематический перебор вершин графа такой, что каждая вершина графа просматривается точно один раз, такие алгоритмы называются поиском пути в графе, или обходом графа. Рассмотрим два метода поиска пути в графе – метод поиска в глубину (англ. depth first search) и метод поиска в ширину (англ. breadth first search). Эти методы входят в основные методики проектирования алгоритмов на графах.

4.1. Метод поиска пути в глубину (для неориентированного графа)

Просмотр вершин графа начинается с некоторой вершины *V*. Выбирается вершина *U*, смежная с *V*, не просмотренная ранее; процесс повторяется с вершиной *U*. Если на очередном шаге мы работаем с вершиной *G*, и нет вершин, смежных с *G*, не просмотренных

ранее, то возвращаемся из G к вершине Q, из которой попали в G. Далее выбираем вершину, смежную с Q, не просмотренную ранее, и т.д. Если по возвращении мы попали в вершину V, и нет вершин, смежных с ней, и не просмотренных ранее, то просмотр закончен. Другими словами, поиск в глубину из вершины V основывается на поиске в глубину из всех вершин, смежных с V.

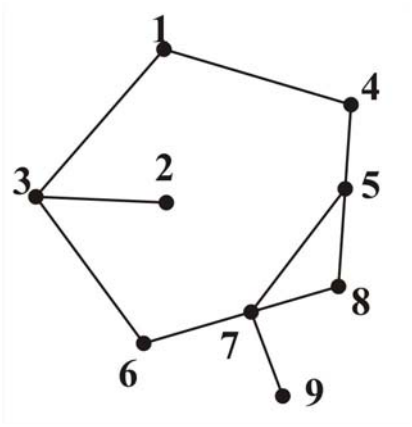


Рис. 36

Пример последовательности анализируемых вершин при просмотре графа в глубину (рис. 36), начиная от вершины 1: 1, 3, 2, 6, 7, 5, 4, 8, 9.

Программы обхода графа в глубину

Описание данных

A – матрица смежности, A:array[1..n,1..n] of 0..1;	Nnew:array[1..n] of boolean;
Apex– массив, где хранятся просмотренные вершины, Apex:array[1..n] of 1..n;	St – стек вершин, запомненных для просмотра (представленный в программе как массив)
Nnew – массив, хранящий сведения о каждой вершине (просмотрена она или нет)	Uk – указатель на позицию элемента массива (вершину стека) для записи и извлечения элемента.

Исходные данные – матрица смежности графа (см. рис. 36).

Результат – последовательность вершин при обходе графа в глубину.

```
{нерекурсивная программа обхода графа в глубину}
program graf_depth;
  uses crt; const n=9;
  type mas=array[1..n] of 1..n;
 mas1=array[1..n,1..n] of 0..1;
 const  a:mas1=((0,0,1,1,0,0,0,0,0),
(0,0,1,0,0,0,0,0,0), (1,1,0,0,0,1,0,0,0),
(1,0,0,0,1,0,0,0,0),(0,0,0,1,0,0,1,1,0),
    if p then
      begin
      uk:=uk+1;  st[uk]:=j;
      nnew[j]:=false; m:=m+1;
      apex[m]:=j;
      end
    else
      uk:=uk-1;
    end;
```

```
(0,0,1,0,0,0,1,0,0),(0,0,0,0,1,1,0,1,1),
(0,0,0,0,1,0,1,0,0),(0,0,0,0,0,0,1,0,0));
  var apex:mas;st:mas; v:byte; i,m:integer;
      nnew:array[1..n] of boolean;
  procedure ver(v:byte);
   var uk,t,j:byte; p:boolean;
  begin
  uk:=1; apex[m]:=v;
  nnew[v]:=false; st[uk]:=v;
  while uk<>0 do
   begin
   t:=st[uk]; p:=false; j:=1;
   repeat
     if (a[t,j]<>0) and (nnew[j]) then
              p:=true  else inc(j);
   until (p) or (j>n);
```

```
    end;
begin
clrscr;
for i:=1 to n do nnew[i]:=true;
v:=1;m:=1; ver(v);
for i:=2 to n do
        if nnew[i] then ver(i);
for i:=1 to n do
        write(apex[i]:4);
end.
```

```
{рекурсивная программа поиска в глубину}
program graf_depth1;
uses crt; const n=9;
type mass=array[1..n,1..n]of integer;
mass1=array[1..n] of boolean;
const a:mass=((0,0,1,1,0,0,0,0,0),
(0,0,1,0,0,0,0,0,0), (1,1,0,0,0,1,0,0,0),
(1,0,0,0,1,0,0,0,0),(0,0,0,1,0,0,1,1,0),
(0,0,1,0,0,0,1,0,0),(0,0,0,0,1,1,0,1,1),
(0,0,0,0,1,0,1,0,0),(0,0,0,0,0,0,1,0,0));
var i,v,m:integer; sw:mass1;
    apex:array[1..n] of integer;
  procedure ser(v:integer);
  var  i,j:integer;
```

```
  begin
   m:=m+1; sw[v]:=true;
   write(v); apex[m]:=v;
   for j:=1 to n do
   if (a[v,j]<>0) and (not sw[j]) then
      ser(j);
  end;
begin
  for i:=1 to n do sw[i]:=false;
  clrscr; write('введите первую вершину
для просмотра');  readln(v);
  ser(v); m:=0;  for i:=1 to n do
  if not sw[i] then ser(i);
end.
```

4.2. Поиск пути в ширину в неориентированном графе

Просмотр вершин графа начинается с некоторой вершины V – первой текущей вершины. Далее рассматриваются все вершины, смежные с выбранной текущей и не просмотренные ранее. Выбирает-

ся следующая текущая вершина, принцип выбора которой такой: выбирается та вершина, которая была ранее всех просмотрена, но не выбиралась текущей. Процесс просмотра вершин заканчивается, когда все просмотренные вершины будут использованы как текущие.

Пример последовательности анализируемых вершин при просмотре графа (см. рис. 36) в ширину, начиная от вершины 1: 1, 3, 4, 2, 6, 5, 7, 8, 9.

Программа обхода графа в ширину

Описание данных

A – матрица смежности, A:array[1..n,1..n] of 0..1; Apex– массив, где хранятся просмотренные вершины, Apex:array[1..n] of 1..n;	Uk1– указатель на позицию постановки элемента в очередь. Пример. Uk1:=5; Turn[Uk1]:='A'; В очередь Turn на пятую позицию поставлена вершина A.
Nnew – массив, хранящий сведения о каждой вершине (просмотрена она или нет), Nnew:array[1..n] of boolean; Turn – очередь вершин, поставленных на просмотр (представленная в программе как массив).	Uk2 – указатель на позицию извлечения элемента массива из очереди. Пример. Uk2:=3; z:=Turn[Uk2]; Из очереди Turn с третьей позиции извлекается вершина для анализа

Исходные данные – матрица смежности графа (см. рис. 36). Результат – последовательность вершин при обходе графа в ширину.

```
program graf;
 const n=9; type mas=array[1..n] of 1..n;
 mas1=array[1..n,1..n] of 0..1;
const  a:mas1=((0,0,1,1,0,0,0,0,0),
(0,0,1,0,0,0,0,0,0), (1,1,0,0,0,1,0,0,0),
(1,0,0,0,1,0,0,0,0),(0,0,0,1,0,0,1,1,0),
(0,0,1,0,0,0,1,0,0),(0,0,0,0,1,1,0,1,1),
 (0,0,0,0,1,0,1,0,0),(0,0,0,0,0,0,1,0,0));
var  apex, st:mas; v, i, m, k, j: byte;
nnew:array[1..n] of boolean;
procedure breadth(v:byte;m1:1..n);
  var uk1, i,t:1..n;turn:array[1..n] of 1..n;
    for i:=1 to n do
    if (a[t,i]<>0) and (nnew[i]) then
      begin
      uk1:=uk1+1; turn[uk1]:=i;
      nnew[i]:=false; m1:=m1+1;
      apex[m1]:=i;{pred[i]:=t;}
      end;
    end;
end;
begin
write('Введите  начальную точку ');
readln(v);
```

```
      uk2:0..n;
begin
  uk1:=1;uk2:=0;turn[uk1]:=v;
  apex[m1]:=v; nnew[v]:=false;
  while uk2<>uk1 do
    begin
    uk2:=uk2+1; t:=turn[uk2];
```

```
for i:=1 to n do  nnew[i]:=true; m:=1;
breadth(v,m);
for i:=2 to n do
     if nnew[i] then breadth(i,m);
for i:=1 to n do  write(apex[i]:4);
end.
```

Замечания

1. Покажем, что алгоритмы просматривают каждую вершину в точности 1 раз. Каждая вершина просматривается не более одного раза, так как просматривается только вершина *V*, для которой Nnew[V]=true. Алгоритм начинает поиск поочередно от каждой еще не просмотренной вершины. Следовательно, просматриваются все вершины графа, даже не обязательно связного.

2. Конечность алгоритмов поиска в глубину и в ширину. Всего в *St* может попасть не более *n* вершин при поиске в глубину. На каждом шаге процедуры одна вершина *t* удаляется из стека, если для нее нет смежной. Следовательно, алгоритм завершит работу, когда стек станет пустым. Аналогичные рассуждения для поиска в ширину. Сложность алгоритмов поиска пути графа в глубину и в ширину равна $O(n+m)$.

3. Оба вида поиска в графе могут быть использованы для нахождения пути между фиксированными вершинами *U* и *V*. Достаточно начать поиск пути в графе с вершины *V* и вести его до момента посещения вершины *U*. Преимуществом поиска в глубину является тот факт, что в момент посещения вершины *U* стек содержит последовательность вершин, определяющих путь от *V* к *U*. Это очевидно, так как каждая вершина, помещаемая в стек, является смежной с вершиной, находящейся в вершине стека.

Используя метод поиска в ширину, можно получить кратчайший путь, так как в очередь помещены сначала вершины, находящиеся на расстоянии 0 от *V*, то есть сама вершина *V*. Затем вершина, находящаяся на расстоянии 1 от вершины *V*, и т. д. Зная предыдущую вер-

шину *U*, а также предыдущие вершины для каждой вершины графа *G*, мы можем найти путь от *V* к *U*. Для этого обход в ширину необходимо начать с вершины *V*, а в программу достаточно вставить оператор pred[i]:=*t*; (в программе этот оператор выделен в фигурные скобки), и в конце программы добавить нижерасположенный фрагмент, описав недостающие переменные: *k* – типа byte, массивы pred[1:n] и z[1:n] – типа mass, *u* – типа byte, где *u* – конечная вершина искомого кратчайшего пути.

```
writeln; write('введите конечную вер-
шину пути '); readln(u);
i:=u; k:=1;z[k]:=u;
while i<>v do
    begin
      i:=pred[i]; k:=k+1; z[k]:=i;
    end;
for i:=k downto 1 do     write(z[i]:4);
```

***Задача* 14.** Разработайте и реализуйте программы, моделирующие процессы поиска пути в ориентированном графе, поиска в глубину и поиска в ширину. Граф задается матрицей смежности или построением диаграммы в графическом окне выбранной системы программирования.

***Задача* 15.** Составьте компьютерную модель процесса поиска пути в неориентированном графе, поиск в ширину, если достигнутая вершина помещается не в очередь, а в стек. Оцените вычислительную сложность программы.

Матрица достижимости

Если существует путь от вершины *V* к *U*, то говорят, что *U* достижима из *V*. Определим матрицу достижимости – это матрица *D* размера $n \times n$.

$$D(U,V) = \begin{cases} 1, \text{ если вершина } U \text{ достижима из } V; \\ 0, \text{ в противном случае.} \end{cases}$$

Программа формирования матрицы достижимости

Пусть граф *G* задан матрицей смежности, матрица *D* – матрица достижимости.

```
Uses crt;
Const
  n=9;
Type
  mass=array[1..n,1..n] of 0..1;
Const a:mass=((0,0,1,1,0,0,0,0,0),
(0,0,1,0,0,0,0,0,0), (1,1,0,0,0,1,0,0,0),
(1,0,0,0,1,0,0,0,0),(0,0,0,1,0,0,1,1,0),
(0,0,1,0,0,0,1,0,0),(0,0,0,0,1,1,0,1,1),
(0,0,0,0,1,0,1,0,0),(0,0,0,0,0,0,1,0,0));
Var    D:mass; i,j:integer;
Procedure reach;
Var   i,j,k:byte;   S,T:set of 1..n;
Begin
  for i:=1 to n do
    begin
      T:=[i];
      repeat
        S:=T;
                for k:=1 to n do
                 if k in S then
                   for j:=1 to n do
                     if A[k,j]<>0 then T:=T+[j];
              until S=T;
              for j:=1 to n do
                if j in S then D[i,j]:=1;
              end;
            end;
Begin
Clrscr;
reach;
for i:=1 to n do
   begin
   for j:=1 to n do
     write(D[I,j]:4);
   writeln;
   end;
End.
```

Вопрос. Каким свойством обладает матрица достижимости связного неориентированного графа?

Замечание. Если вершины графа перенумеровать соответствующим образом, то связные компоненты графа G соответствуют состоящим из единиц квадратным подматрицам матрицы достижимости.

5. Вопросы и задания к семинарским занятиям

Тема «Графы и компьютерное моделирование»

Перефразируйте задания 1 – 6 главы 1, сформулировав предлагаемые задания и вопросы для рассматриваемой темы «Графы и компьютерное моделирование». Выполните задания 1 – 6, ответьте на вопросы, поставленные в этих заданиях. Доложите результаты своей работы в студенческой аудитории в форме презентации.

7. Подберите мотивационные задачи для введения понятия «граф» и демонстрации удобства языка теории графов для описания

информационных и компьютерных моделей практических задач. Разработайте методику использования выбранных мотивационных задач на уроках по данной теме. Апробируйте свои разработки в студенческой аудитории.

8. Отберите содержание учебного материала, разработайте методику изложения выбранного материала по предложенным ниже темам на выбранной ступени непрерывного курса изучения информатики. Разработайте соответствующие фрагменты уроков. Апробируйте в студенческой аудитории один из фрагментов урока.

- Введение основных понятий теории графов (граф, вершина графа, ребро графа, дуга графа, ориентированный и неориентированный граф, смежные вершины и т. д.).
- Машинное представление графов (матрица инцидентности, матрица смежности, перечень ребер, списки связей или списки инцидентностей).
- Связность графа, путь в графе, цикл, элементарный цикл, поиск пути в графе в глубину, поиск пути в графе в ширину, сравнение обхода вершин графа разными способами.
- Эйлеров путь, эйлеров цикл, алгоритм нахождения эйлерова цикла.
- Алгоритм определения уникурсальной линии.

6. Лабораторно-практическая работа

Тема «Графы и компьютерное моделирование»

***Задание* 1**. Отберите содержание темы для решения предложенных в данной главе задач 4, 8 – 14, разработайте методики обучения школьников поиску решения предложенных задач, оформите решения задач. Мотивируйте свой выбор методик поиска решения конкретных задач. Апробируйте разработанные методики поиска решения задач в студенческой аудитории.

Для выполнения заданий 2 и 3 разработайте и реализуйте с использованием выбранного исполнителя учебный проект в виде ком-

пьютерной модели, рассмотрев все этапы составления компьютерных моделей. Подготовьте проекты к защите.

***Задание* 2.** В некоторой стране 110 городов, между некоторыми из них летают самолеты. Авиатрассы расположены так, что из любого города можно перелететь в другой (возможно с пересадками), известна цена перелёта из города *V* в город *U*. Разработайте компьютерную модель процесса нахождения:

- всех маршрутов из города *V* в город *U*, делая не более одной пересадки, и цены каждого из этих маршрутов;
- всех маршрутов из города *V* в город *U* с одной пересадкой в порядке возрастания цены маршрута;
- всех маршрутов из города *V* в город *U* ровно с двумя пересадками;
- всех маршрутов из города *V* в город *U* с двумя пересадками, в порядке убывания цены маршрута;
- всех маршрутов из города *V* в город *U*, делая не более трёх пересадок.

***Задание* 3.** Разработайте учебный проект в виде компьютерной модели, позволяющей получить остовные связные деревья минимального веса для графов с пятью вершинами. Граф в виде схемы задайте в графическом поле, а матрицу смежности, соответствующую заданной схеме, выведите в элемент управления, представляющий собой таблицу [16].

7. Самостоятельная работа

Тема «Графы и компьютерное моделирование»

Для выполнения заданий 1, 2 разработайте и реализуйте с использованием выбранного исполнителя учебные проекты в виде компьютерных моделей, рассмотрев все этапы составления компьютерных моделей. Проекты подготовьте к защите.

***Задание* 1.** Проект «Владимирское метро». Спроектируйте информационную и компьютерную модели линий и станций

Владимирского метро, учитывающую время в пути между соседними станциями и примерное время, которое тратится на пересадки.

***Задание* 2.** Разработайте и реализуйте на выбранном языке программирования компьютерную модель справочного электронного табло для пассажиров одной фиксированной станции проекта «Владимирское метро». Продумайте оформление табло, возможность получения ответов на актуальные вопросы пассажиров.

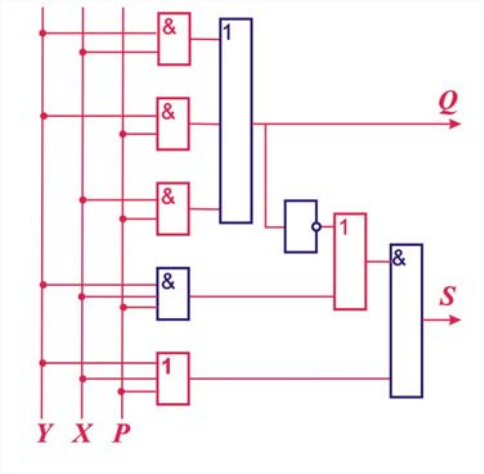
Выполните задания 3, 4, 5. Проделанную и оформленную работу подготовьте для защиты.

***Задание* 3.** Составьте поурочный план изучения данной темы по предмету «Информатика и ИКТ» на одной из ступеней непрерывного курса изучения информатики.

***Задание* 4.** Разработайте тематику, содержание, методы и приёмы проведения лабораторных работ по теме «Графы и компьютерное моделирование» на каждой ступени непрерывного курса изучения информатики.

***Задание* 5.** Разработайте сценарии программ (демонстрационных, обучающих, тренажеров, контролирующих) по данной теме и реализуйте их на одном из языков программирования.

ГЛАВА 3. ЛОГИЧЕСКИЕ ИНФОРМАЦИОННЫЕ И КОМПЬЮТЕРНЫЕ МОДЕЛИ



«Ибо это недостойно совершенства человеческого, подобно рабам тратить часы на вычисления».

Лейбниц

Самой мощной движущей силой развития информатики является стремление освободить человека от бремени однообразной утомительной умственной работы, переложив её на электронно-вычислительную технику. Сфера применения электронно-вычислительной техники обширна и практически охватывает все области человеческой деятельности. Только познав «внутреннюю жизнь» электронно-вычислительной техники (компьютера), подчинённую строгим законам логики и таких наук, как физика, химия, математика, можно полноценно использовать её (его) для решения практических задач. Глубокая идейная связь, столь характерная для современного развития логики и информатики, находит своё выражение во многих новых направлениях, имеющих практическое значение.

Рассмотрим некоторые аспекты изучения темы «Логические информационные и компьютерные модели» в школьном предмете «Информатика и ИКТ». Учебный материал этой темы даёт возможность для реализации основной задачи курса информатики, направленной на развитие у обучаемых информационной культуры. Это формирование умений, необходимых для осуществления всех

этапов процесса исследования, а также приобретение субъективного опыта обучаемого в познавательной деятельности.

Выполнение заданий по теме формирует умения и навыки в овладении основным методом познания окружающей действительности – методом моделирования и, в частности, методом компьютерного моделирования. Способствует развитию навыков логического обоснования выводов из практических задач, использования аналитико-синтетического метода при нахождении решения поставленной задачи, позволяет вести обучение на достаточном уровне сложности, что обусловливает необходимость повышения роли теоретических знаний в содержании образования, умелого применения полученных знаний в новой обстановке; позволяет решать задачи с продолжением. Многообразие способов решения предложенных задач позволяет учащемуся выбрать интересный и посильный для него метод. Ценность теории определяется тем, насколько она применима на практике. Содержание темы, с одной стороны, вбирает основные понятия, опорные знания по математической логике, с другой – способствует сознательному использованию компьютерного моделирования при решении практических задач.

1. От практических задач к компьютерным моделям

При изучении темы «Логические информационные и компьютерные модели» можно предложить рассмотреть такие мотивационные задачи.

***Задача* 1.** Пусть в некотором университете проходит в несколько туров конкурс танца. Четырьмя членами жюри (A, B, C, D) решается вопрос о допуске того или иного участника к следующему туру. Решение положительно тогда и только тогда, когда хотя бы трое членов жюри высказываются за допуск, причем среди них обязательно должен быть председатель жюри A.

Задание. Разработайте электронное устройство для голосования и реализуйте это устройство с использованием выбранного вами исполнителя, включая программы «Конструктор логических схем», для составления компьютерной модели.

Для создания компьютерной модели необходимо разработать информационные модели: *составить таблицу истинности*, *структурную формулу,* соответствующую разрабатываемому устройству, *функциональную схему*.

Решение.

а) Составим *таблицу истинности*, соответствующую работе жюри**:**

A	B	C	D	F (A, B, C, D)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1*
1	1	0	0	0
1	1	0	1	1*
1	1	1	0	1*
1	1	1	1	1*

б) Составим *структурную формулу*, соответствующую разрабатываемому устройству.

Для решения задачи воспользуемся алгоритмом получения СДНФ по таблице истинности

$$F(A,B,C,D) = (A \wedge \overline{B} \wedge C \wedge D) \vee (A \wedge B \wedge \overline{C} \wedge D) \vee$$
$$\vee (A \wedge B \wedge C \wedge \overline{D}) \vee (A \wedge B \wedge C \wedge D).$$

в) Получили логическую функцию *F(A,B,C,D)*, функцию от четырёх переменных, минимизируем её. *Минимизация* булевой функции – это преобразование функции к такому виду, когда она содержит наименьшее возможное число логических переменных и наименьшее возможное число операций над ними. Применим два способа минимизации исследуемой функции.

1-й способ. Для минимизации функции воспользуемся основными законами логики. По закону идемпотентности дизъюнкции можно записать

$$F(A,B,C,D) = \left(A \wedge \overline{B} \wedge C \wedge D\right) \vee \left(A \wedge B \wedge \overline{C} \wedge D\right) \vee \left(A \wedge B \wedge C \wedge \overline{D}\right) \vee$$
$$\vee \left(A \wedge B \wedge C \wedge D\right) \vee \left(A \wedge B \wedge C \wedge D\right) \vee \left(A \wedge B \wedge C \wedge D\right).$$

Попарно сгруппируем выражения в скобках и применим закон дистрибутивности конъюнкции относительно дизъюнкции

$$F(A,B,C,D) = A \wedge C \wedge D \wedge \left(\overline{B} \vee B\right) \vee A \wedge B \wedge D \wedge \left(\overline{C} \vee C\right) \vee$$
$$\vee A \wedge B \wedge C \wedge \left(\overline{D} \vee D\right).$$

По закону исключённого третьего имеем

$$F(A,B,C,D) = A \wedge C \wedge D \vee A \wedge B \wedge D \vee A \wedge B \wedge C.$$

Далее, применив закон дистрибутивности конъюнкции относительно дизъюнкции, получим *структурную формулу*[7] для функции *F – информационную модель*:

$$F(A,B,C,D) = A \wedge (C \wedge D \vee B \wedge D \vee B \wedge C).$$

2-й способ. Воспользуемся картами Карно (диаграммами Вейча) для четырёх переменных[8] (рис. 37).

---

[7] Выражение, задающее логическую функцию, реализуемую логическим устройством, называется структурной формулой.

[8] Карты Карно – графический способ минимизации булевых функций. Карты Карно были изобретены в 1952 году Эдвардом Вейчем и усовершенствованы в 1953 году физиком Морисом Карно.

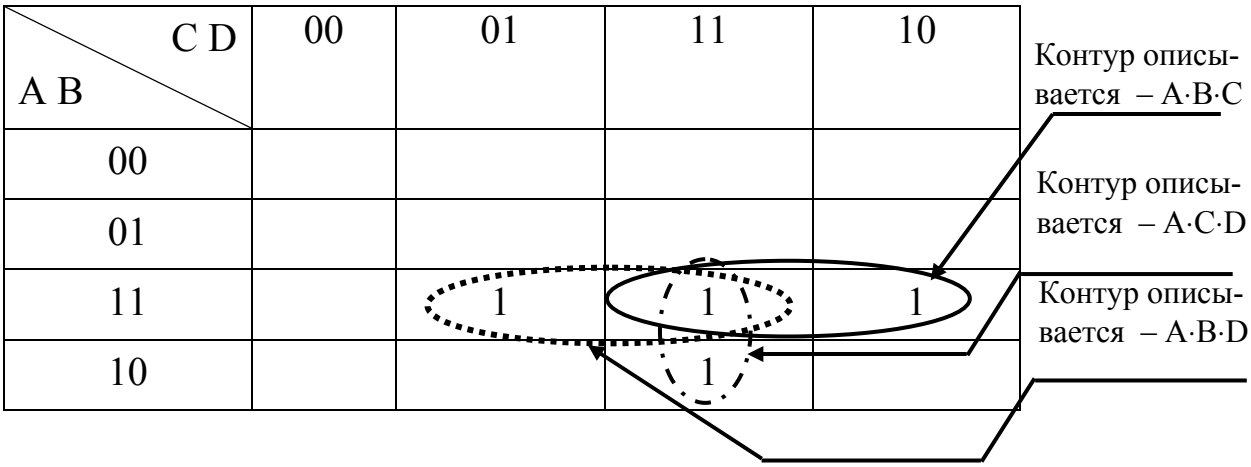


Рис. 37

С использованием карт Карно получим структурную формулу для функции F такого вида:

$$F(A,B,C,D) = A \wedge (C \wedge D \vee B \wedge D \vee B \wedge C).$$

г)

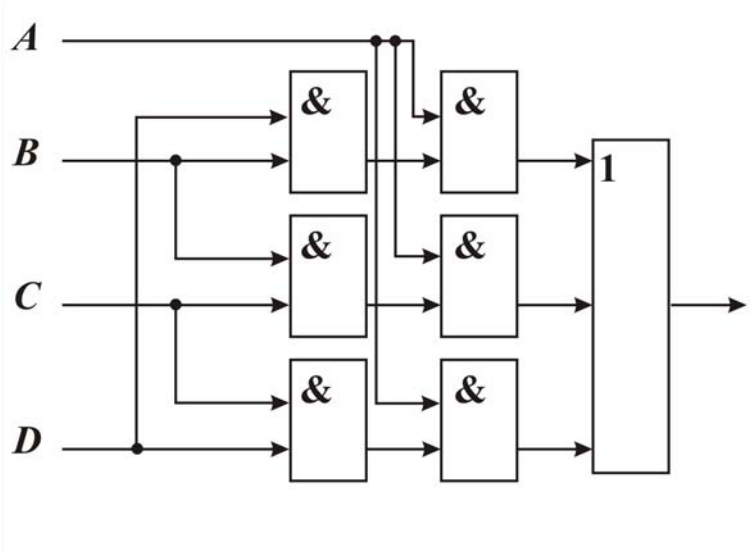


Рис. 38

По полученному выражению, задающему логическую функцию F, составим функциональную схему[9] необходимого устройства, информационную модель процесса голосования (рис. 38).

[9] Схема соединения логических элементов, реализующих логическую функцию, называется функциональной схемой.

д) Составим по функциональной схеме компьютерную модель процесса голосования, например с использованием программы «Конструктор логических схем» (для проведения компьютерного эксперимента).

Схема перехода от практической задачи к компьютерной модели

Реальный объект⟶ Системный анализ⟶ Табличная информационная модель ⟶ Структурная формула (информационная модель) ⟶ ⟶ Функциональная схема (информационная модель)⟶ Компьютерная модель (программа или электронная схема, сконструированная на специальном программном обеспечении, реализующая процесс голосования с помощью выбранного исполнителя).

Задача о расписании

Может ли компьютер оказать помощь человеку в составлении расписания, например, расписания движения поездов, автобусов, расписания всевозможных дежурств, уроков? Это очень трудоёмкий процесс, при составлении расписания необходимо учитывать многие факторы. Проведём эксперимент, попробуем составить расписание уроков с использованием компьютера на один день в восьмом классе общеобразовательной школы.

***Задача* 2.** Составьте расписание уроков на один день занятий, учитывая следующие предварительные пожелания учителей.

- В планируемый день занятий у учащихся должно быть 4 урока.
- Учитель информатики может провести либо 1-й, либо 2-й, либо 3-й уроки.
- Учитель литературы может провести либо 2-й, либо 3-й уроки.
- Учитель математики может провести либо 1-й, либо 2-й уроки.
- Учитель физкультуры согласен проводить только последний урок.

Решение. Решим задачу алгебраическим методом. Алгебраический метод решения логических задач является универсальным. Можно рекомендовать такой алгоритм решения логических задач алгебраическим методом.

- Проанализируй условие задачи.
- Введи систему обозначений для логических высказываний.
- Сконструируй логическую функцию, описывающую логические связи между всеми заданными высказываниями задачи – составь *информационную модель.*
- Упрости выражение, задающее сконструированную функцию.
- Определи значения истинности этой логической функции, для этого составь компьютерную модель, рассчитанную на выбранного исполнителя.
- Из полученных значений истинности функции определи значения истинности введенных логических переменных. По значению переменных сделай заключение о решении задачи.

A. Реализуем первый и второй этапы решения логических задач. Введём следующие обозначения. I1 (I2, I3) – учитель информатики проводит свой урок первым (вторым, третьим) уроком. L2 (L3) – учитель литературы проводит свой урок вторым (третьим) уроком. M1 (M2) – учитель математики проводит свой урок первым (вторым) уроком. F4 – учитель физкультуры проводит свой урок четвёртым уроком. Выбрали восемь логических переменных, с их помощью формализуем решение задачи.

B. Составим *информационную модель* в виде логической функции, описывающей взаимосвязь между исходными простыми высказываниями. Значение формируемого выражения присвоим переменной DAY. Найдём совокупность значений (I1, I2, …, F4), которые обратят уравнение Day = true в верное равенство. В день у учеников должно быть 4 разных урока. Условие того, что урок информатики проведён, запишется в виде некоторого логического выражения, его значение присвоим переменной INF. Для других предметов будем соответственно использовать переменные LT, MT, FZ. О правильности расписания будут говорить истинные значения этих переменных. Кроме того, в правильно составленном расписании не должно быть

наложений. Например, не могут быть одновременно первым уроком информатика и математика. Обозначим высказывание об отсутствии наложений в расписании на первом уроке – Less1, соответственно высказывание об отсутствии наложений в расписании на втором уроке – Less2, на третьем – Less3. Для четвёртого урока такое условие можно не записывать, так как на это время никто не претендует, кроме учителя физкультуры.

DAY=INF и LT и MT и Less1 и Less2 и Less3.

Выведем зависимость каждого отдельного условия от исходных данных. Начнём с урока информатики. Записать пожелания учителя информатики можно логическим выражением INF= I1 or I2 or I3. Высказывание INF должно принимать истинное значение для того, чтобы учитель информатики был доволен расписанием. Аналогично рассуждая, запишем высказывания для других предметов. Пожелания учителя литературы запишем высказыванием LT = L2 or L3. Пожелания учителя математики запишем высказыванием MT=M1 or M2.

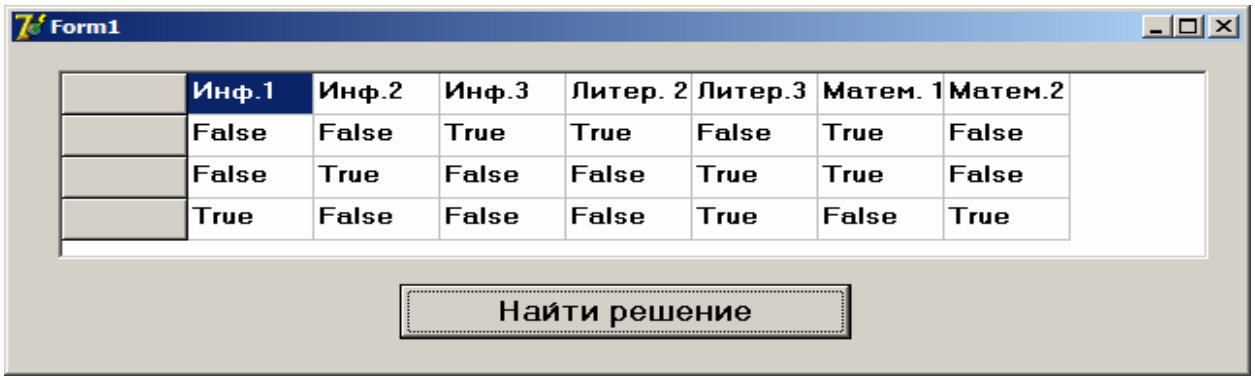
Определим высказывания Less1, Less2, Less3, Less4. Less1 = (I1 and not M1) or (not I1 and M1), Less2 = (I2 and not M2 and not L2) or (not I2 and M2 and not L2) or (not I2 and not M2 and L2), Less3 = (I3 and not L3) or (not I3 and L3). Полученные значения логических переменных поставим в функцию Day, логическую функцию от семи переменных DAY=INF and LT and MT and Less1 and Less2 and Less3. В силу того что на четвертый урок нет претендентов, кроме учителя физкультуры, то четвертый урок можно исключить из рассмотрения.

С. Определим, для какой совокупности значений семи логических переменных I1, I2, I3, L2, L3, M1,M2 функция Day принимает значение true. То есть составим таблицу истинности для полученной функции и выберем из неё строки, где значение Day истинно. Для решения поставленной задачи воспользуемся компьютером. Составим компьютерную модель процесса составления таблицы истинности

функции Day и выбора совокупности значений логических переменных, при которых значение функции истинно.

Компьютерная модель решаемой задачи

```
procedure TForm1.Button1Click(Sender: TObject);
var I1, I2, I3, L2, L3, M1, M2, Day, Less1, Less2, Less3, INF, LT, MT: Boolean;
i:byte;
begin
with stringgrid1 do
begin
cells[1,0]:='Инф.1';cells[2,0]:='Инф.2';cells[3,0]:='Инф.3';
cells[4,0]:='Литер.2';cells[5,0]:='Литер.3';  cells[6,0]:='Матем.1';
cells[7,0]:='Матем.2';
end;
i:=1;
for I1:=False to True do for I2:=False to True do  for I3:=False to True do
for L2:=False to True do for L3:=False to True do for M1:=False to True do
for M2:=False to True do
  begin
  INF:= I1 or I2 or I3;   LT:= L2 or L3;   MT:=M1 or M2;
  Less1:=(I1 and not M1) or (not I1 and M1);
  Less2:=(I2 and not M2 and not L2) or (not I2 and M2 and not L2);
  Less2:=Less2 or (not I2 and not M2 and L2);
  Less3:=(I3 and not L3) or (not I3 and L3);
  DAY:=INF and LT and MT  and Less1 and Less2 and Less3;
  if DAY then
  begin
//добавить новую строку в таблице для вывода результата
  StringGrid1.RowCount:=StringGrid1.RowCount+1;
  with Stringgrid1 do
    begin
    cells[1,i]:=BooltoStr(I1,true);cells[2,i]:=BooltoStr(I2,true);
    cells[3,i]:=BooltoStr(I3,true);cells[4,i]:=BooltoStr(L2,true);
    cells[5,i]:=BooltoStr(L3,true);cells[6,i]:=BooltoStr(M1,true);
    cells[7,i]:=BooltoStr(M2,true);
    end;   i:=i+1; end;  end; end;
```



D. По значению переменных сделаем заключение о решении задачи. Получили три решения.

1-й урок – математика 2-й урок – литература 3-й урок – информатика 4-й урок – физкультура	1-й урок – математика 2-й урок – информатика 3-й урок – литература 4-й урок – физкультура	1-й урок – информатика 2-й урок – математика 3-й урок – литература 4-й урок – физкультура

Схема перехода от практической задачи к компьютерной модели

Реальный объект ⟶ Системный анализ ⟶ Логическая функция (информационная модель) ⟶ Компьютерная модель (программа для выбранного исполнителя).

Поэтапное решение этих задач приводит к необходимости либо уточнять некоторые понятия математической логики, либо вводить эти понятия.

2. Элементы математической логики

Представление логической структуры учебного материала по данной теме

Всякий учебный материал имеет свою логическую структуру. Осознание этой структуры – одно из эффективных средств внесения элементов творчества в учебную деятельность учащихся. Творить – значит создавать новое. Это новое может быть не только неизвестным ученику фактом, а новым приемом мыслительной деятельности, открытием связей между явлениями, нахождение логической структуры учебного материала. В представленной логико-структурной схеме

(рис. 39) учебного материала данной темы указаны компоненты и их взаимосвязь, выстроена иерархия понятий, установлена последовательность их введения.

Рис. 39

2.1. Логические операции

Высказывание – повествовательное предложение, относительно которого можно сказать, истинно оно или ложно. Логические константы – это константы, которые принимают два значения, называемые "истиной" и "ложью". Принято истинное значение обозначать

TRUE или 1, а ложное – FALSE или 0. Высказывания обозначаются заглавными латинскими буквами (A, B, …). Логическая операция – способ построения сложного высказывания из данных высказываний, при котором значение истинности сложного высказывания полностью определяется значением истинности исходных высказываний (табл. 3).

Инверсия – образуется из высказывания с помощью добавления частицы «не» к сказуемому или использования оборота речи «неверно, что…». Обозначается: $\neg$, $\overline{\ }$, not. Инверсия высказывания истинна, когда высказывание ложно, и ложна, когда высказывание истинно.

Инверсия	
A	$\bar{A}$
0	1
1	0

Конъюнкция – логическое произведение – образуется соединением двух высказываний в одно с помощью знаков операции &, $\wedge$, •, и, and. Конъюнкция двух высказываний истинна тогда и только тогда, когда оба высказывания истинны, и ложна, когда хотя бы одно высказывание ложно.

Дизъюнкция – логическое сложение – образуется соединением двух высказываний в одно с помощью знаков операции $\vee$, +, или, or. Дизъюнкция двух высказываний ложна тогда и только тогда, когда оба высказывания ложны, и истинна, когда хотя бы одно высказывание истинно.

Исключающее ИЛИ – образуется соединением двух высказываний в одно с помощью знаков операции $\forall$, $\oplus$, **xor**. Исключающее ИЛИ двух высказываний истинно тогда и только тогда, когда одно высказывание ложно и одно высказывание истинно, и ложно в противном случае.

Импликация – образуется соединением двух высказываний в одно с помощью оборота речи «если…, то…». Обозначается: $\rightarrow$, $\Rightarrow$.

Импликация двух высказываний ложна тогда и только тогда, когда из истинного высказывания следует ложное высказывание.

Таблица 3

Таблица истинности логических операций

A	B	$A\&B$	$A \vee B$	$A \forall B$	$A \to B$	$A \Leftrightarrow B$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Эквивалентность – образуется соединением двух высказываний в одно с помощью оборота речи «… тогда и только тогда, когда …». Обозначается: $\leftrightarrow$, $\Leftrightarrow$, =. Эквивалентность двух высказываний истинна тогда и только тогда, когда оба высказывания истинны или оба высказывания ложны.

Используя основные логические операции, можно построить более сложные высказывания – логические выражения, например,

$$\left(A \wedge B\right) \vee \left(A \wedge B\right) \vee \left(A \wedge \overline{C}\right),\ \left(\overline{A} \vee A\right) \wedge \left(B \to A \wedge C\right).$$

Логические выражения строятся из простых высказываний A, B, C, ..., знаков логических операций и скобок. При отсутствии скобок первой всегда выполняется операция отрицания, затем конъюнкция, дизъюнкция и далее импликация и эквиваленция в порядке следования.

Логическая функция – это функция, определенная на множестве истинностных значений и принимающая значение из того же множества. Например,

$$f(A,B,C) = \overline{\left(A \vee B\right)} \to C\,.$$

2.2. Построение таблицы истинности

Алгоритм построения таблицы истинности

• Определи количество строк в таблице по формуле 2^k , где k количество логических переменных в выражении.

• Определи количество столбцов в таблице – количество логических переменных плюс количество логических операций.

• Построй таблицу истинности. Обозначь столбцы, заполни столбцы значений исходных логических переменных всевозможными различными наборами значений из 2 элементов (0,1) по k, где k – количество данных логических переменных (размещения с повторением из 2 элементов (0 или 1) по k).

• Заполни таблицу до конца, выполняя логические операции в заданной в выражении последовательности (табл. 4).

Составление таблиц истинности логических функций с использованием систем программирования

Обозначения истинности выражений в языках программирования дано в табл. 4.

Таблица 4

Значение истинности	Запись значений			
	Turbo Pascal	Turbo Delphi	Visual Basic	QBasic
Ложь	false	false	false	0
Истина	true	true	true	-1

Программы построения таблиц истинности логических операций: A and B, A or B, not A, A xor B.

Turbo Pascal

```
program table1;
var A,B:   boolean;
begin
writeln ('A  ':7, 'B  ':7, ' A and B ':7, 'not A ' :7, 'A or B ':7,'A xor  B ':7);
for A:=false  to  true do
for B:=false  to  true  do
begin
    writeln (A:7, B:7,(A and B):7, not A  :7,(A or  B):7, (A xor B) :7);
end; end.
```

QBasic

```
REM table
CLS
PRINT USING"\       \"; "      A";"      B";" A and B";"  not A";"A OR B";" A XOR B"
FOR A = 0 TO -1 STEP -1
  FOR B = 0 TO -1 STEP -1
    PRINT USING "########"; A; B; A and B; not A; A or B; A xor B
NEXT B, A
END
```

Turbo Delphi	Visual Basic.Net

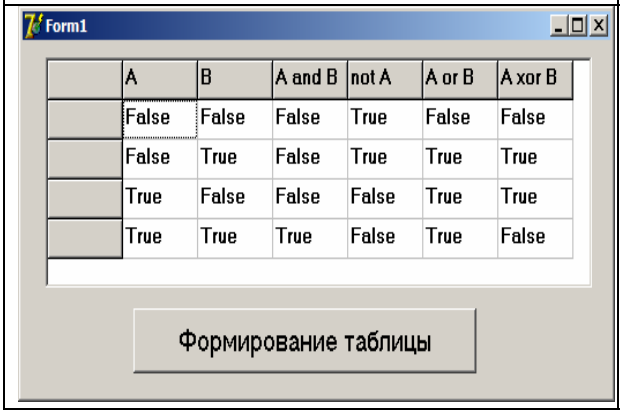
procedure TForm1.Button1Click (Sender: TObject);
const n=6;
var a,b:boolean; i,j:1..n;
begin
with stringgrid1 do
begin
Cells[1,0]:='A';
Cells[2,0]:='B';
Cells[3,0]:='A and B';
Cells[4,0]:='not A';
Cells[5,0]:='A or B';
Cells[6,0]:='A xor B';
end;
i:= 1;
For a:= false To true do
For b:= false To true do
begin
with StringGrid1 do
begin
j:= 1; Cells[j, i]:= BoolToStr(a,true);
j:= 2; Cells[j, i]:= BoolToStr(b,true);
j:= 3; Cells[j, i]:= BoolToStr(a And b, true);
j:= 4; Cells[j, i]:= BoolToStr(Not a,true);
j:= 5; Cells[j, i]:= BoolToStr(a Or b, true);
	Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
	Dim a, b As Boolean
	Dim i, j, k, l As Integer
	With (DataGridView1)
	.Rows.Add(3)
	.Columns(0).HeaderText = "A"
	.Columns(1).HeaderText = "B"
	.Columns(2).HeaderText = "A and B"
	.Columns(3).HeaderText = "not A"
	.Columns(4).HeaderText = "A or B"
	.Columns(5).HeaderText = "A xor B"
	End With
	i = 0
	For k = 0 To -1 Step -1
	If k = -1 Then a = True Else a = False
	For l = 0 To -1 Step -1
	If l = -1 Then b = True Else b = False
	With DataGridView1
	j = 0 : .Item(j, i).Value = a
	j = 1 : .Item(j, i).Value = b
	j = 2 : .Item(j, i).Value = a And b
	j = 3 : .Item(j, i).Value = Not a
	j = 4 : .Item(j, i).Value = a Or b
	j = 5 : .Item(j, i).Value = a Xor b

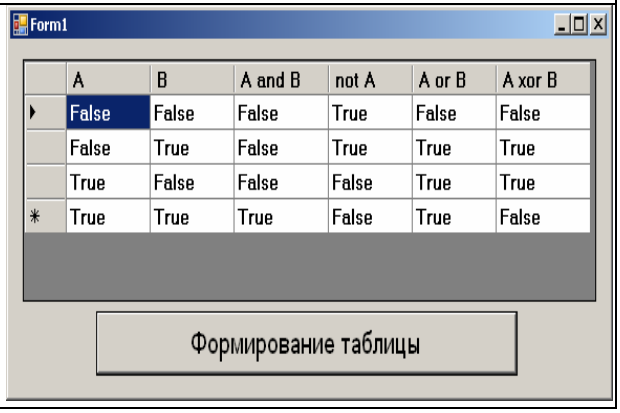
```
   j:= 6; Cells[j, i]:= BoolToStr(a Xor b, true);
   i:= i + 1;
   end;
   end;
end;
```

```
    i = i + 1
   End With
   Next
   Next
End Sub
```

Результат формирования таблиц

Delphi	Visual Basic.NET





2.3. Упрощение логических выражений

Алгоритмы получения СДНФ И СКНФ по таблице истинности

СДНФ	СКНФ
1. Отметьте звёздочкой строки таблицы, в которых стоят единицы в столбце значений логической функции. 2. Для каждой выделенной строки выпишите конъюнкцию всех переменных: если значение переменной в данной строке равно 1, то в конъюнкцию включите саму эту переменную, а если 0, то её отрицание.	1. Отметьте звёздочкой строки таблицы, в которых стоят нули в столбце значений логической функции. 2. Для каждой выделенной строки выпишите дизъюнкцию всех переменных: если значение переменной в данной строке равно 0, то в дизъюнкцию включите саму эту переменную, а если 1, то её отрицание.

3. Все конъюнкции свяжите в дизъюнкцию. Получите СДНФ исследуемой функции.	3. Все дизъюнкции свяжите в конъюнкцию. Получите СКНФ исследуемой функции

Основные законы логики

Законы коммутативности $A \vee B = B \vee A,\ A \wedge B = B \wedge A$ Законы дистрибутивности $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C),$ $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ Важные формулы (I – истина, L – ложь). $A \vee I = I,\ A \vee \overline{A} = I,\ A \wedge I = A,$ $A \wedge L = L,\ \overline{A} \vee (A \vee B) = \overline{A} \wedge B,$ $A \vee \overline{A} \wedge B = A \vee B, A \wedge \overline{A} = L.$	Законы ассоциативности $(A \vee B) \vee C = A \vee (B \vee C),$ $(A \wedge B) \wedge C = A \wedge (B \wedge C).$ Законы идемпотентности $A \vee A = A,\ A \wedge A = A.$ Законы поглощения $A \vee A \wedge B = A,\ A \wedge (A \vee B) = A.$ Законы де Моргана $\overline{A \vee B} = \overline{A} \wedge \overline{B},\ \overline{A \wedge B} = \overline{A} \vee \overline{B}.$
Инфолюция $\overline{\overline{A}} = A,\ A \to B = \overline{A} \vee B,$ $A \leftrightarrow B = (A \wedge B) \vee (\overline{A} \wedge \overline{B}) = (A \to B) \wedge (B \to A) = (\overline{A} \vee B) \wedge (\overline{B} \vee A)$	

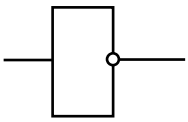
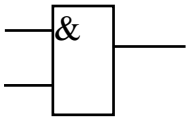
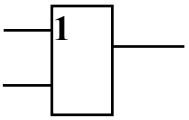
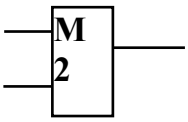
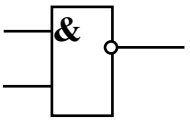
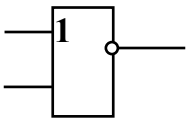
3. Базовые логические элементы функциональных схем, реализующие логические операции

Для составления функциональных схем используют базовые логические элементы, простейшие преобразователи информации[10], являющиеся графическими представлением часто используемых логических операций, табл. 5. Преобразователи могут иметь один или несколько входов и только один выход. По входам преобразователь получает информацию, на выходах демонстрирует результат преобразования. На входах и выходах появляются сигналы только булевского

[10] Преобразователи, которые могут, получая сигналы об истинности отдельных простых высказываний, обработать их и в результате выдать значение логической операции, называются логическими элементами [9].

типа. Сигнал, выработанный одним логическим элементом, можно подавать на вход другого элемента, что даёт возможность образовывать цепочки из отдельных логических элементов. Каждая цепочка из логических элементов, где выходы одних логических элементов являются входами других, реализует определённую логическую функцию. Графическое представление логической функции называется функциональной схемой функции.

Таблица 5

Инвертор	Конъюнктор	Дизъюнктор
		
Элемент XOR	Элемент Вебба, И – НЕ	Элемент Шеффера, ИЛИ – НЕ
		

4. Информационные и компьютерные модели решения логических содержательных задач

Решение логических содержательных задач – это практикум в составлении информационных и компьютерных моделей, это практикум в формировании умений и навыков формализации условия задачи и процесса её решения. Существуют различные способы решения логических содержательных задач, различные способы составления информационных моделей: алгебраический, табличный, графический и др. Каждый из этих методов обладает своими достоинствами. Так, например, при применении *алгебраического метода* наиболее трудным считается перевод текста задачи на язык формул, но далее, если вы знаете методы упрощения логических выражений или умеете составлять таблицы истинности на компьютере, решение задачи сводится к формальным преобразованиям и приводит сразу к ответу, кото-

рый остается лишь расшифровать, исходя из принятых вами обозначений.

Табличный метод очень нагляден, но предназначен для решения только определенного класса задач, он требует анализа находящейся в таблице информации, умения сравнивать и сопоставлять.

Граф позволяет наглядно представить связи между объектами, о которых идет речь в задаче, и определить, какие из них не противоречат условиям задачи.

Метод *диаграмм Эйлера — Венна* позволяет графически решать математические задачи на основе применения теории множеств.

Как правило, задачу можно решить несколькими способами (методами). Чтобы выбрать наиболее простой и эффективный способ для каждой конкретной задачи, необходимо знать все эти способы.

4.1. Алгебраический метод решения логических задач

Задача. Алеша, Боря и Гриша нашли в земле сосуд. Рассматривая удивительную находку, каждый высказал по два предположения:

Алеша сказал: «*Это сосуд греческий и изготовлен в V веке».* Боря сказал: «*Это сосуд финикийский и изготовлен в III веке».* Гриша сказал: «*Это сосуд не греческий и изготовлен в IV веке».* Учитель истории сказал ребятам, что каждый из них прав только в одном из двух предположений. Где и в каком веке изготовлен сосуд?

Указание. Используйте поэтапное решение задачи с составлением информационных и компьютерных моделей, предложенное на с. 104.

Решение

А. Реализуем первый и второй этапы решения задачи. В условии даны высказывания школьников и учителя. Введём следующие обозначения: *G* – «*Это сосуд греческий»; F* – «*Это сосуд финикийский»; V_3 – «*Сосуд изготовлен в III в.»; V_4* – «*Сосуд изготовлен в IVв.»; V_5* – «*Сосуд изготовлен в V в.»*

В. Условие задачи запишем в выбранных обозначениях, формализуем условие задачи. Из высказывания учителя следует, что Алеша

прав только в чем-то одном: или $G=1$, или $V_5=1$. Таким образом, тождественно истинным будет высказывание – $G\cdot\overline{V_5}\vee\overline{G}\cdot V_5$. Аналогично из слов Бори и учителя следует, что $F\cdot\overline{V_3}\vee\overline{F}\cdot V_3=1$, а из слов Гриши и учителя – $\overline{G}\cdot\overline{V_4}\vee G\cdot V_4=1$. Кроме того, ясно, что сосуд может быть изготовлен только в одном из веков и только в одной из стран. Эти условия можно записать так:

$$V_3\cdot\overline{V_4}\cdot\overline{V_5}\vee\overline{V_3}\cdot V_4\cdot\overline{V_5}\vee\overline{V_3}\cdot\overline{V_4}\cdot V_5=1,\ F\cdot\overline{G}\vee\overline{F}\cdot G=1.$$

Получено пять тождественно истинных высказываний. Составим конъюнкцию этих высказываний. Получим логическую функцию

$$H(G,F,V_3,V_4,V_5)=\left(G\cdot\overline{V_5}\vee\overline{G}\cdot V_5\right)\&\left(F\cdot\overline{V_3}\vee\overline{F}\cdot V_3\right)\&\left(\overline{G}\cdot\overline{V_4}\vee G\cdot V_5\right)\&$$
$$\&\left(F\cdot\overline{G}\vee\overline{F}\cdot G\right)\&\left(V_3\cdot\overline{V_4}\cdot\overline{V_5}\vee\overline{V_3}\cdot V_4\cdot\overline{V_5}\vee\overline{V_3}\cdot\overline{V_4}\cdot V_5\right).$$

С. Упростим функцию *H*. Применим сначала закон коммутативности – второе и третье выражения в конъюнкции поменяем местами, а затем закон дистрибутивности конъюнкции относительно дизъюнкции к выражениям, определяющим функцию.

$$H=\left(G\cdot\overline{V_5}\cdot\overline{G}\cdot\overline{V_4}\vee G\cdot\overline{V_5}\cdot G\cdot V_4\vee\overline{G}\cdot V_5\cdot\overline{G}\cdot\overline{V_4}\vee\overline{G}\cdot V_5\cdot G\cdot V_4\right)\&$$
$$\&\left(F\cdot\overline{V_3}\cdot F\cdot\overline{G}\vee F\cdot\overline{V_3}\cdot\overline{F}\cdot G\vee\overline{F}\cdot V_3\cdot F\cdot\overline{G}\vee\overline{F}\cdot V_3\cdot\overline{F}\cdot G\right)\&$$
$$\&\left(V_3\cdot\overline{V_4}\cdot\overline{V_5}\vee\overline{V_3}\cdot V_4\cdot\overline{V_5}\vee\overline{V_3}\cdot\overline{V_4}\cdot V_5\right).$$

Преобразуем выражения в первой и второй скобках, учитывая, что $G\cdot\overline{G}=0$, $G\cdot G=G$, $\overline{G}\cdot\overline{G}=\overline{G}$, получим:

$$H=\left(G\cdot\overline{V_5}\cdot V_4\vee\overline{G}\cdot V_5\cdot\overline{V_4}\right)\&\left(F\cdot\overline{V_3}\cdot\overline{G}\vee\overline{F}\cdot V_3\cdot G\right)\&$$
$$\&\left(V_3\cdot\overline{V_4}\cdot\overline{V_5}\vee\overline{V_3}\cdot V_4\cdot\overline{V_5}\vee\overline{V_3}\cdot\overline{V_4}\cdot V_5\right).$$

Применяя к выражениям, определяющим функцию, закон дистрибутивности конъюнкции относительно дизъюнкции и, упрощая их, последовательно получаем:

$$H=\left(G\cdot\overline{F}\cdot V_3\cdot V_4\cdot\overline{V_5}\vee\overline{G}\cdot F\cdot\overline{V_3}\cdot\overline{V_4}\cdot V_5\right)\&$$
$$\&\left(V_3\cdot\overline{V_4}\cdot\overline{V_5}\vee\overline{V_3}\cdot V_4\cdot\overline{V_5}\vee\overline{V_3}\cdot\overline{V_4}\cdot V_5\right)=G\cdot\overline{F}\cdot V_3\cdot V_4\cdot\overline{V_5}\cdot V_3\cdot\overline{V_4}\cdot\overline{V_5}\vee$$
$$\vee G\cdot\overline{F}\cdot V_3\cdot V_4\cdot\overline{V_5}\cdot\overline{V_3}\cdot V_4\cdot\overline{V_5}\vee G\cdot\overline{F}\cdot V_3\cdot V_4\cdot\overline{V_5}\cdot\overline{V_3}\cdot\overline{V_4}\cdot V_5\vee$$
$$\vee\overline{G}\cdot F\cdot\overline{V_3}\cdot\overline{V_4}\cdot V_5\cdot V_3\cdot\overline{V_4}\cdot\overline{V_5}\vee\overline{G}\cdot F\cdot\overline{V_3}\cdot\overline{V_4}\cdot V_5\cdot\overline{V_3}\cdot V_4\cdot\overline{V_5}\vee$$
$$\vee\overline{G}\cdot F\cdot\overline{V_3}\cdot\overline{V_4}\cdot V_5\cdot\overline{V_3}\cdot\overline{V_4}\cdot V_5=\overline{G}\cdot F\cdot\overline{V_3}\cdot\overline{V_4}\cdot V_5.$$

D. Определим, для какой совокупности значений пяти логических переменных G, F, V_3, V_4, V_5 функция H принимает значение *true.* Дизъюнкция истинных высказываний есть истинное высказывание, следовательно, H=1, но $H = \overline{G} \cdot F \cdot \overline{V_3} \cdot \overline{V_4} \cdot V_5$, отсюда $\overline{G} \cdot F \cdot \overline{V_3} \cdot \overline{V_4} \cdot V_5 = 1$, а следовательно, и $\overline{G} = 1$, $F = 1$, $\overline{V_3} = 1$, $\overline{V_4} = 1$, $V_5 = 1$.

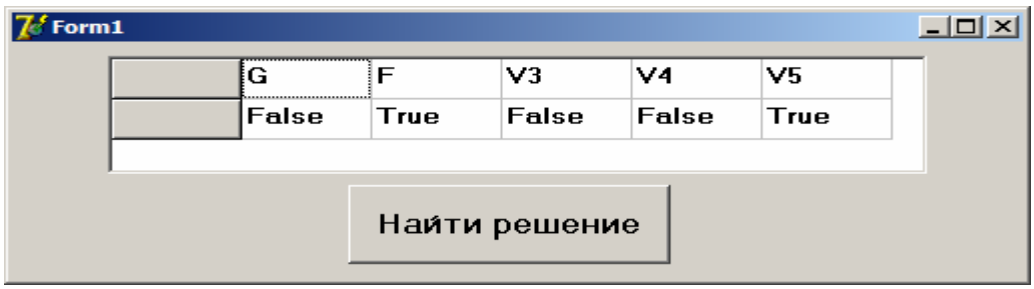
Таким образом, сосуд финикийский и изготовлен в V веке.

Программная реализация этапов C – D решения поставленной задачи

Сущность предлагаемого способа состоит в том, чтобы, выполнив этапы A и B, составить компьютерную модель процесса составления таблицы истинности для функции H и выбрать совокупности значений логических переменных G, F, V_3, V_4, V_5, при которых значение функции H равно *true*. Выберем для компьютерной модели исполнителя Turbo Delphi.

Оболочку программы можно взять из компьютерной модели на с. 105, изменив ядро программы, чтобы найти значения логических переменных G, F, $V3$, $V4$, $V5$, при которых значение функции H истинно.

Результат выполнения программы



4.2. Применение графов к решению логических задач

Задача. Из трех человек, стоящих рядом, один всегда говорит правду (правдивый), другой всегда лжет (лжец), а третий, смотря по обстоятельствам, говорит правду или ложь («дипломат»). У стоящего слева спросили: «Кто стоит рядом с тобой?» Он ответил: «Правдолюб». Стоящему в центре человеку, задали вопрос: «Кто ты?», и он

ответил: «Я дипломат». Когда у стоящего справа человека, спросили: «Кто стоит рядом с тобой?», то он ответил: «Лжец». Кто, где стоял? Составьте информационную модель в виде графа и проведите исследования модели для нахождения решения.

Решение. Если в данной задаче ребро графа будет соответствовать месту, занимаемому тем или иным человеком, то могут представиться следующие возможности их расположения (рис. 40).

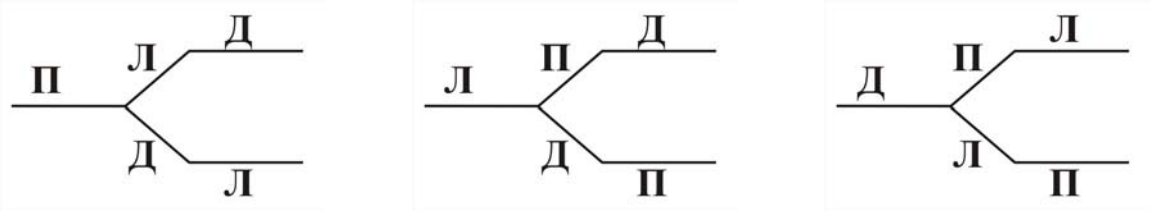


Рис. 40

Решение. Рассмотрим первую возможность. Если «правдолюб» стоит слева, то рядом с ним, судя по его ответу, также стоит «правдолюб». У нас же стоит лжец. Следовательно, эта расстановка не удовлетворяет условию задачи. Рассмотрев все остальные возможности, приходим к выводу, что расстановка «дипломат», «лжец», «правдолюб» удовлетворяет задаче. Действительно, если «правдолюб» стоит справа, то по его ответу, рядом с ним стоит «лжец», что выполняется. Стоящий в центре заявляет, что он «дипломат», и, следовательно, лжет. Таким образом, все условия задачи выполнены.

4.3. Табличные информационные модели решения логических содержательных задач

Задача. Беседуют трое друзей: Белокуров, Рыжов и Чернов. Брюнет сказал Белокурову: «Любопытно, что один из нас блондин, другой брюнет, третий — рыжий, но ни у кого цвет волос не соответствует фамилии». Какой цвет волос у каждого из друзей?

Пояснение к решению. Для решения задачи можно предложить составить информационную модель в виде табл. 6. Интерпретируем следующее предложение: «Ни у кого из друзей цвет волос не соответствует фамилии». Поставим в соответствующих клетках символ «–».

Между множеством фамилий участников беседы и множеством цветов волос должно быть установлено взаимно-однозначное соответствие. Если определили цвет волос у участника беседы, то в соответствующую клетку поставьте знак « +». В каждой ячейке таблицы должен быть поставлен один из знаков «+» или «–».

Таблица 6

Фамилии	Цвет волос		
	рыжие	чёрные	русые
Белокуров			–
Чернов		–	
Рыжов	–		

5. Вопросы и задания к семинарским занятиям

Тема «Логические информационные и компьютерные модели»

Перефразируйте задания 1 – 6 главы 1, сформулировав предлагаемые задания и вопросы для рассматриваемой темы «Логические информационные и компьютерные модели». Выполните задания 1 – 6, ответьте на вопросы, поставленные в этих заданиях. Доложите результаты своей работы в студенческой аудитории в форме презентации.

7. Подберите мотивационные задачи для введения понятий, выделенных в составленной логико-структурной модели учебного материала по теме. Апробируйте свои разработки в студенческой аудитории.

8. Отберите содержание и разработайте методику изложения выбранного учебного материала по предложенным ниже темам. Проведите презентацию проделанной работы в студенческой аудитории:

8.1. Роль математической логики в информатике.

8.2. Высказывания. Логические константы. Логические переменные. Логические операции. Логические выражения, формулы, функции.

8.3. Упрощение сложных высказываний. Законы логики. Совершенная дизъюнктивная нормальная форма и совершенная конъюнктивная нормальная форма. Карты Карно.

8.4. Программирование построения таблиц истинности логических функций.

8.5. Построение информационных и компьютерных моделей при решении логических содержательных задач.

8.6. Простейшие преобразователи информации. Построение информационных и компьютерных моделей для исследования логических функций.

8.7. Информационные и компьютерные модели типовых логических устройств компьютера.

9. На выбранной ступени непрерывного курса изучения информатики и информационно-коммуникационных технологий по темам пункта 8 смоделируйте серию последовательных уроков. Апробируйте в студенческой аудитории фрагмент одного из разработанных уроков.

10. Разработайте по выбранной схеме технологическую карту одного из уроков по выбранной теме содержательной линии «Моделирование и формализация». Проведите презентацию своей разработки в студенческой аудитории.

6. Лабораторно-практическая работа

Информационные и компьютерные модели, составленные при выполнении заданий 1, 2, представьте в студенческой аудитории в форме презентации.

***Задание* 1.** Составьте информационные и компьютерные модели электронных логических устройств, выполняющих суммирование двоичных чисел: табличную модель, логическую функцию, функциональную схему, реализуйте информационные модели на соответствующем программном обеспечении:

а) одноразрядного полусумматора – устройства с двумя входами *X* и *Y* и двумя выходами *Q* и *S*;

б) одноразрядного сумматора – устройства с тремя входами *X*, *Y*, *P* и двумя выходами *Q* и *S*.

Примерный алгоритм решения задач а) и б):

- активизируйте правила сложения двоичных чисел;
- представьте правила сложения двоичных чисел в виде таблицы истинности (составьте табличную информационную модель работы сумматоров);
- используя СДНФ или СКНФ, составьте логические функции для нахождения Q и S (информационные модели работы сумматоров в виде логических функций от двух и трёх переменных соответственно), упростите полученные выражения;
- составьте функциональные схемы по соответствующим структурным формулам и компьютерные модели работы сумматоров для выбранных исполнителей (Ершол, QBasic, Turbo Pascal, Visual Basic.Net, Turbo Delphi, Microsoft Excel, Конструктор логических схем);
- проведите вычислительный эксперимент на выбранном программном обеспечении.

***Задание* 2.** Составьте информационную модель работы асинхронного RS-триггера в виде таблицы истинности и исследуйте его работу с использованием компьютерной модели (рис. 41).

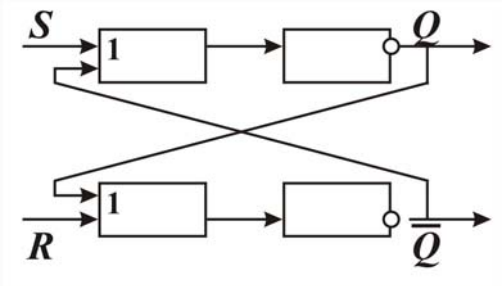


Рис. 41

***Задание* 3.** Разработайте и реализуйте с использованием выбранного исполнителя учебный проект в виде компьютерной модели

на тему «Конструктор логических схем», рассмотрите при проектировании все этапы составления компьютерных моделей. Подготовьте проект к защите.

***Задание* 4.** Составьте с использованием электронных таблиц и систем программирования компьютерные модели для решения отобранных вами логических содержательных задач. Результаты выполнения задания оформите в виде презентации и защитите в студенческой аудитории.

***Задание* 5.** Составьте логические содержательные задачи, для решения которых необходимо использовать компьютерные модели, моделирующие процесс поиска решения задач. Разработайте методику обучения школьников поиску решения предложенных задач, оформите решения составленных задач. Апробируйте разработанные методики в студенческой аудитории.

7. Самостоятельная работа

Выполнение заданий 1 – 4 и задач 1 – 2 подготовьте для проверки при индивидуальном собеседовании.

***Задание* 1.** Логическая функция $F(A,B,C)$ задана таблицей значений, определите выражение этой функции через логические операции $\neg$, $\wedge$,$\vee$, используя СДНФ и СКНФ. Упростите полученное выражение с использованием законов логики и карт Карно:

а)

A	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1
C	0	1	0	1	0	1	0	1
F	0	0	0	1	0	1	0	1

б)

A	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1
C	0	1	0	1	0	1	0	1
F	0	0	1	1	0	0	1	1

***Задание* 2.** Упростите заданные логические функции F. Упрощённый вид этих функций должен содержать не более трёх логических операций. Составьте для упрощённых функций F функциональные схемы:

а) $F(A,B,C)=\big(A\to(B\vee C)\big)\leftrightarrow(A\to B)\vee\overline{(A\to C)}$,

б) $F(A,B,C)=\Big(\big(\overline{A}\leftrightarrow\overline{B\wedge C}\big)\to\overline{C}\Big)\to\big(\overline{A}\vee\overline{C}\leftrightarrow\overline{B}\big)$.

***Задание* 3.** Составьте логическое выражение, реализуемое приведенной ниже функциональной схемой. Вычислите значение логического выражения при $a=b=1$; $a=b=0$; $a=0$, $b=1$. Упростите полученное логическое выражение. Составьте функциональную схему, соответствующую упрощенному выражению (рис. 42).

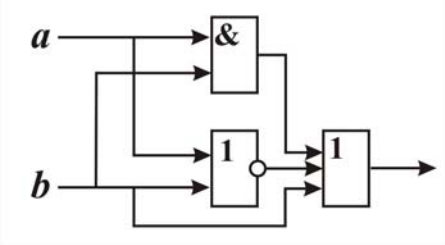


Рис. 42

***Задание* 4.** Разработайте сценарии программ (демонстрационных, обучающих, тренажеров, контролирующих) по данной теме и реализуйте их на одном из языков программирования.

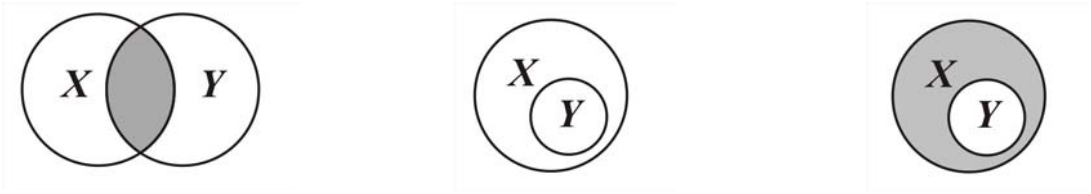
***Задача* 1.** В некотором университете проходит в несколько туров конкурс вокала. Тремя членами жюри (A, B, C) решается вопрос о допуске того или иного участника к следующему туру. Решение положительно тогда и только тогда, когда большинство членов жюри высказывается за допуск. Разработайте компьютерную модель для голосования и реализуйте её с использованием соответствующего программного обеспечения.

***Задача* 2.** Решите поставленную задачу, составив информационную модель с помощью кругов Эйлера – Венна. В школе из 100 учеников английский язык изучают 28, немецкий – 30, французский – 42, английский и немецкий – 8, английский и французский – 10, немец-

кий и французский – 5, немецкий, английский и французский – 3. Сколько учеников не изучают ни одного языка? Сколько человек изучают только немецкий язык, только английский язык, только французский?

Пояснение к решению задачи

Нужно использовать наглядную геометрическую иллюстрацию объёмов понятий и отношений между ними, выделив следующие виды отношений: пересечение, включение, дополнение (рис. 43).



Пересечение $X \cap Y$ Включение $X \subset Y$ Дополнение $X - Y$

Рис. 43

ПРИЛОЖЕНИЯ

Примеры учебных[11] компьютерных моделей процессов, реализованных в решении практических задач

Приложение 1. Компьютерные модели процесса нахождения эйлерового цикла в графе

В прил. 1 даны компьютерные модели нахождения эйлерова цикла в графе, для которого выполняются необходимые и достаточные условия существования этого цикла (см. с. 79). Для того чтобы использовать модели для определения существования эйлерова пути (определения, является ли диаграмма исследуемого графа уникурсальной линией), необходимо дополнить модели процедурами с учётом замечания 1.

Алгоритм нахождения эйлерова цикла

Для реализации процесса нахождения эйлерового цикла от выбранной вершины организуйте два стека: стек *ST* и *ST1*. Так как стеки в предложенных моделях организованы в виде массивов, то необходимо выбрать два указателя *uk* и *uk1* типа byte на соответствующие вершины стеков.

1. Введите в память компьютера матрицу смежности *A* либо с клавиатуры, либо с использованием типизированной константы-массива, либо из текстового файла.

2. Введите начальную вершину эйлерового цикла – *V*. Поместите её в стек *ST*.

3. Пока стек *ST* не пуст, выполняйте шаги 4, 4.1, 4.2, иначе идите на шаг 5.

[11] В данном пособии приведены примеры учебных компьютерных моделей, они реализуют ядро в программировании моделируемого процесса. Желательно, чтобы читатель видоизменил программу в соответствии с особенностями используемой для программирования системы, личными знаниями и вкусами, обогатил программу мультимедийными объектами, защитил проект от неквалифицированного пользователя.

4. Просматривайте граф «в глубину», начиная от вершины графа *T*, находящейся в вершине стека *ST*:

4.1. Если найдена вершина *J*, смежная с рассматриваемой вершиной *T*, то поместите её в стек *ST* и удалите рёбро {*T*, *J*} из графа, т.е. соответствующим элементам матрицы смежности *A* присвойте значение нуль: a[t,j]:=0, a[j,t]:=0 и идите на шаг 3, иначе идите на шаг 4.2.

4.2. Если нет в преобразованном графе вершин *J*, смежных с вершиной *T*, находящейся в вершине стека *ST* , то вершину *T* поместите в стек *ST*1, а из стека *ST* удалите вершину *T* и идите на шаг 3. В этом случае из графа был удалён цикл, и вершины этого цикла находятся в стеке *ST*.

5. Выведите на экран или в файл вершины из стека *ST*1, определяющие эйлеров цикл, начиная с выбранной вами вершины. Закончите исполнение алгоритма.

Компьютерная модель 1 процесса нахождения эйлерового цикла[12]. Компьютерная модель реализована в системе Turbo Pascal. Граф задаётся матрицей смежности, элементы которой вводятся из текстового файла. Полное имя текстового файла и номер начальной вершины цикла нужно ввести с клавиатуры в процессе работы программы.

```
Program cycle; {Поиск Эйлерова цикла}
 uses Crt;
 const n = 7; m = 12; k=14;
 type mass = array [1..n, 1..n] of byte;
 type mass1 = array [1..k] of byte;
{ если система Turbo Pascal позволяет использовать
типизированные константы, то для отладки программы
можно этим воспользоваться:
 const
  a:mass = ((0,1,0,0,1,0,0),(1,0,1,0,1,1,0),
  (0,1,0,1,0,1,1), (0,0,1,0,1,1,1), (1,1,0,1,0,1,0), (0,1,1,1,1,0,0),   (0,0,1,1,0,0,0));}
  var
     t,v,j,uk,uk1,uk2,i,j1:byte;a:mass;
     st1,st,st2: mass1; pr: boolean;
```

[12] Компьютерные модели работают со *связными* графами, у которых все вершины *чётной* степени.

```
Procedure input;
 Var
  f: text; i,j,b: byte; name: string [20];
 Begin
  write ('введите имя текстового файла,');
  write ('содержащего элементы матрицы смежности');    readln (name);
  assign (f,name);
  reset (f);
  for i:=1 to n do
   for j:=1 to n do
    begin
     read(f,b); { чтение числа b из файла в ОЗУ}
     a[i,j]:=b; {присвоение текущему элементу матрицы смежности значение b}
    end;
  close (f);
 end;
Procedure print1 (x1:mass1; uk2:byte);
 {процедура печати элемента стека, определяемого именем X1}
 var apex:byte;uk3:byte;x2:mass1;
begin
 uk3:=0;
 while uk2<>0 do
  begin
   apex:=x1[uk2]; uk2:=uk2-1; uk3:=uk3+1; x2[uk3]:=apex;
  end;
 while uk3>0 do
  begin write (x2[uk3], ';'); uk3:=uk3-1; end;
 writeln;
end;
begin {основная программа}
 clrscr;  input; {ввод матрицы смежности};
 write('введите первую вершину для просмотра');  readln(v);
 uk1:=0; uk:=1; st [uk]:=v; {инициализация стека ST}
 while (uk <> 0) do {обход графа «в глубину» с удалением просмотренных рёбер}
  begin
   t:=st[uk];    j:=1;    pr:=false;
   repeat
    if a[t,j]<>0 then
     pr:=true
    else
     j:=j+1
   until (pr) or (j>n);
  {найдено новое ребро или все ребра, инцидентные с вершиной V, просмотрены}
  {обработка поиска}
```

```
 If pr then
  begin
   uk:=uk+1;     st[uk]:=j;     a[t,j]:=0;     a[j,t]:=0; Write('стек st:   ');print1(st,uk);
  end
 else
  begin
   uk1:=uk1+1; st1[uk1]:=t;  Write('стек st1:   ');print1(st1,uk1);     uk:=uk-1
  end
end;
writeLn('*************************************');
writeLn ('Эйлеров цикл от вершины ', v);  print1(st1,uk1);
writeLn('*************************************')
End.
```

Компьютерная модель 2 процесса нахождения эйлерового цикла. Компьютерная модель реализована в электронных таблицах MS Office Excel с использованием языка VBA. Матрица смежности вводится из текстового файла, полное имя которого вы должны ввести с клавиатуры и ввести номер начальной вершины цикла.

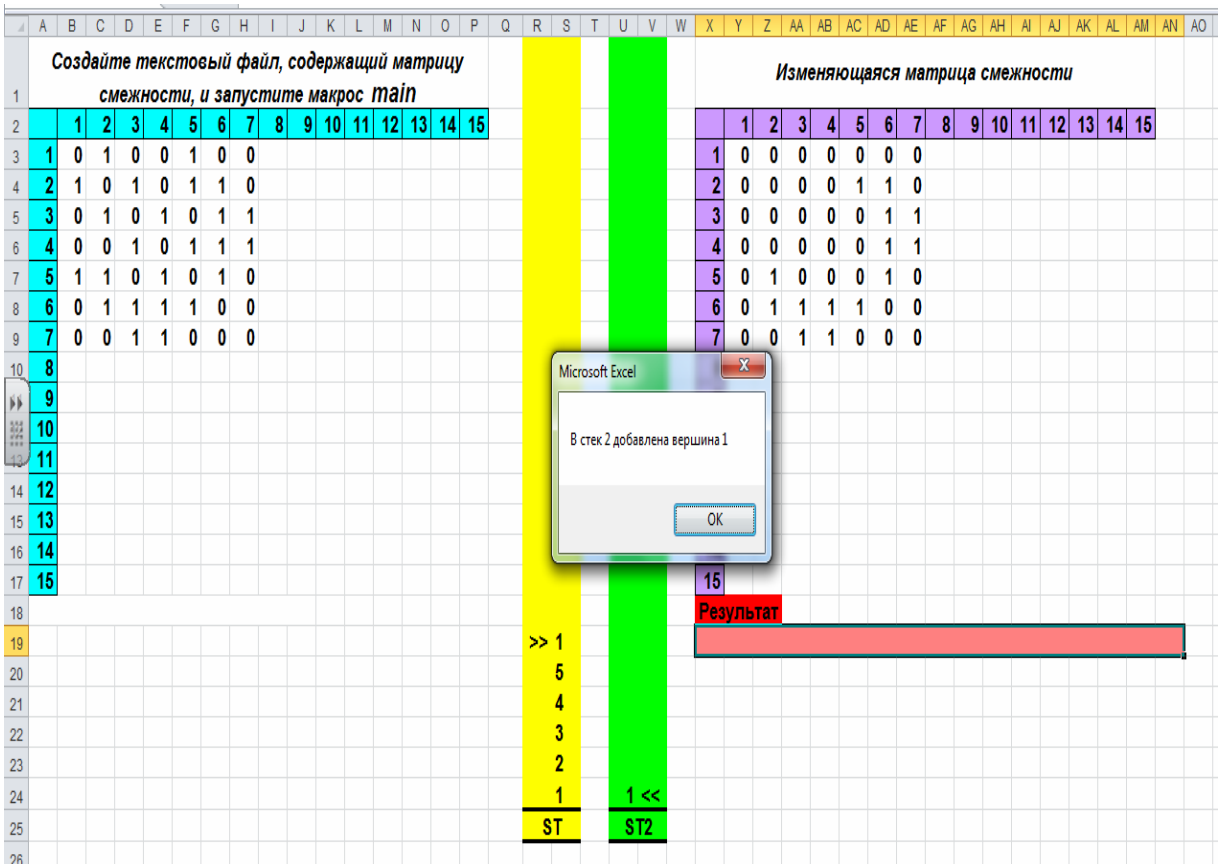
```
Dim n, fn As Integer, uk1, uk, a(15, 15), st(15), st1(15), st2(15) As Byte
Sub main()
n = InputBox("введите количество вершин графа")
vvod
unikurs
End Sub

Sub vvod()
'очистка
Range("B3:P15").Select:    Selection.ClearContents
Range("Y3:AM15").Select:   Selection.ClearContents
Range("R1:V24").Select:    Selection.ClearContents
Range("X19:AN19").Select:  Selection.ClearContents
'ввод матрицы смежности
fn = FreeFile:    s = InputBox("Введите полный путь к файлу ")
Open s For Input As fn
For i = 1 To n
   For j = 1 To n
      c = Input(1, fn)
      If (c <> "0") And (c <> "1") Then j = j - 1: GoTo 1
```

```
        Cells(i + 2, j + 1).Value = c
        a(i, j) = Cells(i + 2, j + 1).Value
1:      Next j
Next i
Close fn
For i = 1 To n
    For j = 1 To n
        Cells(i + 2, j + 24).Value = a(i, j)
Next j, i
End Sub
Sub unikurs()
v = InputBox("Введите первую вершину для просмотра")
'инициализация стека  st
uk1 = 0:     uk = 1:     st(uk) = v
Cells(25 - uk, 19).Value = st(uk)
'просмотр в глубину
While uk <> 0
 t = st(uk):    j = 1:      pr = False
 Do
    If a(t, j) <> 0 Then pr = True Else j = j + 1
 Loop Until (pr) Or (j > n)
'найдено новое ребро или все ребра, инцидентные с вершиной v, просмотрены
'обработка поиска
If pr Then
  uk = uk + 1:     st(uk) = j
  a(t, j) = 0:     Cells(t + 2, j + 24).Value = 0 :   a(j, t) = 0:     Cells(j + 2, t + 24) = 0
  Cells(25 - uk, 19).Value = st(uk):    Cells(25 - uk, 18).Value = ">>"
  Cells(25 - uk + 1, 18).Value = ""
  For q = 1 To 20000000
  Next q
Else
  uk1 = uk1 + 1:    st1(uk1) = t:    Cells(25 - uk1, 21).Value = st1(uk1)
  Cells(25 - uk1, 22).Value = "<<":     Cells(25 - uk1 + 1, 22).Value = ""
  MsgBox ("В стек 2 добавлена вершина " & st1(uk1))
  For q = 1 To 20000000
  Next q
  uk = uk - 1
End If
```

```
Wend
print1 (uk1)  'печать результата
End Sub
Sub print1(uk As Byte)
'процедура вывода на экран маршрута обхода графа,
'удовлетворяющего требуемому условию
Dim uk2 As Byte
uk2 = 0
While uk <> 0
   apex = st1(uk):     uk = uk - 1:     uk2 = uk2 + 1:     st2(uk2) = apex
Wend
i = 1
While uk2 > 0
   Cells(19, i + 23).Value = st2(uk2):     uk2 = uk2 - 1:     i = i + 1
Wend
End Sub
```

Вид окна MS Excel в промежуточный момент исполнения программы



Замечание 1. Чтобы использовать модели для определения существования эйлерова пути в графе G (определения, является ли диаграмма исследуемого графа уникурсальной линией), необходимо дополнить модели процедурами:

1. Процедурой определения степени вершин графа G, используя для нахождения степеней вершин графа, например, матрицу смежности. Если вершин с нечётной степенью в графе G больше, чем две, то выдайте ответ, что не существует в графе G эйлерова пути, диаграмма графа G не является уникурсальной линией.

2. Процедурой определения связности графа G, используя для проверки связности графа G, например, матрицу достижимости. Если граф G несвязный, то выдайте ответ, что не существует в графе G эйлерова пути, диаграмма графа не является уникурсальной линией.

3. Если есть не более двух вершин нечётной степени у исследуемого связного графа G, например, вершины U и V, то образуйте граф G_1 [13]:

а) добавьте в граф G новое ребро $\{U,V\}$, если $\{U,V\}$ не принадлежит графу G;

б) дополните граф G новой вершиной R и рёбрами $\{U,R\}$ и $\{R,V\}$, если ребро $\{U,V\}$ принадлежит графу G;

в) воспользуйтесь предложенными моделями для нахождения эйлерова цикла в графе G_1, начиная от вершины или U, или R, или V;

г) в полученном эйлеровом цикле уберите добавленные рёбра. Получите эйлеров путь, который начинается либо от вершины U, либо от вершины V.

[13] Эйлеровы пути в графе G_1 находятся во взаимном соответствии с эйлеровыми путями в графе G.

Приложение 2. Компьютерные модели «Электронный кассир», имитирующие работу кассира по продаже билетов в кинотеатр на один сеанс

Компьютерная модель 1. «Электронный кассир», реализованная в системе QBasic. Пример разработки алгоритма и реализации некоторых процедур программы описан на с. 23.

```
REM Casier:
DECLARE SUB delay1 ()
DECLARE SUB clrscr1 () : DECLARE SUB Write2 ()
DECLARE SUB clrscr2 ()
DECLARE SUB delay () : DECLARE SUB Sale () : DECLARE SUB Selection ()
```

Основная программа

```
DIM SHARED o AS INTEGER: DIM SHARED time2 AS LONG
DIM SHARED TIME1 AS LONG: DIM SHARED time3 AS LONG
DIM t AS INTEGER: DIM SHARED f AS INTEGER
DIM SHARED fl1 AS INTEGER: DIM SHARED I AS INTEGER
DIM SHARED j AS INTEGER: DIM SHARED r AS INTEGER
DIM SHARED m AS INTEGER: DIM SHARED n AS INTEGER
DIM SHARED n1 AS INTEGER: DIM SHARED sum AS INTEGER
DIM SHARED z AS INTEGER
r = 5: m = 7: f = 1
DIM SHARED card(r, m + 1) AS INTEGER
 DIM SHARED cost(r) AS INTEGER
RANDOMIZE TIMER
CLS
sum = 0   'sum-количество проданных билетов
'формирование массива card
FOR I = 1 TO r
 FOR j = 1 TO m
   card(I, j) = INT(RND * 2)
   IF card(I, j) = 1 THEN sum = sum + 1
 NEXT j
 card(I, m + 1) = 1
NEXT I
'формирование массива cost
'цена билетов по рядам
t = INT(RND * 100)
cost(1) = t: cost(2) = t: cost(r - 1) = t: cost(r) = t: t = INT(RND * 200)
FOR I = 3 TO r - 2
 cost(I) = t + I * 10
NEXT I
```

```
'вывод расположения мест и цены билетов
Write2
time2 = TIMER
TIME1 = 0
'организация диалога покупателей и кассира
WHILE (sum < r * m) AND (TIME1 <= 300)
  o = 0
  LOCATE 11, 10: PRINT "Диалог с "; f; " покупателем:"
  LOCATE 12, 10: INPUT "Хотите купить билеты?(1-да,2-нет)"; o1
  IF o1 = 1 THEN
   LOCATE 13, 10: INPUT "Сколько билетов Вам нужно?"; n
   IF n > 0 THEN Sale
  END IF
  'анализ диалога с покупателем
  f = f + 1
  time3 = TIMER
  time1 = time3 - time2
  IF o1 <> 1 THEN
   LOCATE 22, 3: PRINT "Не хотите покупать билеты! До свидания!"
   GOTO 6
  END IF
  IF o = 1 THEN Write2
  IF o = 2 THEN
     LOCATE 22, 3: PRINT "К сожалению, других свободных мест нет!
                          Извините!  До свидания!"
  END IF
  IF fl1 = 0 THEN
       LOCATE 22, 3: PRINT "К сожалению, вместе расположенных"; n; " мест
       нет"
  END IF
  6 : delay
  clrscr2
WEND
IF TIME1 > 300 THEN
  LOCATE 21, 8: PRINT "Извините, сеанс уже начался.  До свидания!"
END IF
IF sum = r * m THEN
  LOCATE 20, 8: PRINT "Извините, все билеты проданы! До свидания!"
END IF
delay1
END
```

Подпрограммы

```
SUB clrscr1
'очистка экрана с 15 по 23 строку
DIM i1 AS INTEGER
FOR i1 = 15 TO 23
LOCATE i1, 1: PRINT SPC(78); " "
NEXT i1
END SUB
```

```
SUB clrscr2
'очистка экрана с 11 по 23 строку
DIM i1 AS INTEGER
FOR i1 = 11 TO 23
LOCATE i1, 1: PRINT SPC(78); " "
NEXT i1
END SUB
```

```
SUB delay
'задержка до нажатия любой клавиши
LOCATE 23, 5: PRINT "Для продолжения нажмите любую клавишу."
SLEEP
LOCATE 23, 5: PRINT SPC(70); " "
END SUB
```

```
SUB delay1
delay
clrscr2
END SUB
```

```
SUB Sale
DIM fl AS INTEGER
DIM kol AS INTEGER
'отыскание n1 подряд расположенных, непроданных билетов
PRINT
fl1 = 0
kol = 0
FOR I = 1 TO r
 FOR j = 1 TO m
  IF card(I, j) = 0 THEN
    fl = 0
    n1 = 0
    WHILE card(I, j) = 0
      n1 = n1 + 1
      z = j
      j = j + 1
      fl = 1
    WEND
    IF fl = 1 THEN j = j - 1
    IF n1 >= n THEN
```

```
SUB Selection
'диалог с покупателем
DIM k AS INTEGER, p AS INTEGER
fl1 = 1
LOCATE 15, 8
PRINT "Свободные места с ";
PRINT z - n1 + 1; " по "; z; " "; I; " ряда."
LOCATE 16, 8
PRINT "Стоимость билетов в этом ряду ";
cost(I);
PRINT  "р. за билет."
LOCATE 17, 8
PRINT "Вам подходит данный ряд?(1-да,2-нет)";
INPUT o
IF o = 1 THEN
 LOCATE 20, 8: PRINT SPC(70); " "
 IF n1 = n THEN
  FOR k = z - n + 1 TO z
   card(I, k) = 1
  NEXT k
  sum = sum + n
```

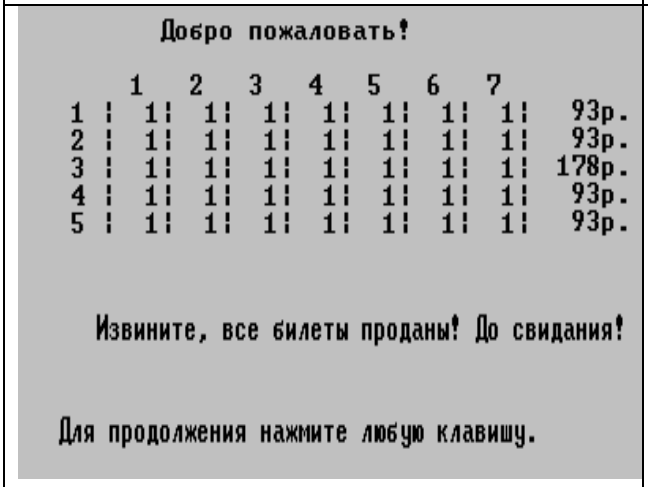
```
      kol = kol + 1
   IF (kol > 0) AND (kol <= r * m)
THEN
    IF o = 2 THEN
     LOCATE 20, 8: PRINT "Смотрим
следующие свободные места:"
    END IF
   END IF
   Selection
  END IF
 END IF
 IF o = 1 THEN GOTO 5
 NEXT j
NEXT I
5 : delay1
END SUB
******************************
**
SUB Write2
'вывод таблицы состояния продажи
билетов  и таблицы цены билетов
на данный сеанс
DIM i1 AS INTEGER
DIM j1 AS INTEGER
CLS
LOCATE 2, 17: PRINT "Добро
пожаловать!"
LOCATE 4, 12
FOR i1 = 1 TO m
  PRINT USING "####"; i1;
NEXT i1
FOR i1 = 1 TO r
  LOCATE 4 + i1, 10
  PRINT i1;
  FOR j1 = 1 TO m
   PRINT USING "|###"; card(i1, j1);
  NEXT j1
  PRINT "|";
  PRINT USING "#####p."; cost(i1)
NEXT i1
END SUB
```

```
    LOCATE 18, 8: PRINT " Продано! До
     свидания!"
  delay
 ELSE
  11 : LOCATE 18, 8:
  PRINT "С какого места хотите взять
  билеты?";
  INPUT p
  IF p > z - n + 1 THEN
     LOCATE 19, 8: PRINT "С этого места
     нет "; n;
    PRINT " рядом расположенных
     свободных мест"
    delay
    LOCATE 18, 1: PRINT SPC(70); " "
    LOCATE 19, 1: PRINT SPC(70); " "
    GOTO 11
  END IF
  FOR k = p TO p + n - 1
    card(I, k) = 1
  NEXT k
  sum = sum + n
  LOCATE 19, 8: PRINT "Продано!До
     свидания!"
  delay
 END IF
END IF
clrscr2
END SUB
```

Вид окна ввода-вывода системы QBasic в промежуточные моменты исполнения программы компьютерной модели

Кадр 1

Кадр 2



Кадр 3

Кадр 4

Компьютерная модель 2. «Электронный кассир», реализованная в электронных таблицах MS Office Excel с использованием языка VBA.

```
Dim a(50, 50), c(50), i, j, s, r, m As Integer
Sub касссир()
r = 10: m = 12: Randomize Timer
'задание начальной конфигурации
For i = 1 To r
   For j = 1 To m
```

```
        k = Int(Rnd * 2)
        If k < 1 Then a(i, j) = 1 Else a(i, j) = 0
Next j, i
Call оформление
For i = 1 To r
    For j = 1 To m
        If a(i, j) = 0 Then
            ThisWorkbook.Worksheets(1).Cells(i + 3, j + 3).Select
            Selection.Interior.ColorIndex = 0
        Else
            ThisWorkbook.Worksheets(1).Cells(i + 3, j + 3).Select
            Selection.Interior.ColorIndex = 16
        End If
Next j, i
'задание стоимости билетов
For i = 3 To r - 2
    c(i) = 200
Next i
c(1) = 100: c(2) = 100: c(r - 1) = 100: c(r) = 100
'диалог с покупателем
ThisWorkbook.Activate
Call проверка
While (MsgBox("будете покупать билеты?", vbYesNo) = vbYes) And (s < r * m)
    Call проверка
    If s = 120 Then
        MsgBox ("Извините, все билеты проданы ")
        End
    End If
    Do
    kol = InputBox("введите необходимое количество билетов (1,2,3)")
    Loop While (kol <= 0) Or (kol > 3)
    Do
    n = InputBox("в каком ряду вы бы хотели сидеть?")
    Loop While (n <= 0) Or (n > 12)
    i = n:    If kol = 1 Then Call место1
    If kol = 2 Then Call место2
    If kol = 3 Then Call место3
Wend
```

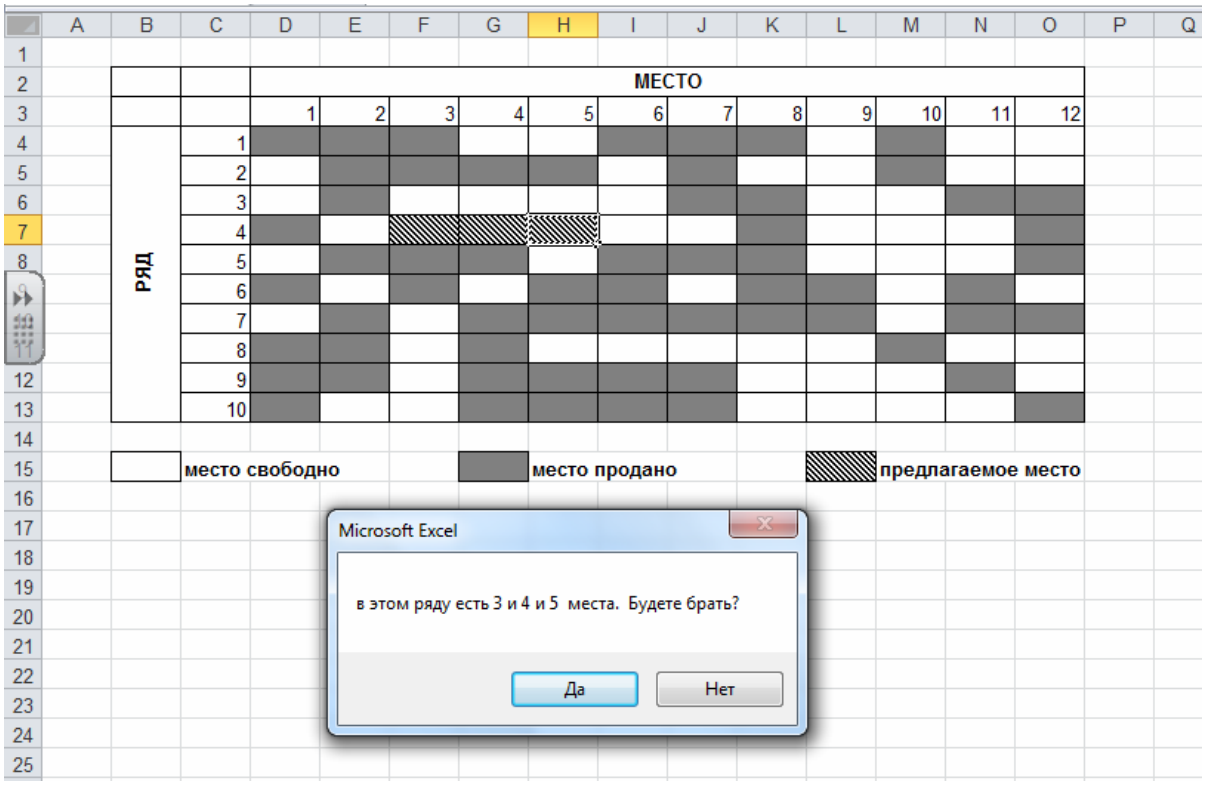
```
End Sub
Sub проверка()
s = 0
For q = 1 To r
   For w = 1 To m
      s = s + a(q, w)
Next w, q
End Sub
Sub оформление()
Range("a1:z30").Delete: Rows.RowHeight = 15: Columns.ColumnWidth = 6
Range("b4:b13").Select
With Selection
   .Merge
   .Orientation = 90
   .Font.Bold = True
   .HorizontalAlignment = xlCenter
   .VerticalAlignment = xlCenter
End With
Selection = "РЯД"
For q = 1 To 10
   Cells(q + 3, 3) = q
Next q
Range("d2:o2").Select
With Selection
   .Merge
   .Font.Bold = True
   .HorizontalAlignment = xlCenter
   .VerticalAlignment = xlCenter
End With
Selection = "МЕСТО"
ThisWorkbook.Worksheets(1).Cells(15, 2).Select
Selection.Interior.ColorIndex = 0: Selection.Borders.LineStyle = 1
Range("c15:e15").Select:    Selection.Merge:   Selection.Font.Bold = True
Selection = "место свободно"
ThisWorkbook.Worksheets(1).Cells(15, 7).Select
Selection.Interior.ColorIndex = 16:      Selection.Borders.LineStyle = 1
Range("h15:j15").Select:    Selection.Merge:    Selection.Font.Bold = True
Selection = "место продано"
```

```
ThisWorkbook.Worksheets(1).Cells(15, 12).Select
Selection.Interior.Pattern = xlDown:   Selection.Borders.LineStyle = 1
Range("m15:o15").Select:     Selection.Merge:    Selection.Font.Bold = True
Selection = "предлагаемое место"
For q = 1 To 12
   Cells(3, q + 3) = q
Next q
Worksheets(1).Range("b2:o13").Borders.LineStyle = xlContinuous
End Sub
Sub место1()
fl = 0
 For j = 1 To 12
       If a(i, j) = 0 Then
          fl = 1
          Cells(i + 3, j + 3).Select:   Selection.Interior.Pattern = xlDown
          If MsgBox("в этом ряду есть " & j & " место. Будете брать?", vbYesNo) =
vbYes Then
             a(i, j) = 1
             Cells(i + 3, j + 3).Select
             Selection.Interior.Pattern = xlSolid:  Selection.Interior.ColorIndex = 16
             MsgBox ("С Вас " & c(i) & " руб."):   GoTo 1
          Else
             Cells(i + 3, j + 3).Select:  Selection.Interior.ColorIndex = 0
          End If
       End If
    Next j
  If fl = 1 Then
       MsgBox ("Извините, больше ничего предложить не могу ")
  Else
       MsgBox ("В этом ряду нет необходимого Вам количества мест")
    End If
1: End Sub
Sub место2()
fl = 0
For j = 1 To 11
       If (a(i, j) = 0) And (a(i, j + 1) = 0) Then
          fl = 1
          Cells(i + 3, j + 3).Select: Selection.Interior.Pattern = xlDown
```

```
        Cells(i + 3, j + 4).Select:  Selection.Interior.Pattern = xlDown
        If MsgBox("в этом ряду есть " & j & " è " & j + 1 & _
        "  места. Будете брать?", vbYesNo) = vbYes Then
           a(i, j) = 1:  a(i, j + 1) = 1
           Cells(i + 3, j + 3).Select
           Selection.Interior.Pattern = xlSolid:   Selection.Interior.ColorIndex = 16
           Cells(i + 3, j + 4).Select
           Selection.Interior.Pattern = xlSolid:   Selection.Interior.ColorIndex = 16
           MsgBox ("С Вас " & 2 * c(i) & " руб."):    GoTo 2
        Else
           Cells(i + 3, j + 3).Select:  Selection.Interior.ColorIndex = 0
           Cells(i + 3, j + 4).Select:  Selection.Interior.ColorIndex = 0
        End If
     End If
  Next j
  If fl = 1 Then
     MsgBox ("Извините, больше ничего предложить не могу")
  Else
     MsgBox ("В этом ряду нет необходимого Вам количества мест")
  End If
2: End Sub
Sub место3()
fl = 0
For j = 1 To 10
     If (a(i, j) = 0) And (a(i, j + 1) = 0) And (a(i, j + 2) = 0) Then
        fl = 1
        Cells(i + 3, j + 3).Select:  Selection.Interior.Pattern = xlDown
        Cells(i + 3, j + 4).Select:  Selection.Interior.Pattern = xlDown
        Cells(i + 3, j + 5).Select:  Selection.Interior.Pattern = xlDown
        If MsgBox("В этом ряду есть " & j & " è " & j + 1 & " è " & j + 2 & _
        "  места. Будете брать?", vbYesNo) = vbYes Then
           a(i, j) = 1:   a(i, j + 1) = 1:   a(i, j + 2) = 1
           Cells(i + 3, j + 3).Select
           Selection.Interior.Pattern = xlSolid:   Selection.Interior.ColorIndex = 16
           Cells(i + 3, j + 4).Select
           Selection.Interior.Pattern = xlSolid:   Selection.Interior.ColorIndex = 16
           Cells(i + 3, j + 5).Select
           Selection.Interior.Pattern = xlSolid:   Selection.Interior.ColorIndex = 16
```

```
            MsgBox ("С Вас " & 3 * c(i) & " руб."):    GoTo 3
        Else
            Cells(i + 3, j + 3).Select:   Selection.Interior.ColorIndex = 0
            Cells(i + 3, j + 4).Select:   Selection.Interior.ColorIndex = 0
            Cells(i + 3, j + 5).Select:   Selection.Interior.ColorIndex = 0
        End If
      End If
   Next j
   If fl = 1 Then
      MsgBox ("Извините, больше ничего предложить не могу")
   Else
      MsgBox ("В этом ряду нет необходимого Вам количества мест")
   End If
3: End Sub
```

Вид окна MS Excel в промежуточный момент исполнения программы



БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. **Андреева, Е. В.** Математические основы информатики. Элективный курс: метод. пособие / Е.В. Андреева, Л.Л. Босова, И. Н. Фалина. – М. : БИНОМ; Лаборатория знаний, 2007. – 312 с. – ISBN 5-94774-138-5.

2. **Босова, Л. Л.** Информатика и ИКТ. 5 – 7 классы: метод. пособие / Л. Л. Босова, А. Ю. Босова. – М. : БИНОМ; Лаборатория знаний, 2011. – 479 с. – ISBN 978-5-9963-0457-8.

3. **Гейн, А. Г.** Информатика: учеб. пособие для 10 – 11 кл. общеобразоват. учреждений / А. Г. Гейн, А. И. Сенокосов, И. А. Юнерман. – М. : Просвещение, 2011. – 225 с. – ISBN 5-09-010486-7.

4. **Семакин, И. Г.** Информатика. Базовый курс. 7 – 9 классы / И. Г. Семакин [и др.]. – М. : Лаборатория базовых знаний, 2003. – 390 с. – ISBN 5-94774-082-6.

5. **Информатика и ИКТ.** Задачник-практикум. В 2 т. Т. 1 / Л. А. Залогова [и др.]; под ред. И. Г. Семакина, Е. К. Хеннера. – М. : БИНОМ; Лаборатория знаний, 2011. – 309 с. – ISBN 978-5-9963-0476-9 (Т. 1), ISBN 978-5-9963-0475-2.

6. **Информатика и ИКТ.** Задачник-практикум. В 2 т. Т. 2 / Л. А. Залогова [и др.]; под ред. И. Г. Семакина, Е. К. Хеннера. – М. : БИНОМ; Лаборатория знаний, 2011. – 294 с. –ISBN 978-5-9963-0477-9 (Т. 2), ISBN 978-5-9963-0475-2.

7. **Информатика и ИКТ.** Практикум 8 – 9 классы / под ред. проф. Н. В Макаровой. – СПб. : Питер, 2008. – 384 с. – ISBN 978-5-469-01622-9.

8. **Макарова, Н. В.** Информатика: учебник для 7-9 кл. Ч. 1 (Теория) / Н. В. Макарова [и др.]. – СПб. : Питер, 2012. – 432 с. – ISBN 978-5-496-00010-9.

9. **Касаткин, В. Н.** Информация, алгоритмы, ЭВМ: пособие для учителя / В. Н. Касаткин. – М. : Просвещение, 1991. – 192 с. – ISBN 5-09-003398-6.

10. **Кушниренко, А. Г.** Информатика. 7 – 9 классы : учеб. для общеобразоват. учеб. заведений / А. Г. Кушниренко, Г. В. Лебедев, Я. Н. Зайдельман. – М. : Дрофа, 2000. – 336 с. – ISBN 5-7107-3109-9.

11. **Лапчик, М. П.** Теория и методика обучения информатике / М. П. Лапчик [и др.]. – М. : Academia, 2008. – 592 с. – ISBN 978-5-7695-4748-5.

12. **Николаева, И. В.** Теория и методика обучения информатике. Содержательная линия «Алгоритмизация и программирование»: учеб. пособие / И. В. Николаева, Е. П. Давлетярова. – Владимир : Изд-во ВлГУ, 2012. – 225 с. – ISBN 978-8-9984-0250-0.

13. **Николаева, И. В.** Численные методы и компьютерное моделирование / И. В. Николаева. – Владимир : ВГПУ, 2005. – 62 с.

14. **Основы информатики и вычислительной техники:** проб. учебник для 10 – 11 кл. сред. шк. / А. Г. Гейн [и др.]. – М. : Просвещение, 1992. – 254 с. – ISBN 5-09-003852-X.

15. **Семакин, И. Г.** Информатика и ИКТ. Базовый уровень: практ. для 10 – 11 кл. / И. Г. Семакин, Е. К. Хеннер, Т. Ю. Шеина. – М. : БИНОМ; Лаборатория знаний, 2011. – 120 с. – ISBN 978-5-9963-0596-4.

16. **Угринович, Н. Д.** Информатика и ИКТ. Профильный уровень: учебник для 11 кл. / Н. Д. Угринович. – М. : БИНОМ; Лаборатория знаний, 2010. – 308 с. – ISBN 978-5-99663-0328-1.

17. **Угринович, Н. Д.** Информатика и ИКТ. Профильный уровень: учебник для 10 кл. / Н. Д. Угринович. – М. : БИНОМ; Лаборатория знаний, 2008. – 387с. – ISBN 978-5-94774-828-4.

18. **Угринович, Н. Д.** Информатика и ИКТ. Базовый уровень: учебник для 11 кл. / Н. Д. Угринович. – М. : БИНОМ; Лаборатория знаний, 2010. – 187 с. – ISBN 978-5-9963-0244-4.

19. **Угринович, Н. Д.** Информатика и ИКТ. Базовый уровень: учебник для 10 кл. / Н. Д. Угринович. – М. : БИНОМ; Лаборатория знаний, 2010. – 212 с. – ISBN 978-5-9963-0243-7.

Учебное издание

НИКОЛАЕВА Ирина Васильевна
МАРТЫНОВА Анна Александровна

ТЕОРИЯ И МЕТОДИКА ОБУЧЕНИЯ ИНФОРМАТИКЕ
Содержательная линия
«Моделирование и формализация»

Учебное пособие

Подписано в печать 18.03.13.
Формат 60x84/16. Усл. печ. л. 8,37. Тираж 300 экз.
Заказ
Издательство
Владимирского государственного университета имени
Александра Григорьевича и Николая Григорьевича Столетовых
600000, Владимир, ул. Горького, 87.

144