№ 40 (377) 23–31 октября 2002

ПОДПИСКА: (095) 249-47-58

Еженедельная газета Издательского дома «ПЕРВОЕ СЕНТЯБРЯ»

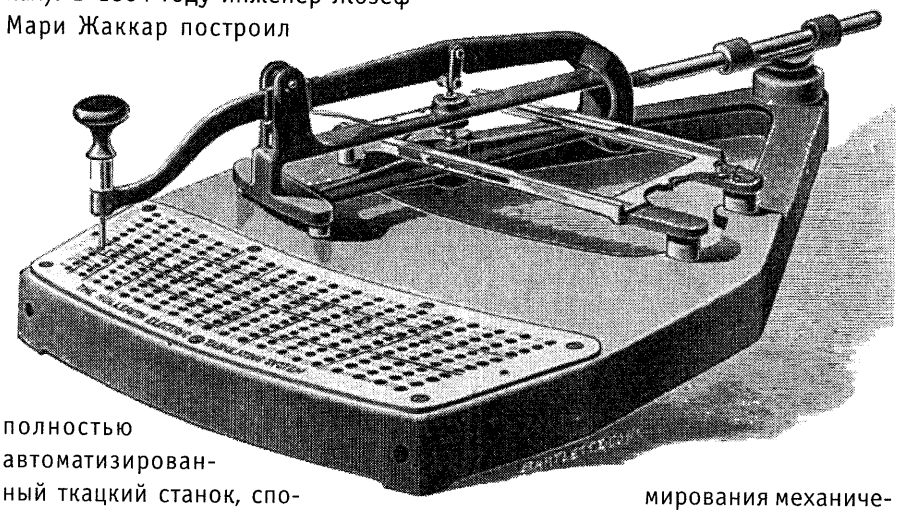
ИНФОРМАТИК

Перфокарты и перфоленты

105 лет назад, в 1897 году, состоялась первая Всероссийская перепись населения. Для обработки ее данных использовались машины основоположника счетно-перфорационной техники Германа Холлерита, который семью годами раньше впервые применил перфокарты для вычислений — в своей суммирующей машине (табуляторе).

Многие из устройств, которые первыми использовались в компьютерах для ввода и вывода данных, были изобретены еще до появления компьютеров, а потом их просто приспособили к этим машинам. Главный недостаток подобных приспособлений заключался в низком быстродействии [1]. Какая польза от того, что компьютер может выполнить тысячи операций в секунду, если для ввода и вывода данных требуются минуты, часы и даже дни?

Наиболее важное значение среди приспособленных таким образом устройств ввода информации имели те, которые работали с перфокартами, применявшимися еще в XVIII веке для автоматизации производства шелковых тканей (на французских фабриках). В 1804 году инженер Жозеф Мари Жаккар построил



полностью автоматизированный ткацкий станок, способный воспроизводить сложнейшие узоры. Работа станка программировалась при помощи перфокарт, каждая из которых управляла одним ходом челнока. Для замены рисунка оператору нужно было просто заменить один набор перфокарт другим.

Несколько позже английский математик Чарльз Бэббидж и его помощница Августа Ада Байрон (графиня Лавлейс) решили воспользоваться перфокартами Жаккара для программирования механического компьютера, названного Аналитической машиной [1—4]. Эту машину удалось построить лишь частично, однако система ввода данных с перфокарт нашла впоследствии широкое применение в различных устройствах обработки информации.

В 1890 году американец, сын немецких эмигрантов, Герман Холлерит сделал электромеханическую счетную машину (табулятор), в которой перфокарты использовались для обработки данных переписи населения США. Табулятор Холлерита вышел победителем в соревновании с несколькими другими системами, и с изобретателем был заключен контракт на проведение переписи 1890 года. Машина Холлерита действовала настолько быстро, что предварительные подсчеты удалось выполнить за шесть недель (полный статистический анализ занял два с половиной года). Для обработки результатов переписи населения потребовалось примерно в три раза меньше времени, чем для обработки результатов предыдущей переписи. Как заметил один из современников Холлерита, результаты были получены "тогда, когда они еще представляли интерес" [1].

Машины Холлерита применялись для обработки данных переписи населения также в Австрии, Канаде, Норвегии и некоторых других странах.

Холлерит получил несколько премий, а также звание профессора Колумбийского университета (в котором он в свое время учился).

Вверху: перфокарта Г.Холлерита.

В центре: ручной механический перфоратор Г.Холлерита.

Окончание читайте на с. 32

Страницы повышения квалификации 3–15

В.А. Козлова, М.А. Плаксин. ЭТО МЫ НЕ ПРОХОДИЛИ?.. ОБЗОР МЕТОДИК ПРЕПОДАВАНИЯ И ПРОГРАММНОЙ ПОДДЕРЖКИ КУРСА ИНФОРМАТИКИ В 1—7-х КЛАССАХ
ЛЕКЦИЯ № 4. ИНФОРМАТИКА В НАЧАЛЬНОЙ ШКОЛЕ С МАШИННОЙ ПОДДЕРЖКОЙ

План публикаций лекций см. на с. 3.

Знаете ли вы, что... эффективность обучения можно существенно повысить за счет использования различных каналов получения ребенком информации?

В очередной лекции рассматриваются четыре курса: курс И.Б. Мыловой "Информатика в младших классах. Машина Поста"; курс А.А. Дуванова "Азы информатики"; курс А.Л. Семенова и др. "Информатика для начальной школы" и начальный курс информатики Н.В. Макаровой и др. Все перечисленные курсы имеют компьютерную поддержку, что и послужило критерием их отбора для рассмотрения в данной лекции. В ряде случаев компьютерная поддержка создавалась специально для конкретного курса, иногда авторы предлагают использовать известные программные продукты других производителей. Авторы лекции стараются прежде всего обратить внимание на ключевые особенности рассматриваемых курсов и на проблемные вопросы. В частности, отмечена крайне любопытная попытка И.Б. Мыловой использовать различные каналы получения ребенком информации: визуальный ("глазами"), аудиальный ("ушами") и кинестетический ("руками"). Не прошли авторы лекции и мимо вопроса, который А.А. Дуванов относил к своему курсу, но который, вообще говоря, является достаточно общим: курсы-конструкторы — это благо или источник потенциальных проблем?

А.П. Шестаков. КОМПЬЮТЕРНОЕ МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ
ЛЕКЦИЯ № 4. МОДЕЛИРОВАНИЕ В ЕСТЕСТВЕННЫХ НАУКАХ. НЕСКОЛЬКО КЛАССИЧЕСКИХ МОДЕЛЕЙ ЭКОЛОГИИ

План публикаций лекций см. на с. 9.

Знаете ли вы, что... экологические модели взаимодействия популяций просты, наглядны и являются источником большого количества красивых задач, которые можно решать в различных программных средах?

В лекции рассматриваются некоторые классические модели "старой" экологии, как науки о взаимодействии популяций. Это обусловлено тем, что, с одной стороны, эти модели достаточно просты, наглядны и хорошо изучены, с другой — в познавательном плане модели взаимодействия популяций достаточно интересны для школьников. Исследование моделей "современной" экологии, например связанных с загрязнением окружающей среды, требует привлечения весьма сложного математического аппарата, сами модели тоже не вполне устоялись, да и для их визуализации обычно требуется приложить значительные усилия.

На стенд в кабинете информатики 16–17

УЧЕБНО-РАЗВЛЕКАТЕЛЬНАЯ СТЕНГАЗЕТА "ИНФОРМАТИК@" № 4

Стенгазеты публикуются в рубрике "На стенд в кабинете информатики" через номер.

Знаете ли вы, что... в программах популярного пакета Microsoft Office спрятаны игрушки?

Из очередного выпуска стенгазеты, помимо информации о компьютерных игрушках, спрятанных разработчиками в программах MS Word и MS Excel, ребята узнают о первом алгоритме игры в шахматы, который принадлежит Алану Тьюрингу, и о первой шахматной программе, написанной для компьютера MANIAC I.

Предлагаю коллегам 18–27

Е.А. Еремин. MMIX — УЧЕБНЫЙ RISC-ПРОЦЕССОР НОВОГО ТЫСЯЧЕЛЕТИЯ ОТ ДОНАЛЬДА КНУТА

Знаете ли вы, что... Дональд Кнут "проапгрейдил" свой легендарный компьютер MIX?

"Многие читатели, без сомнения, думают: "Почему Кнут заменил MIX другой машиной, вместо того чтобы просто перейти к языку программирования высокого уровня? Едва ли кто-то использует сейчас ассемблер".
Эти люди имеют право на свое мнение, и им не стоит утруждать себя чтением частей моей книги, связанных с машинным языком. Тем не менее причины применения машинного языка, которые я приводил в предисловии к первому тому, написанному в начале 60-х, остаются в силе и сегодня..." (Дональд Кнут)
Что это за причины? И зачем Дональду Кнуту все-таки понадобился новый компьютер? И, наконец, что он собой представляет, этот компьютер, названный MMIX?

Задачи 28–31

Е.В. Андреева, Ю.Е. Егоров. ВЫЧИСЛИТЕЛЬНАЯ ГЕОМЕТРИЯ НА ПЛОСКОСТИ

Продолжение. Начало см. в № 39/2002.

Знаете ли вы, что... основу большей части геометрических задач, которые встречаются в задачниках по информатике и нередко предлагаются на олимпиадах разных уровней, составляет не такое большое количество элементарных подзадач?

Цель настоящей статьи — показать подходы к решению геометрических задач на плоскости, которые позволяют достаточно быстро и максимально просто получать решения большинства элементарных подзадач.

Это мы не проходили?..

Обзор методик преподавания и программной поддержки курса информатики в 1–7-х классах

В.А. Козлова, М.А. Плаксин,
г. Пермь

ЛЕКЦИЯ 4
ИНФОРМАТИКА В НАЧАЛЬНОЙ ШКОЛЕ С МАШИННОЙ ПОДДЕРЖКОЙ

План лекции

4.1. Курс И.Б. Мыловой "Информатика в младших классах. Машина Поста"
4.2. Курс А.А. Дуванова "Азы информатики" и "Роботландия.RU"
4.3. Курс А.Л. Семенова и др. "Информатика для начальной школы", "Клавиатор" и "ПервоЛого"
4.4. Начальный курс информатики Н.В. Макаровой и "ЛогоМиры"
4.5. Координаты разработчиков и поставщиков программно-методических средств
Использованные источники информации

Эту лекцию мы посвятили авторским курсам для начальной школы, предполагающим проведение компьютеризованных уроков информатики.

Первые два курса — И.Б. Мыловой и А.А. Дуванова — можно с полным правом назвать *программно-методическими комплексами*, поскольку их программная поддержка разрабатывалась с учетом авторских концепций пропедевтического курса информатики для реализации конкретных педагогических задач.

Курс А.Л. Семенова, по утверждению самого автора, "не обязательно связан с компьютером, его можно изучать, даже если в школе нет ни одного компьютера". Поэтому его опора на "ПервоЛого" носит весьма условный характер, хотя и накладывает некоторый отпечаток на содержание и построение курса.

Напротив, в курсе Н.В. Макаровой обучение ориентировано в первую очередь на освоение Windows-технологий и работу в среде программирования "ЛогоМиры". Поэтому он является скорее курсом компьютерных информационных технологий с элементами пропедевтики понятий информатики, таких, как "информация", "алгоритм", "виды алгоритмов", "моделирование". Впрочем, автор в рекомендациях учителю непосредственно представляет этот курс как начальный, подготовительный к курсу "Информатика. 6—7-е классы", определяя его целью "формирование базиса компьютерной грамотности учащегося".

План публикаций лекций курса "Это мы не проходили?.. Обзор методик преподавания и программной поддержки курса информатики в 1—7-х классах" на "Страницах повышения квалификации" в первом полугодии 2002/2003 учебного года

Номер лекции	Номер газеты
1	34/2002
2	36/2002
3	38/2002
4	40/2002
5	42/2002
6	44/2002
7	46/2002
8	48/2002

Данный курс рассчитан на два полугодия. Полный график публикации лекций курса см. в следующих лекциях.

4.1. Курс И.Б. Мыловой "Информатика в младших классах. Машина Поста"

Учебный курс "Информатика в младших классах. Машина Поста" и методика его преподавания разработаны Ириной Борисовной Мыловой (сейчас — заведующая Центром информатизации образования Санкт-Петербургского университета педагогического мастерства). Целью курса является знакомство учащихся с основами алгоритмики (понятия "алгоритм", "команда", "программа", составление алгоритмов разных видов) и развитие логического мышления детей.

Включение в концепцию начального курса информатики развивающих целей автор объясняет следующим образом. Согласно теории Ж.Пиаже, в 7—8 лет ребенок находится на стадии "конкретных операций", а на стадию "формальных операций" переходит только годам к 14. Задача, которую поставила перед собой И.Б. Мылова: стимулировать более быстрый переход ребенка со стадии "конкретных операций" на стадию "формальных", а для этого развивать его умственные и творческие способности. В результате "Информатика в младших классах" включила в себя очень мощные компоненты "развивающего обучения".

Курс рассчитан на два года, 2—3-й классы. По рекомендации автора, начинать курс надо тогда, когда дети уже умеют читать. Годовой курс состоит из 16 уроков по 2 часа каждый, которые проводятся раз в две недели. "Парная" организация уроков связана, видимо, с проведением занятий на компьютере, хотя автор допускает как безмашинный вариант курса, так и проведение одного часового урока в неделю.

В 1994 г. частной школой "Эпиграф" (Санкт-Петербург) изданы методические пособия по курсу: две книги для учителя (по одной на год) [1, 2], "пресс-тетрадь" (тетрадь-учебник, тетрадь-раскраска) для ученика и пресс-тетрадь с дополнительным материалом на два года обучения. Книги для учителя содержат подробнейшие поурочные планы, применение которых не требует от учителя каких-либо усилий.

Информацию об этом курсе один из авторов обзора получил непосредственно от И.Б. Мыловой достаточно давно, в 1996 году. С тех пор нам не встречались публикации по этому курсу. Однако мы сочли необходимым включить его в обзор сразу по нескольким причинам:

1. Как нам кажется, именно эта авторская разработка, вместе с "Уроками развития" В.В. Дубininой, послужила основой для роста целой ветви "развивающей информатики" в предметной области "информатика". Но это не просто развивающий курс, это в первую очередь курс алгоритмики, а уже затем — развивающего обучения.
2. В отличие от многих, последовавших за ним, курс И.Б. Мыловой обладает концептуальным единством содержания, методики преподавания и компьютерной практики.
3. Крайне любопытной (и крайне редко встречающейся в методических разработках) является попытка автора методики сознательно использовать различные каналы получения ребенком информации: визуальный ("глазами"), аудиальный ("ушами") и кинестетический ("руками").

Компьютерная поддержка курса

В качестве компьютерной поддержки используется модель машины Поста (отсюда — название курса). Специально для курса разработана под ДОС программа POST (автор — В.А. Духнякова). Однако машина Поста используется на уроках и в виде предметной модели.

Исполнитель "Машина Поста" имеет разделенную на ячейки ленту и считывающую "головку". Он умеет передвигать головку вдоль ленты (каждый раз ровно на одну ячейку), класть в ячейку или забирать из ячейки кружочки, проверять, пуста ли ячейка. В данной программной реализации в роли головки выступает обычное ведро, и на ленту из него выкладываются не кружки, а яблоки.

В качестве учебного исполнителя машина Поста была выбрана, исходя из соображений простоты. В чем-то она похожа на Черепашку, Муравья, Кенгуренка и прочих ползающе-прыгающих исполнителей. Но они все двигаются по плоскости, в двухмерном пространстве, а головка машины Поста — в одномерном. Естественно, уследить за ней ребенку много проще.

Методика преподавания

Материал, планируемый на каждый урок, достаточно объемен. Связано это с "парной" организацией занятий. При этом И.Б. Мылова успешно справляется с весьма сложной задачей: удержать внимание малышей в течение целой пары. Достигается это объединением в рамках одного урока весьма различающихся частей.

Рассмотрим в качестве примера план первых четырех уроков курса.

Урок 1

Цели: На игровом материале познакомить учащихся с понятиями "команда" и "программа". Активизировать познавательные процессы. Познакомить с компьютерным классом.

Ход урока

1-й этап. Введение. История развития счетных устройств.

2-й этап. Физкультминутка. Учитель командует, используя стихотворение-считалку. Дети должны выделить отдельные команды и сосчитать их количество. Цель: освоить понятие команды и программы (совокупности команд).

3-й этап. Решение задач. Определить последовательность действий (команд), которые необходимо выполнить капитанам кораблей, чтобы разойтись в узком проливе.

4-й этап. Игра "Волшебные палочки": сложить из спичек фигуру, затем убрать или переместить несколько спичек так, чтобы получилась другая фигура.

Урок 2

Цели: Закрепление понятий "команда", "программа", выработка навыков позиционных преобразований. Знакомство с машиной Поста. Активизация познавательных процессов.

Ход урока

1-й этап. Повторение. Игра "Волшебные палочки".

2-й этап. Физкультминутка, проводится аналогично первому уроку.

3-й этап. Игра "Поиграем в роботов". Знакомство с предметной моделью машины Поста (в качестве моделей служат пластинка с отверстиями и кружки).

4-й этап. Работа на компьютерах.

Урок 3

Цели: Обучение анализу исходных данных, развитие логического мышления, обучение умению рассуждать и обосновывать свои рассуждения. Обучение выполнению программ на модели машины Поста и на компьютере.

Ход урока

1-й этап. Разминка. Решение стихотворных вычислительных задач. Поиск закономерности в расположении картинок и определение недостающей картинки.

2-й этап. Решение задач путем логических рассуждений с оформлением в виде таблицы "Условия/Суждения/Выводы".

3-й этап. Физкультминутка. Сколько команд было отдано? Какие это команды? Повторялись ли они? (Первый шаг к введению циклов.)

4-й этап. Практическая работа на предметной модели машины Поста.

5-й этап. Работа на компьютерах.

Урок 4

Цели: активизация мыслительных процессов, обучение умению рассуждать и доказывать, обосновывать и делать выводы. Рассмотрение случаев "поломок" машины Поста.

Ход урока

1-й этап. Разминка. Стихотворные вычислительные задачи. Задачи на развитие внимания и наблюдательности (найди лишнюю фигуру, лишнее слово).

2-й этап. Решение задач путем логических рассуждений.

3-й этап. Физкультминутка. Сколько команд было отдано? Какие команды повторялись?

4-й этап. Работа на предметной модели машины Поста.

5-й этап. Работа на компьютерах.

На каждом уроке автор методики задействует все три канала для восприятия информации: визуальный, аудиальный и кинестетический. Разные люди имеют различную степень восприимчивости к информации, идущей по тому или иному каналу. Существуют специальные психологические тесты, позволяющие разделить "визуалов", "аудиалов" и "кинестетиков". Внутри "кинестетиков" проводят еще одну градацию: кто-то лучше воспринимает информацию, получаемую "от пальцев", а кому-то надо задействовать мышцы всего тела.

И.Б. Мылова в своей методике постаралась сознательно учесть это различие. В течение урока одну и ту же информацию она преподносит несколько раз, но так, чтобы каждый раз информация шла по новому каналу. Для достижения образовательной цели использовано все: стихи, рисунки, спички (работа "пальчиками"). Даже физкультминутка имеет вполне конкретную методическую нагрузку. На уроках Мыловой физкультура — это способ задействовать кинестетический канал передачи информации для тех детей, кто лучше воспринимает "двигательную" информацию мышцами всего тела.

4.2. Курс А.А. Дуванова "Азы информатики" и "Роботландия.RU"

Новая форма работы авторского коллектива "Роботландии" — Роботландский сетевой университет (RU) [3], существующий с 1997 года, — позволяет разработчикам постоянно ощущать потребности преподавателей информатики и оперативно реагировать на запросы этой аудитории. Вслед за первыми курсами, основанными на "досовских" программных продуктах "Роботландии", в рамках RU появились курсы, поддержанные так называемыми "гипертекстовыми учебниками".

Гипертекстовые учебники Роботландского университета представляют собой локальные гипертекстовые приложения, работающие под управлением браузера. Осваивая приемы работы с таким учебником, ребенок знакомится фактически с современными технологиями Интернета.

В одном программном продукте, благодаря технологиям web-программирования, оказывается возможным соединить:

- учебник для учащегося,
- его программную поддержку в виде многочисленных тренажеров, исполнителей, испытателей и конструкторов,
- блоки контроля эффективности обучения в виде электронных тестов по пройденному материалу,
- поурочные комментарии для учителя.

Проект А.А. Дуванова под названием "Азы информатики", реализуемый с прошлого учебного года в рамках RU, представляет собой такой гипертекстовый учебник, то есть фактически — комплексную среду для организации учебного процесса.

"Азы информатики" — это гипертекстовый интерактивный курс информатики для младших школьников. Сохраняя методические идеи Роботландии, новый курс предлагает школьнику и педагогу современные средства для реализации педагогических задач, а именно:

- Windows-технологии для работы с программами и для обработки текста,
- Интернет-технологии для взаимодействия с интерактивным гипертекстовым приложением, каковым и являются "Азы информатики".

В курсе автор, А.А. Дуванов, планирует представить 8 тем, каждая из которых будет поддержана отдельной гипертекстовой книгой:

1. Знакомство с компьютером.
2. В мире информации.
3. Пишем на компьютере.
4. Рисуем на компьютере.
5. Выходим в Интернет.
6. Алгоритмические этюды.
7. "Черные ящики".
8. Исполнители.

Сейчас реализованы три первые книги, работа над остальными продолжается. "Бумажные версии" гипертекстовых книг "Азов" по мере их появления печатает "Информатика" [4—6]. Демо-версию учебника можно скачать с сайта Роботландии [3].

Чтобы читатели почувствовали разницу между "Азами информатики" и старой доброй "Роботландией", приведем здесь хотя бы перечень программных модулей (исполнителей) для поддержки только первой книги "Знакомство с компьютером":

1. *Бука* — выполняет алгоритм подсчета заданной буквы в тексте. Служит наглядно-действенной иллюстрацией понятия "алгоритм".
2. *Извозчик* — в игровой форме помогает освоить действия с мышью: щелчок, двойной щелчок, перетаскивание.
3. *Испытатели для работы с курсорами разной формы и для изменения размеров экранного объекта.*
4. *Пиктограмма* — способствует формированию представления об одноименных объектах и их назначении, а также развитию способностей к сравнению и анализу.
5. *Испытатели для выполнения действий с окном:* перетаскивание, сворачивание, распахивание, протяжка содержимого окна.
6. *Испытатели для работы с компьютерными меню разных видов:* текстовый список, пиктографическое меню, меню на обычных и радиокнопках, меню на флажках, разворачивающийся список, иерархическое меню.
7. *Привет* — простейший тренажер для ввода текста, сочиняющий забавные сказки. Служит наглядно-действенной иллюстрацией конкретного алгоритма.
8. *Сочинитель* — работая по конкретному алгоритму, способствует самостоятельному осмыслению ребенком проблемы "Умеет ли компьютер думать?".

В рамках той же первой книги, знакомясь с устройством и назначением компьютера, с основными приемами управления компьютером, дети осваивают понятия информатики: "механизация" и "автоматизация", "меню" и "интерфейс", "иерархия", "алгоритм" и "программа". К этим понятиям автор возвращается и в других книгах курса, постоянно связывая их с компьютерной практикой. В результате освоение компьютерных технологий отходит на второй план, занимая подобающее им место средства, а не цели обучения.

"Азы информатики" должны послужить основой для нового программно-методического продукта под рабочим названием "Роботландия.RU". Из общения с А.А. Дувановым у нас сложилось представление о будущей "Роботландии.RU", как о комплексе, включающем в себя:

- множество исполнителей, сгруппированных по темам, определенным в "Азах", как реализованных средствами web-программирования, так и существующих вне браузера (текстовые редакторы, к примеру);
- задачник по всем темам;
- базу контрольных вопросов по всем темам для построения зачетов;
- конструктор уроков, позволяющий собрать из вышеперечисленных элементов урок по усмотрению учителя.

В такой комплекс "Азы информатики" войдут в качестве примера готового учебного курса, по усмотрению автора построенного из тех же элементов и подкрепленного книгой для учащегося и методическими рекомендациями учителю.

Потребность в адаптации учебного курса к реальным условиям возникает практически у каждого преподавателя: к примеру, по-разному приходится строить урок в сильном и слабом классах или в классах с разной специализацией. Да и количество учебных часов, отводимых под изучение информатики, может существенно отличаться в разных школах. Поэтому возможная вариативность учебного курса была бы большим плюсом программного комплекса.

Отдавая должное идее построения курса-конструктора, А.А. Дуванов отмечает тем не менее проблему, которая в ней кроется:

"Движение в этом направлении будет означать движение по оси нового качества создаваемого продукта, придавая ему вариативность и мобильность, наделяя его покладистым характером. Последнее содержит в себе опасность: кое-кто может собрать из модулей такой курс, под которым Дуванов никогда не подписался бы! (Дом из кирпичей получится только у профессиональных строителей.)".

Пожелаем же не только автору успешной и скорейшей реализации всех его замыслов, но и преподавателям — высокого уровня профессионализма, необходимого для дальнейшего совершенствования курса информатики!

4.3. Курс А.Л. Семенова и др. "Информатика для начальной школы", "Клавиатор" и "ПервоЛого"

Авторский коллектив:

А.Л. Семенов, Т.А. Рудченко, О.В. Щеглова.

Вот как представлен этот курс на сайте Института новых технологий обучения (ИНТ) [7]:

"Курс А.Л. Семенова и др. "Информатика для начальной школы" интегрирует теоретическую информатику, социальную информатику и информационные технологии. Его программа рассчитана на обучение в начальной школе в объеме 34—68 часов в год (34 часа — математические основы информатики, 34 часа — информационные технологии, интегративные проекты).

Курс не обязательно связан с компьютером, его можно изучать, даже если в школе нет ни одного компьютера. Однако интеграция в курс изучения информационных технологий дает учащимся навыки использования компьютера и другие информационно-технологические навыки и информационно-социальные знания, которые могут (и должны) немедленно применяться учащимися при изучении различных предметов. Разделы теоретической информатики предпочтительно интегрировать с изучением математики, а основы слепой компьютерной машинописи в рамках курса русского языка.

Одной из важных составляющих программы являются интегративные проекты, в ходе которых происходит изучение и активное усвоение материалов курса.

В комплект учебных печатных материалов для каждого класса входят учебник, задачи, собранные в отдельные рабочие тетради на печатной основе (издания до 2000 г.) или включенные в учебник (позднее), и тетрадь проектов, а также методическое пособие — книга для учителя [9—13].

Компьютерная составляющая базируется на использовании интегрированной творческой среды "ПервоЛого" и тренажера машинописи и правописания "Клавиатор".

Курс вошел в работу, удостоенную премии президента Российской Федерации в области образования, отвечает требованиям рекомендаций ЮНЕСКО по информатике в начальном образовании. Прошел апробацию в сотнях школ Москвы и России. С 2001 года используется в рамках федерального эксперимента по 12-летней школе".

Особенности курса (по мнению авторов обзора, а не из рекламы курса на сайте ИНТ):

1. Слепой клавиатурный ввод. Уже в программе 2-го класса ставится задача освоения детьми слепого клавиатурного ввода с помощью тренажера машинописи и правописания "Клавиатор", на что отводится "от 10 до 15 часов в случае интеграции с уроками русского языка". Конечно, за отведенное время задачу в полном объеме решить невозможно. Однако практики отмечают [14], что:

- приемлемая скорость письма с помощью клавиатуры, сравнимая со скоростью письма ручкой, достигается при значительно меньших временных затратах на обучение;

- умение второклассников достаточно уверенно использовать клавиатуру позволяет успешно использовать компьютер как в формировании грамотности учащегося, так и в творчестве.

2. Интегрированность учебного процесса. Предполагаемая автором курса супервысокая степень интегрированности учебного процесса в начальных классах едва ли достижима в рядовой школе на сегодняшний день. Считается, что:

"На проектную деятельность могут быть использованы как часы информатики, так и часы других предметов. Информационные технологии изучаются в рамках проектной и другой учебной деятельности в различных предметах начальной школы. При выделении на курс информатики одного часа в неделю в кабинете информационных технологий необходимо проводить уроки, сочетая компьютерные занятия информационными технологиями с занятиями теоретической информатикой так, чтобы время работы с компьютером не превышало санитарные нормы".

Такая идеалистическая картина требует не только от учителя начальных классов совершенного владения техникой, но также и почти неограниченно свободного доступа к компьютерному классу либо наличия хотя бы одного хорошо оснащенного компьютера в каждой классной комнате. К сожалению, сегодня это напоминает утопию.

3. Отсутствие связи теоретической информатики с компьютерной практикой. Предлагаемый в курсе объем материала в разделе "Теоретическая информатика" весьма обширен и ценен, как с позиций информатики, так и с позиций математики: знакомство со структурами данных, многоаспектное применение древовидных структур, понятие объекта, его свойства и признаки, классификация и сортировка объектов, множество и его элемент, а кроме того, понятия "все" и "каждый", истинность и ложность утверждений и т.д.

Это — далеко не полный список тем, затрагиваемых в курсе. Но — в безмашинной его части, характеризуемой как "теоретическая информатика", или "математическая информатика", которую автор рекомендует прямо интегрировать с изучением математики, но на которую все же отводится 34 часа в год. Авторской компьютерной поддержки теоретической информатики не предлагается, хотя такая разработка вполне возможна даже в среде "ПервоЛого", работа в которой положена в основу изучения информационных технологий в этом же курсе.

4. Освоение компьютерных информационных технологий в среде "ПервоЛого". Интегрированная творческая среда "ПервоЛого" призвана поддержать только раздел "Информационные технологии". Выбор компьютерной среды накладывает существенный отпечаток на тематику раздела: в основе — информационные объекты (рисунок, текст, звук, простейшая гиперструктура), для создания которых и предназначена среда "ПервоЛого". Средствами же среды осуществляются интеграция отдельных объектов в единый мультимедийный проект и превращение его в web-сайт, пригодный для размещения в Интернете.

Компьютерная поддержка курса

"Клавиатор", тренажер машинописи и орфографии, разработан для создания уроков клавиатурного тренажа и тренировки правописания в процессе обучения учащихся слепому десятипальцевому методу печати с использованием различного лексического материала. Открытая архитектура *"Клавиатора"* позволяет учителю с помощью простого языка самостоятельно подготавливать упражнения как для старших школьников и взрослых, так и для первоклассников; ориентировать уроки на учащихся с разными способностями; органично встраивать обучение учащихся слепому десятипальцевому способу печати в различные варианты курсов информатики. Основной отличительной чертой данного программного продукта является встроенный в программу для учителя макроязык СЦЕНОР, позволяющий учителям и методистам создавать свои собственные сценарии уроков [15].

"ПервоЛого" — интегрированная творческая среда, предназначенная для применения в дошкольном и начальном школьном образовании. Программа управляется с помощью простого графического меню, причем смысл почти всех действий более или менее очевидно описывается соответствующей пиктограммой, поэтому работать с ней могут и не читающие дети. Запуская программу, ребенок открывает компьютерный альбом, в котором может рисовать картинки, создавать музыку и мультфильмы, управлять исполнителями-черепашками.

"ПервоЛого 2.0" обладает всеми **мультимедийными** и **интернет**-возможностями своего "старшего брата" — программы "ЛогоМиры 2.0". В частности, вы можете опубликовать свой проект в сети Интернет. Установив так называемый "web-плейер", любой посетитель вашей страницы сможет увидеть ваш проект в действии прямо в сети через браузер. Кроме того, в программу входят: графический редактор (тысячи цветов), музыкальный редактор, до сотни черепашек, 64 полноцветных формы для них, параллельные процессы, встроенный справочник — и многое другое. Таким образом, вы, ваши дети и ученики получаете мощный инструмент для создания собственных проектов, в том числе мультимедийных презентаций на любую тему" [16].

С сайта ИНТ [16] можно скачать демо-версию "ПервоЛого".

Требования к компьютеру

- 486DX (рекомендуется 66 МГц или лучше).
- 8 Мб RAM.
- Графика SVGA 256 цветов (рекомендуется 16-бит ный цвет).
- Звуковая карта Sound Blaster или совместимая.

4.4. Начальный курс информатики Н.В. Макаровой и "ЛогоМиры"

Авторский коллектив

Макарова Наталья Владимировна,
Николайчук Галина Семеновна,
Титова Юлияна Францевна,
Симонова Ирина Владимировна.

Авторы ориентируют учебный материал данного курса на 68 часов изучения информатики детьми 9—12 лет, прежде не умевшими работать на компьютере. При одном часе в неделю материал рассчитан на 2 года обучения, при двух часах его можно освоить за один год. Курс поддержан учебником [17] под редакцией Н.В. Макаровой, содержащим изложение материала, контрольные вопросы и компьютерные задания, а также задания для самостоятельной работы на компьютере.

Как уже отмечалось, данный курс является прежде всего курсом компьютерных информационных технологий, в котором акцент ставится на освоение Windows-технологий и работу в среде программирования "ЛогоМиры". Об этом свидетельствует и содержание учебника, состоящее всего из трех разделов:

1. Учимся работать на компьютере.
2. Компьютерная графика.
3. Среда программирования "ЛогоМиры".

Тем не менее авторы не забывают о пропедевтике понятий информатики, необходимых для успешного использования компьютерных технологий: "информация", "алгоритм", "виды алгоритмов", "моделирование".

В качестве отличительной особенности курса необходимо отметить тщательный подбор большого количества компьютерных заданий с учетом возрастных особенностей детей. Задания основаны на качественном литературном и привлекательном графическом материале, а главное — вполне посильны даже для учащихся начальной школы и тем более — для тинейджеров. Задания по программированию в среде "ЛогоМиры" вполне можно вписать по времени в требуемый санитарными нормами промежуток в 15—20 минут, что является свидетельством того, что курс хорошо отработан на большом количестве учащихся.

Компьютерная поддержка курса

ОС Windows 95/98.

Стандартные программы Блокнот и Paint.

"ЛогоМиры" [16] — интегрированная творческая среда, предоставляющая все те возможности, которые описаны выше для "ПервоЛого". Имеет несколько отличный интерфейс, позволяет создавать более сложные мультимедийные проекты. Ребенку, "выросшему" из "ПервоЛого", не составит труда перейти к работе в "ЛогоМирах 2.0". Более того, ребенок сможет открыть в "ЛогоМирах" проект, созданный в "ПервоЛого", и продолжить над ним работу.

С сайта ИНТ [16] можно скачать демо-версию "ЛогоМиров".

4.5. Координаты разработчиков и поставщиков программно-методических средств

"Информатика в младших классах. Машина Поста" — Мылова Ирина Борисовна, заведующая Центром информатизации образования Санкт-Петербургского университета педагогического мастерства. E-mail: *upm_inf@mail.ru*.

Распространением гипертекстовых учебников курса "Азы информатики" занимается предприятие "Роботландия". Заявки принимаются по адресу: *kurs@robotland.botik.ru*. Можно заполнить форму и на сайте "Роботландии" *http://www.botik.ru/~robot*.

Заявку на приобретение печатных материалов по курсу А.Л. Семенова вы можете оформить на сайте ИНТ: *http://www.int-edu.ru/soft/zayavka_inf.html*. Специалисты ИНТ осуществляют постоянную методическую поддержку курса, совместно с Центром информационных технологий и учебного оборудования МКО организуют семинары и консультации. Записаться на семинар, сообщить о своем опыте работы с курсом или задать вопрос авторам или методистам вы можете по адресу: *int@mtu-net.ru* или по телефону 915-13-94 (методический отдел).

Учебники по курсу информатики Н.В. Макаровой можно заказать через электронный каталог издательства "Питер": *http://shop.piter.com/display.phtml?study=107*.

Использованные источники информации

1. *Духнякова В.Л., Мылова И.Б.* Информатика в младших классах. Машина Поста. 1-й год обучения. Книга для учителей. СПб.: Частная школа "Эпиграф", 1994.

2. *Мылова И.Б.* Информатика в младших классах. Машина Поста. 2-й год обучения. Книга для учителей. СПб.: Частная школа "Эпиграф", 1994.

3. *http://www.botik.ru/~robot/ru/index.htm* — официальный сайт "Роботландии", страница Роботландского сетевого университета.

4. *Дуванов А.А.* Азы информатики. Книга 1. Знакомство с компьютером. Информатика № 1, 2/2002.

5. *Дуванов А.А.* Азы информатики. Книга 2. В мире информации. Информатика № 5—9, 11, 12, 14—20/2002.

6. *Дуванов А.А.* Азы информатики. Книга 3: Пишем на компьютере. Информатика № 31/2002.

7. *Семенов А.Л. и др.* Курс "Информатика" — *http://www.int-edu.ru/soft/inf1_3.html*

8. *Семенов А.Л., Рудченко Т.А., Щеглова О.В.* Учебная программа "Информатика 1—3". Информатика и образование № 6, 1998.

9. *Семенов А.Л., Рудченко Т.А., Щеглова О.В.* Информатика: Учебник для 1-го класса. М.: Просвещение, 2002.

10. *Семенов А.Л., Рудченко Т.А., Щеглова О.В.* Информатика: Рабочая тетрадь для 1-го класса № 1, 2. М.: Просвещение, 2002.

11. *Семенов А.Л., Рудченко Т.А., Щеглова О.В.* Информатика: Учебник-тетрадь для 2-го класса. М.: Просвещение, 2002.

12. *Семенов А.Л., Рудченко Т.А., Щеглова О.В.* Информатика: Тетрадь проектов для 2-го класса. М.: Просвещение, 2002.

13. *Семенов А.Л., Рудченко Т.А., Щеглова О.В.* Информатика-1, 2: Книга для учителя. М.: Просвещение, 2002.

14. *Муранов А.А., Мухай М.А.* Обучение первоклассников клавиатурному письму. Первые результаты. Информатика и образование № 6, 1998.

15. Клавиатор. Тренажер машинописи и орфографии. *http://www.int-edu.ru/soft/klav.html*

16. Программные продукты Лого. *http://www.int-edu.ru/logo/products.html*

17. Информатика: основы компьютерной грамоты. Начальный курс / Под ред. Н.В. Макаровой. СПб.: Питер, 2000.

Профильное обучение информатике в старших классах средней школы (10—11-е классы) на основе курса "Компьютерное математическое моделирование" (КММ)

А.П. Шестаков,
г. Пермь

ЛЕКЦИЯ 4. МОДЕЛИРОВАНИЕ В ЕСТЕСТВЕННЫХ НАУКАХ. НЕСКОЛЬКО КЛАССИЧЕСКИХ МОДЕЛЕЙ ЭКОЛОГИИ

Методические замечания

При изучении этого раздела ограничимся некоторыми классическими моделями "старой" экологии [1], как науки о взаимодействии популяций, что обусловлено следующими причинами. Во-первых, они достаточно просты и изучены, постановка их вполне очевидна и в познавательном плане интересна и полезна. Во-вторых, модели "современной" экологии, например распространения загрязнений окружающей среды, требуют использования весьма сложного математического аппарата, да и сами еще не вполне устоялись. Проблемы охраны окружающей среды чрезвычайно важны, но их обсуждение выходит за пределы нашего курса. С другой стороны, в случае профильной дифференциации по биологии и (или) экологии возможно расширение этого раздела до самостоятельного курса, как это сделано в работе И.И. Данилиной [2].

Математические модели в экологии используются практически с момента возникновения этой науки. И, хотя поведение организмов в живой природе гораздо труднее адекватно описать средствами математики, чем самые сложные физические процессы, модели помогают установить некоторые закономерности и общие тенденции развития отдельных популяций, а также сообществ. Кажется удивительным, что люди, занимающиеся живой природой, воссоздают ее в искусственной математической форме, но есть веские причины, которые стимулируют эти занятия. Вот некоторые цели создания математических моделей в классической экологии [1]:

1. Модели помогают выделить суть или объединить и выразить с помощью нескольких параметров важные разрозненные свойства большого числа уникальных наблюдений, что облегчает экологу анализ рассматриваемого процесса или проблемы.

2. Модели выступают в качестве "общего языка", с помощью которого может быть описано каждое уникальное явление, и относительные свойства таких явлений становятся более понятными.

3. Модель может служить образцом "идеального объекта" или идеализированного поведения, при сравнении с которым можно оценивать и измерять реальные объекты и процессы.

Модель считается адекватной рассматриваемому явлению только в том случае, если она выполняет одну из указанных выше функций.

Привлечение компьютеров существенно раздвинуло границы моделирования экологических процессов. С одной стороны, появилась возможность всесторонней реализации сложных математических моделей, не допускающих аналитического исследования, с другой — возникли принципиально новые направления, и прежде всего имитационное моделирование.

Переход от изучения физических моделей к экологическим представляется наиболее органичным и закономерным, поскольку речь ниже по-прежнему пойдет о динамических процессах, где важнейшей характеристикой вновь будет скорость, на этот раз — изменения численности популяций.

Разговор о моделях классической экологии начнем с рассмотрения популяций с дискретным размножением, далее плавно перейдем на популяции с непрерывным размножением. Некоторые из изучаемых моделей обсудим более детально.

В качестве источника задач можно по-прежнему использовать [3], кроме того, интересные идеи есть в [4], поэтому представленные там задачи по экологии могут быть адаптированы и для профильного курса.

План публикаций лекций курса "Компьютерное математическое моделирование" на "Страницах повышения квалификации" в первом полугодии 2002/2003 учебного года

Номер лекции	Номер газеты
1	34/2002
2	36/2002
3	38/2002
4	40/2002
5	42/2002
6	44/2002
7	46/2002
8	48/2002

Данный курс рассчитан на одно полугодие.

Содержательный материал

Введение. Некоторые основные понятия

В 1869 году немецкий естествоиспытатель Эрнст Геккель предложил составной термин "экология" ("эко" — дом, жилище, местопребывание и "логос" — наука, знание) как название ставшего самостоятельным раздела биологии. Классическая экология — наука о взаимодействии организмов и окружающей среды. Сегодня, говоря об экологии, чаще всего имеют в виду не классическую, а так

называемую "социальную" экологию, оформившуюся как научное направление и направление общественно-политической деятельности на 100 лет позднее, занимающуюся проблемами охраны окружающей среды, взаимодействием с ней человеческого сообщества.

Остановимся на некоторых понятиях, которые будут встречаться в этой лекции.

Под *особью* понимается отдельный индивидуум, отдельный организм. *Популяция* — это совокупность особей одного вида, существующих в одно и то же время и занимающих определенную территорию. И, наконец, *сообщество* — это совокупность совместно сосуществующих популяций.

В классической экологии рассматриваются взаимодействия нескольких видов:

1. Взаимодействие организма и окружающей среды.
2. Взаимодействие особей внутри популяции.
3. Взаимодействие между особями разных видов (между популяциями).

Будем моделировать два последних взаимодействия.

При построении моделей в математической экологии используется опыт математического моделирования механических и физических систем, но с учетом специфических особенностей биологических систем:

- сложность внутреннего строения каждой особи;
- условия жизнедеятельности организмов зависят от многих факторов внешней среды;
- экологические системы являются незамкнутыми;
- огромный диапазон внешних характеристик, при которых сохраняется жизнеспособность систем.

Перейдем к рассмотрению отдельных задач моделирования экологических систем.

МОДЕЛЬ 1. Внутривидовая конкуренция для популяции с дискретным размножением

Рассмотрим простейшую модель для вида с дискретными периодами размножения, в которой численность популяции в момент времени t равна N_t, и численность изменяется во времени согласно величине чистой скорости воспроизводства R. Такими видами являются, например, большая часть растений, некоторые виды насекомых; у них разные поколения четко разнесены во времени. Коэффициент R характеризует количество особей, которое воспроизводит отдельная особь, а также выживание уже существующих. Например, если $R = 3$, то особь может воспроизвести трех потомков и при этом погибнуть, или же воспроизвести двух потомков и при этом выжить. Данная модель может быть выражена уравнением

$$N_{t+1} = N_t \cdot R, \qquad (1)$$

решение которого имеет вид

$$N_t = N_0 \cdot R^t, \qquad (2)$$

где N_0 — начальная численность популяции. Эта модель, однако, описывает популяцию, в которой отсутствует конкуренция и в которой R является константой; если $R > 1$, то численность популяции будет бесконечно увеличиваться. В реальности таких популяций нет, хотя в экспериментальных условиях, т.е. условиях идеальных, такое возможно. В литературе приводится немало интересных примеров быстрого роста численности популяций, если бы для их размножения существовали идеальные условия. Особенно это относится к насекомым, растениям и микроорганизмам, которые могли бы покрыть земной шар толстым слоем, если им создать благоприятные условия для размножения. Но в действительности такого роста популяций, где их численность увеличивается в геометрической прогрессии, практически не наблюдается (хотя нашествия саранчи выглядят довольно убедительно!).

Следовательно, в первую очередь необходимо изменить уравнения таким образом, чтобы чистая скорость воспроизводства зависела от внутривидовой конкуренции.

Конкуренцию можно определить как использование некоего ресурса (пищи, воды, света, пространства) каким-либо организмом, который тем самым уменьшает доступность этого ресурса для других организмов. Если конкурирующие организмы принадлежат к одному виду, то взаимоотношения между ними называют *внутривидовой* конкуренцией; если же они относятся к разным видам, то их взаимоотношения называют *межвидовой* конкуренцией.

На *рис.* 1 показана зависимость скорости воспроизводства от численности популяции. Точка A отражает ситуацию, в которой численность популяции близка к нулю, конкуренция при этом практически отсутствует, и фактическую скорость воспроизводства вполне можно описывать параметром R в его первоначальном виде. Следовательно, при низких величинах плотности уравнение (1) вполне пригодно. В преобразованном виде оно выглядит так: $\frac{N_t}{N_{t+1}} = \frac{1}{R}$.

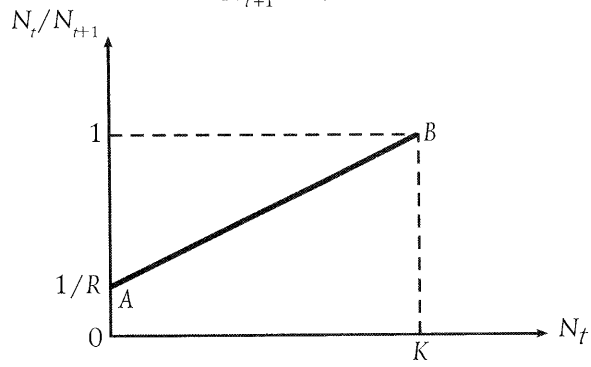


Рис. 1. Простейшая линейная зависимость, в соответствии с которой с ростом численности популяции растет величина, обратная ее приросту

Точка B, напротив, отражает ситуацию, в которой численность популяции высока и где в значительной степени проявляется внутривидовая конкуренция. Фактическая скорость воспроизводства в результате конкуренции настолько снижена, что популяция в целом может не более чем восстанавливать в каждом поколении свою численность, потому что количество родившихся особей уравновешивается количеством погибших. N_{t+1} в точности соответствует N_t и $\frac{N_t}{N_{t+1}} = 1$, а численность попу-

ляции при этом называется "предельной плотностью насыщения K". Прямая AB описывает изменение скорости воспроизводства с ростом численности популяции.

Согласно *рис.* 1 запишем уравнение прямой AB (так как известны координаты точек $A(0; 1/R)$ и $B(K; 1)$) (промежуточные выкладки предлагаем проделать читателю самостоятельно):

$$N_{t+1} = \frac{N_t \cdot R}{1 + a \cdot N_t} \qquad (3)$$

($\frac{R-1}{K}$ обозначено через a).

Это уравнение представляет собой модель роста популяции, ограниченного внутривидовой конкуренцией. Суть этой модели в том, что константа R в уравнении (1) заменена на фактическую скорость воспроизводства, т.е. $\frac{R}{1 + a \cdot N_t}$, которая уменьшается по мере роста численности популяции N_t. Достоинство полученного уравнения заключается в его простоте. Такой тип конкуренции приводит к саморегуляции численности популяции, изображённой на *рис.* 2 (для некоторого набора параметров модели).

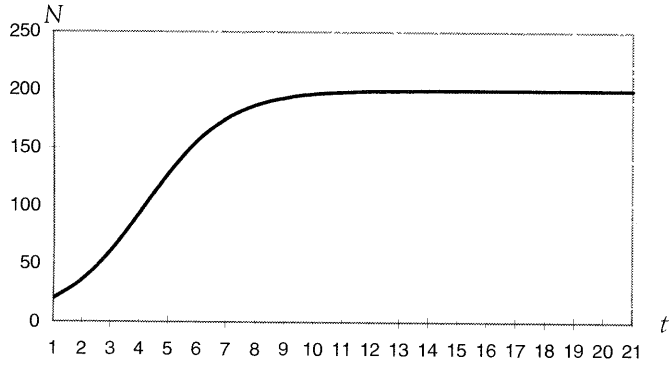


Рис. 2. Изменение численности популяции согласно (3) при $R = 2$, $K = 200$, $N_0 = 20$

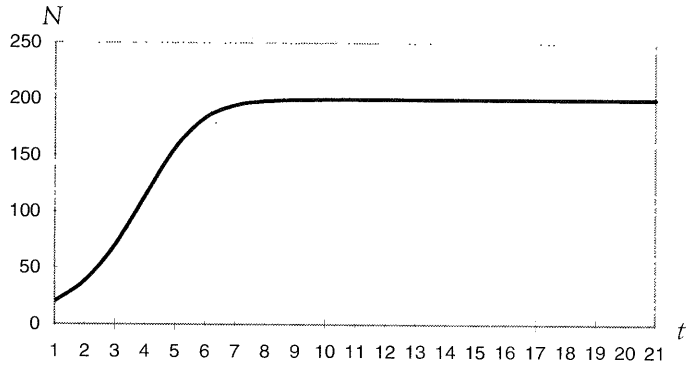
Замечание. Поскольку изучается популяция с дискретным размножением, то в действительности график функции $N(t)$ является точечным. На *рис.* 2 и 3 эти точки соединены линиями для лучшего восприятия.

После несложного изменения в уравнении (3) может быть получена более реалистичная модель. Действительно, простейшая из возможных зависимостей падения скорости роста популяции от ее численности, изображенная на *рис.* 1, является не законом природы, а всего лишь удобной гипотезой. Далеко не всегда следующая из нее динамика численности популяции, определяемая внутривидовой конкуренцией, качественно согласуется с изображенной на *рис.* 2. Более общая гипотеза о законе падения скорости роста популяции в зависимости от ее численности приводит к уравнению

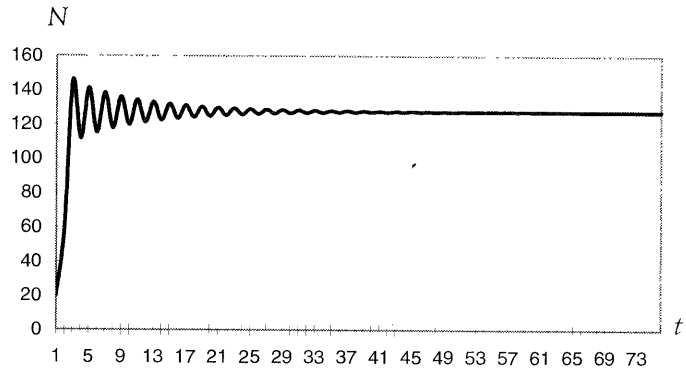
$$N_{t+1} = \frac{N_t \cdot R}{1 + (a \cdot N_t)^b}. \qquad (4)$$

Набор величин a, b, R можно использовать для сравнения и противопоставления сильно различающихся ситуаций. Другим положительным качеством уравнения (4) является его способность освещать новые стороны реального мира. Путем анализа кривых динамики популяций, полученных с помощью уравнения, можно прийти к предварительным выводам относительно динамики природных популяций.

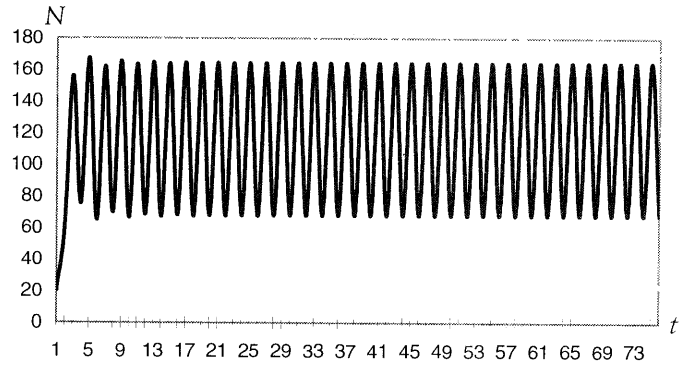
На *рис.* 3 показаны различные варианты динамики численности популяций, полученные с помощью уравнения (4) при разном сочетании параметров b и R ($N_0 = 20$).



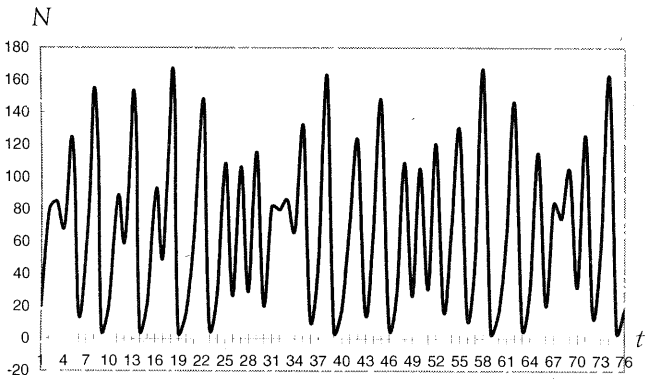
а) Монотонное установление стационарной численности популяции при $b = 1{,}4$, $R = 2$



б) Колебательное установление стационарной численности популяции при $b = 2{,}85$, $R = 3$



в) Устойчивые предельные циклы изменения численности популяции при $b = 3{,}7$, $R = 3$



г) Случайные изменения численности популяции при $b = 5{,}6$, $R = 4$

Рис. 3

Интересной возможностью работы с уравнением (4) является построение на фазовой плоскости b, R границ (т.е. линий), которые разделяют монотонное затухание, затухающие колебания, устойчивые предельные циклы и случайные (хаотические) изменения. Для этого нужно задаться значениями a и N_0, производить расчеты, изменяя параметры b, R. Различить каждый из возможных режимов программным путем алгоритмически трудно, поэтому проще делать это визуально, например, выполнять построение на экране компьютера графиков изменения численности популяции и, если происходит переход от одного режима к другому, запоминать соответствующие значения параметров b, R. Монотонное установление стационарной численности популяции и затухающие периодические колебания еще можно различить программным путем, а различение устойчивых предельных циклов и случайных изменений численности является само по себе интересной задачей, поразмышлять над которой предлагаем читателям.

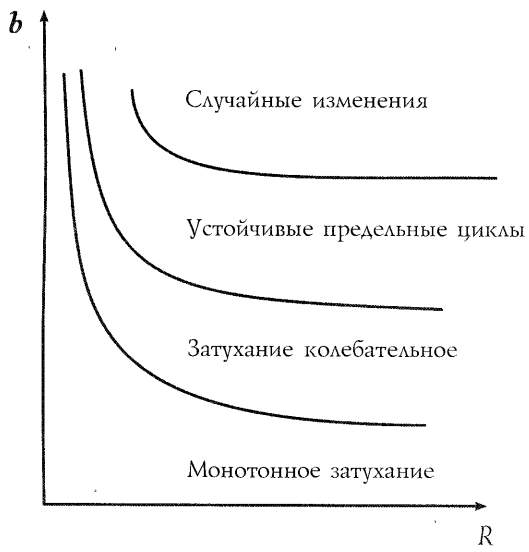


Рис. 4. Схематическое изображение фазовой диаграммы динамики численности популяции с дискретным размножением [5]

В связи с уравнениями (3), (4) сделаем замечание общего характера, справедливое для всех моделей экологии. В отличие от физики, где "модель" часто является точно установленным законом природы, в экологии модель гораздо более условна. Ее адекватность реальности (в той мере, в какой моделирование вообще на это претендует) проверяется лишь экспериментально. Поскольку экологи много раз наблюдали каждую из динамик, изображенных на рис. 3, можно сделать вывод о полезности соответствующей модели — иначе она была бы лишь математическим упражнением.

МОДЕЛЬ 2. Логистическое уравнение, внутривидовая и межвидовая конкуренция

В природе встречаются популяции, где рождение и гибель организмов происходят непрерывно. Рассмотрим популяцию с непрерывным размножением и построим модель изменения ее численности. Математическим аппаратом здесь, как в любом случае, где со временем происходят изменение, движение, являются дифференциальные уравнения. Скорость роста в этом случае можно обозначить $\frac{dN}{dt}$, тогда средняя скорость увеличения численности в расчете на одну особь определяется величиной $\frac{dN}{dt}\cdot\frac{1}{N}$. Без учета внутривидовой конкуренции получаем $\frac{dN}{dt} = rN$. Через r обозначена мгновенная удельная скорость роста численности, т.е. приращение численности за единицу времени в пересчете на одну особь. Согласно этой модели при $r > 0$ будет наблюдаться экспоненциальный рост.

Теперь попробуем учесть внутривидовую конкуренцию, предположив, что, когда численность популяции близка к нулю, скорость ее роста определяется приведенным выше законом, когда же при возрастании N достигается значение K (предельной плотности насыщения), скорость роста популяции снижается до нуля. Если принять, что скорость роста численности популяции в пересчете на одну особь при увеличении N от нуля до K уменьшается от r до 0 по линейному закону, приходим к уравнению $\frac{dN}{dt} = rN\cdot\left(\frac{K-N}{K}\right)$.

Последнее уравнение известно под названием "логистического". В истории экологии логистическое уравнение сыграло значительную роль, оказав большое влияние на ее развитие, применение в экологии математических методов. С другой стороны, простота уравнения накладывает ограничения на область его применения, так как с его помощью можно описать немногие настоящие популяции. Адаптируем модель к имитации более реалистичных взаимодействий между популяциями.

Сущность межвидовой конкуренции заключается в том, что у особей одного вида уменьшаются плодовитость, выживаемость и скорость роста в результате использования того же ресурса особями другого вида, причем влиянию конкуренции в той или иной мере подвергаются особи обоих видов. Но конкуренция воз-

никает лишь в том случае, когда потребляемый ресурс ограничен.

Так как рассматриваются две различных популяции, введем соответствующие обозначения [1]. Пусть N_1 — численность первой популяции, а N_2 — второй. Предельные плотности насыщения и максимальные врожденные скорости роста популяций обозначим соответственно K_1, K_2, r_1 и r_2.

Обратимся к логистическому уравнению и учтем в нем межвидовую конкуренцию, предположив, что M особей второго вида оказывают такое же воздействие на вид один, как одна особь вида один. Константу $\frac{1}{M}$ в этом случае называют "коэффициентом конкуренции". Обозначается она обычно α_{12}. Таким образом, чтобы отразить суммарное воздействие на вид один, нужно в логистическом уравнении в числителе дроби вместо N записать $N_1 + \alpha_{12} \cdot N_2$. Аналогично получается уравнение для исследования численности второй популяции. В результате получаем систему двух дифференциальных уравнений:

$$\begin{cases} \dfrac{dN_1}{dt} = r_1 N_1 \cdot \dfrac{K_1 - N_1 - \alpha_{12} N_2}{K_1} \\ \dfrac{dN_2}{dt} = r_2 N_2 \cdot \dfrac{K_2 - N_2 - \alpha_{21} N_1}{K_2} \end{cases} \qquad (5)$$

Модель межвидовой конкуренции, выраженная системой (5), названа в честь ее авторов "моделью Лотки — Вольтерры".

Заметим, что если коэффициенты α_{12} или α_{21} больше единицы, то влияние со стороны конкурирующей популяции на особей данного вида сильнее, чем со стороны особей своего вида.

Соответствующие конечно-разностные уравнения здесь таковы (верхние индексы — номера популяций):

$$\begin{cases} N_n^{(1)} = N_{n-1}^{(1)} + \left(r_1 N_{n-1}^{(1)} \cdot \dfrac{K_1 - N_{n-1}^{(1)} - \alpha_{12} N_{n-1}^{(2)}}{K_1} \right) \cdot \Delta t \\ N_n^{(2)} = N_{n-1}^{(2)} + \left(r_2 N_{n-1}^{(2)} \cdot \dfrac{K_2 - N_{n-1}^{(2)} - \alpha_{21} N_{n-1}^{(1)}}{K_2} \right) \cdot \Delta t \end{cases} \qquad (6)$$

Начальные условия для уравнений (5), (6) определяются так: $N^{(1)}(0) = N_0^{(1)}$, $N^{(2)}(0) = N_0^{(2)}$.

Главный вопрос, который интересует исследователя межвидовой конкуренции: "При каких условиях увеличивается или уменьшается численность каждого из видов?" Для ответа на этот вопрос нужно построить диаграммы, где были бы изображены все возможные сочетания численностей популяций обоих видов. На таких диаграммах число особей одного вида откладывают по горизонтальной оси, а другого — по вертикальной. При одних сочетаниях численностей будет отмечаться рост выбранной для наблюдения популяции, при других — уменьшение ее численности. Также для каждого из видов можно провести изоклины — линии, вдоль которых не наблюдается ни увеличения, ни уменьшения численности (из системы (5) — это прямые с уравнениями $N_1 = K_1 - \alpha_{12} \cdot N_2$ и $N_2 = K_2 - \alpha_{21} \cdot N_1$).

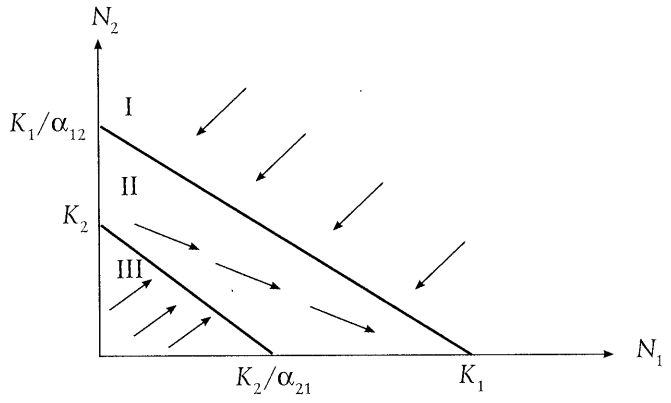
Для решения поставленной выше задачи объединим в одной фазовой плоскости изоклины для обоих видов и будем одновременно исследовать динамику их численностей. Изоклины относительно друг друга располагаются четырьмя различными способами, что дает различные исходы конкуренции [1] (*рис.* 5).

Вообще, как можно заметить, проанализировав систему (5) и приведенные на *рис.* 5 диаграммы, есть три существенно различных исхода конкуренции:

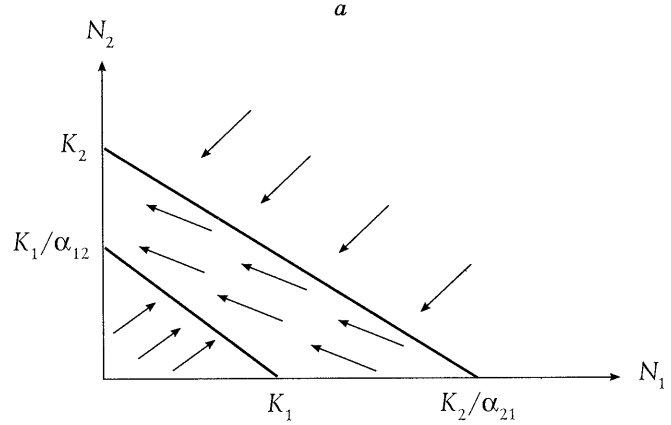
1) мирное сосуществование популяций с выходом их численностей на некоторое стационарное состояние (возможно лишь при $\alpha_{12} \cdot \alpha_{21} < 1$);

2) вытеснение первой популяцией второй;

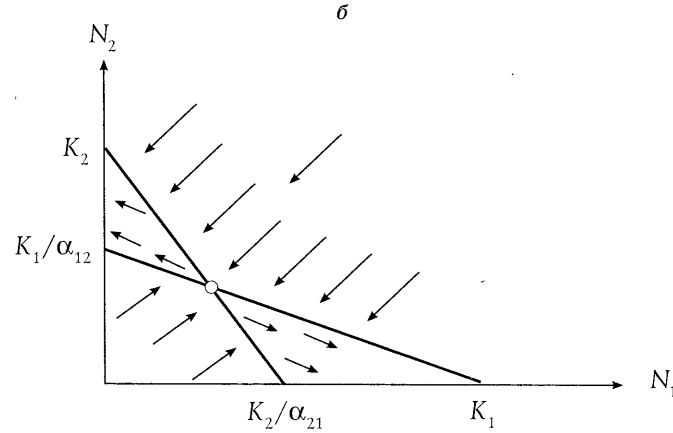
3) вытеснение второй популяцией первой.



а



б



в

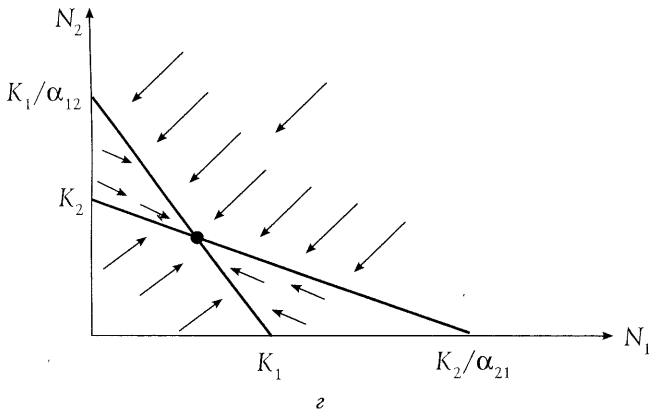


Рис. 5. **Примерные результаты конкуренции, полученные с помощью модели Лотки — Вольтерры при различных параметрах**

На *рис. 5а* в зоне I численность обеих популяций падает; в зоне II численность первой популяции растет, второй — уменьшается; в зоне III численность обеих популяций увеличивается. (В каждом из случаев результаты получены для нескольких наборов начальных условий.)

Возможны дальнейшие усовершенствования приведенных выше моделей. Например, в логистическом уравнении можно принять закон изменения скорости роста в пересчете на одну особь не линейным, а квадратичным либо экспоненциальным, построить и исследовать получившуюся модель, сравнить результаты для случая линейного закона. В модели межвидовой конкуренции можно рассмотреть не две, а три и более конкурирующих популяции, провести исследование.

МОДЕЛЬ 3. Динамика численностей популяций хищника и жертвы

Рассматривая динамику численностей популяций хищника и жертвы (под хищниками будем понимать особей одной популяции, которые в качестве пищи используют особей другой популяции — жертв), экологи прежде всего стремятся понять их общие закономерности. В простейших моделях хищник и жертва рассматриваются изолированно, безотносительно влияния на них других видов. Одна из самых первых и простых моделей с непрерывным размножением была предложена, как и модель межвидовой конкуренции, Лоткой и Вольтеррой и носит их имя.

Модель состоит из двух компонентов: C — численность популяции хищника и N — численность популяции жертвы. Предполагается, что в отсутствие хищника популяция жертвы растет экспоненциально. Чем больше численность той и другой популяций, тем чаще происходят встречи. Число встреченных и успешно съеденных жертв будет зависеть от эффективности, с которой хищник находит и ловит жертву. Если обозначить через a "эффективность поиска", то скорость поедания жертвы будет равна $a \cdot C \cdot N$, и окончательно для численности жертвы получаем:

$$\frac{dN}{dt} = r \cdot N - a \cdot C \cdot N.$$

В отсутствие пищи отдельные особи хищника голодают и гибнут. Предположим вновь, что численность хищника в отсутствие пищи будет уменьшаться экспоненциально: $\frac{dC}{dt} = -q \cdot C$ (q — смертность). Скорость рождения новых особей в данной модели полагается зависящей от двух обстоятельств: скорости потребления пищи $a \cdot C \cdot N$ и эффективности f, с которой эта пища переходит в потомство хищника. Итак, для численности хищника окончательно получаем:

$$\frac{dC}{dt} = f \cdot a \cdot C \cdot N - q \cdot C \cdot$$

Так как процессы нужно рассматривать вместе, объединим уравнения в систему:

$$\begin{cases} \frac{dN}{dt} = r \cdot N - a \cdot C \cdot N \\ \frac{dC}{dt} = f \cdot a \cdot C \cdot N - q \cdot C \end{cases} \quad (7)$$

Конечно-разностные уравнения:

$$\begin{cases} N_n = N_{n-1} + (r \cdot N_{n-1} - a \cdot C_{n-1} \cdot N_{n-1}) \cdot \Delta t \\ C_n = C_{n-1} + (f \cdot a \cdot C_{n-1} \cdot N_{n-1} - q \cdot C_{n-1}) \cdot \Delta t \end{cases} \quad (8)$$

Начальные условия: $N(0) = N_0$, $C(0) = C_0$.

Как и в предыдущей модели, свойства полученной можно исследовать, построив изоклины и совместив их на одном чертеже, после чего в этой плоскости (N, C) отслеживая одновременное изменение численностей популяций хищника и жертвы [1]. В результате получим картину взаимодействия популяций. Модель предполагает лишь получение так называемых "нейтральных устойчивых циклов", т.е. численности популяций хищника и жертвы совершают периодические колебания: при увеличении численности хищников уменьшается численность популяции жертвы и наоборот. Такие колебания численности будут продолжаться в соответствии с моделью до тех пор, пока какое-либо внешнее воздействие не изменит численность популяций, после чего произойдет переход в новое устойчивое состояние. Еще один возможный исход взаимодействия популяций: хищники полностью поедают жертв, после чего в отсутствие пищи гибнут сами.

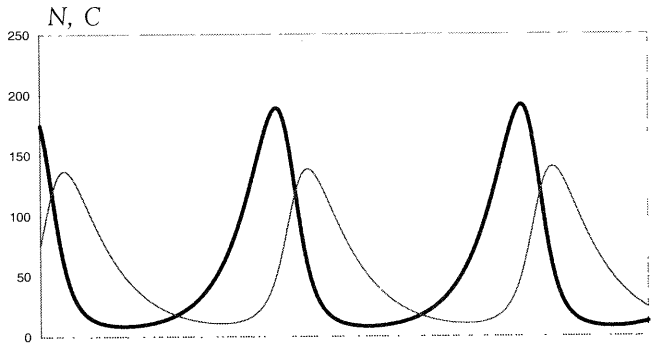


Рис. 6. **Численности популяций хищника (тонкая линия) и жертвы (толстая линия) совершают периодические колебания при $r = 5$, $q = 3{,}5$, $f = 0{,}6$, $a = 0{,}1$, $N_0 = 175$, $C_0 = 75$**

Дальнейшие возможные усовершенствования модели (7), (8): "хищник" и две конкурирующие популяции "жертв", "хищник" и две не конкурирующие популяции "жертв", два вида "хищников" и совместно потребляемая ими популяция "жертв".

На этом завершаем обзор моделей классической экологии. Для расширения круга экологических задач рекомендуем обратиться к [1], [5], [6], [8].

По завершении изучения данного раздела *учащиеся должны знать:*

- *отличительные особенности и специфику компьютерного математического моделирования в классической экологии; основные понятия классической экологии: особи, популяции, сообщества, конкуренция, хищничество и др.;*
- *модели динамики численности популяций с дискретным размножением;*
- *модели динамики численности популяций с непрерывным размножением и внутривидовой и межвидовой конкуренцией;*
- *модели динамики численности популяций хищника и жертвы.*

Учащиеся должны уметь:

- *исследовать динамику численности популяций с дискретным размножением; строить фазовые диаграммы;*
- *исследовать динамику численности популяций с непрерывным размножением, анализировать влияние на ее поведение внутривидовой и межвидовой конкуренции;*
- *исследовать систему "хищник—жертва", анализировать взаимное влияние численностей популяций;*
- *строить, исследовать и анализировать имитационные модели экологических систем.*

Контрольные вопросы и задания (для учащихся)

1. В чем отличие классической экологии от современной?
2. Какие проблемы рассматриваются в классической экологии?
3. Какие виды взаимодействия организмов принято рассматривать в классической экологии?
4. Какие цели преследуются при составлении математических моделей в экологии?
5. В чем выражается специфика биологических систем в отличие от рассмотренных ранее физических и механических систем (см. лекции 2—3)?
6. Что понимают под конкуренцией в биологии? внутривидовой конкуренцией? межвидовой конкуренцией? Каковы источники конкуренции и как конкуренция учитывается в приведенных моделях?
7. Какие результаты могут быть получены с помощью простейшей модели роста численности популяции с дискретным размножением? Как изменятся эти результаты, если учесть интенсивность конкуренции?
8. Как построить фазовую диаграмму динамики численности популяции с дискретным размножением?
9. Составить программу, позволяющую получить все четыре способа изменения численности популяции в модели (4). Воспроизвести полученные в лекции результаты. Использовать программу для продолжения исследования модели.
10. Как выводится логистическое уравнение? Каково аналитическое решение этого уравнения? Как в нем учитывается внутривидовая конкуренция? По какому принципу записывается модель межвидовой конкуренции?
11. Какие результаты могут быть получены с помощью модели межвидовой конкуренции? Подобрать начальные условия и параметры модели для получения каждого из трех возможных исходов межвидовой конкуренции.
12. Какие факторы необходимо учесть при разработке модели системы "хищник—жертва"? Какие результаты могут быть получены с помощью модели "хищник—жертва"?
13. Получить результаты для указанного в примере набора параметров (см. *рис.* 6). Определить период колебаний численностей популяций хищника и жертвы. Варьируя заданные параметры, добиться вымирания популяций хищника и жертвы (популяции вымирают, если численность жертв становится меньше единицы).

Вопросы для обсуждения (для учителей)

1. Насколько целесообразным представляется изучение экологических моделей в курсе КММ? Не следует ли развить этот раздел, дополнив его не слишком сложными имитационными моделями современной экологии?
2. Нужно ли включить в программу этого раздела работу с программами, имитирующими процессы как классической, так и современной экологии?

Литература

1. *Бигон М., Харпер Дж., Таунсенд К.* Экология. Особи, популяции и сообщества: В 2 т. М.: Мир, 1989. Т. 1, 667 с.
2. *Данилина И.И.* Обучение информатике в условиях профильной дифференциации (на примере курса экологической направленности). // Диссертация кандидата педагогических наук. Екатеринбург, 1998, 143 с.
3. Информатика. Задачник-практикум в 2 т. / Под ред. И.Г. Семакина, Е.К. Хеннера: Т. 2. М.: Лаборатория Базовых Знаний, 1999, 280 с.
4. Информатика. 7—9-е классы. Базовый курс. Практикум-задачник по моделированию. / Под ред. Н.В. Макаровой. СПб.: Питер, 2001, 176 с.
5. *Хеннер Е.К., Шестаков А.П.* Математическое моделирование: пособие для учителя. Пермь: Пермский государственный пед. университет, 1995, 259 с.
6. *Вершинин О.Е.* За страницами учебника информатики. М.: Просвещение, 1992, 352 с.
7. *Горстко А.Б.* Познакомьтесь с математическим моделированием. М.: Знание, 1991, 160 с.
8. *Горстко А.Б., Угольницкий Г.А.* Введение в моделирование эколого-экономических систем. Ростов: Изд-во РГУ, 1990, 112 с.
9. *Шолоховичь В.Ф.* Информационные технологии обучения: дидактические основы, проблемы разработки и использования. Екатеринбург: Уральский государственный пед. университет, 1995, 128 с.
10. *Риклефс Р.* Основы общей экологии. М.: Мир, 1979.

№ 4 (выходит два раза в месяц)
Учебно-развлекательная стенгазета
ИНФОРМАТИК@

Игры в Office

Можно с уверенностью сказать, что Microsoft Office — один из самых популярных программных пакетов в России. Его последние версии предоставляют пользователям богатый набор возможностей для работы с текстами, слайдами и электронными таблицами. Но, оказывается, помимо привычных функций форматирования текста, проверки орфографии, работы с рисунками и слайдами, в MS Office есть еще и встроенные игры! Они не документированы и скорее всего являются плодом развлечений разрабатывавших MS Office программистов.

Поскольку игры "неофициальные", то, естественно, в Microsoft Office нет ни кнопок, ни команд, их вызывающих. Как же узнали об их существовании? Да очень просто — со слов программистов — разработчиков MS Office, поделившихся секретами своих детищ.

В MS Excel 95 есть игра наподобие Doom. Чтобы вызвать ее, нужно выполнить следующие действия:

— открыть MS Excel с чистой книгой;

— прокрутить лист (мышью или клавишами) до строки с номером 95;

— выделить всю 95-ю строку, щелкнув мышью в служебной колонке слева от ее номера;

— нажать клавишу Tab;

— открыть пункт меню "Help | About Microsoft Excel" ("Справка | О программе");

— удерживая одновременно клавиши Ctrl, Alt и Shift, щелкнуть мышью на кнопке "Technical Support" ("Техническая поддержка").

После этого откроется небольшое окно, где запускается игра. С помощью клавиш перемещения курсора можно передвигаться по залу.

В MS Excel 97 встроена игра-"леталка", так что вы сможете полетать и полюбоваться прекрасными видами. Вот как можно ее открыть:

— запустите MS Excel и проследите, чтобы в пункте меню "Сервис | Параметры | Общие" была выключена опция "Стиль ссылок R1C1";

— создайте чистую таблицу. Нажмите F5 и введите в строке ссылки для перемещения "X97:L97" (т.е. переход к ячейкам от X97 до L97 и их выделение). Нажмите Enter;

— теперь вы перейдете к строке с номером 97. Нажмите Tab, а затем, удерживая Ctrl и Shift, щелкните левой кнопкой мыши на значке "Мастер диаграмм" и подождите, пока открывающееся окно не заполнит весь экран, — и вы в игре.

Теперь все управляется мышью. Движения мыши задают направление: левая кнопка — ускорение вперед, правая — назад. Выход из игры — Esc.

Встроенная в MS Word 97 игра PinBall отличается простотой как правил, так и оформления. Ваша задача — удержать на наклонном поле шарик, ударяя его двумя лопаточками, расположенными в нижней части этого поля. Чтобы вызвать игру, запустите MS Word 97, создайте новый документ, напечатайте слово "Blue" и выделите его. Затем с помощью меню ("Формат | Шрифт") установите для этого слова полужирное начертание шрифта и синий цвет. Поместите курсор в конец слова "Blue", введите один пробел, выберите пункты "Help | About" ("Справка | О программе") и, удерживая клавиши Ctrl и Shift, щелкните левой кнопкой мыши на пиктограмме в части появившегося окна.

Появившаяся на экране игра управляется всего тремя клавишами: Z — удар левой лопаточкой, M — правой, Esc — выход.

Если в Microsoft Excel 97 есть игра-"леталка", то в следующую версию этой программы уже включены автогонки. Но об этом — в следующем выпуске.

По материалам журнала "Мир ПК"

Первый шахматный алгоритм...

Имя английского математика Алана Тьюринга достаточно известно. Он — один из основоположников вычислительной техники. В 30-х годах XX века Тьюринг создал теоретическую базу по созданию первых машин-автоматов, занимающихся математическими расчетами.

Менее известно, что именно он в 1947—1950 гг. разработал первую специализированную шахматную программу. В годы Второй мировой войны Тьюринг руководил группой ученых, взламывавших коды, сгенерированные немецкой шифровальной машиной Enigma. Шахматы были любимой игрой математика. Однако он, как ни странно, оставался в этой области практически дилетантом, показывая весьма посредственный уровень игры. Так или иначе, алгоритм для игры в шахматы Тьюрингу создать удалось. Алгоритм был построен на основе изучения теоретической литературы и показывал бы в игре результаты, не худшие, чем его автор. Почему "показывал бы"? Да потому что в то время еще не было компьютеров, способных такой алгоритм выполнить. Отладка и прогонка его осуществлялись вручную, по длинным рулонам блок-схем. Тьюринг сам выполнял роль процессора, водя указкой по стрелкам логических переходов; каждый реальный ход "машины" занимал полчаса реального времени. Однако алгоритм все-таки работал!

и первая шахматная программа

В 1965 году американские ученые-ядерщики получили в свое распоряжение одну из первых ЭВМ — машину под названием MANIAC I. Она состояла из нескольких тысяч радиоламп и обладала быстродействием около 11 тыс. операций в секунду. Но самое главное — она была программируемой.

Разумеется, инженеры (профессия "программист" появилась несколько позже) тут же принялись испытывать новый, сверхсовременный вычислитель. В качестве тестовых программ было алгоритмизировано немало игр, в том числе и шахматы. Программированием занялся талантливый математик Алекс Берштейн. Однако, здраво оценивая способности (возможности?) машины, он серьезно упростил правила. Вместо классических 8×8 клеток игра шла на доске 6×6. При этом пожертвовали, понятное дело, парой пешек и парой фигур, а именно: слонов с каждой стороны.

Программе (ей даже не удосужились дать какое-либо название) удалось сыграть три полноценные партии, прежде чем MANIAC I целиком погрузился в расчеты по своему "прямому" назначению. Первую партию машина провела сама с собой — белые, как и логично было предположить, выиграли. Во второй партии программе противостоял настоящий сильный шахматист, давший электронному противнику фору, сняв с доски своего ферзя. После десяти часов упорной борьбы (в среднем раздумья над одним ходом у программы занимали около 12 минут!) машина все-таки не сумела воспользоваться своим преимуществом — и проиграла. Третий матч программа провела с новичком — секретаршей лаборатории, которую обучили передвигать по доске фигуры перед самым испытанием. Машина разгромила бедняжку в двадцать три хода. Пусть на уменьшенной доске, пусть без слонов, пусть противник не был силен, но это все-таки был первый в истории вычислительной техники случай победы машины над человеком в игре, требующей напряжения интеллекта.

По материалам журнала "ПЛ: Компьютеры"

MMIX — учебный RISC-процессор нового тысячелетия от Дональда Кнута

Е.А. Еремин,
г. Пермь

Наверное, нет человека, интересующегося компьютерными вычислениями, которому не был бы известен фундаментальный многотомный труд Дональда Кнута "Искусство программирования на ЭВМ". В нем проведено всестороннее исследование практически всех наиболее важных вычислительных алгоритмов. В качестве примера достаточно упомянуть, что в одном из разделов книги обсуждается влияние временных характеристик движения магнитной ленты на эффективность сортировки больших массивов информации. Такой детальный подход к исследованию вычислительных задач невозможен без анализа времени исполнения машинных операций, а значит, без рассмотрения конкретной системы команд ЭВМ. Д.Кнут в своей книге предложил условную вычислительную машину собственной конструкции, которую назвал **MIX**[1]. MIX является виртуальной машиной, но она очень похожа на наиболее распространенные ЭВМ того времени. С лукавым юмором Кнут заявляет, что MIX есть римская запись числа 1009, получающегося как средний номер моделей всех машин, которые он проанализировал при создании своей модели [1]:

(360 + 650 + 709 + 7070 + U3 + SS80 +
+ 1107 + 1604 + G20 + B220 + S2000 +
+ 920 + 601 + H800 + PDP4 + II)/16

Несмотря на то что модель MIX содержит минимальное количество технических деталей, за четыре десятилетия она все-таки устарела. И вот недавно Кнут решил ее модернизировать и заменить на более современную, которую он назвал **MMIX**[2]. Здесь снова есть столь любимый Кнутом шутливый подтекст: если название считать состоящим из римских цифр, то из них образуется число 2009, которое автор предполагает годом выпуска окончательной редакции своего многотомника.

Возможно, у вас на языке вертится недоуменный вопрос: а надо ли на современном этапе развития вычислительной техники знать внутренние основы ее работы? Приведем подробный ответ самого Дональда Кнута (цитируется по тексту официальной домашней страницы MMIX [2]; перевод автора статьи):

"Многие читатели, без сомнения, думают: "Почему Кнут заменил MIX другой машиной, вместо того чтобы просто перейти к языку программирования высокого уровня? Едва ли кто-то использует сейчас ассемблер".

Эти люди имеют право на свое мнение, и им не стоит утруждать себя чтением частей моей книги, связанных с машинным языком. Тем не менее причины применения машинного языка, которые я приводил в предисловии к первому тому, написанному в начале 60-х, остаются в силе и сегодня:

- Одна из принципиальных целей моей книги состоит в том, чтобы показать, как конструкции высокого уровня на самом деле реализуются в машинах, а не просто показать, как они применяются. Я объясняю взаимодействие соподпрограмм, древовидные структуры, генерацию случайных чисел, арифметику повышенной точности, перевод из одной системы счисления в другую, уплотнение информации, комбинаторный поиск, рекурсию и т.д. с учетом мельчайших деталей.
- Программы, которые требуются в моей книге, как правило, такие короткие, что их главные моменты могут быть легко осознаны.
- Людям, которые действительно интересуются компьютерами, следует знать по крайней мере некоторые идеи о том, что представляет собой используемое ими оборудование. В противном случае программы, которые они напишут, будут достаточно причудливыми.
- Машинный язык необходим, во всяком случае, как выходной для многих программ, которые я описываю.
- Представление основных методов, таких, как, например, алгоритмы сортировки и поиска, на машинном языке дает возможность провести всестороннее исследование влияния на них размера кэша и ОЗУ, а также других аппаратных характеристик (быстродействие памяти, конвейеризация, multiple issue, буферы с записью lookaside, размер блоков кэша и т.д.), сравнивая различные схемы.

Кроме того, если бы я действительно использовал язык высокого уровня, какой это был бы язык? В 60-х я, вероятно, выбрал бы Algol-W[3]; затем в 70-х мне пришлось бы переписать мои книги, используя Pascal; в 80-х я непременно изменил бы все на Си; в 90-х я был бы вынужден переключиться на C++ и, вероятно, на Java. В 2000-х еще один язык, несомненно, будет *de rigueur*. Я не могу позволить себе тратить время на переписывание моих книг потому, что языки входят в моду и выходят из моды; языки не являются предметом моих книг, их предметом скорее является то, что вы можете делать на своем любимом языке. Мои книги фокусируются на "вечных истинах".

Поэтому я буду продолжать использовать в **TAOCP**[4] английский как язык высокого уровня и буду продолжать использовать язык низкого уровня, чтобы показывать, как машина на самом деле считает. Читателям, которые хотят видеть только алгоритмы, уже готовые к употреблению и написанные на наиболее часто используемом языке, следует покупать книги других людей".

---

[1] Mix переводится с английского как "смесь".

[2] Сам автор рекомендует читать это сокращение как "эм-микс", причем первая буква "М" "скромно" обозначает *millennium* — т.е. новое тысячелетие.

[3] Версия Алгола, реализованная Виртом.

[4] Первые буквы слов в английском названии книги "Искусство программирования на ЭВМ".

Да простят меня читатели за длинную цитату, но она очень хорошо описывает суть дела.

Итак, MMIX — это учебная модель компьютера, созданная Д.Кнутом для изучения эффективности работы различных численных методов. Тем не менее это не единственно возможное ее применение. Благодаря хорошему соответствию между данной моделью и реальными процессорами (Кнут упоминает, что при разработке он во многом ориентировался на процессоры AMD), MMIX может быть использован для целей обучения, например, для знакомства с основными принципами устройства и функционирования современных компьютеров. Именно поэтому автор считает достаточно актуальной публикацию данной статьи в педагогическом, а не в техническом издании. Возможностям использования модели в учебном процессе посвящен специальный раздел.

Статья имеет еще одну цель: дать некоторую доступную информацию на русском языке об интересной и масштабной разработке Кнута. Дело в том, что пока описание MMIX существует лишь в виде обновляемых авторских материалов, публикуемых на Интернет-сайте Дональда Кнута [2], что, к сожалению, для многих непригодно по причине языковых трудностей. Кроме того, Кнут не принадлежит к числу популяризаторов науки — скорее это серьезный профессор, излагающий свои знания академически строго и сжато в предположении некоторой предварительной подготовки читателей. Напротив, автор данной статьи надеется, что она будет понятна любому, кто интересуется работой компьютера: от школьника до преподавателя вуза.

Основные характеристики MMIX

Ниже приводится краткий перечень основных характеристик RISC-процессора MMIX:

Максимальная разрядность данных	64 бита
Допустимая разрядность данных	8 / 16 / 32 / 64 бита
Адресное пространство	64 бита
Адресация памяти	косвенная, через регистры
Система команд	трехадресная
Длина команды	32 бита
Количество операций	256
Максимальное число адресуемых регистров	256
Общее число регистров	более 256
Дополнительные внутренние регистры	на данный момент 32
Разрядность регистров	64 бита
Разрядность целочисленной арифметики	64 бита
Форматы целых чисел	знаковый / без знака
Умножение и деление	встроенные
Вещественная арифметика	встроенная
Разрядность вещественной арифметики	64 бита
Разрядность мантиссы	52 бита + скрытый бит + знак
Стандарт вещественных чисел	IEEE
Количество битовых логических операций	16
Смещение в условных переходах	16 бит
Смещение в безусловных переходах	24 бит
Переход к подпрограмме	двух типов

Приведенные данные дают нам некоторое первоначальное представление о MMIX. Сразу становится очевидным, что это очень подробная модель, полное освоение которой явно выходит за рамки данной ознакомительной статьи.

Архитектура MMIX

Основными устройствами для хранения информации в процессоре MMIX являются **регистры**. Именно с их содержимым в той или иной степени оперируют все инструкции процессора. MMIX содержит большое число регистров: единовременно их может быть подключено до 256 (это, кстати, неизбежно следует из факта, что под номер каждого регистра в командах отводится один байт, а значит, номер регистра может изменяться от 0 до 255). Тем не менее архитектура машины позволяет дополнительно увеличить общее количество регистров за счет специального механизма переключения. Иными словами, хотя в каждый момент времени доступны только 256 регистров, но они могут с помощью специальных команд выбираться из гораздо большего количества.

Не вдаваясь в подробности, можно считать все регистры процессора равноправными. При более детальном чтении документации оказывается, что это не совсем так, но в данной статье мы не будем обращать внимание на эти детали.

Помимо описанных выше, MMIX имеет также целый ряд внутренних вспомогательных регистров, которые при первоначальном знакомстве особого интереса не представляют.

Итак, для учебной модели регистров в MMIX больше чем достаточно. И хотя начинающим для упражнений даже половина из имеющихся регистров вряд ли потребуется, тем не менее имеет смысл разобрать методы взаимодействия процессора MMIX с оперативной памятью: все-таки на практике исходные данные берутся именно из ОЗУ, и туда же помещается результат их обработки, а в регистрах хранится лишь промежуточная информация.

Память в MMIX имеет **байтовую организацию**, т.е. минимальным количеством информации при чтении и записи является 1 байт. Чтобы обеспечить возможность обращения к любому байту памяти, все они пронумерованы и, следовательно, каждый из них имеет свой собственный номер (**адрес**).

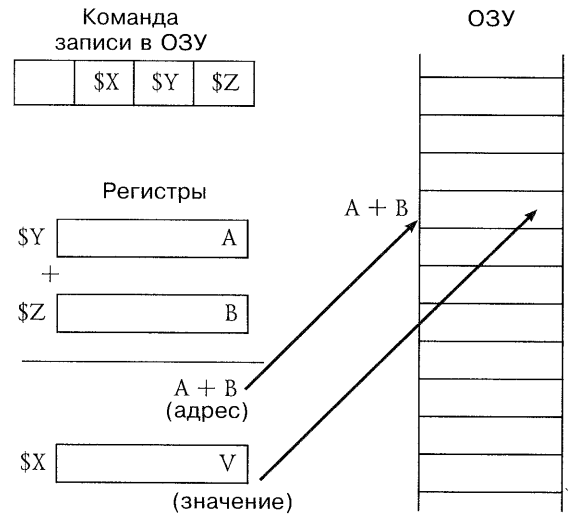
Для хранения данных, занимающих несколько байт, необходимое их количество объединяют. Так, для числа в MMIX требуется 8 последовательно расположенных байт. По существующей традиции *адресом всей информации в таких случаях принято считать адрес самого пер-*

вого байта, т.е. наименьший из всех адресов. Кроме того, для MMIX **обязательно** требуется, чтобы адрес данных был кратен их длине (делился на нее без остатка): иными словами, адрес числа должен делиться на 8, а команды — на 4; адрес однобайтовых данных, естественно, может быть любым.

Рассмотрим теперь порядок записи байтов данных в память. Очевидно, что здесь возможны два альтернативных способа: либо записывать данные "слева направо", т.е. так, чтобы самому старшему байту соответствовал минимальный номер, либо, наоборот, начать с младшего байта. Первый способ получил название **big-endian**, а второй — **little-endian**[5]. MMIX использует более естественный для человека big-endian метод хранения информации. Для справки заметим, что IBM PC, напротив, имеет little-endian архитектуру запоминающего устройства.

Для того чтобы прочитать из ОЗУ требуемую информацию, необходимо предварительно в одном из регистров задать ее адрес, а затем выполнить команду обмена с памятью, содержащую ссылку на номер используемого регистра. Подобный механизм адресации данных, когда адрес интересующей ячейки памяти хранится не в самой команде, а в регистре процессора и регистр, в свою очередь, указывается в качестве операнда, называется **методом косвенной адресации**. Именно он лежит в основе работы всех современных микропроцессоров.

Однако косвенным методом возможности адресации данных в MMIX не исчерпываются. Адрес ячейки памяти на самом деле вычисляется несколько более сложным способом: он является суммой двух величин — вычисляется как содержимое одного из регистров плюс содержимое другого или константа.



Рассмотрим для иллюстрации две инструкции записи содержимого регистра $1 (знаком "$" Кнут помечает номера регистров, чтобы не путать их с числами) в ОЗУ:

AE 01 02 03 — сохраняет содержимое регистра $1 в 8 последовательных байт, начиная с адреса, равного сумме значений регистров $2 и $3;

AF 01 02 03 — единственное отличие данной команды от предыдущей состоит в том, что начальный адрес ОЗУ равен сумме значения регистра $2 и *константы* 3.

Нетрудно сообразить, что если в последнем случае положить константу равной нулю, то получится метод косвенной адресации "в чистом виде", когда адрес ОЗУ определяется *только одним* регистром. Подчеркнем также, что представление адреса операнда в виде суммы двух значений необычайно удобно для доступа к последовательно расположенным данным — массивам (см. задачу 3 в разделе "Примеры простых программ").

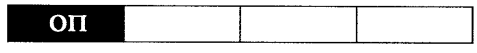
Форматы команд MMIX

Структура команд процессора для выполнения различных операций может несколько отличаться: например, заранее ясно, что инструкции сложения и условного перехода функционально не похожи настолько сильно, что вполне могут по-разному хранить внутри себя необходимую информацию. Распределение битов команды между отдельными ее частями часто называют **форматом** команды.

MMIX использует 5 форматов операций. Опишем их, сохраняя введенную Д.Кнутом нумерацию. Напомним читателю, что каждая команда занимает 32 бита, т.е. 4 байта.

Формат 0

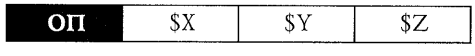
Используется довольно редко, только для немногочисленных системных инструкций (я насчитал всего три, и все достаточно экзотические). Существенным является только код операции, остальная часть команды произвольна.



При первом знакомстве этот формат смело можно пропустить, поэтому никаких примеров и дополнительных сведений здесь не приводится.

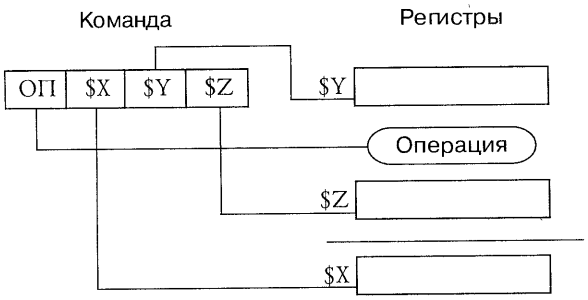
Формат 1

Данный формат в MMIX является основным. Помимо кода операции, задаются адреса трех регистров: в первый из них ($X) заносится результат выполнения операции, а второй и третий ($Y и $Z) позволяют сослаться на исходные данные (операнды).



В наиболее общем виде рассматриваемый тип операций можно сформулировать в виде *"установить регистр $X в соответствии с результатом операции над регистрами $Y и $Z"*.

[5] Терминология восходит к книге Д.Свифта "Путешествие Гулливера", где описаны две антагонистические партии, споривших, с какого конца необходимо разбивать яйцо — с тупого (по-английски big end) или острого (little end).



Операция может быть любой: арифметической, логической или даже записью или чтением из ячейки памяти. Вот несколько примеров:

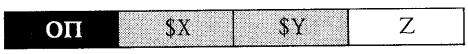
20 03 01 02 — сложить регистры $1 и $2, результат поместить в $3;

C8 06 04 05 — выполнить логическую операцию И над регистрами $4 и $5, результат поместить в $6;

AC 07 08 09 — сохранить значение регистра $7 в ячейку памяти с адресом, равным сумме значений регистров $8 и $9.

Формат 2

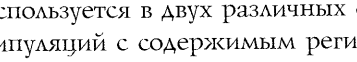
Фактически является разновидностью предыдущего формата. Единственное отличие состоит в том, что вместо последнего регистра указывается константа — беззнаковое целое значение от 0 до 255.



Например: 21 03 01 02 — к содержимому регистра $1 прибавить 2, результат поместить в $3 (сравните с первым примером из предыдущего пункта). Обратите внимание на то обстоятельство, что большинство инструкций MMIX допускает оба формата, причем коды одноименных операций форматов 1 и 2 всегда отличаются на единицу.

Формат 3

Кроме кода операции, задается единственный регистр $X и двухбайтовая константа от 0 до 65 535, обозначенная на рисунке YZ.



Формат используется в двух различных случаях:

- для манипуляций с содержимым регистра $X с использованием двухбайтовой константы YZ;
- для реализации переходов.

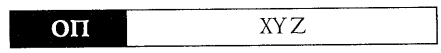
В первом случае непосредственно заданная в команде константа YZ указанным в коде операции способом воздействует на содержимое регистра $X. Например, команда с кодом E1 03 FFFF устанавливает значение регистра $3 равным 0000 FFFF 0000 0000, т.е. в определенную пару байт заносится указанная константа FFFF, а остальные байты обнуляются.

Во втором случае анализируется содержимое регистра $X и в случае выполнения условия, определяемого кодом операции, к адресу текущей команды добавляется 4 · YZ байт, т.е. осуществляется условный переход через заданное число инструкций. Так инструкция с кодом 42 05 0004 в случае нулевого содержимого регистра $5 пропустит 4 следующих после нее команды программы.

Подчеркнем, что значение смещения YZ в командах перехода MMIX может быть как положительным, так и отрицательным, а коды инструкций "вперед по программе" или "назад" отличаются на единицу.

Формат 4

Объединяет 3 последних байта в одну большую беззнаковую константу.



Используется довольно редко: для реализации "длинных" безусловных переходов и в двух достаточно сложных системных командах. Примеры рассматривать не будем.

Форматы данных

В данном разделе будут описаны форматы данных, с которыми может работать MMIX.

Начнем с того, что для большинства операций существуют инструкции, которые могут работать с данными двух типов: **числами со знаком и без знака**. Обозначение команд второго типа Кнут дополняет буквой U[6], например, ADD — сложение чисел со знаком (код операции 20) и ADDU — сложение операндов без знака (код 22). Наиболее важное формальное отличие между такими операциями состоит в том, что в модификации с литерой U отсутствует анализ переполнения разрядной сетки машины.

Для того чтобы лучше представить себе разницу между данными без знака и со знаком, рассмотрим самые простые однобайтовые данные. Беззнаковые значения начинаются с 0 и достигают максимума при 255. Чисел со знаком тоже 256, но они "сдвинуты" вдоль числовой оси: их диапазон составляет от −128 до +127 включительно. Рассмотрим следующую таблицу:

Код	**0**	**1**	**2**	**...**	**7F**	**80**	**81**	**...**	**FE**	**FF**
Число без знака	0	1	2	...	127	128	129	...	254	255
Число со знаком	0	1	2	...	127	−128	−127	...	−2	−1

Для положительных чисел она достаточно тривиальна, а вот отрицательная часть таблицы с непривычки вызывает недоумение. Чтобы разобраться в ней, отметим два важных факта. Во-первых, старший бит чисел является знаковым; он равен 1 для отрицательных чисел и 0 для остальных. Во-вторых, отрицательные числа представляются специальным образом (математики называют его **дополнительным кодом**). Более подробно с данным способом представления чисел и его свойствами можно познакомиться, например, в книгах [3, 4]. В частности, в [3] очень хорошо сказано, что

[6] От англ. *Unsign* — без знака.

"к понятию дополнительного кода, ...весьма неудобному для человеческого восприятия, можно подойти естественным путем, выполняя... вычитание из меньшего числа большего. Этот подход не избавляет от неудобств, но может служить определенным утешением, так как показывает, что неудобства не придуманы людьми, а являются неотъемлемым свойством двоичного представления чисел".

Аналогично описанному выше, знаковое и беззнаковое представление информации возможно и для многобайтовых данных. Особо подчеркнем, что изложенная теория справедлива для любой ЭВМ, а не является особенностью MMIX.

Другое деление данных по форматам принято для **целых и вещественных чисел**. Известно (см., например, [4]), что способы их хранения в памяти ЭВМ различны, отсюда имеются два набора арифметических операций — для целых и для вещественных чисел. Например, есть операция сложения двух целых чисел (уже упоминавшаяся ранее инструкция ADD с кодом 20) и аналогичная для вещественных — FADD (ее код — 04). Аналогично обстоит дело и с остальными арифметическими операциями.

Наконец, последнее разнообразие форматов данных имеется в командах чтения из ОЗУ и записи в него. В MMIX существуют инструкции для 1 байта (их мнемоника содержит букву "B"), 2 байт ("W"), 4 байт ("T") и 8 байт ("O"). Интересно, что при описании этой стороны работы MMIX Кнут связывает однобайтовые данные с ASCII-символами, двухбайтовые — с символами нового стандарта Unicode (в данный момент 16-битные данные чаще связывают с устаревшим стандартом целых чисел), а полная 8-байтовая ячейка ассоциируется с числами.

Примеры операций записи в ОЗУ с данными всех четырех типов приведены в следующем разделе.

Наиболее важные команды MMIX

Подобно любому реальному процессору, MMIX обладает не очень сложной системой команд. Не знаю, как вас, уважаемые читатели, а меня этот факт до сих пор продолжает удивлять: глядя на огромное разнообразие столь сложных по поведению программ, трудно себе представить, что в их основе лежит всего пара сотен (а порой и гораздо меньше) разновидностей достаточно простых инструкций процессора!

Под код операции в модели MMIX отводится один байт. Из этого немедленно следует, что количество операций в ее системе команд не может превышать 256. Кнут использовал все 256 возможностей, причем, как он заявил в одном из интервью [5], это не случайно: он принципиально считает, что ни один из кодов не должен оставаться незадействованным.

Очевидно, что не все команды используются одинаково часто и не все одинаково важны. Поэтому давайте познакомимся только с наиболее важными группами команд, которые способен выполнять MMIX. Именно их мы сможем как-то использовать в учебном процессе (речь, разумеется, не идет о подготовке компьютерных специалистов высокого уровня, когда необходимо изучать *полную* систему команд процессора).

Команды переписи информации

Начнем с наиболее простых команд — команд переписи информации. Эти инструкции не производят никакой обработки информации, а только переписывают информацию из одного места в другое. В качестве источника и приемника информации могут выступать как регистры, так и ячейки памяти компьютера. В MMIX команды пересылки спроектированы следующим образом.

Прежде всего выделены операции считывания данных из памяти и их записи в память. Рассмотрим для примера запись в память. Полное содержимое конкретного регистра $X или его части (8, 16, 32 или 64 бита) можно записать, используя одну из операций, код которой находится в диапазоне A0 — AF. Конкретный код зависит от того, сколько байт требуется записать в память, а также нужно ли при этом учитывать знак числа или нет. Кроме того, как уже упоминалось при обсуждении форматов команд, адрес ячейки памяти может вычисляться одним из двух возможных способов:

регистр + *регистр* или *регистр* + *константа*.

В итоге получаем 16 инструкций записи информации в ОЗУ:

Код	Мнемоника	Действие
A0	STB $X, $Y, $Z	Записать 1 байт со знаком из $X в $Y + $Z
A1	STBI $X, $Y, Z	Записать 1 байт со знаком из $X в $Y + Z
A2	STBU $X, $Y, $Z	Записать 1 байт без знака из $X в $Y + $Z
A3	STBUI $X, $Y, Z	Записать 1 байт без знака из $X в $Y + Z
A4	STW $X, $Y, $Z	Записать 2 байта со знаком из $X в $Y + $Z
...		
A8	STT $X, $Y, $Z	Записать 4 байта со знаком из $X в $Y + $Z
...		
AC	STO $X, $Y, $Z	Записать 8 байтов со знаком из $X в $Y + $Z
...		

Примечание. В таблице специально пропущены некоторые строчки. Они легко поддаются восстановлению чисто логическим путем. Советую читателям проделать это в качестве несложного упражнения.

Отметим, что запись в память чисел со знаком и без знака осуществляется немного по-разному, но отличия пока рассматривать не будем.

Совершенно аналогично строится таблица для операций чтения данных из памяти.

Наконец, обсуждая команды переписи информации, нельзя не упомянуть столь необходимые на практике инструкции для задания значений регистров. Они со-

держат двухбайтовую константу (см. формат 3), которая и заносится в заданную кодом операции "четверть" регистра. Например, команда SETH (код E0) определяет старшие 2 байта, а SETL (код E3) — младшие. При этом все остальные байты обнуляются. Инструкции E4 — E7 делают то же самое, что E0 — E3, но значения неопределяемых байтов сохраняются без изменений. Наличие таких инструкций позволяет задавать содержимое регистра "по частям", используя несколько последовательных команд.

Чтобы закрепить все эти многочисленные возможности, предлагаем читателю внимательно разобрать приведенную ниже последовательность команд

Код	Мнемоника	Действие	Комментарий
E0 01 1234	SETH $1, 1234	$1 ← 1234 0...0	Константа
E5 01 5678	INCMH $1, 5678	$1 ← 1234 5678 0...0	(Дополнить ее)
E3 02 0100	SETL $2, 0100	$2 ← 0...0 0100	Адрес
E5 02 2000	INCMH $2, 2000	$2 ← 2000 0000 0100	(Дополнить его)
AD 01 02 00	STOI $1, $2, 0	($2+0) ← $1	Записать в ОЗУ

и убедиться, что они заносят 8-байтовую константу 1234 5678 0000 0000 в память начиная с адреса 0000 2000 0000 0100.

Арифметические операции

Другую, не менее важную, группу команд образуют арифметические операции (вспомним, что своим названием ЭВМ были обязаны именно вычислительным действиям).

Арифметические инструкции в MMIX реализуются достаточно просто и единообразно. В соответствии с форматом 1 или 2 выполняется необходимое вычислительное действие над двумя регистрами или регистром и константой, а результат всегда помещается в регистр, указанный в команде первым. Например, инструкция вычитания двух чисел с учетом знака 24 01 02 03 (ее мнемоника SUB) выполняется так: из содержимого регистра $2 вычитается значение, находящееся в $3, и результат записывается в $1.

Примеры некоторых арифметических операций приведены в следующей таблице.

Код	Мнемоника	Действие
20	ADD $X, $Y, $Z	$X ← $Y + $Z (целые числа со знаком)
24	SUB $X, $Y, $Z	$X ← $Y − $Z (целые числа со знаком)
18	MUL $X, $Y, $Z	$X ← $Y * $Z (целые числа со знаком)
1C	DIV $X, $Y, $Z	$X ← $Y / $Z (целые числа со знаком)
04	FADD $X, $Y, $Z	$X ← $Y + $Z (вещественные числа)
06	FSUB $X, $Y, $Z	$X ← $Y − $Z (вещественные числа)
10	FMUL $X, $Y, $Z	$X ← $Y * $Z (вещественные числа)
14	FDIV $X, $Y, $Z	$X ← $Y / $Z (вещественные числа)

Следует иметь в виду, что в таблице приведены далеко не все арифметические операции MMIX, но для решения простейших задач содержимого таблицы вполне достаточно.

Особо в этой группе хочется выделить команды сравнения данных. При сравнении из $Y вычитается $Z и **знак** результата записывается в $X: так, при $Y < $Z в $X записывается −1; при $Y > $Z результат равен +1, а при $Y = $Z — нулю.

В качестве образца решения на MMIX вычислительной задачи см. пример 1 в следующем разделе.

Логические операции

Обычно процессоры "обучены" выполнять 3 базовых логических операции (И, ИЛИ, НЕ). Нередко этот набор дополняет исключающее ИЛИ. MMIX в этом смысле выглядит гораздо мощнее: он умеет выполнять 16(!) логических операций, включая, разумеется, все базовые.

Очень оригинально реализована в MMIX операция логического отрицания НЕ. Обычно данная инструкция отличается от остальных, поскольку у нее не два, а только один операнд. Кнут предлагает реализовать НЕ как частный случай другой более сложной комплексной операции. Например, можно взять за основу инструкцию с кодом C5, которая вычисляет NOT ($Y OR Z), положив Z = 0. Нетрудно видеть, что при этом фактически получается требуемая нам операция NOT $Y.

Особо хотелось бы подчеркнуть, что логические инструкции всегда являются *поразрядными*, т.е. выполняются над каждым разрядом (битом) в отдельности независимо от результатов обработки всех остальных битов. Поэтому часто обсуждаемые операции используются для выделения отдельных битов кода (вы, конечно, помните, что отдельно взятый бит нельзя извлечь из ОЗУ!), а также для сброса или установки отдельных битов информации. Вот типичный пример: логическая инструкция И с кодом C9 01 01 20 сбрасывает 5-й бит в коде символа и тем самым преобразует любую букву в заглавную. Команда ИЛИ, имеющая код C1 01 01 20, напротив, устанавливает этот бит в 1, что соответствует строчным буквам.

Операции управления

В эту обширную группу обычно относят довольно большое число операций: прежде всего всевозможные переходы, а также команды, связанные с остановкой процессора или изменением режима его работы.

Начнем рассмотрение с инструкций переходов. Для реализации реальных программ, команд переписи и разнообразных действий по обработке информации явно недостаточно. В самом деле, с их помощью мы можем реализовать лишь линейный алгоритм обработки данных. Для того чтобы написать разветвляющуюся или циклическую программу, требуется иметь дополнительные команды, позволяющие изменять порядок исполнения инструкций программы. С этой целью в любом процессоре есть команды переходов.

Переходы могут быть **безусловные** и **условные**. В первом случае переход выполняется всегда и вне зависимости от результатов предыдущих действий. Во втором случае поведение процессора будет более сложным. Если

результат предыдущей операции удовлетворяет определенному условию, то переход "срабатывает", а в противном случае он игнорируется и обычным порядком выполняется *следующая* за условным переходом инструкция. Читатели, знакомые хотя бы с одним языком программирования, без сомнения, узнали в описанном алгоритме логику условной конструкции IF — THEN. Разумеется, это не случайно, ибо первые языки программирования строились по аналогии с внутренним языком процессора, а значит, не могли не быть на него похожи.

В MMIX существует достаточно большой ассортимент **условных переходов**. Некоторые из них приведены в таблице:

Код	Мнемоника	Условие	Действие
40	BN $X, S	< 0	Переход "вперед" при < 0 (if Negative)
41	BNB $X, S	< 0	Переход "назад" при < 0 (if Negative)
42	BZ $X, S	$= 0$	Переход "вперед" при $= 0$ (if Zero)
44	BP $X, S	> 0	Переход "вперед" при > 0 (if Positive)
48	BNN $X, S	≥ 0	Переход "вперед" при ≥ 0 (if NonNegative)

Всего в MMIX 16 переходов такого типа; из них ровно половина является переходами "вперед по программе" (выражаясь более точно, в сторону увеличения адресов), а остальные "назад". Коды переходов "назад" всегда на единицу больше.

В каждой команде указывается регистр, содержимое которого MMIX проверяет в соответствии с кодом операции на справедливость определенного условия. Как нетрудно сообразить, таких условий ровно 8: из них 6 — общепринятые проверки на неравенство ($< 0, = 0, > 0, \geq 0, <>0, \leq 0$), а оставшиеся контролируют четность и нечетность.

Последним параметром команды перехода является смещение S, которое показывает, сколько команд, считая от команды перехода, надо пропустить; при переходе "назад" $S < 0$.

Конкретные примеры команд перехода можно посмотреть в задачах 2, 3 следующего раздела.

Помимо описанных выше 16 условных переходов, Кнут вводит еще 16 функционально аналогичных переходов (коды 50—5F), но с другим временем исполнения, а также 32 условных переписи (коды 60—7F). Подробности этих операций рассматривать не будем.

Команда **безусловного перехода** (F0 "вперед" и F1 "назад") имеет особый формат (см. формат 4), что позволяет задавать смещение не в двух байтах, а в трех. Все остальные правила полностью аналогичны рассмотренным ранее для смещения условных переходов.

Завершая обсуждение группы управляющих инструкций, поговорим подробнее о **проблеме окончания программы**. Раньше (в машинах первого и второго поколений) вопроса о том, что делать после окончания решения задачи, просто не стояло — ЭВМ пассивно ожидала, пока оператор введет следующую задачу. Поэтому каждая программа кончалась специальной командой, завершающей работу (на программистском жаргоне ее почему-то называли "**останов**"). В ответ на данную инструкцию ЭВМ прекращала всякую деятельность, и из состояния ожидания ее выводил оператор, нажимая те или иные кнопки на пульте управления машины. В третьем поколении ЭВМ, когда машины стали коллективными и многозадачными, такое поведение стало неразумным. Появилась операционная система, автоматически управляющая прохождением задач; поэтому, завершая свою работу, каждая программа стала "возвращать[7] управление" ОС. В персональных компьютерах данная философия завершения программ сохранилась (а кстати, попробуйте представить, как бы вам работалось, если бы после выхода из текстового редактора процессор каждый раз останавливался!).

MMIX, претендующий на роль процессора нового тысячелетия, не имеет команды останова в принципе. Для завершения любой программы Кнут рекомендует передавать управление несуществующей пока операционной системе под названием NNIX: для этого последней командой программы пишется специальная системная инструкция TRAP 0, имеющая нулевой код.

Отметим, что, используя модификацию TRAP 1, Кнут предлагает в данный момент решать проблему обмена с внешними устройствами: специальных команд ввода/вывода MMIX также не имеет.

Примеры простых программ

Приведем несколько примеров простейших программ для MMIX. В описании Кнута, по-моему, подобных примеров очень не хватает (будем надеяться, что в окончательном варианте текста книги они все же появятся).

Представленные примеры содержат линейный, разветвляющийся и циклический фрагменты, а также демонстрируют простейшие приемы работы с байтовыми данными и массивами. Такой подбор задач позволяет автору надеяться, что, просмотрев их, читатель получит некоторое представление о программе для MMIX.

Во всех примерах дополнительно ассемблерная мнемоника, предложенная Д.Кнутом. Хотя ее подробное обсуждение выходит за рамки данной статьи, я все же решил сохранить данный столбец, поскольку он во многом интуитивно понятен и частично облегчает разбор кода. В частности, пользуясь мнемоникой, легко видеть, где стоит номер регистра, а где константа.

1. Пример на вычисления

Пусть мы хотим вычислить сумму двух целых чисел: A, которое хранится с адреса 110_{16}, и B (118_{16}). Программа для решения этой простейшей задачи на MMIX приведена на с. 25:

[7] Именно *возвращать* управление, поскольку при запуске программа стартовала по инициативе ОС.

Напомним читателям, что команды в MMIX занимают по 4 байта (отсюда нумерация в первом столбике), а целые числа — 8 байт.

Интересно, что для нахождения суммы вещественных чисел достаточно просто изменить код операции сложения в предпоследней команде.

Адрес	Код	Мнемоника	Действие	Комментарий
0...0	E3 01 0110	SETL $1, 0110	$1 ← 0...110	Адрес A в $1
0...4	8D 00 01 00	LDO $0, $1, 0	$0 ← ($1+0)	Считать A из ОЗУ в $0
0...8	8D 02 01 08	LDO $2, $1, 8	$2 ← ($1+8)	Считать B из ОЗУ в $2
0...C	20 00 00 02	ADD $0, $0, $2	$0 ← $0 + $2	A + B в $0
0...10	00 000000	TRAP 0	Выход в ОС	Завершение задачи

2. Модуль числа

Вычислим модуль целого числа N, хранящегося в регистре $1, и поместим результат в регистр $2. Вспомним, что для неотрицательных чисел модуль равен самому числу, а в противном случае результат равен разности 0 — N.

Адрес	Код	Мнемоника	Действие	Комментарий
0...0	21 02 01 00	ADDI $2, $1, 0	$2 ← $1 + 0	Скопировать $1 в $2
0...4	48 02 0002	BNN $0, +2	к 0...C при $2 ≥ 0	Обход, если $2 ≥ 0
0...8	34 02 00 02	NEG $2, 0, $2	$2 ← 0 − $2	Получить −N
0...C	00 000000	TRAP 0	Выход в ОС	Завершение задачи

Как оказалось, MMIX не имеет специальных инструкций для переписи содержимого из одного регистра в другой. Поэтому в первой команде программы ее пришлось искусственно создать путем сложения с нулем (в описании Кнута мне не попадалось никаких рекомендаций по этому поводу, но применяемая хитрость выглядит вполне "по-кнутовски").

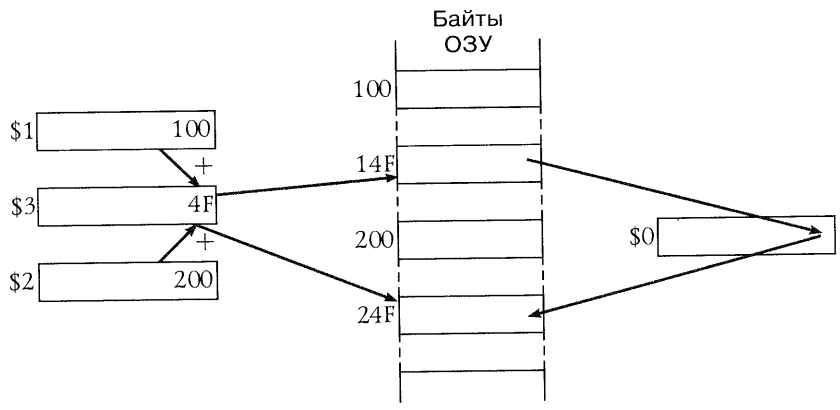
3. Копирование массива

Скопируем байты 100—14F в область 200—24F.

С технической точки зрения оказывается удобнее копировать массив начиная с бóльших адресов. Начальный адрес исходного массива помещается в $1, а адрес копии — в $2. Для смещения по массиву используется регистр $3, пробегающий значения от 4F до 0 (последний копируемый байт). При следующем вычитании в $3 получается отрицательный результат, и цикл завершается.

Адрес	Код	Мнемоника	Действие	Комментарий
0...0	E3 01 0100	SETL $1, 0100	$1 ← 0...100	Адрес I массива
0...4	E3 02 0200	SETL $2, 0200	$2 ← 0...200	Адрес II массива
0...8	E3 03 004F	SETL $3, 004F	$3 ← 0...4F	Для последнего адреса
0...C	82 00 01 03	LDBU $0, $1, $3	$0 ← ($1 + $3)	Прочитать байт
0...10	A2 00 02 03	STBU $0, $2, $3	($2 + $3) ← $0	Записать байт
0...14	25 03 03 01	SUBI $3, $3, 1	$3 ← $3 − 1	−1 из счетчика
0...18	48 03 FFFD	BNN $3, −3	К 0...C при $3 ≥ 0	К повторению
0...10	00 000000	TRAP 0	Выход в ОС	Завершение задачи

Регистр $0, как уже догадался читатель, использован в качестве рабочего для временного хранения значения очередного байта.



Чем может быть полезен MMIX нам?

Завершим на этом техническую часть рассказа. Конечно, многое еще осталось за кадром, но для первого знакомства вполне достаточно. Настало время выполнить обещание и обсудить вопросы, связанные с применением модели в учебном процессе. Итак, как же можно использовать MMIX в учебном процессе?

Начнем с анализа обязательного минимума содержания курса информатики [6], который является важнейшим нормативным документом и определяет тематику **школьного курса информатики**. Как известно, он предлагает два уровня: А и Б, различающиеся количеством часов и, соответственно, степенью подробности изложения материала. Уровень А рекомендован для школ и классов, где к курсу не предъявляются высокие требования, например, в случае гуманитарного профиля обучения. Для остальных предназначается уровень Б, который дополнительно предполагает достаточное обеспечение учебного заведения компьютерами.

В списке тем, предлагаемых для **уровня А**, можно выделить два круга вопросов, где знакомство с теми или иными аспектами работы MMIX может быть полезно: представление информации и программное управление работой компьютера.

Представление информации

"Кодирование. Двоичная форма представления информации. Количество и единицы измерения информации".

Модель MMIX, которая оперирует с различными форматами данных, вполне подходит при изучении данной тематики. Читатель, внимательно ознакомившийся с разделом о форматах данных MMIX, наверняка и сам заметил это. Работа с одно-, двух-, четырех- и восьмибайтовой информацией, возможность хранения ASCII- и Unicode-символов, целые числа со знаком и без знака, вещественные числа — вот основа для рассмотрения теоретических вопросов и задач по кодированию информации на базе MMIX. К сожалению, в описании MMIX явно не оговорены детали работы с графической и другими современными видами мультимедийной информации. Но, с другой стороны, анализ имеющихся материалов показывает, что это возможно. Более того, в Интернете уже имеется Windows-версия MMIX с поддержкой черно-белой графики [7], но, к сожалению, мне пока не удалось в ней полностью разобраться (во всяком случае, демонстрационный пример, рисующий окружность, она выполняет!).

По мнению автора, более тесное соединение вопросов представления информации с фундаментальными основами устройства компьютера может стать достаточно плодотворным, так как добавляет интересный материал и задачи, имеющие практический выход.

Программное управление работой компьютера

Начиная с учебника А.П. Ершова [8], где очень четко и в доступной ученикам форме сформулирован основной алгоритм работы процессора, авторы многих учебников по информатике уделяют этому материалу достаточное внимание (см., например, [9], [10]). Рассмотрение данного вопроса ведется либо на примере "настоящей" ЭВМ [10], либо с помощью учебной модели компьютера [9]. Учитывая, что удобные для изучения машины с архитектурой PDP-11 ("ДВК" "БК", "УКНЦ") уже ушли в прошлое, а придумывавшие процессор Intel инженеры, мягко говоря, не очень позаботились об удобстве его изучения, путь с учебной моделью сейчас становится все более предпочтительным. Отсюда MMIX как хорошо продуманная учебная модель тоже может быть использована при изложении данного вопроса.

Перейдем теперь к **уровню Б**. Помимо уже рассмотренных выше вопросов, сюда входят некоторые дополнительные темы, интересные с точки зрения нашего обсуждения. По сравнению с уровнем А, здесь добавился целый раздел: *Системы счисления и основы логики.* В него вошли несколько описываемых ниже проблем.

Системы счисления

"Двоичная система счисления. Двоичная арифметика. Системы счисления, используемые в компьютере".

Связь данных вопросов с учебной моделью MMIX довольно очевидна: достаточно назвать представление данных и команд в двоичной и шестнадцатеричной системах счисления, кодирование отрицательных чисел, выполнение арифметических операций.

Логические выражения

"Основные понятия и операции формальной логики. Логические выражения и их преобразование. Построение таблиц истинности логических выражений".

MMIX с его 16 логическими операциями подойдет здесь как нельзя лучше. А как было бы замечательно проверить построенную "вручную" таблицу истинности на работающем имитаторе MMIX!

Хочется отметить, что часто логические выражения обсуждаются применительно к логическим элементам (микросхемам). Если это сопровождается какой-то практической работой, например, изучением таблицы истинности для 1—3 логических микросхем с подключенными к ним светодиодами, такой путь представляется замечательным! А если логический элемент только нарисован мелом на доске? Не знаю как вас, уважаемые читатели, а меня это не убеждает. В этом смысле вариант с проверкой составленной в тетради таблицы с помощью учебной ЭВМ мне кажется более впечатляющим.

Примечание. Изучение логических операций на основе составления запросов к базам данных хотя и является очень наглядным, но в такой ситуации операции не являются *битовыми.*

Основные логические элементы компьютера (регистр, сумматор)

Хотя данное предложение и может вызвать некоторые возражения, но рассказ о многочисленных регистрах MMIX здесь может быть хорошим дополнением.

Есть еще одно направление курса, которое можно связать с моделью учебного компьютера, хотя явной ссылки в тексте не содержится. Речь идет о понятии исполнителя, а точнее — о системе его команд. В подтверждение данного положения сошлюсь на рекомендуемую формулировку билета № 17 [11], где есть важная ключевая фраза: "Компьютер как формальный исполнитель алгоритмов (программ)". И действительно, очень важно понять, в чем состоит сходство и в чем особенности системы команд компьютера по сравнению, например, с Кенгуренком или Роботом. Поэтому изучение наиболее типичных машинных инструкций с этой точки зрения также представляет определенный интерес.

Завершая обсуждение возможностей применения MMIX в учебном процессе, хочется внести еще два дополнительных предложения по использованию учебных ЭВМ в курсе информатики. Первое касается возможности **показать реализацию всех базовых алгоритмических структур** (следование, развилка, цикл) на уровне инструкций процессора. Имеющийся в данной статье раздел с примерами программ иллюстрирует, как, по мнению автора, это можно сделать. И вторая, менее очевидная возможность: **связать**

изучение учебной модели компьютера с темой *Программное обеспечение ЭВМ*. Дело в том, что знакомство с готовым программным обеспечением при традиционных методах изложения кажется никак не связанным с изучением теоретических основ информатики — всех этих наборов нулей и единиц, логических операций и многого другого. Именно таким абсолютно несвязанным образом основная масса учеников и воспринимает теорию, которую им рассказывают. Неудивительно, что по поводу необходимости ее изучения они совершенно искренне заявляют: "А зачем все это нужно?" В качестве попытки перебросить мостик между фундаментальными основами информатики и программными продуктами на базе учебной модели компьютера сошлюсь на публикации [12], [13]. Аналогичный подход в принципе может быть применен к архиваторам, простейшим программам поиска и обработки текстов и многим другим.

Таким образом, имеется довольно много точек потенциального применения модели MMIX (или другой, которая вам больше по душе) даже в рамках *школьного* курса информатики. Что касается **более детальных курсов**, например, для углубленного изучения или в средних и высших учебных заведениях, то там возможностей еще больше. По определению напрашивается применение MMIX при изучении численных методов (помните основную цель Дональда Кнута при создании MMIX?). То же самое можно сказать по поводу всевозможных курсов по основам вычислительной техники и программирования на ассемблере.

Автор планирует продолжать работу по внедрению компьютера MMIX в учебный процесс, включая создание действующего имитатора упрощенной модели. Все материалы будут опубликованы на сайте [14], посвященном учебным моделям компьютеров (возможен также доступ по переадресуемой ссылке [15]). Все пожелания, комментарии и предложения, в том числе о возможном сотрудничестве, будут с благодарностью приняты.

Литература

1. *Кнут Д.* Искусство программирования на ЭВМ. Т. 1. М.: Мир, 1976, 736 с.

2. Официальная страница MMIX (на английском языке). *http://www-cs-faculty.stanford.edu/~knuth/mmix.html*

3. *Бруснецов Н.П.* Микрокомпьютеры. М.: Наука, 1985, 208 с.

4. *Андреева Е., Фалина И.* Системы счисления и компьютерная арифметика. М.: Лаборатория Базовых Знаний, 1999, 256 с.

5. MMIX — RISC-компьютер для нового тысячелетия: аудиозапись интервью с Дональдом Кнутом (на английском языке). *http://www.technetcast.com/tnc_play_stream.html?stream_id=199*

6. Обязательный минимум содержания среднего (полного) общего образования по информатике./В сб. Программно-методические материалы: Информатика. 7—11-е кл. / Сост. — Л.Е. Самовольнова. М.: Дрофа, 2001. С. 6—18.

7. *Боетпчер А.* Windows-версия имитатора MMIX с видеопамятью (на английском языке). *http://www.informatik.fh-muenchen.de/~mmix/mmixwin/mmixwin.html*

8. Основы информатики и вычислительной техники: Пробное учебное пособие для средних учебных заведений. В 2 ч. Ч. 2 / А.П. Ершов, В.М. Монахов, А.А. Кузнецов и др.; Под ред. А.П. Ершова, В.М. Монахова. М.: Просвещение, 1986, 143 с.

9. Основы информатики и вычислительной техники: Пробное учебное пособие для 10—11-х классов средней школы / В.А. Каймин, А.Г. Щеголев, Е.А. Ерохина, Д.П. Федюшкин. М.: Просвещение, 1989, 272 с.

10. *Кушниренко А.Г. и др.* Информатика. 7—9-е кл.: Учебник для общеобразовательных учебных заведений / А.Г. Кушниренко, Г.В. Лебедев, Я.Н. Зайдельман. М.: Дрофа, 2000, 336 с.

11. *Кузнецов А.А., Угринович Н.Д., Цветкова М.С.* Материалы для подготовки и проведения итоговой аттестации выпускников IX классов общеобразовательных учреждений в 2001/2002 учебном году. Информатика и образование № 1, 2002. С. 10—12.

12. *Еремин Е.А.* Компилятор? Это очень просто. Компьютер УКНЦ. М.: Компьютика № 3, 1995. С. 25—33.

13. *Еремин Е.А.* Компилятор? Это довольно просто! Информатика № 40, 2001, с. 7—17; № 43, с. 7—14; № 45, с. 21—29; № 46, с. 19—25; № 47, с. 8—10.

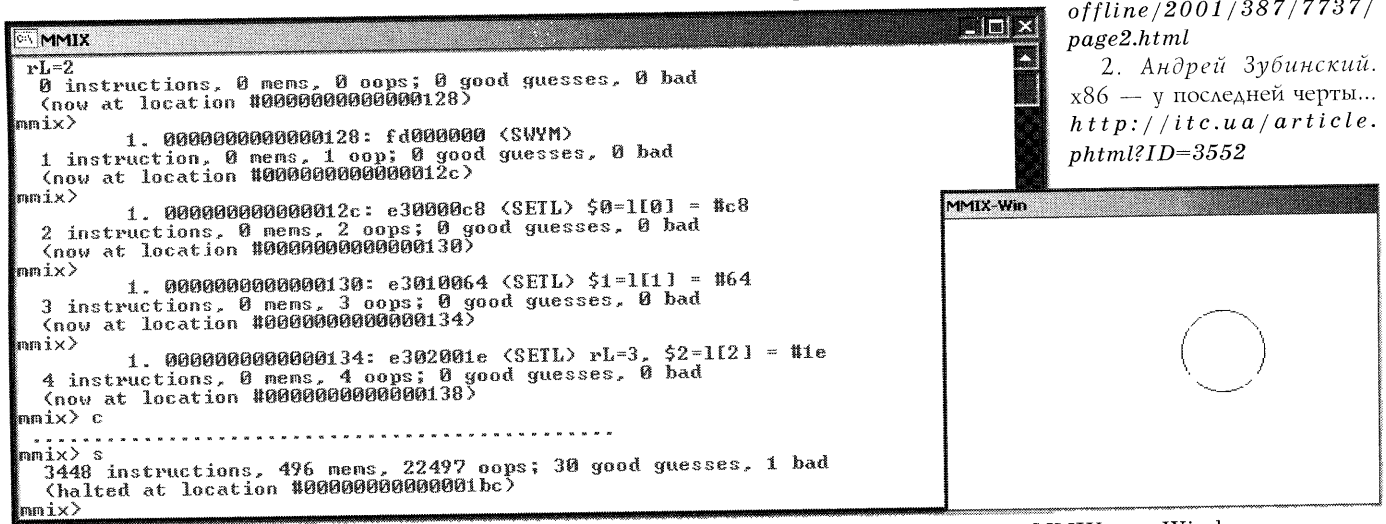
14. Учебные модели компьютера. *http://emc.km.ru*

15. Проект rEd-MMI. *http://emmi.4u.ru*

Дополнительный материал

1. *Игорь Гордиенко.* Путь без конца. *http://www.computerra.ru/offline/2001/387/7737/page2.html*

2. *Андрей Зубинский.* x86 — у последней черты... *http://itc.ua/article.phtml?ID=3552*



Результат выполнения программы построения окружности на интерпретаторе MMIX для Windows.

Вычислительная геометрия на плоскости

Е.В. Андреева, Ю.Е. Егоров,
Москва

Продолжение. Начало в № 39/2002

Взаимное расположение точек и фигур

2.1. Расположение точки относительно прямой, луча или отрезка.

В первую очередь в этой задаче нас интересует принадлежность данной точки $P(x, y)$ указанному геометрическому объекту, уравнение которого нам известно (либо может быть легко получено). Чтобы ответить на этот вопрос для прямой, достаточно подставить координаты заданной точки в уравнение прямой (3) или (4). Равенство нулю значения полученного выражения (для вещественных координат или коэффициентов уравнения проверку на равенство нулю необходимо осуществлять с учетом погрешности) означает, что точка принадлежит данной прямой. Если значение выражения меньше нуля, то точка лежит в одной полуплоскости от прямой, если больше нуля — в другой. Действительно, уравнение (3) получено в предположении, что прямая проходит через точки $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$. Заметим, что левая часть (3) — это значение косого произведения векторов $\overrightarrow{P_1P}$ и $\overrightarrow{P_1P_2}$. Его знак определяет ориентацию этой пары векторов, иначе говоря, принадлежность точки P одной из полуплоскостей (а равенство нулю — принадлежность прямой).

Если прямая изначально задана двумя своими точками, то для решения данной задачи достаточно вычислить значение указанного косого произведения, а уравнение прямой выписывать не нужно.

В случае проверки принадлежности точки лучу при равенстве $[\overrightarrow{P_1P}, \overrightarrow{P_1P_2}]$ нулю (здесь P_1 — начало луча, а P_2 — любая точка, принадлежащая лучу) полезно вычислить и скалярное произведение этих же векторов. Если оно меньше нуля, то P_1 лежит на прямой между P_2 и P, следовательно, P лучу не принадлежит. Чтобы в аналогичной ситуации убедиться в принадлежности точки P отрезку P_1P_2, необходимо вычислить еще и значение скалярного произведения $(\overrightarrow{P_2P_1}, \overrightarrow{P_2P})$. Если оно неотрицательно, то точка P лежит на отрезке.

Пусть теперь нам требуется определить, на каком расстоянии находится заданная точка P от определенной прямой, луча или отрезка. Формула для расстояния от точки до прямой получается из сопоставления двух способов вычисления площади треугольника: $2S = h \cdot |P_1P_2| = |[PP_1, PP_2]|$ (см. *рис.* 8). То есть расстояние h от точки P до прямой, заданной координатами точек P_1 и P_2, можно подсчитать как отношение модуля косого произведения векторов $\overrightarrow{PP_1}$ и $\overrightarrow{PP_2}$ к длине отрезка P_1P_2.

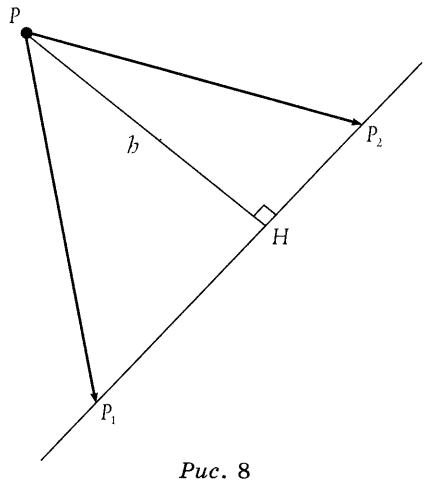


Рис. 8

Для луча или отрезка указанный способ нахождения расстояния нужно слегка подкорректировать. Точка H (*рис.* 8) принадлежит лучу P_1P_2 в том и только том случае, когда скалярное произведение $(\overrightarrow{P_1P_2}, \overrightarrow{P_1P}) \geq 0$. Для отрезка P_1P_2, конечно, нужно еще и выполнение условия $(\overrightarrow{P_2P_1}, \overrightarrow{P_2P}) \geq 0$. Тогда применима формула расстояния от точки до прямой. В противном случае расстояние от точки до луча (отрезка) будет равно расстоянию от точки до начала луча (до ближайшего конца отрезка).

2.2. Взаимное расположение двух точек относительно прямой.

Обычно требуется просто выяснить, по одну или разные стороны от определенной прямой лежат две точки P_1 и P_2, заданные своими координатами. Выберем на прямой две произвольные несовпадающие точки — P_3 и P_4 (ими могут быть как раз те точки, которыми определена наша прямая). Тогда, если косые произведения $[\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}]$ и $[\overrightarrow{P_3P_2}, \overrightarrow{P_3P_4}]$ имеют разный знак, то есть $[\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}] \cdot [\overrightarrow{P_3P_2}, \overrightarrow{P_3P_4}] < 0$, то точки лежат по разные стороны от прямой; если знаки совпадают — то по одну сторону, а если одно из них равно 0, то соответствующая точка лежит на исходной прямой. Вообще же эта задача является частью следующей, более сложной, задачи.

2.3. Взаимное расположение двух прямых или прямой и отрезка.

Легко выяснить, пересекаются ли две прямые или параллельны. Напомним еще раз, что условие коллинеарности двух векторов — это равенство нулю их косого произведения. Если прямые заданы уравнениями $a_1x + b_1y + c_1 = 0$ и $a_2x + b_2y + c_2 = 0$, то удобно перейти к их нормалям $\boldsymbol{n}_1 = (a_1, b_1)$ и $\boldsymbol{n}_2 = (a_2, b_2)$. Тогда условие

коллинеарности нормалей (а значит, и параллельности прямых): $[\boldsymbol{n}_1, \boldsymbol{n}_2] = a_1 b_2 - a_2 b_1 = 0$. Если прямые заданы парами точек, то таким же способом проверяется коллинеарность направляющих векторов. Проверка наличия пересечения прямой и отрезка нами фактически уже была произведена при решении задачи 2.2.

Пусть прямая и отрезок пересекаются в одной точке. Найдем ее.

Обозначим концы отрезка $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$. Пусть $P_3(x_3, y_3)$ и $P_4(x_4, y_4)$ — две точки на прямой, а M — искомая точка пересечения (*рис.* 9). Опустим из концов отрезка на прямую перпендикуляры P_1H_1 и P_2H_2. Треугольники $P_3P_1P_4$ и $P_3P_2P_4$ имеют общее основание. Следовательно, отношение их площадей равно отношению их высот $|P_1H_1|$ и $|P_2H_2|$. Но площадь (ориентированная) треугольника $P_3P_1P_4$ — это $\frac{1}{2}[\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}]$, аналогично выражается площадь $P_3P_2P_4$. То есть

$$\frac{|[\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}]|}{|[\overrightarrow{P_3P_2}, \overrightarrow{P_3P_4}]|} = \frac{|P_1H_1|}{|P_2H_2|} = \frac{|P_1M|}{|P_2M|}.$$

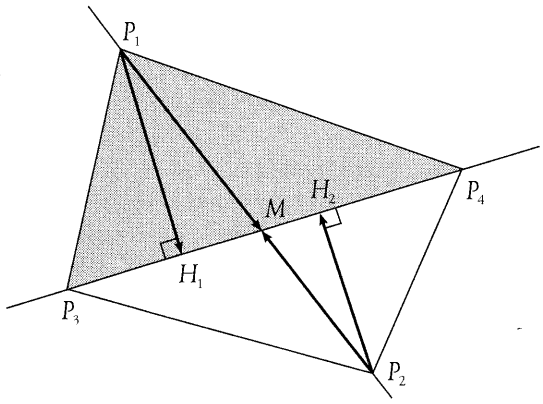


Рис. 9

Последнее равенство вытекает из подобия треугольников P_1H_1M и P_2H_2M. Отсюда получаем

$$[\overrightarrow{P_3P_2}, \overrightarrow{P_3P_4}] \cdot \overrightarrow{P_1M} = [\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}] \cdot \overrightarrow{P_2M}. \qquad (12)$$

В этой формуле учтено и то, что одна из площадей может быть равна нулю, и то, что указанные площади имеют одинаковые знаки в точности тогда, когда точки P_1 и P_2 лежат по одну сторону от прямой P_3P_4 (то есть когда векторы $\overrightarrow{P_1M}$ и $\overrightarrow{P_2M}$ сонаправлены). Мы нашли, в каком отношении делится этот отрезок точкой M (внутренним или внешним образом). С другой стороны, $\overrightarrow{P_2M} = \overrightarrow{P_1M} - \overrightarrow{P_1P_2}$. Подставим это соотношение в (12). Далее выразить $\overrightarrow{P_1M}$ — дело техники:

$$\overrightarrow{P_1M} = \frac{[\overrightarrow{P_3P_4}, \overrightarrow{P_3P_1}]}{[\overrightarrow{P_3P_2}, \overrightarrow{P_3P_4}] - [\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}]} \overrightarrow{P_1P_2} = \qquad (13)$$

$$= \frac{[\overrightarrow{P_3P_4}, \overrightarrow{P_3P_1}]}{[\overrightarrow{P_3P_2} - \overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}]} \overrightarrow{P_1P_2} = \frac{[\overrightarrow{P_3P_4}, \overrightarrow{P_3P_1}]}{[\overrightarrow{P_1P_2}, \overrightarrow{P_3P_4}]} \overrightarrow{P_1P_2}.$$

(Здесь мы использовали свойство линейности косого произведения: $[\overrightarrow{P_3P_2}, \overrightarrow{P_3P_4}] - [\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}] = [\overrightarrow{P_3P_2} - \overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}]$, которое легко устанавливается, если записать это равенство в координатах.) Теперь, зная координаты точки P_1 и вектор $\overrightarrow{P_1M}$, находим точку M.

Таким образом, мы нашли точку пересечения прямой и отрезка. Для нахождения пересечения двух прямых на одной из них выбираются произвольные точки P_1 и P_2, на другой — P_3 и P_4.

Может быть, вывод (13) не совсем элементарен, но саму формулу легко запомнить, если понять ее геометрический смысл. Так, если отрезки P_1P_2 и P_3P_4 пересекаются, то площадь четырехугольника $P_1P_2P_3P_4$ по известной формуле есть половина произведения диагоналей на синус угла между ними. То есть по абсолютной величине косое произведение векторов $\overrightarrow{P_1P_2}$ и $\overrightarrow{P_3P_4}$, стоящее в знаменателе формулы (13), — это удвоенная площадь четырехугольника $P_1P_2P_3P_4$. В числителе же мы имеем удвоенную площадь треугольника $P_3P_4P_1$.

2.4. Определение взаимного расположения двух отрезков или лучей.

Проверить наличие пересечения у двух отрезков (а зачастую нас интересует лишь сам факт пересечения) несложно опять же с использованием косого произведения. Пусть первый отрезок задан точками P_1 и P_2, а второй — P_3 и P_4. Отрезки пересекаются тогда, когда, во-первых, пересекаются ограничивающие их прямоугольники и, во-вторых, концы каждого отрезка лежат по разные стороны от прямой, которой принадлежит другой отрезок. Первое из этих условий позволяет не рассматривать отдельно тот случай, когда отрезки лежат на одной прямой.

Обозначим x_{max1} и x_{min1} — максимальную и минимальную из x-координат первого отрезка, x_{max2} и x_{min2} — второго отрезка. Для y-координат имеем аналогично y_{max1}, y_{min1}, y_{max2} и y_{min2}. Тогда условия пересечения отрезков формально можно записать так:

1) $x_{max1} \geq x_{min2}$, $x_{max2} \geq x_{min1}$, $y_{max1} \geq y_{min2}$ и $y_{max2} \geq y_{min1}$;

2) косые произведения $[\overrightarrow{P_1P_3}, \overrightarrow{P_1P_2}]$ и $[\overrightarrow{P_1P_4}, \overrightarrow{P_1P_2}]$ имеют разный знак, точнее, $[\overrightarrow{P_1P_3}, \overrightarrow{P_1P_2}] \cdot [\overrightarrow{P_1P_4}, \overrightarrow{P_1P_2}] \leq 0$;

3) аналогично $[\overrightarrow{P_3P_1}, \overrightarrow{P_3P_4}] \cdot [\overrightarrow{P_3P_2}, \overrightarrow{P_3P_4}] \leq 0$.

Если факт наличия пересечения нами уже установлен, то для отрезков, находящихся на пересекающихся прямых, точка пересечения ищется, как и в задаче 2.3. Для отрезков одной прямой их пересечение (точку или отрезок) можно найти путем подсчета значения двух скалярных произведений аналогично задаче 2.1 или с помощью сравнения координат концов отрезков.

Введем теперь понятие расстояния между двумя непересекающимися отрезками как минимальное среди расстояний между всеми парами точек двух отрезков. Несложно понять, что оно равно расстоянию от конца одного из отрезков до другого отрезка (задача 2.1). Поэтому для решения задачи достаточно подсчитать

соответствующее расстояние для каждой из четырех концевых точек и выбрать из них минимальное.

Для проверки наличия пересечения двух лучей P_1P_2 и P_3P_4 следует изучить взаимное расположение соответствующих прямых. Равенство нулю косого произведения $[\overrightarrow{P_1P_2}, \overrightarrow{P_3P_4}]$ означает принадлежность лучей параллельным прямым. Если эти прямые различны, то векторы $\overrightarrow{P_1P_3}$ и $\overrightarrow{P_1P_2}$ неколлинеарны и, значит, косое произведение $[\overrightarrow{P_1P_3}, \overrightarrow{P_1P_2}]$ отлично от нуля. В этом случае лучи не пересекаются. Когда лучи лежат на одной прямой, с помощью знака скалярного произведения $(\overrightarrow{P_1P_2}, \overrightarrow{P_3P_4})$ можно понять, в одну или в разные стороны они направлены. В первом случае скалярное произведение будет положительным, а во втором — отрицательным. Если лучи сонаправлены, то определить, какой из лучей является их пересечением, можно, подсчитав значение скалярного произведения $(\overrightarrow{P_1P_2}, \overrightarrow{P_1P_3})$. Когда оно больше нуля, пересечением является луч P_3P_4, в противном случае — луч P_1P_2. В случае же противоположной направленности лучей их пересечение — либо отрезок P_1P_3, и тогда начало любого из двух лучей лежит внутри другого луча: $(\overrightarrow{P_1P_2}, \overrightarrow{P_1P_3}) > 0$, либо одна точка $P_1 = P_3$: $(\overrightarrow{P_1P_2}, \overrightarrow{P_1P_3}) = 0$, либо оно пусто: $(\overrightarrow{P_1P_2}, \overrightarrow{P_1P_3}) < 0$.

Наконец, если прямые P_1P_2 и P_3P_4 пересекаются в одной точке M ($[\overrightarrow{P_1P_2}, \overrightarrow{P_3P_4}] \neq 0$), то найти эту точку можно так же, как в задаче 2.3. Затем следует проверить, что M принадлежит каждому из лучей: $(\overrightarrow{P_1P_2}, \overrightarrow{P_1M}) \geq 0$ и $(\overrightarrow{P_3P_4}, \overrightarrow{P_3M}) \geq 0$.

2.5. Взаимное расположение окружности и прямой.

Прямая может пересекать окружность в двух точках, касаться ее или не иметь с окружностью общих точек. Эти случаи легко определяются. Достаточно найти расстояние от центра окружности до данной прямой (по формуле расстояния от точки до прямой, см. 2.1). Если это расстояние (обозначим его l) меньше радиуса окружности r, то прямая пересекает окружность в двух точках, если равно ему, то прямая касается окружности, а если оно больше радиуса, то общих точек нет. В последнем случае нас может интересовать и расстояние от прямой до окружности. Оно равно $l - r$.

Более сложной является задача поиска общих точек прямой и окружности. Случай касания был рассмотрен нами в п. 1.7. Пусть теперь прямая и окружность пересекаются в двух точках. Координаты этих точек можно найти по следующему алгоритму (*рис.* 10). Найдем вектор n нормали к прямой. Отложим в направлении этого вектора вектор $\overrightarrow{OA}$ длины l. Вычислим расстояние $|AP_1| = |AP_2| = \sqrt{r^2 - l^2}$. От точки A вдоль прямой отложим в обе стороны векторы длины $|AP_1|$. Их концы дадут нам две искомые точки — P_1 и P_2. Каждый из шагов этого алгоритма в отдельности нами уже рассматривался ранее. Заметим только, что на первом шаге необходимо правильно выбрать одно из двух возможных направлений нормали к прямой. Для этого достаточно проверить, что скалярное произведение $(\boldsymbol{n}, \overrightarrow{OM}) \geq 0$, где M — произвольная точка прямой.

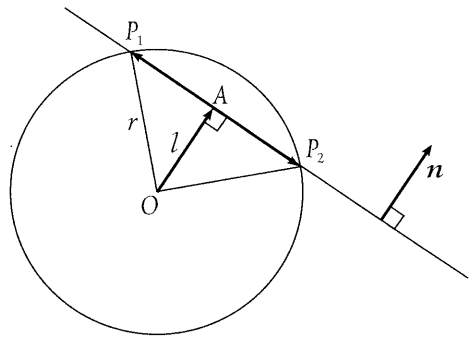


Рис. 10

2.6. Взаимное расположение двух окружностей.

Две различные окружности также могут пересекаться в двух точках, касаться друг друга или не иметь общих точек. В последнем случае одна из окружностей может располагаться внутри другой (назовем такие окружности вложенными) или каждая из окружностей лежит вне другой.

Проверка наличия пересечения или касания аналогично предыдущей задаче осуществляется путем сравнения расстояния между центрами окружностей (обозначим его l) с их радиусами. Если $l > r_1 + r_2$ или $l < |r_1 - r_2|$, то окружности общих точек не имеют. Второе условие обозначает вложенность одной окружности в другую. При замене знака любого из этих двух неравенств на равенство мы получим случай касания окружностей (внешнего или внутреннего). Если же $|r_1 - r_2| < l < r_1 + r_2$, то окружности имеют ровно две точки пересечения.

Координаты точки касания окружностей найти очень просто. Ведь центры окружностей $O_1(x_1, y_1)$ и $O_2(x_2, y_2)$ задают прямую, на которой лежит и точка касания $P(x_3, y_3)$. Будем считать, что точка O_1 является центром окружности большего радиуса. Тогда вектор $\overrightarrow{O_1P}$ сонаправлен с вектором $\overrightarrow{O_1O_2}$. Длины обоих векторов также известны, поэтому искомые координаты равны $\left(\frac{x_1 + (x_2 - x_1)r_1}{l}, \frac{y_1 + (y_2 - y_1)r_1}{l}\right)$. Проверьте, что если $r_1 < r_2$, то в случае вложенности окружностей полученная формула нуждается в корректировке.

При поиске координат двух точек пересечения окружностей воспользуемся механизмом, уже описанным в задаче 1.7. Рассмотрим треугольник O_1O_2P (*рис.* 11). В нем нам известны длины всех трех сторон ($r_1 > r_2$ и l). Проведем в треугольнике высоту PH. В получившемся прямоугольном треугольнике O_1HP неизвестны длины катетов. Найдем O_1H, записав теорему косинусов для

треугольника O_1O_2P: $r_2^2 = r_1^2 + l^2 - 2lr_1 \cdot \cos\varphi = r_1^2 + l^2 - 2l \cdot |O_1H|$. Отсюда $|O_1H| = \frac{(r_1^2 + l^2 - r_2^2)}{2l}$. По теореме Пифагора $|HP| = \sqrt{r_1^2 - |OH_1|^2}$. Теперь последовательно находим: вектор $\overrightarrow{O_1H} = \frac{|O_1H|}{l}\overrightarrow{O_1O_2}$, точку H по известной точке O_1 и вектору $\overrightarrow{O_1H}$, вектор $\boldsymbol{n} = (y_2 - y_1, x_1 - x_2)$, перпендикулярный O_1O_2, вектор $\overrightarrow{HP} = \frac{|HP|}{|\boldsymbol{n}|}\boldsymbol{n}$, наконец, точку P по известной точке H и вектору $\overrightarrow{HP}$. Заменив в последнем действии вектор $\overrightarrow{HP}$ на противоположный, получим и вторую точку пересечения окружностей.

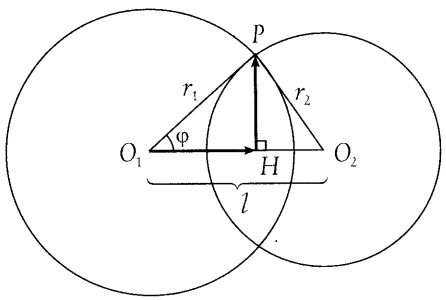


Рис. 11

Заметим, что при решении задач 1.7, 2.5 и 2.6 мы используем "пошаговые" алгоритмы. А именно, анализируя чертеж, выделяем цепочку объектов (векторов, точек и т.п.), каждый из которых за один элементарный шаг получается из уже найденных объектов. Фактически мы последовательно восстанавливаем параметры объектов, выделенных нами на чертеже. Цепочка вычислений должна привести нас к ответу, при этом каждый шаг предельно прост и вполне очевиден. Не исключено, что у задачи имеется более короткое аналитическое решение (например, решение системы уравнений). Однако "пошаговый" метод в силу своей наглядности прозрачен и позволяет контролировать каждый шаг решения, а именно это качество может сыграть решающую роль при проверке алгоритма и отладке программы.

2.7. Проверка принадлежности точки внутренней области многоугольника.

Пусть M — некоторая точка плоскости. Требуется определить ее местонахождение относительно замкнутой ломаной, являющейся границей многоугольника. Рассмотрим сначала случай выпуклого многоугольника. Пусть заданные своими координатами вершины многоугольника $P_0, P_1, \ldots, P_{n-1}$ перечислены в порядке его обхода против часовой стрелки. При таком обходе многоугольник лежит слева от границы. И, значит, если точка M лежит внутри многоугольника, то ориентированный угол между векторами $\overrightarrow{P_iM}$ и $\overrightarrow{P_iP_{i+1}}$ отрицателен. Поэтому нам достаточно подсчитать величину косых произведений $[\overrightarrow{P_iM}, \overrightarrow{P_iP_{i+1}}]$, $i = 0, 1, \ldots, n-1$; значение $i + 1$ берется по модулю n. Если все полученные при этом значения отрицательны, то точка M внутренняя. Если же одно из них равно нулю, а все остальные отрицательны, то M принадлежит границе многоугольника (убедитесь, что просто равенства нулю одного из значений не достаточно). В противном же случае точка M лежит вне нашего многоугольника.

Рассмотрим теперь произвольный многоугольник. Проведем горизонтальный луч из точки M, например, влево. Так как многоугольник ограничен, то всегда легко указать на этом луче точку $P(x, y)$, заведомо ему не принадлежащую. Далее подсчитаем количество пересечений отрезка PM с границей многоугольника. Если оно равно нулю или четно, то точка M лежит вне многоугольника, в противном случае — внутри него.

Количество пересечений отрезка PM с границей мы подсчитаем, рассмотрев по очереди пересечение отрезка PM с каждым из звеньев ломаной. При этом возможны следующие особые случаи.

1. Одно из звеньев ломаной целиком содержится внутри отрезка PM.
2. Звено ломаной касается отрезка PM.
3. M лежит на одном из звеньев ломаной.

В последнем случае M принадлежит границе многоугольника, и в подсчете общего числа пересечений необходимости нет. Для двух первых случаев поступим следующим образом. В первом случае пересечение будем игнорировать. Во втором — дополнительно проверим, "нижним" или "верхним" концом звено ломаной касается горизонтального отрезка PM. Если точкой касания является "нижний" конец звена, то пересечение игнорируется, а если "верхний" — то засчитывается. С учетом этого соглашения касание отрезка PM границы многоугольника в одних точках игнорируется, а в других точках считается дважды. Это не изменит четности числа пересечений, а только она важна при поиске ответа на вопрос данной задачи. Если же отрезок действительно пересекает ломаную в ее вершине, то, по нашему соглашению, число пересечений как раз увеличится на единицу (пересечение с верхним ребром засчитано не будет, а с нижним — будет). Например, на *рис.* 12 количество пересечений для верхней из исследуемых точек будет равно четырем (касание засчитано дважды), а для нижней точки — трем (касание не учтено, а пересечение в вершине ломаной учтено один раз).

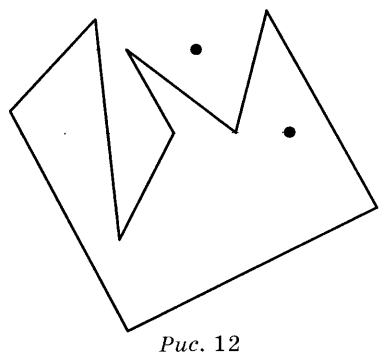


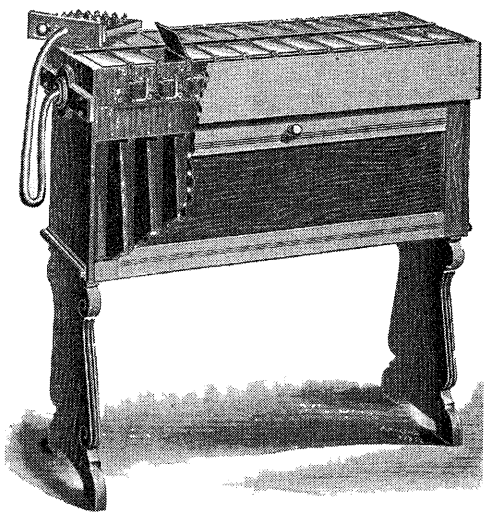
Рис. 12

Продолжение см. в № 43

Перфокарты и перфоленты

Окончание. Начало на с. 1

В 1896 году Холлерит организовал фирму, специализирующуюся на выпуске счетно-перфорационных машин и перфокарт. Такие машины стали широко применяться, например, на транспорте, в торговле, в статистических управлениях, страховых обществах. Этому предприятию сразу же сопутствовал успех, и в дальнейшем оно становилось все более преуспевающим [1, 5]. С годами предприятие менялось, причем последнее изменение при жизни Холлерита произошло в 1924 году, когда оно было преобразовано в фирму IBM (*International Business Machines*), которая сейчас, как известно, является крупнейшим разработчиком компьютерных систем.



В центре: машина для сортировки перфокарт.

Вскоре началась компьютерная эра, и перфокарты очень долго употреблялись в качестве носителей информации для компьютеров, причем обычно перфокарты вводились в них колодами, как это делалось и в табуляционных машинах [1].

Первой альтернативой перфокартам как носителям входных и выходных данных была перфорационная бумажная лента. Ее преимущество заключалось в том, что закодированные данные записывались на "непрерывный" носитель, а не на отдельные карты большой колоды. Предшественниками компьютерной перфоленты являлись движущиеся бумажные ленты, применявшиеся на телеграфных приемных аппаратах в XIX веке. Приспособленная к табуляторам, а затем и к компьютерам, перфолента выполняла те же функции, что перфокарты, и данные считывались с нее похожими устройствами. (Кстати, один из первых действующих компьютеров с программным управлением, Марк-1, появившийся в 1944 году, главным разработчиком которого был Говард Эйкен из Гарвардского университета [1, 3, 4, 6], создавался на базе стандартных деталей перфорационных устройств, выпускавшихся в то время фирмой IBM, при этом для управления машиной употреблялись команды, вводимые с помощью перфоленты.) Однако бумажная лента имела и серьезный недостаток: она могла рваться...

Литература

1. Знакомьтесь: компьютер: Пер. с англ. М.: Мир, 1989.
2. Язык компьютера: Пер. с англ. М.: Мир, 1989.
3. *Частиков А.П.* От калькулятора до суперЭВМ // Новое в жизни, науке, технике. Серия "Вычислительная техника и ее применение" № 1/88.
4. *Леонов А.Г., Четвергова О.В.* История компьютеров // Информатика № 35/98.
5. Перфоратор и табулятор // Информатика № 39/2000.
6. *Гутер Р.С., Полунов Ю.Л.* От абака до компьютера. Изд. 2-е, испр. и доп. М.: Знание, 1981.

Гл. редактор
С.Л. Островский
Зам. гл. редактора
А.И. Сенокосов
Редакция:
Е.В. Андреева
Н.Л. Беленькая
Л.Н. Картвелишвили
Н.П. Медведева
Дизайн и верстка:
Н.И. Пронская
Корректоры:
Е.Л. Володина,
С.М. Подберезина

Адрес редакции и издателя:
121165, Киевская, 24
тел. 249-48-96
Отдел рекламы
тел. 249-98-70

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта изданий 32744

Учредитель: ООО "Чистые пруды"

Зарегистрировано в Министерстве РФ по делам печати. ПИ № 77-7230 от 12.04.2001.
Отпечатано в ОИД "Медиа-Пресса", 125993, ГСП-3, Москва, А-40, ул. "Правды", 24.
Тираж 6500 экз.
Срок подписания в печать по графику 16.10.2002.
Номер подписан 16.10.2002.
Заказ № 16730
Цена свободная

Тел.: (095)249-31-38, 249-33-86. Факс (095)249-31-84

Internet: inf@1september.ru
WWW: http://www.1september.ru