№ 16 (209) апрель 1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Электронные версии газеты “Первое сентября” и приложений http://www.1september.ru

К XX ГОДОВЩИНЕ ПЕРЕВОРОТА

Представьте себе, что вам предложили по цене подержанной, но вполне приличной автомашины приобрести синхрофазотрон. Ну, не такой, конечно, какой используется в крупных научных центрах и занимает гектары площади, а портативный, умещающийся на вашем рабочем столе...

Несмотря на абсурдность подобного, примерно такая ситуация сложилась при выпуске в продажу первых персональных компьютеров, о которых большинство было наслышано только в связи с оборонными и космическими программами. Ну, в лучшем случае как о непозволительной роскоши, доступной только крупным корпорациям.

Разумеется, какая-то небольшая часть ученых и инженеров сразу оценила подобное новшество, но зачем нужна такая дорогостоящая игрушка экономистам, бухгалтерам, менеджерам, секретаршам в конце концов, было совершенно непонятно. И недаром даже такой гигант, как фирма IBM, совершил грубейший просчет,

Окончание на с. 16

НАШИ ДЕТИ БУДУТ ЖИТЬ В XXI ВЕКЕ

Системы счисления и компьютерная арифметика

Е.В. АНДРЕЕВА, И.Н. ФАЛИНА

Продолжение. Начало в № 14, 15/99.

Ранее были опубликованы разделы части I “Представление информации (базовый курс)”:

Тема 1. Системы счисления как разновидность информационных систем.

Тема 2. История систем счисления.

Тема 3. Позиционные системы счисления.

Тема 4. Двоичное кодирование информации.

Тема 5. Двоичная арифметика.

Тема 6. Перевод чисел из двоичной системы счисления в десятичную и обратно.

В этом номере:

Тема 7. Информация и компьютер.

Тема 8. Компьютерное представление целых чисел.

Тема 9. Компьютерное представление вещественных чисел.

Начата публикация части II (“Системы счисления”):

Глава 1. “Позиционные системы счисления”.

Продолжение следует

Первое знакомство с объектно-ориентированным программированием

А.А. СЕМЕНОВ, А.Г. ЮДИНА

Окончание. См. № 14, 15/99

Занятие № 5

Попробуем создать графический объект произвольной формы `SimSprite`, своего рода спрайт, только примитивный. Он может передвигаться только на черном фоне, т.к. наследует алгоритм движения предка (стирает свое изображение, перерисовывая себя черным цветом). Попробуем создать синего жучка. Его изображение вполне можно разместить в матрице 9×9 точек.

Для задания спрайта определим тип `ImageArr` — массив 9×9 байт.

```
{$R-,Q-}
uses Graph13h,MyObj,crt;
type
ImageArr=array[0..8,0..8] of byte;

SimSprite=object(GOb)
  image: ImageArr;
  constructor Init(var aimage:ImageArr;ax,ay:integer;
                   acolor:byte);
  procedure Draw(acolor:byte); virtual;
end;

constructor SimSprite.Init(var image:ImageArr;ax,ay:integer;
                           acolor:byte);
begin
 inherited Init(ax,ay,acolor);
 Image:=aImage;
end;

procedure SimSprite.Draw(acolor:byte);
var
 i,j: byte;
begin
 for i:=0 to 8 do
   for j:=0 to 8 do
      if acolor<>0 then PutPixel(x+j,y+i,image[i,j])
      else PutPixel(x+i,y+j,0)
end;

const max=50;
var a:array[1..max] of SimSprite;
       i:byte;

const bug:ImageArr=((0,0,0,0,9,0,0,0,0),
                    (0,0,1,1,1,1,1,0,0),
                    (1,0,1,1,9,1,1,0,1),
                    (0,1,1,6,9,6,1,1,0),
                    (0,1,1,1,9,1,1,1,0),
                    (0,1,1,1,1,1,1,1,0),
                    (1,0,1,1,1,1,1,0,1),
                    (0,0,0,1,1,1,0,0,0),
                    (0,0,0,0,14,0,0,0,0));
begin
 Screen13h;
 Randomize;
 for i:=1 to max do
  a[i].Init(bug,20+random(280),20+random(160),1);
 for i:=1 to max do a[i].Show;
 readln;
 while not keypressed do
   for i:=1 to max do
      a[i].Move(random (3)-2,random(3)-2);
 readln
end.
```

Получилось! Жучки целеустремленно бегут к левому верхнему углу экрана, где и пропадают с тем, чтобы вновь появиться в правом нижнем углу.

Примечание

Вопрос: Как сделать, чтобы "жучок" бегал по травке?

Ответ: Для этого придется усовершенствовать метод `Draw`. Перед выводом спрайта необходимо запомнить изображение под ним в массиве 9×9 (придется добавить еще одно поле в объект `SimSprite`, его можно назвать, например, `Background`). Для того чтобы сохранить картинку под спрайтом, ее можно просканировать по точкам с помощью процедуры `GetPixel`, возвращающей цвет точки экрана с координатами `x`, `y`.

```
Function GetPixel(x,y: integer): byte;
begin
  GetPixel:=Mem[$A000:(y*320 + x)];
end;
```

Для того чтобы стереть спрайт, достаточно вывести по точкам массив `Background`.

Если нужно нарисовать спрайт, мы выведем только те точки спрайта, цвет которых отличается от "прозрачного". Номер прозрачного цвета принимают, как правило, за нуль. Можно написать так:

```
...
if image[i,j]<>0 then PutPixel(x+j,y+i,image[i,j])
```

Домашнее задание

Разработать методы `LT90` и `RT90`, поворачивающие спрайт налево и направо на 90 градусов. Как вы уже догадались, мы хотим научить жучков поворачиваться в процессе движения и бежать все время головой вперед, как поступают все нормальные насекомые. В следующий раз наш объект получит новые методы и будет моделировать поведение живых существ!

Занятие № 6

На этом занятии наша задача — моделирование движения живого существа (жучка), меняющего направление своего движения, но при этом передвигающегося всегда головой вперед.

Для этого прежде всего нужно научиться "поворачивать" матрицу, задающую изображение.

Для поворота спрайта достаточно поменять местами строки и столбцы матрицы, изменив их последовательность на противоположную (иначе получится просто зеркальное отражение). Поэкспериментируйте на бумаге с простым спрайтом 4×4 точки. Для формирования повернутой картинки используем временный массив `TempArr`.

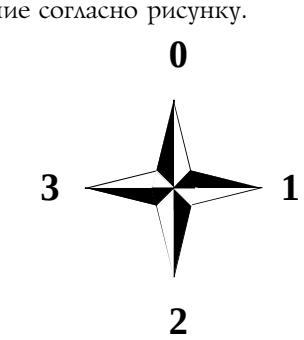
```
procedure Bug.RT90;
var
  i,j:byte;
  TempArr:ImageArr;
begin
  for i:=0 to 8 do
   for j:=0 to 8 do TempArr[i,j]:=image[8-j,i];
  for i:=0 to 8 do
   for j:=0 to 8 do image[i,j]:=TempArr[i,j];
end;

procedure Bug.LT90;
var
  i,j:byte;
  TempArr:ImageArr;
begin
  for i:=0 to 8 do
   for j:=0 to 8 do TempArr[i,j]:=image[j,8-i];
  for i:=0 to 8 do
   for j:=0 to 8 do image[i,j]:=TempArr[i,j];
end;
```

Теперь попробуем создать специализированного потомка `GOb` по имени `Bug`.

Данный объект умеет перемещаться в направлении `Direction` (`0≤Direction≤3`) со скоростью `velocity`.

Характеристика `Direction` определяет направление согласно рисунку.




```
{$R-,Q-}
uses Graph13h,MyObj,crt;

type
ImageArr=array[0..8,0..8] of byte;

Bug=object(GOb)
  image: ImageArr;
  direction,
  velocity:integer;
  constructor Init(var aimage:ImageArr;ax,ay:integer;
                   acolor,avelocity:byte);
  procedure Draw(acolor:byte); virtual;
  procedure RT90;
  procedure LT90;
  procedure Move;
end;

constructor Bug.Init(var aimage:
ImageArr;ax,ay:integer;acolor,avelocity:byte);
begin
 inherited Init(ax,ay,acolor);
 Image:=aImage;
 direction:=0; {все жучки в момент рождения повернуты головой на север}
 velocity:=avelocity;
end;

procedure Bug.Draw(acolor:byte);
var
 i,j: byte;
begin
 for i:=0 to 8 do
   for j:=0 to 8 do
      if acolor<>0 then PutPixel(x+j,y+i,image[i,j])
      else PutPixel(x+j,y+i,0)
end;
```

```
procedure Bug.RT90;
var
  i,j:byte;
  TempArr:ImageArr;
begin
  for i:=0 to 8 do
   for j:=0 to 8 do TempArr[i,j]:=image[8-j,i];
  for i:=0 to 8 do
   for j:=0 to 8 do image[i,j]:=TempArr[i,j];
   direction:=direction+1;
   if direction>3 then direction:=0;
   {наши методы RT90 и LT90 не только поворачивают спрайт,
    но и меняют направление движения}
end;

procedure Bug.LT90;
var
  i,j:byte;
  TempArr:ImageArr;
begin
  for i:=0 to 8 do
   for j:=0 to 8 do TempArr[i,j]:=image[j,8-i];
  for i:=0 to 8 do
   for j:=0 to 8 do image[i,j]:=TempArr[i,j];
   direction:=direction-1;
   if direction<0 then direction:=3;
end;

procedure Bug.Move;
begin
   case direction of
   0: inherited move(0,-velocity);
   1: inherited move(velocity,0);
   2: inherited move(0,velocity);
   3: inherited move(-velocity,0);
   end;
end;
const max=30;
var a:array[1..max] of Bug;
    i:integer;
const {спрайт-матрица 9x9 точек:}
 BugSprite:ImageArr=((0,0,0,0,9,0,0,0,0),
                     (0,0,6,1,1,1,6,0,0),
                     (1,0,1,1,1,1,1,0,1),
                     (0,1,1,1,9,1,1,1,0),
                     (0,1,1,9,9,9,1,1,0),
                     (0,1,1,1,9,1,1,1,0),
                     (1,0,1,1,1,1,1,0,1),
                     (0,0,0,1,1,1,0,0,0),
                     (0,0,0,0,14,0,0,0,0));
begin
 Screen13h;
 Randomize;
 for i:=1 to max do
 a[i].Init(BugSprite,20+random(280),20+random(160),1,random(3)+1);
 for i:=1 to max do a[i].Show;
 readln;
 while not keypressed do
  begin
    a[1+random(max)].RT90;a[1+random(max)].LT90;
    for i:=1 to max do
     begin
       a[i].Move;
       delay(2);
     end;
  end;
 readln
end.
```

Теперь вы наконец можете отдохнуть. Ваши ученики будут долго с увлечением создавать аквариумы с рыбками или танкодромы. А может быть, кто-то из них замахнется на Тетрис.

Заключение

Ну, вот и все. Цель достигнута. Теперь для нас прояснился смысл эпиграфа статьи. Действительно, при разработке стандартных элементов сложной программы было бы нерационально писать весь код “с нуля”. Достаточно взять за основу ряд объектов, разработанных профессиональными программистами, и дополнить их своими полями данных и методами, чтобы они удовлетворяли вашим требованиям. Некоторые стандартные методы можно переписать заново (перекрыть), если они вас не устраивают. Так проявляются два фундаментальных свойства объектов:

- *наследование* (способность использовать методы и поля предка, причем доступ к его исходному тексту необязателен);
- *инкапсуляция* (способность объекта включать в себя новые поля, методы и даже объекты). В последнем случае объект называют группой.

В этой статье мы не смогли рассмотреть третье важное свойство объектов — *полиморфизм*. Оно упоминалось в неявном виде, когда шла речь о конструкторе и виртуальных методах. Для демонстрации полиморфизма необходимо научиться создавать и уничтожать объекты во время выполнения программы, что требует владения техникой работы с динамическими переменными. Но это, наверное, тема отдельной публикации.

Если вас заинтересовала статья, пишите в редакцию. Послать вопросы и пожелания авторам вы можете по адресу: **andrew_semenov@mtu-net.ru** или **sasha@udina.mccme.ru**.

Продолжение дискуссии

Уважаемые коллеги, мы выражаем свое несогласие с мнением А.Г. Гейна и А.И. Сенокосова, авторов критических статей об учебнике “Информатика” для 6–7-х классов под ред. проф. Н.В. Макаровой.

Хотелось бы поделиться с вами общим впечатлением и некоторыми результатами, которые мы получили, проводя занятия в школе по этому учебнику. К сожалению, последующие книги (для 7–8-х и 10–11-х кл.) из созданного коллективом авторов — преподавателей школ под руководством проф. Н.В. Макаровой комплекта учебников вышли значительно позже и мы не смогли полностью апробировать их в учебном процессе. Предполагаем использовать их по базовому курсу информатики.

В течение ряда лет в нашей гимназии информатика преподается с начальной школы и до 11-го класса. Проблема с учебниками стояла всегда.

В старших классах (9–11-х) обычно рекомендовались книги справочного характера для пользователей компьютеров. Затем появилась книга Ю.А. Шафрина “Основы компьютерной технологии”, которая объединила многие вопросы применения компьютеров под одной обложкой, но учебником по информатике, на наш взгляд, не стала. Рекомендованные же Министерством образования учебники по информатике для старших классов либо предполагают безмашинный вариант преподавания, либо опираются на язык программирования. Они, безусловно, приносят пользу, но использовать их как учебники по аналогии с другими дисциплинами мы не можем.

В младших и в 5-х классах мы частично стали применять тетради, подготовленные авторским коллективом под руководством А.И. Горячева. А вот для 6–8-х классов не было ничего приемлемого.

Какие бы учебники мы хотели иметь? Учебник должен обязательно быть интересной для ученика книгой, которой он может воспользоваться в любое время и в школе, и дома. Особенно это актуально, когда ученик пропустил уроки и ему предстоит за короткий срок догнать своих одноклассников. С помощью такого учебника учащиеся должны знакомиться с различными способами обработки информации и возможностями современных компьютерных инструментов. Это обычные требования. Помимо этого, нам бы хотелось, чтобы материал был так преподнесен, чтобы учащиеся могли самостоятельно проводить исследования в разных программных средах, приобретая навыки познания и применяя их в других дисциплинах.

Обсуждаемый учебник очень обрадовал нас.

Во-первых, выбран единый подход к изучению — системный, объектно-ориентированный. Имея в руках еще два учебника из комплекта, видно, что на этой идее строится изложение всего материала как по базовому курсу, так и по предпрофессиональной подготовке. Работа с объектами с учетом среды их обитания позволяет прививать ученику общие навыки исследования.

Во-вторых, в книге предлагается множество вопросов, практических заданий, задач, стимулирующих познавательный интерес учащихся. Учебник хорошо работает как на уроках, так и дома.

В-третьих, учебник написан достаточно серьезным, но вместе с тем доступным языком, благодаря чему дети чувствуют уважение к себе.

В-четвертых, имеется глоссарий; многократные введения одних и тех же понятий с разных точек зрения помогают разобраться в новом материале.

В-пятых, учебник состоит из 5 разделов, каждый из которых является отдельным модулем (блоком), которые можно изучать достаточно независимо. Об этом более подробно.

Уже сейчас можно говорить о разных вариантах использования учебника. В 5-м классе — это книга для чтения. Пятиклассники с интересом разглядывают ее, учитель рекомендует читать отдельные темы (обычно об информации, назначении и аппаратном обеспечении компьютера). Любопытно, что в книгу заглядывают и родители. Если дома есть компьютер, то они вместе с детьми по учебнику выполняют задания, работая с графическим редактором или в среде Logo (семейное обучение!).

В 6-м или 7-м (первый год обучения) — это уже учебник. Мы внимательно разбираем раздел 1 “Представление об объектах и информации”. Рисуем или делаем макеты различных сред, описываем объекты, которые обитают в этих средах. Место и роль компьютера в современном обществе воспринимается учащимися ясно. Раздел “Аппаратное и программное обеспечение компьютера” несет практическую нагрузку. Простая формула этапов обработки информации компьютером, представление об устройствах ввода-вывода, передачи и хранения информации усваиваются легко. Более сложную тему о структуре персонального компьютера можно отложить на второй год обучения. Столь же естественно воспринимаются понятия о программном обеспечении. Названия некоторых программных систем и пакетов дети знают понаслышке и с удовлетворением знакомятся с ними более подробно.

Дальнейшая работа может пойти разными путями. Приведем несколько нам известных примеров.

В школе г. Всеволожска Ленинградской области далее начинают изучать и описывать объекты в программной среде Logo. В нашей гимназии шести- и семиклассники сначала знакомятся со средой Windows, затем исследуют графические объекты приложения “Графический редактор”. В 108-й школе объектный подход (после работы с разделами 1 и 2) применяется при изучении графического конструктора, вообще не описанного в данном учебнике.

На первом году обучения предполагается исследование двух сред. На втором году мы предполагаем еще раз вернуться к рассмотрению 1 и 2-го разделов, более подробно изучать раздел 3 “Системная среда Windows” и программную среду Logo. У нас еще есть два учебника “Информатика” (7—8-й кл., 9-й кл.), позволяющие перейти к более серьезным вопросам в области программирования, технологии работы в пользовательских средах и моделирования, но об этом в следующий раз.

Мы заметили в учебнике ряд неточностей и опечаток, но предполагаем, что они будут исправлены, а на занятиях можно явно указать на них. Некоторые определения и термины вызывают споры среди учителей.

Мы считаем, что истина как раз и рождается в спорах, тем более в нашей области не все можно определить однозначно, да еще с учетом возрастных категорий учащихся. Учебник не диктует раз навсегда заданные определения, а пытается акцентировать внимание ученика. Он предлагает новый на сегодняшний день подход к обучению информатике.

Мы благодарим весь коллектив авторов, на наш взгляд, самого современного комплекта учебников “Информатика” под ред. проф. Н.В. Макаровой и не сомневаемся, что они помогут учителю в организации учебного процесса и будут интересны и полезны ученикам.

Учителя информатики
гимназии № 470, г. Санкт-Петербург,
Л.Р. Кустанович,
С.Е. Снеткова;
учитель информатики
108-й школы, г. Санкт-Петербург,
А.А. Шперхт

P.S. А вот письмо, которое мы получили от наших коллег из Нижнего Новгорода:

Учебник “Информатика. 6—7” под редакцией Н.В. Макаровой появился у нас в продаже. Ребятам и нам он понравился. Пробуем использовать в учебном процессе.

Очень хорошо написана глава “Программирование в среде LogoWriter”.

Лично мне понравилось структурное изложение глав учебника, а также его насыщенность материалом и приятное оформление.

Н.ГОРОДЕЦКАЯ
и сотрудники УКЦ
Н.Новгорода

ПИСЬМО В РЕДАКЦИЮ 3

А.Г. ЮДИНА

Практикум по информатике

Избранные главы — в среде LogoWriter

Окончание. См. № 15/99

Глава 8. МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ

§ 32. Прямолинейное равномерное движение

Вспомним* рекурсивные процедуры, заставлявшие Черепашку "вечно" (до прерывания) двигаться по различным траекториям, форма которых зависела от угла поворота. Если каждый поворот равен 90 градусам, то траектория имеет квадратную форму; 60 — шестиугольную; 30 — двенадцатиугольную и т.д. С уменьшением угла поворота (если только угол равен 360/*N*, где *N* — натуральное число) траектория движения все больше приближается к окружности и одновременно увеличивается в размерах.

Другими словами, кривизна траектории Черепашки с уменьшением угла поворота делается все меньше, а когда угол станет равным нулю, Черепашка будет двигаться просто по прямой:

```
ЭТО ДВИЖЕНИЕ_1 :V
FD :V
ДВИЖЕНИЕ_1 :V
END
```

Процедура ДВИЖЕНИЕ_1 заставляет Черепашку пройти V шагов и вызывает сама себя. При каждом новом вызове Черепашка продвигается по прямой на одно и то же расстояние.

Поскольку вызовы процедуры следуют через одинаковые (и очень небольшие) промежутки времени, вы увидите на экране прямолинейное и почти равномерное движение. Меняя параметр V, можно ускорить или замедлить движение Черепашки. Если принять промежуток времени от одного вызова процедуры до другого за единицу измерения, то V будет скоростью прямолинейного равномерного движения Черепашки, измеренной в черепашьих шагах за условную единицу времени.

Удобные для наблюдения значения V — примерно от —5 до 5.

В процедуре ДВИЖЕНИЕ_1 можно предусмотреть остановку Черепашки, когда она достигнет некоторой границы. Пусть Черепашка может двигаться только в пределах замкнутого контура. Поле экрана вне этого контура закрасим каким-либо цветом (например, синим) и предусмотрим проверку цвета поля под Черепашкой при каждом вызове:

```
ЭТО ДВИЖЕНИЕ_1 :V
FD :V
IF COLORUNDER  = 5 [STOP]
ДВИЖЕНИЕ_1 :V
END
```

Черепашка остановится, дойдя до синего поля. Правильно работать такое условие будет только в том случае, когда Черепашка двигается, не оставляя следа.

Задания

32.1. Подготовьте "полигон" для Черепашки, на границе которого она будет останавливаться. Для этого напишите процедуру, рисующую поле прямоугольной формы; область между границами "полигона" и границами экрана надо закрасить.

В процедуру ДВИЖЕНИЕ_1 *нужно внести номер цвета как условие остановки.*

32.2. Проведите "испытания" процедуры ДВИЖЕНИЕ_1 *с различными скоростями.*

После того как Черепашка остановилась на границе "полигона", можно "включить задний ход", т.е. запустить процедуру с отрицательной скоростью.

§ 33. Прямолинейное равноускоренное движение

Процедура ДВИЖЕНИЕ_1 каждый раз вызывает сама себя с тем же значением параметра, поэтому скорость перемещения Черепашки остается неизменной.

Если же параметр при вызове будет меняться, то и характер движения изменится. Попробуем запустить следующую процедуру:

```
ЭТО ДВИЖЕНИЕ_2  :V
FD  :V
IF COLORUNDER = 5 [STOP]
ДВИЖЕНИЕ_2  :V + 0,01
END
```

Полигон для Черепашки должен быть ограничен синим полем, перо поднято.

Пусть Черепашка получила команду ДВИЖЕНИЕ_2 1. За первую единицу времени (условную) она пройдет один черепаший шаг. Следующее перемещение будет уже чуть больше — 1,01 шага. Затем 1,02, 1,03, 1,04 и т.д.

Значит, скорость Черепашки (в шагах за условную единицу времени) станет все время увеличиваться. Причем изменение скорости будет равномерное — на 0,01 за каждую единицу времени.

Другими словами, Черепашка станет двигаться прямолинейно с постоянным ускорением. Так двигается тело, когда на него действует постоянная сила. Черепашка достигнет границы со скоростью, которая существенно больше, чем начальная.

А что получится, если параметр не увеличивать, а уменьшать? Например:

```
ЭТО ДВИЖЕНИЕ_2  :V
FD :V
IF COLORUNDER  = 5 [STOP]
ДВИЖЕНИЕ_2  :V - 0,01
END
```

После запуска с положительной скоростью (скажем, равной 2) Черепашка будет двигаться все медленнее. Через некоторое время скорость станет равной нулю, а затем, продолжая уменьшаться, начнет принимать отрицательные значения. И Черепашка "поедет" в обратном направлении все быстрее, пока не вернется на старт.

Примерно то же происходит с подброшенным вверх мячом: его скорость постепенно уменьшается, оказывается равной нулю в самой верхней точке, а затем мяч начинает падать и падает все быстрее, пока не достигнет поверхности. На протяжении всего времени движения на мяч действовала постоянная сила притяжения Земли, и его движение происходило с постоянным отрицательным ускорением (для Земли оно равно примерно 9,8 м/с^2).

В нашей модели есть возможность менять ускорение (например, можно сравнить движение подброшенного тела на планетах с разным притяжением).

Чтобы было удобно проводить испытания с различными значениями скорости и ускорения, сделаем ускорение вторым параметром процедуры:

```
ЭТО ДВИЖЕНИЕ_2  :A  :V
FD :V
IF COLORUNDER = 5 [STOP]
ДВИЖЕНИЕ_2  :A  :V + :A
END
```

При каждом вызове меняется скорость (V), а ускорение (A) остается неизменным (движение равноускоренное).

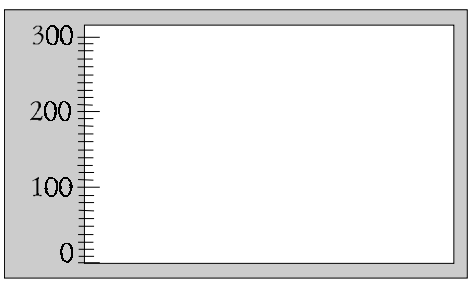
При отрицательных значениях A ускорение будет направлено в сторону, противоположную движению, т.е. движение будет равнозамедленное.

Задания

33.1. Проведите испытания модели равноускоренного движения с различными значениями скорости и ускорения.

33.2. Подготовьте форму небольшого шарика для Черепашки. Напишите процедуру для установки шарика на старт — на нижней границе полигона ближе к левой стороне.

33.3. Напишите процедуру для рисования на левой границе поля линейки с делениями через 10 шагов. Затем в режиме надписей (вход — F8, выход — Esc) проставьте соответствующие числа (см. рисунок).



Чтобы числа получились около соответствующих штрихов, нулевое деление разметки должно иметь ординату, кратную 20 (например, —160). Кроме того, необходимо, чтобы у нулевого деления и шарика на старте была одна и та же ордината. Картинку полезно сохранить (запись картинки — SAVEPIC*, загрузка —* LOADPIC*).*

33.4. Проведите с помощью процедуры из задания 33.1 серию опытов, моделирующих подбрасывание шарика в поле тяготения с неизменной начальной скоростью и с разными ускорениями (начальное значение скорости должно быть положительным, а ускорение — отрицательным). Сформулируйте, как зависит высота подъема шарика от ускорения поля тяготения.

33.5. Проведите серию опытов по подбрасыванию шарика с неизменным ускорением и с разными значениями начальной скорости. Сформулируйте, как зависит высота подъема шарика от начальной скорости.

33.6. Подберите согласующуюся с опытными данными формулу зависимости высоты подъема подброшенного вертикально тела от начальной скорости и ускорения.

Продолжение на с. 13

* Соответствующие алгоритмы рассматриваются в главах пособия, которые не вошли в данную публикацию. Вы можете ознакомиться с ними, например, в № 35/97.

УЧЕБНИКИ

10. Существуют ли системы счисления с основаниями P и Q, в которых $12_P > 21_Q$?

Решение. Да, существуют. Для этого необходимо, чтобы $P+2>2Q+1$ (по формуле (1.7) мы записали каждое из чисел в развернутой форме), то есть $P>2Q-1$. Например, $P=20$, $Q=10$.

11. Для десятичного числа 371 найти основание P традиционной позиционной системы счисления, в которой данное число будет представлено теми же цифрами, но записанными в обратном порядке, т.е. $371=173_P$.

Решение. Согласно формуле (1.7) получаем уравнение $P^2+7P+3=371$. Полученное уравнение имеет один целый положительный корень — 16, значит, искомой является шестнадцатеричная система счисления.

12. В вопросах и заданиях к теме 3 первой части приведена задача из книги Якова Перельмана "Занимательная математика" о чудаке математике. В какой системе счисления указан возраст чудака в момент окончания университета?

Решение. Основание этой системы определяется фразой "...спустя год (после 44 лет) 100-летним молодым человеком...". Если при прибавлении 1 к числу 44 мы получаем 100, то цифра 4 — наибольшая в этой системе, как 9 — в десятичной. Следовательно, основанием системы является 5.

13. *(Московский государственный университет экономики, статистики и информатики, 1997 г.)*

Существует ли такая система счисления, в которой $3+4=7$, и $3\cdot4=13$, и $39+29=70$?

Решение. Предположим, что такая система счисления существует, обозначим ее основание через P. Тогда справедлива следующая система уравнений относительно P:

$$\begin{cases} 3P^0+4P^0=7P^0, \text{ то есть } 3+4=7 \text{ и } P>7; \\ 3P^0 \cdot 4P^0=P+3P^0; \\ 3P+9P^0+2P+9P^0=7P; \end{cases}$$

Из двух последних уравнений следует, что $P=9$.

Но в позиционной системе счисления с основанием 9 нет цифры 9, т.е. в этой системе счисления нельзя записать числа 29 и 39.

Ответ: такой системы счисления не существует.

1.8. Контрольные вопросы и упражнения

1. Выпишите базис двадцатеричной системы счисления. Сколько "весит" каждая цифра в пятом разряде этой системы?

2. С применением каких позиционных систем счисления вы встречались?

3. Запишите год своего рождения с помощью римских цифр.

4. Придумайте и выпишите алфавит для 50-ричной системы счисления.

5. Придумайте свою позиционную систему счисления, базис которой не образует геометрической прогрессии.

6. Сколько и каких требуется цифр, чтобы можно было любое число записать в пятеричной системе счисления? А в двенадцатеричной?

7. Запишите в 6-ричной системе счисления число, следующее по порядку за числом 5.

8. Какое число следует за числом 111_{14} в 14-ричной системе счисления?

9. Какое число предшествует числу 10_{18} в 18-ричной системе счисления?

10. Продолжите последовательность членов арифметической прогрессии: 1, 4, 7, *A*... Объясните свой ответ.

11. Запишите в развернутом виде числа 65_7; 1998_{10}; $0{,}15A_{16}$; $1AF1H{,}A9_{20}$.

12. Какое минимальное основание может иметь система счисления, если в ней записаны все четыре числа — 432, 120, 111, 2331?

13. Какие из следующих чисел имеют ошибки в записи: 211_3, 183_8, $A9A_{11}$, 341_5?

14. Какое из чисел больше: 5_{10} или 10_5; 1000_2 или 10_8?

15. В каком случае при прибавлении единицы к числу в P-ичной системе счисления количество цифр в результате возрастет по сравнению с исходным числом? Может ли количество цифр возрасти больше чем на одну?

16. Как будет выглядеть в двоичной системе счисления десятичное число 0,125?

17. Запишите в системе счисления с основанием 240 числа 241, 242, 243, 250, 251.

18. Подсчитайте количество двоичных чисел в диапазоне от 10_2 до 1000_2.

19. Подсчитайте количество троичных чисел в диапазоне от 12_3 до 1000_3.

20. Назовем круглыми все числа, записываемые одной цифрой и несколькими нулями (быть может, одним). Выпишите все двузначные и трехзначные круглые числа в 5-ричной системе счисления.

21. В какой (каких) системе счисления $2_P \cdot 2_P = 10_P$?

22. Во сколько увеличится число 32_4, если справа к нему приписать три нуля?

23. Выполните следующие действия:

$$100000_P : 1000_P;\ 201_P : 100_P.$$

Задачи повышенной сложности

24. В каких системах счисления 10 является нечетным числом?

25. Опишите позиционную систему счисления, базис которой составляют числа Фибоначчи.

26. Верно ли, что для любого трехзначного числа задача 11 из параграфа 1.7 имеет решение?

27. Определите, в какой системе счисления произведена операция сложения $2102+211=10020$. Докажите, что найденная система счисления единственная.

28. Докажите, что в любой позиционной системе счисления с основанием $P \geq 3$ число 121_P является полным квадратом.

29. В каких P-ичных системах счисления $\frac{1}{2}$ будет точно представлена конечной P-ичной дробью? А $\frac{1}{3}$?

Информатика — это наука, которая занимается вопросами представления и обработки информации.
(Манфред Брой, профессор, лауреат премии Лейбница в области информатики)

Системы счисления и компьютерная арифметика

Е.В. АНДРЕЕВА, И.Н. ФАЛИНА

Продолжение. Начало см. в № 14, 15/99

Тема 7
Информация и компьютер

Как было сказано ранее, в компьютере в двоичном виде может храниться числовая, текстовая, графическая и звуковая информация.

7.1. Представление чисел

Напоминаем, что все числовые данные хранятся в компьютере в двоичном виде, т.е. в виде последовательности нулей и единиц, однако формы хранения целых и действительных чисел различны.

Целые числа хранятся в форме ***с фиксированной запятой***, действительные числа хранятся в форме с ***плавающей запятой***. В темах 8 и 9 можно прочитать подробное описание способов представления чисел в компьютерах. Заметим, что термин действительные числа в компьютерной терминологии заменяется на термин вещественные числа.

В этом параграфе отметим, что необходимость различного представления целых и действительных чисел вызвана тем, что скорость выполнения арифметических операций над числами с плавающей запятой существенно ниже скорости выполнения этих же операций над числами с фиксированной запятой.

Существует большой класс задач, в которых не используются действительные числа. Например, в задачах экономического характера, при решении которых данными служат количество деталей, акций, сотрудников и т.д., необходимы только целые числа. Текстовая, графическая и звуковая информация, как это будет показано ниже, также кодируется в компьютере с помощью целых чисел.

Для повышения скорости выполнения программ и используется представление целых чисел в форме с фиксированной запятой. Для решения математических и физических задач, в которых невозможно обойтись только целыми числами, используется представление в форме с плавающей запятой.

В современных персональных компьютерах процессоры выполняют операции только над целыми числами в форме с фиксированной запятой.

Для выполнения операций над числами в форме с плавающей запятой в персональном компьютере должен быть сопроцессор или заменяющие его специальные подпрограммы. В последнем случае существенно замедляется скорость выполнения задачи.

7.2. Представление текстовых данных

Любой текст состоит из последовательности символов. Символами могут быть буквы, цифры, знаки препинания, знаки математических действий, круглые и квадратные скобки и т.д. Особо обратим внимание на символ "пробел", который используется для разделения слов и предложений между собой. Хотя на бумаге или экране дисплея "пробел" — это пустое, свободное место, этот символ ничем не "хуже" любого другого символа. На клавиатуре компьютера или пишущей машинки символу "пробел" соответствует специальная клавиша.

Текстовая информация, как и любая другая, хранится в памяти компьютера в двоичном виде. Для этого каждому символу ставится в соответствие некоторое неотрицательное число, называемое ***кодом символа***, и это число записывается в память ЭВМ в двоичном виде. Конкретное соответствие между символами и их кодами называется ***системой кодировки***.

В современных ЭВМ, как правило, используются 8-разрядные коды символов (16-разрядные — в Windows 95, 98, NT). Использование 8-разрядных кодов позволяет закодировать 256 различных символов, чего вполне достаточно для практических нужд. В этом случае код символа занимает ровно один байт памяти.

В персональных компьютерах обычно используется система кодировки **ASCII** (*American Standard Code for Information Interchange* — американский стандартный код для обмена информацией). В ней для национальных алфавитов, таких, как русский, отводятся коды со 128-го по 255-й. К сожалению, исторически

сложилось так, что наш алфавит оказался закодирован пятью разными способами. Соответственно, мы имеем пять русскоязычных вариантов этой системы. Чаще всего используется вариант, известный под названием “Альтернативная кодировка”.

7.3. Представление графической информации

Мониторы современных компьютеров могут работать в двух режимах: текстовом и графическом.

В текстовом режиме экран обычно разбивается на 25 строк по 80 символов в строке. Каждая позиция экрана называется ***знакоместом***, и туда может быть помещен один символ. Режим предназначен для вывода на экран монитора текстов и простых рисунков, составленных из символов псевдографики. Всего на экране 25×80=2000 знакомест. Как уже было сказано, в каждом знакоместе находится ровно один символ (пробел, напоминаем, — равноправный символ, имеющий собственную кодировку). Символ этот может быть высвечен одним из 16 цветов.

Можно изменять и цвет фона (8 цветов), на котором рисуется символ, а кроме того, символ может мерцать. Для представления цвета символа нам требуется 4 бита (2^4= 16), для представления цвета фона требуется 3 бита (2^3= 8) и один бит — для представления мерцания (0 — не мерцает, 1 — мерцает).

Следовательно, для описания каждого знакоместа нам требуется 2 байта: первый байт — символ, второй байт — его цветовые характеристики. Таким образом, содержимое экрана в текстовом режиме монитора занимает в памяти компьютера ***(в видеопамяти)*** 2000×2 байта = 4000 байт ≈ 4 Кбайта.

В графическом режиме экран разделяется на отдельные светящиеся точки ***(пиксели)***, количество которых определяет разрешающую способность монитора и зависит от его типа и режима.

Любое графическое изображение хранится в памяти в виде информации о каждом пикселе на экране. Состояние каждого пикселя описывается последовательностью нулей и единиц, соответствующих кодировке его цвета.

Такую форму представления графических изображений называют ***растровой***. В зависимости от того, сколькими цветами мы можем высветить каждый пиксель, рассчитывается отводимый под него размер информации. Если монитор может работать с 16 цветами, то цвет каждого пикселя описывается 4 битами (2^4=16). Для работы с 256 цветами под каждый пиксель надо отвести 8 бит, или 1 байт (2^8=256).

Подсчитаем, сколько байт занимает в видеопамяти картинка, если на экран можно вывести 640×480 пикселей и монитор поддерживает 256 цветов:

640×480×1 байт =307 200 байтам = 300 Кбайтам.

7.4. Представление звуковой информации

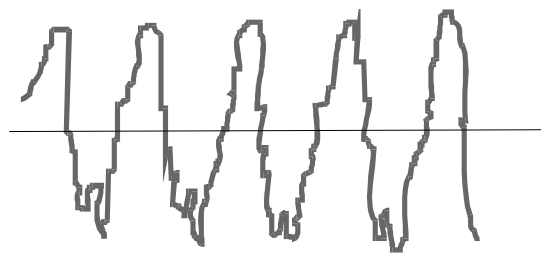
Развитие аппаратной базы современных компьютеров параллельно с развитием программного обеспечения позволяет сегодня записывать и воспроизводить на компьютерах музыку и человеческую речь. Существует два способа звукозаписи:

- ***Цифровая запись***, когда реальные звуковые волны преобразуются в цифровую информацию путем измерения звука тысячи раз в секунду. Этот процесс называется ***дискретизацией***, и, например, “частота дискретизации 44 кГц” означает, что измерения производятся 44 тысячи раз в секунду;
- ***MIDI-запись***, которая, вообще говоря, является не реальным звуком, а записью определенных команд-указаний (какие клавиши надо нажимать, например, на синтезаторе). MIDI-запись является электронным эквивалентом нотной записи.

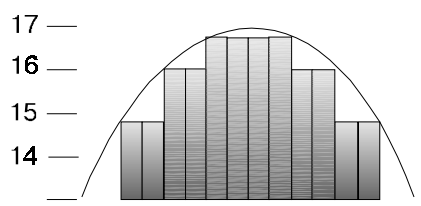
Первый из указанных способов можно реализовать, если в компьютере есть звуковая плата.

Звук представляет собой вибрации, которые формируют волну с соответствующими амплитудой и периодом.

Звуковая плата преобразует звук на входе в цифровую информацию путем измерения его характеристик (период, амплитуда) несколько тысяч раз в секунду. Получившиеся значения записываются в виде нулей и единиц в память компьютера (в файлы с расширением WAV). При воспроизведении звука специальное устройство звуковой карты преобразует цифры обратно в аналоговый сигнал, подающийся на самые обычные звуковые усилители, а оттуда на колонки. Хранение звука в виде цифровой записи занимает достаточно много места в памяти компьютера.



Реальные звуковые волны имеют весьма сложную форму, и для получения их высококачественного цифрового представления требуется высокая частота квантования



число 629,5 в двоично-десятичной системе будет записано в виде: 110 0010 1001,0101_{2-10}=
= $2^2 10^2 + 2\cdot 10^2 + 2\cdot 10 + 2^3 + 1 + 2^2 10^{-1} + 10^{-1}$.

Опустить здесь можно лишь первый незначащий ноль в двоично-десятичной записи первой цифры 6, двоичное же представление остальных десятичных цифр должно быть дополнено слева нулями до четырех двоичных разрядов.

Особый интерес в P-Q-ичных системах представляет случай, когда $Q=P^m$, где m — натуральное число, большее единицы. Тогда представление числа в P-ичной системе счисления совпадает с его представлением в P-Q-ичной системе. Системы с такими основаниями P и Q назовем ***смешанными*** позиционными системами счисления. Подробно они будут рассмотрены в главе 4.

В заключение еще раз проанализируем недостатки непозиционных систем счисления, которые помешали их широкому использованию в современном обществе.

- при работе с большими числами приходилось бы придумывать новые символы (цифры), причем этот процесс может продолжаться бесконечно (такая проблема у позиционных систем счисления отсутствует);
- правила формирования чисел даже из уже перечисленных цифр достаточно сложны, и далеко не очевидно, какая, например, из следующих форм записи числа 1998 в римской системе счисления верна: MCMXCVIII или MXMVIII (а действительно, какая из них верна?);
- встает проблема представления дробных частей чисел, особенно у иррациональных чисел (рациональные дроби можно записывать в виде отношения целых числителя и знаменателя);
- тем не менее запись чисел в римской системе находит ограниченное применение в настоящее время, в том числе при нумерации разделов в книгах и веков в исторических трудах, при оформлении циферблатов часов и т.д.

1.7. Примеры решения задач

1. Запишите в развернутом виде число $10203{,}405_7$.

Решение. Из формул (1.7) и (1.8) следует, что
$10203{,}405_7 = 1\cdot 7^4 + 2\cdot 7^2 + 3 + 4\cdot 7^{-1} + 5\cdot 7^{-3} =$

$$= 1\cdot 7^4 + 2\cdot 7^2 + 3 + \frac{4}{7} + \frac{5}{7^3}.$$

2. Покажите, что любое натуральное число может быть представлено в виде суммы различных неотрицательных степеней числа 2.

Решение. Это следует из возможности перевода любого натурального числа в двоичную систему счисления, вид числа в которой согласно формуле (1.7) и есть сумма степеней 2, включая нулевую степень, т.е. единицу.

3. В каких системах счисления $5_p + 5_p \neq 10_p$?

Решение. Заметим, что в любом случае основание в таких системах счисления должно быть больше 5, так как в системах с основаниями 2, 3, 4 и 5 цифра 5 в алфавите отсутствует, и выражение не имеет смысла. В системах с P=6, 7, 8, 9 — $5_p + 5_p > 10_p$, а при P=11, 12, …, ∞ — $5_p + 5_p < 10_p$, т.к. всегда $10_p = P$. Таким образом, равенство достигается лишь в десятичной системе счисления, а в остальных, с основанием $P>5$, нет.

4. В каких системах счисления $2_p + 2_p = 4_p$?

Решение. При решении этой задачи мы можем рассматривать лишь те системы счисления, в которых основание $P>4$, так как во всех их цифра 4 входит в алфавит, а 2 и 3 меньше $P-1$. В этих системах, дважды прибавляя 1 к двойке, мы всегда получим 4 (правило (1) для прибавления 1 к натуральному числу в P-ичной системе счисления). Следовательно, исходное равенство достигается при любом $P>4$.

5. Записать в системе счисления с основанием 234 число 235.

Решение. Несмотря на то, что вид всех цифр в подобной системе счисления может быть самым разнообразным, данное задание выполнить можно. Так как $234 = 10_{234}$, то, прибавив к нему 1, получим $235 = 11_{234}$.

6. Во сколько раз увеличится число 325_6, если приписать к нему справа один ноль?

Решение. Добавление справа одного нуля к любому числу, записанному в P-ичной системе счисления, соответствует умножению на $10_p = P$, значит, в нашем случае число возрастет в 6 раз.

7. Как изменится запись P-ичной дроби с нулевой целой частью, если ее разделить на P^2?

Решение. После запятой следует дописать два нуля, так как деление на $P^2 = 100_p$ соответствует умножению на $0{,}01_p$. Например: $0{,}03_5$ при делении на $25 = 100_5$ превратится в $0{,}0003_5$.

8. Будут ли справедливы признаки делимости натуральных чисел на 2, 3, 5, 9, 10, сформулированные для десятичной системы счисления, и в других системах?

Решение. Как правило, нет. Например, наличие последнего нуля в P-ичной записи числа говорит о его делимости на P, а не на 10. В системах счисления с четными основаниями четность последней цифры в записи числа указывает на четность самого числа, а в остальных системах счисления это не так (данный факт легко доказать, используя формулу (1.7)). Можно привести и ряд других примеров.

9. Число, записанное в десятичной системе счисления, оканчивается цифрой 5. Будет ли оно делиться на 5, если записать его в троичной системе счисления?

Решение. Будет, так как оно делится на 5 по признаку делимости в десятичной системе. Свойство же делимости числа никак не зависит от его записи в какой бы то ни было системе счисления.

1.6. Примеры применения недесятичных систем счисления

С точки зрения математики все позиционные системы счисления принципиально друг от друга ничем не отличаются, но на практике в разных ситуациях удобнее пользоваться разными системами.

Так, в повседневной жизни мы обычно пользуемся десятичной системой счисления (P=10). Очевидно, что эту систему мы предпочитаем остальным позиционным системам счисления лишь потому, что количество пальцев на руках у человека равно десяти, а именно пальцы первоначально служили основным "инструментом" для счета.

Таблицы сложения и умножения в десятичной системе, основные арифметические операции и правила переноса заучиваются нами еще в раннем детстве, и в результате повседневной практики мы оперируем ими чуть ли не подсознательно. По этой причине многие люди даже и не догадываются, что существуют другие системы счисления.

Приведем несколько примеров использования позиционных систем счисления, отличных не только от десятичной, но и от традиционных.

1. История подтверждает такой факт: шведский король Карл XII увлекался математикой и, в частности, считал восьмеричную систему счисления более удобной, чем десятичная. Карл XII намеревался королевским указом ввести восьмеричную систему счисления как общегосударственную, и только его неожиданная смерть помешала осуществлению этого необычного намерения.

2. В современных электронных вычислительных машинах для организации арифметических операций используются двоичная и смешанная двоично-шестнадцатеричная системы счисления. Подробнее вы можете прочитать об этом в части III данной книги.

3. В главе 4 приведена классическая задача на взвешивания, которую достаточно просто решить, если выбрать подходящую систему счисления. В данном случае — вариант троичной. Использование этого варианта троичной системы счисления легло и в основу конструирования вычислительной машины "Сетунь". Особенности этой машины до сих пор привлекают к ней пристальное внимание. Есть ряд серьезных соображений, которые позволяют считать инженерные поиски в создании троичных машин незаконченными. Подробнее о так называемой уравновешенной троичной системе счисления рассказывается в параграфе 4.4.

4. В реальной жизни для описания определенных явлений мы пользуемся мерами или масштабами, которые можно соотнести с нетрадиционными позиционными системами счисления.

Так, мы ежедневно сталкиваемся с измерением времени. Одной из форм измерения времени является следующая: *дни, часы, минуты, секунды, десятые доли секунды, сотые доли секунды.* Для записи времени в секундах можно воспользоваться системой счисления со следующим базисом: ...10^{-2}, 10^{-1}, 1, 60, 600, 3600, 86400, 864000, ...

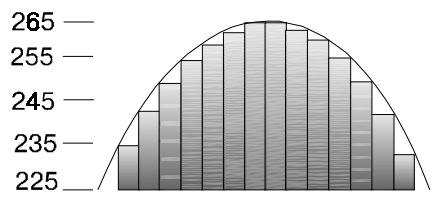
Дни (десятки)	$10 \cdot 24 \cdot 60 \cdot 60$
Дни (единицы)	$24 \cdot 60 \cdot 60$
Часы	$60 \cdot 60$
Минуты	60
Секунды	1
Десятые доли секунды	10^{-1}
Сотые доли секунды	10^{-2}

Такая система имеет 3 различных основания:
P_1=10 для записи долей секунд и количества дней в интересующем нас промежутке времени (дней может быть как угодно много);
P_2=24 для представления часов;
P_3=60 для минут и секунд (подразумевается, что любое количество целых минут или секунд, не превышающее 59, составляет одну цифру в соответствующем разряде такой системы).
Итак, если нам требуется подсчитать количество секунд в 12 днях 14 часах 35 минутах и 16,97 секунды, необходимо использовать следующую формулу: $1 \cdot 864000 + 2 \cdot 86400 + 14 \cdot 3600 + 35 \cdot 60 + 16 + 9 \cdot 10^{-1} + 7 \cdot 10^{-2}$.

5. До 1971 года в Англии в денежной системе использовались следующие денежные единицы:
1 фунт стерлингов = 20 шиллингам;
1 шиллинг = 12 пенсам.
Аналогично подсчету времени в секундах для подсчета значения некоторой суммы денег в пенсах использовалась позиционная система счисления с базисом 1, 12, 240, 2400, ...
В настоящее же время, когда один фунт считается равным 100 пенсам, в английской денежной системе используется традиционная десятичная система счисления.

6. Во многих ЭВМ первых поколений применялась двоично-десятичная система счисления, являющаяся примером ***P*-*Q*-ичной** позиционной системы записи чисел (здесь P<Q). В такой системе каждая цифра числа, заданного в Q-ичной системе, заменяется ее представлением в P-ичной системе. Стало быть, в двоично-десятичной системе каждая десятичная цифра будет записана в двоичной системе. При этом количество разрядов, отводимых под запись каждой цифры Q-ичной системы в ее P-ичном представлении, равно числу P-ичных цифр в максимальной цифре Q-ичной системы счисления.

Пример 1.11. В двоично-десятичной системе максимальная цифра 9 из десятичной системы представляется четырьмя цифрами двоичной (9_{10}=1001_2). Тогда



Число разрядов, используемое для создания цифрового звука, определяет качество звучания

MIDI-запись была разработана в начале 80-х годов (MIDI — *Musical Instrument Digital Interface* — интерфейс цифровых музыкальных инструментов). MIDI-информация представляет собой команды — инструкции синтезатору, а не параметры звуковых волн. MIDI-команды гораздо удобнее для хранения музыкальной информации, чем цифровая запись. Однако для записи MIDI-команд требуется устройство, выполняющее роль клавишного синтезатора, которое воспринимает MIDI-команды и при их получении может генерировать соответствующие звуки.

Вопросы и задания

1. Какой объем памяти в байтах будет занимать текст этого вопроса в компьютере?
2. Выпишите первую половину (128 символов) таблицы ASCII.
3. В графическом режиме на экране дисплея пиксель можно отображать одним из 1024 цветов. Сколько бит требуется для описания цвета каждого пикселя, если разрешение экрана составляет 800×600 пикселей?
4. Как вы понимаете термин оцифровывание звука?

Тема 8
Компьютерное представление целых чисел

Более подробно материал этой темы изложен в главе 6 третьей части.

Целые числа являются простейшими числовыми данными, с которыми оперируют ЭВМ. Для целых чисел существуют два представления: беззнаковое (только для неотрицательных целых чисел) и со знаком. Очевидно, что отрицательные числа можно представлять только в знаковом виде.

Целые числа в машине хранятся в формате ***с фиксированной запятой***. Для беззнакового представления все разряды ячейки отводятся под представление самого числа.

Для представления со знаком самый старший (левый) разряд отводится под знак числа, остальные разряды — под само число. Если число положительное, то в знаковый разряд помещается 0, если число отрицательное — 1.

Различие в представлении целых чисел со знаком и без знака вызвано тем, что в ячейках одного и того же размера в беззнаковом типе можно представить больше различных положительных чисел, чем в знаковом.

Например, в байте (8 разрядов) можно представить беззнаковые числа от 0 до 255, а знаковые — только до 127. Поэтому если известно заранее, что некоторая числовая величина является неотрицательной, то выгоднее оперировать с ней как с беззнаковой.

8.1. Прямой код числа

Представление числа в форме "знак" — "величина", когда старший разряд ячейки отводится под знак, а остальные разряды ячейки — под запись числа в двоичной системе, называется ***прямым кодом*** двоичного числа.

Например, прямой код двоичных чисел 1001_2 и -1001_2 для 8-разрядной ячейки равен 00001001 и 10001001 соответственно.

Положительные числа в ЭВМ всегда представляются с помощью прямого кода. Прямой код числа полностью совпадает с записью самого числа в ячейке машины. Прямой код отрицательного числа отличается от прямого кода соответствующего положительного числа лишь содержимым знакового разряда. Но отрицательные целые числа представляются в ЭВМ с помощью совсем другого кода, который называется ***дополнительным кодом***.

8.2. Дополнительный код числа

Дополнительный код положительного числа равен прямому коду этого числа.

Дополнительный код отрицательного числа m равен $2^k-|m|$, где k — количество разрядов в ячейке, а $|m|< 2^k$.

(Для особо дотошных заметим, что в компьютерной k-разрядной арифметике $2^k \equiv 0$, так как двоичная запись этого числа состоит из одной единицы и k нулей, а в ячейку из k разрядов может уместиться только k цифр, в данном случае они все — нули. Таким образом, дополнительный код отрицательного числа — это дополнение $|m|$ до 2^k (или до нуля в k-разрядной арифметике: $2^k-|m|+|m|=0$).)

Как уже было сказано, при представлении неотрицательных чисел в беззнаковом формате все разряды ячейки отводятся под само число. Например, запись числа 243 в одном байте при беззнаковом представлении будет выглядеть следующим образом:

1	1	1	1	0	0	1	1

— так как 243_{10}=11110011_2.

При представлении целых чисел со знаком самый старший (левый) разряд отводится под знак числа, и под собственно число остается на один разряд мень-

ше. Поэтому если приведенное выше состояние ячейки рассматривать как запись целого числа со знаком в дополнительном коде, то для компьютера в этой ячейке записано число −13 (так как 243+13=256=2^8).

Но если это же отрицательное число записать в ячейку из 16 разрядов, то содержимое ячейки будет следующим:

1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

↑ знаковый разряд

Из приведенных примеров видно, что представление отрицательного числа зависит от разрядности ячейки, в которую записывается число.

С какой целью отрицательные числа записывают в виде дополнительного кода и как получить дополнительный код отрицательного числа?

Дополнительный код используется для упрощения выполнения арифметических операций. Если бы вычислительная машина работала с прямыми кодами положительных и отрицательных чисел, то при выполнении арифметических операций следовало бы выполнять ряд дополнительных действий.

Например, при сложении нужно было бы проверять знаки обоих операндов и определять знак результата. Если знаки одинаковые, то вычисляется сумма операндов и ей присваивается тот же знак. Если знаки разные, то из большего по абсолютной величине числа вычитается меньшее и результату присваивается знак большего числа.

То есть при таком представлении чисел (в виде только прямого кода) операция сложения реализуется через достаточно сложный алгоритм. Если же отрицательные числа представлять в виде дополнительного кода, то операция сложения, в том числе и чисел разного знака, сводится только лишь к их поразрядному сложению. Желающие могут это проверить на нескольких примерах.

Для компьютерного представления целых чисел обычно используется один, два или четыре байта, т.е. ячейка памяти будет состоять из восьми, шестнадцати или 32 разрядов соответственно.

8.3. Алгоритм получения дополнительного кода отрицательного числа

Для получения дополнительного k-разрядного кода отрицательного числа необходимо

1) модуль числа представить прямым кодом в k двоичных разрядах;
2) значения всех битов **инвертировать**: все нули заменить на единицы, а единицы на нули (таким образом получается k-разрядный **обратный код** исходного числа);
3) к полученному обратному коду, трактуемому как k-разрядное неотрицательное двоичное число, прибавить единицу.

Пример 29. Получим *восьмиразрядный* дополнительный код числа −52:

00110100 — число |−52|=52 в прямом коде;
11001011 — число −52 в обратном коде;
11001100 — число −52 в дополнительном коде.

Напомним, что при различных значениях k вид отрицательного числа в дополнительном коде также будет различным.

Пример 30. Получим дополнительный код числа −52 в *шестнадцати разрядах*:

0000 0000 0011 0100 — прямой код числа |−52|;
1111 1111 1100 1011 — обратный код числа −52;
1111 1111 1100 1100 — дополнительный код числа −52.

8.4. Диапазон значений целых чисел

Рассмотрим диапазон значений целых чисел в беззнаковом представлении в зависимости от разрядности ячейки. Понятно, что минимальное число, которое можно записать в k битах, будет состоять из k нулей, т.е. это число 0. Максимальное число будет состоять из k единиц. Это число (k единиц в двоичной системе счисления) в десятичной системе счисления равно 2^k-1. Действительно, запишем это двоичное число в развернутой форме (3):

$$2^{k-1}+2^{k-2}+\ldots+2^2+2+1=\frac{2^{k-1}\cdot 2-1}{2-1}=2^k-1$$

(здесь использована формула суммы конечной геометрической прогрессии).

Таким образом, для беззнаковых чисел нижняя граница диапазона значений всегда равна нулю, а верхнюю границу диапазона допустимых значений можно подсчитать, зная количество разрядов, занимаемых данным числом.

При наличии одинакового числа разрядов в ячейке максимальное значение для знакового представления будет практически в два раза меньше, чем для беззнакового. Это связано с тем, что один разряд в знаковом представлении отводится под знак числа, и тем самым для представления самого числа отводится на один разряд меньше.

Выпишем таблицу границ диапазонов для знаковых и беззнаковых представлений в ячейках с различной разрядностью.

Для записи чисел, больших P в P-ичной системе счисления, воспользуемся следующими несложными правилами, описывающими, как по известной P-ичной форме записи натурального числа a_p получить запись следующего натурального числа a_p+1:

1) если последняя (крайняя справа) цифра числа a_p меньше, чем $P-1$, то в следующем по порядку натуральном числе все цифры, кроме последней, будут совпадать с цифрами числа a_p, а последняя цифра числа a_p+1 будет на единицу больше последней цифры числа a_p;

2) если последняя (крайняя справа) цифра числа a_p равна $P-1$, то последняя цифра числа a_p+1 будет равна 0, а остальные цифры будут представлять число, состоящее из первых цифр числа a_p (начиная с крайней левой цифры и заканчивая предпоследней справа), увеличенное на единицу по правилам (1)−(2); если же первые цифры в записи a_p отсутствуют, то число a_p+1 будет равно 10_p.

Покажем, как, используя правила (1)−(2), перечислять натуральные числа в различных системах счисления.

Пример 1.6. В двоичной системе первые 8 чисел будут иметь следующий вид:
0=0_2;
1=1_2;
2=10_2 (правило (2));
3=11_2 (правило (1));
4=100_2 (дважды примененное правило (2));
5=101_2 (правило (1));
6=110_2 (правила (2) и (1));
7=111_2 (правило (1));
8=1000_2 (трижды примененное правило (2));
и т.д.

Пример 1.7. Приведем, но уже без подробных комментариев, некоторые числа в 16-ричной системе счисления:
10=A_{16};
11=B_{16};
12=C_{16};
13=D_{16};
14=E_{16};
15=F_{16};
16=10_{16};
17=11_{16};
18=12_{16};
19=13_{16};
20=14_{16};
...
31=$1F_{16}$;
32=20_{16};
...
255=FF_{16};
256=100_{16}.

Перейдем теперь к представлению P-ичных дробей. Рассмотрим обыкновенные дроби, записанные с помощью отношения числителя и знаменателя, наибольший общий делитель которых равен 1. Как и в десятичной системе счисления, такие дроби будут точно представимы ***конечной*** P-ичной дробью, если существует такое натуральное число m, что при умножении на него знаменателя дроби можно получить некоторую натуральную степень числа P. Если же такого числа не существует, то в P-ичной системе счисления дробь окажется бесконечной периодической.

Данный факт следует непосредственно из формулы (1.8). Например, в десятичной системе счисления

$$\frac{1}{4}=\frac{1\cdot 25}{4\cdot 25}=\frac{25}{100}=0{,}25 \text{ (здесь } m=25).$$

Кроме того, из развернутого представления дробной части числа следует, что в любой системе счисления с основанием P верны равенства:

$$\frac{1}{P}=0{,}1_P;\ \frac{1}{P^2}=0{,}01_P;\ldots;\ \frac{1}{P^k}=0,\underbrace{0\ldots0}_{k-1}1_P. \qquad (1.10)$$

Пусть для нашей дроби такое натуральное число m существует. При умножении знаменателя на m получим k-ю степень числа P. Умножим и числитель дроби на m, представив результат умножения в P-ичной системе счисления. Дополним его, если потребуется, до k цифр нулями слева. Полученное число, записанное после запятой, и будет являться конечной P-ичной дробью для исходной рациональной.

Пример 1.8. $\frac{5}{16}$ запишем в двоичной системе. Так как 16 уже является четвертой степенью двойки, то переведем 5 в двоичную систему (5=101_2) и дополним до четырех цифр — 0101.

В результате получаем: $\frac{5}{16}=0{,}0101_2$.

Пример 1.9. $\frac{5}{6}$ запишем в 12-ричной системе. Для того чтобы знаменатель стал первой степенью двенадцати, его нужно умножить на 2. Тогда, умножая числитель на 2 и представляя его с помощью одной цифры 12-ричной системы, получим $\frac{5}{6}=0{,}A_{12}$.

В главе 3 будут даны способы перевода произвольных дробей из одной системы в другую. Однако уже сейчас можно заметить, что конечные десятичные дроби могут оказаться периодическими P-ичными дробями, и наоборот, периодические десятичные дроби могут стать в некоторых системах счисления конечными P-ичными дробями.

Так, например, $0{,}1_{10}=0{,}0(0011)_2$, а $0{,}(3)_{10}=0{,}1_3$. Но любая конечная двоичная дробь всегда будет конечной в десятичной системе счисления, так как ее знаменатель, являющийся некоторой степенью двух, всегда можно умножить на число m, равное такой же степени пятерки, и получить степень десяти.

Пример 1.10. $0{,}011_2=\frac{11_2}{1000_2}$ запишем в десятичной системе. Для того чтобы знаменатель, равный сейчас 2^3, оказался степенью десяти, его нужно умножить на 5^3. Таким образом, производя необходимые действия в десятичной системе счисления, получим $\frac{11_2}{1000_2}=\frac{3}{8}=\frac{3\cdot 5^3}{8\cdot 5^3}=\frac{375}{1000}=0{,}375$.

Отрицательные числа в P-ичных системах счисления представляются с помощью знака "—" перед выражением вида (1.5) для модуля отрицательного числа. Поэтому далее в части II данной книги мы будем рассматривать только положительные числа.

Назовем формулу (1.5) ***основной формулой для P-ичных систем счисления***. Именно их мы и будем рассматривать далее. Позиционные системы, базис которых не составляет геометрическую прогрессию, будут рассмотрены в главе 4.

Как следует из принципа позиционности, P-ичная система счисления позволяет с помощью заранее ограниченного набора цифр записать как сколь угодно большое, так и сколь угодно малое число.

Любое число в позиционной системе счисления с основанием P можно записать и просто путем последовательного перечисления его значащих цифр начиная со старшей, запятой отделяя целую часть от дробной. То есть разложению вида (1.5) вещественного числа a по степеням P соответствует запись вида

$$a = a_n \dots a_1 a_0, \; a_{-1} \dots a_{-k} \dots \qquad (1.6)$$

Определение. Представление числа в P-ичной системе счисления в виде (1.5) называется ***развернутой формой*** записи числа (эта форма в основном используется при решении задач).

Определение. Представление числа в P-ичной системе счисления в виде (1.6) называется ***свернутой формой*** записи числа (именно так чаще всего изображаются числа в позиционных системах счисления).

Так, натуральное число a в P-ичной системе счисления можно записать как

$$a = a_n P^n + a_{n-1} P^{n-1} + \dots + a_1 P + a_0 = a_n a_{n-1} \dots a_1 a_0 \quad (1.7)$$

Правильную же конечную P-ичную дробь b можно записать в виде

$$b = b_{-1} P^{-1} + b_{-2} P^{-2} + \dots + b_{-k} P^{-k} = \\ = 0, b_{-1} b_{-2} \dots b_{-k} \qquad (1.8)$$

При использовании развернутой формы для записи числа в P-ичной системе счисления основание P обычно записывают в десятичной системе, а цифры — в P-ичной. При записи чисел в P-ичной системе счисления в свернутой форме цифры также записывают в P-ичной системе, а основание P, записанное в десятичной системе, приписывают к числу в качестве его нижнего индекса.

Исключение может составлять лишь десятичная система, при записи чисел в которой индекс часто опускается.

Из теоремы 1 следует, что любое натуральное число можно записать в какой угодно позиционной системе счисления, причем единственным образом.

Пример 1.4. Десятичное число 23 можно записать
в двоичной системе счисления как 10111_2;
в троичной системе счисления как 212_3;
в четверичной системе как 113_4;
в шестнадцатеричной системе как 17_{16};
в 23-ичной системе как 10_{23}.

В системах счисления с основанием большим, чем 23, данное число будет представлено одной цифрой (это будет, например, буква латинского алфавита N или некий другой символ. Подробнее о виде цифр в P-ичных системах счисления было рассказано в параграфе 1.3).

При вычислении значения P-ичного числа с использованием развернутой формы удобна ***схема Горнера***, позволяющая получить результат за минимальное число следующих трех арифметических операций: +, ×, : (операция возведения в степень не используется). Подробнее о применении схемы Горнера см. в параграфах 3.1 и 3.2 главы 3.

Говорить об однозначном представлении дробных чисел в P-ичных системах счисления (в том числе и в десятичной), вообще говоря, некорректно.

Покажем, что каждую конечную P-ичную дробь $b = 0, b_{-1} b_{-2} \dots b_{-k}$ можно записать и в форме бесконечной периодической дроби с периодом, состоящим из максимальной цифры соответствующей системы счисления.

Непериодическая часть такой дроби равна числу $b_{-1} b_{-2} \dots b_{-k}$, уменьшенному на единицу. Так как $b_{-k} > 0$, то на единицу будет уменьшена именно эта цифра, а остальные останутся неизменными.

В развернутой форме эта дробь будет иметь вид:

$$b_{-1} P^{-1} + b_{-2} P^{-2} + \dots + (b_{-k} - 1) P^{-k} + (P-1) P^{-(k+1)} + \\ + (P-1) P^{-(k+2)} + \dots \qquad (1.9)$$

Факт равенства значения выражения (1.9) числу b непосредственно следует из формулы суммы бесконечной геометрической прогрессии со знаменателем $q = P^{-1} < 1$, которая соответствует периодической части дроби (1.9). В данном случае первый член прогрессии — это $(P-1) P^{-(k+1)}$, а сама сумма равна

$$\frac{(P-1)P^{-(k+1)}}{1 - P^{-1}} = \frac{P(P-1)P^{-(k+1)}}{P-1} = 1 \cdot P^{-k}.$$

То есть всю периодическую часть дроби (1.9) можно заменить на единицу в разряде $(-k)$ и, следовательно, период можно отбросить, добавляя при этом единицу к непериодической части дроби (1.9), в результате получаем дробь $b = 0, b_{-1} b_{-2} \dots b_{-k}$, что и требовалось доказать.

Пример 1.5

$1_{10} \equiv 0,(9)_{10}$; $0,001_2 \equiv 0,000(1)_2$; $0,9AF_{16} \equiv 0,9AE(F)_{16}$.

1.5. Представление чисел в *P*-ичной системе счисления

Напомним (см. п. 3.2 части I), что, прежде чем приниматься за арифметические действия в какой-либо системе счисления, необходимо научиться перечислять в ней идущие по порядку натуральные числа и представлять дробные части вещественных чисел.

Как следует из параграфа 1.4, в любой P-ичной системе счисления натуральные числа, меньшие ее основания P, представляются с помощью одной цифры данной системы. Для чисел же, больших или равных P, требуются по крайней мере две цифры. Само же число P в системе счисления с основанием P записывается в виде 10_p, что следует из развернутой формы записи числа P в P-ичной системе: $P = 1 \cdot P + 0$.

Разрядность	8	16	32
Минимум (без знака)	0	0	0
Максимум (без знака)	255	65535	4294967295
Минимум (со знаком)	−128	−32768	−2147483648
Максимум (со знаком)	127	32767	2147483647

Вопросы и задания

1. От чего зависят границы диапазона представимых в компьютере целых чисел?
2. Как будут представлены в 8-битном знаковом типе числа −1, −10, −120?
3. Запишите следующие двоичные числа в прямом, обратном и дополнительном коде для 8-разрядной ячейки: −1000; −11101; −1; −1111111.
4. Объясните, как дополнительный код позволяет заменить операцию вычитания операцией сложения.

Тема 9
Компьютерное представление вещественных чисел

Более подробно материал этой темы изложен в главе 7 третьей части.

В предыдущей теме было рассказано, что существуют два основных типа представления чисел в компьютере, называемые представлениями ***с фиксированной*** и ***с плавающей запятой***. При представлении чисел с фиксированной запятой все разряды ячейки, кроме знакового разряда — если он есть, служат для изображения разрядов числа. Причем каждому разряду ячейки соответствует всегда один и тот же разряд числа. Именно поэтому такое ***представление*** получило название ***с фиксированной запятой***, так как фиксируется место запятой перед определенным разрядом (для целых чисел запятая находится после младшего разряда, т.е. вне разрядной сетки). Такая система упрощает выполнение арифметических действий, но сильно ограничивает диапазон чисел, которые можно записать в ячейку при таком представлении.

Для представления вещественных чисел в современных компьютерах принят способ представления ***с плавающей запятой***. Этот способ представления опирается на ***нормализованную (экспоненциальную) запись действительных чисел***.

Как и для целых чисел, при представлении действительных чисел в компьютере используется двоичная система счисления, следовательно, предварительно десятичное число должно быть переведено в двоичную систему.

9.1. Нормализованная запись числа

Определение. ***Нормализованной*** называется запись отличного от нуля действительного числа в виде $m \cdot P^q$, где q — целое число (положительное, отрицательное или ноль), а m — правильная P-ичная дробь, у которой первая цифра после запятой не равна нулю, т.е. $1/P \le m < 1$. При этом m называется ***мантиссой*** числа, q — ***порядком*** числа.

Пример 31. Приведем примеры нормализации чисел.

1) $3,1415926 = 0,31415926 \cdot 10^1$;
2) $1000 = 0,1 \cdot 10^4$;
3) $0,123456789 = 0,123456789 \cdot 10^0$;
4) $0,0000107_8 = 0,107_8 \cdot 8^{-4}$;
5) $1000,0001_2 = 0,10000001_2 \cdot 2^4$.

Заметим, что число "ноль" не может быть записано в нормализованной форме так, как она была определена. Поэтому относительно нормализованной записи нуля приходится прибегать к особым соглашениям. Условимся, что запись нуля является нормализованной, если и мантисса, и порядок равны нулю, т.е. $0 = 0,0 \cdot 10^0$.

9.2. Представление чисел с плавающей запятой

При представлении чисел с плавающей запятой часть разрядов ячейки отводится для записи порядка числа, остальные разряды — для записи мантиссы. По одному разряду в каждой группе отводится для изображения знака порядка и знака мантиссы.

Например, можно представить себе такое распределение разрядов ячейки памяти:

s_p	s_q	b_k	b_{k-1}	...	b_2	b_1	a_1	a_2	...	a_{n-1}	a_n

Первые два разряда служат для изображения знаков порядка и мантиссы соответственно: 0 — если знак плюс, 1 — знак минус. Следующие k разрядов используются для изображения абсолютной величины порядка числа, остальные n разрядов используются для изображения абсолютной величины мантиссы. Порядок и мантисса записаны в системе счисления с основанием 2.

Тогда изображенному на схеме состоянию ячейки соответствует число $(-1)^{s_q} 2^p \left(\frac{a_1}{2} + \frac{a_2}{2^2} + \dots + \frac{a_n}{2^n}\right)$, где $p = (-1)^{s_p} (b_k \cdot 2^{k-1} + b_{k-1} \cdot 2^{k-2} + \dots + b_2 \cdot 2 + b_1)$.

При такой системе записи наибольшее по абсолютной величине число, которое может быть представлено в машине, равно $2^{2^k - 1}(1 - 2^{-n})$, а наименьшее по абсолютной величине, отличное от нуля, равно $2^{-(2^k - 1)} \cdot 2^{-n} = 2^{-2^k - n + 1}$.

На точность вычислений оказывает влияние длина мантиссы, а количество разрядов, отводимых под порядок, влияет на допустимый диапазон представимых чисел. Очевидно, чем большая точность нам требуется, тем более "длинную" ячейку придется использовать.

9.3. Выполнение арифметических операций над числами с плавающей запятой

Использование в компьютере или калькуляторе представления чисел с плавающей запятой усложняет схему АЛУ. Для того чтобы понимать, с какой точностью и в каком диапазоне значений мы можем производить арифметические операции над действительными числами на том или ином калькуляторе или компьютере, необходимо знать, как эти операции выполняются.

При сложении и вычитании чисел сначала производится подготовительное действие, называемое ***выравниванием порядков***. Порядок меньшего по модулю числа при этом становится равным порядку большего. Для этого мантисса числа с меньшим порядком сдвигается вправо на количество разрядов, равное разности порядков данных чисел. После этой операции одинаковые разряды чисел оказываются расположенными в одних и тех же по номеру разрядах ячеек, отведенных под операнды, и теперь уже сложение или вычитание мантисс выполняется достаточно просто, так же, как над целыми числами.

Пример 32. Предположим, что в ячейке памяти калькулятора можно записать один десятичный разряд порядка и пять десятичных разрядов мантиссы. Требуется выполнить сложение следующих чисел:

$$\begin{array}{rl} & 10^{2}\cdot 0{,}23619 \\ + & 10^{-2}\cdot 0{,}71824 \end{array}$$

Перед сложением производится выравнивание порядков. Число с меньшим порядком преобразуется в число с порядком, равным порядку другого слагаемого (меньший порядок "приводится" к большему). В данном случае второе слагаемое будет преобразовано к виду $10^{2}\cdot 0{,}00071824$, после чего выполняется сложение:

$$\begin{array}{l} 10^{2}\cdot 0{,}23619 \\ 10^{2}\cdot 0{,}00071824 \\ \hline 10^{2}\cdot 0{,}23690824. \end{array}$$

Результат получили с восемью значащими цифрами, в то время как в ячейку памяти можно записать только пять, поэтому он округляется и записывается в виде:

$$10^{2}\cdot 0{,}23691.$$

При умножении двух целых чисел с плавающей запятой их порядки необходимо просто сложить, а мантиссы — перемножить без предварительного выравнивания.

При делении из порядка делимого надо вычесть порядок делителя, а мантиссу делимого разделить на мантиссу делителя. Помимо потери точности в результате операций над мантиссами, в данном случае может произойти переполнение порядка, то есть сумма или разность порядков может оказаться как больше максимально представимого порядка, так и меньше минимально возможного.

Пример 33. С помощью описанного в предыдущем примере калькулятора выполним умножение $0{,}2\cdot 10^{5}\cdot 0{,}5\cdot 10^{5}=0{,}1\cdot 10^{10}$.

Поскольку в нашем калькуляторе для порядка отведен лишь один десятичный разряд, результат умножения двух представимых в нем чисел не может быть в нем записан.

После операций над порядками и мантиссами мы получаем порядок и мантиссу результата. Однако мантисса результата может не удовлетворять ограничениям, накладываемым на мантиссы нормализованных чисел. Так как от результата арифметических операций в машине требуется, чтобы он также был нормализованным числом, необходимо дополнительное преобразование результата, называемое ***нормализацией***.

В зависимости от величины получившейся мантиссы результата она сдвигается вправо или влево так, чтобы ее первая значащая цифра попала в первый разряд после запятой. Одновременно порядок результата увеличивается или уменьшается на число, равное величине сдвига.

Заметим, что над мантиссами в арифметико-логическом устройстве должны выполняться все четыре арифметических действия, а также операции сдвига, тогда как над порядками производятся только действия сложения и вычитания. А если отрицательные порядки записать в дополнительном коде, то и операция вычитания сводится к операции сложения.

Вопросы и задания

1. Приведите к нормализованному виду следующие числа, оставляя их в той же системе счисления, в которой они записаны: $-0{,}000001011101_2$; 9876543210_{10}; $100{,}01_2$.
2. Запишите в естественной форме с фиксированной запятой следующие нормализованные числа: $0{,}1011_2\cdot 2^{1}$; $0{,}1011_2\cdot 2^{11}$; $0{,}12345_{10}\cdot 10^{-3}$.
3. Объясните, что и когда "плавает" в форме представления чисел "с плавающей запятой"? Приведите убедительные примеры.

   Перечислите ошибки, возникающие при работе с вещественными числами в конечном числе разрядов. Проиллюстрируйте с помощью вашего калькулятора.

В результате получим, что

$$X = a_n\cdot P^{n} + a_{n-1}\cdot P^{n-1} + \ldots + a_1\cdot P + a_0,$$

где $0\le a_i<P$, $0\le i\le n$, $a_n\ne 0$.

Единственность. Для доказательства воспользуемся методом от противного.

Предположим, что некоторое натуральное число имеет два различных представления вида (1.1):

$$X_1 = a_n\cdot P^{n} + a_{n-1}\cdot P^{n-1} + \ldots + a_1\cdot P + a_0 \qquad (1.4A)$$

и

$$X_2 = b_m\cdot P^{m} + b_{m-1}\cdot P^{n-1} + \ldots + b_1\cdot P + b_0 \qquad (1.4B)$$

Покажем, что если $m>n$, то $X_2>X_1$. Для доказательства оценим X_2 снизу: $X_2\ge 1\cdot P^{m}+0\cdot P^{m-1}+\ldots+0\cdot P+0=P^{m}$. (Число b_m — единственное, которое не может быть нулем. Мы выбираем его минимальным — равным 1.)

Оценим X_1 сверху, выбрав максимально возможные коэффициенты: $a_n=\ldots=a_1=a_0=P-1$. Тогда:

$$X_1\le (P-1)P^{n}+(P-1)P^{n-1}+\ldots+(P-1)P+(P-1)=P^{n+1}-1<P^{n+1}.$$

Здесь для вычисления суммы использовалась формула суммы членов конечной геометрической прогрессии.

Так как по предположению $m\ge n+1$, то $X_1<P^{n+1}\le X_2$, т.е. $X_1<X_2$.

Следовательно, если $X_1=X_2$, то $n=m$.

Пусть далее существует такое k, что $a_i=b_i$ при $k+1\le i\le n=m$, но $a_k\ne b_k$. Сравним числа

$$Y_1 = X_1 - a_n\cdot P^{n} - \ldots - a_{k+1}\cdot P^{k+1}$$

и

$$Y_2 = X_2 - b_n\cdot P^{n} - \ldots - b_{k+1}\cdot P^{k+1}.$$

Получим

$$Y_1 = a_k\cdot P^{k} + \ldots + a_1\cdot P + a_0,\ Y_2 = b_k\cdot P^{k} + \ldots + b_1\cdot P + b_0,$$

где $a_k\ne b_k$. Тогда $Y_1\ne Y_2$, и, следовательно, $X_1\ne X_2$.

Получили противоречие с исходным предположением о равенстве представлений (1.4*a*) и (1.4*б*) — значит, $a_i=b_i$ при $0\le i\le n=m$ и представление вида (1.1) для любого натурального числа единственно.

Теорема доказана.

Пример 1.3. Построим представление десятичного числа $X=3056$ в виде степенных рядов при различных значениях P:

1) $P=10$.

Очевидно, что $10^{3}\le 3056<10^{4}$, то есть в представлении (1.1) $n=3$.

Разделив интервал $[10^{3};10^{4}]$ на 9 равных частей, получим, что $3\cdot 10^{3}\le 3056<4\cdot 10^{3}$ и $a_3=3$.

Тогда $Y=3056-3\cdot 10^{3}=56$, и так как $10\le 56<10^{2}$, то $a_2=0$.

Далее получаем: $5\cdot 10\le 56<6\cdot 10$ и $a_1=5$.

Оставшееся число $Z=Y-5\cdot 10=56-50=6<10$, следовательно, $a_0=6$ и построение закончено.

В результате $3056=3\cdot 10^{3}+5\cdot 10+6$.

2) $P=16$.

Так как $16^{2}\le 3056<16^{3}$, то в представлении (1.1) $n=2$.

Разделив интервал $[16^{2};16^{3}]$ на 15 равных частей, получим, что $11\cdot 16^{2}\le 3056<12\cdot 16^{2}$ и $a_2=11$.

Тогда $Y=3056-11\cdot 16^{2}=240$. Далее аналогично $15\cdot 16\le 240<16^{2}$ и $a_1=15$.

И так как $240-15\cdot 16=0$, то построение окончено, а $a_0=0$.

В результате $3056=11\cdot 16^{2}+15\cdot 16$.

3) $P=2$.

$2^{11}\le 3056<2^{12}$, то есть в представлении (1.1) $n=11$.

В отличие от рассмотренных случаев (1)–(2), делить найденный интервал на более мелкие уже не требуется, и $a_{11}=1$.

Далее $3056-2^{11}=1008$ и $2^{9}\le 1008<2^{10}$, т.е. $a_{10}=0$, $a_9=1$;

$1008-2^{9}=496$ и $2^{8}\le 496<2^{9}$, т.е. $a_8=1$;

$496-2^{8}=240$ и $2^{7}\le 240<2^{8}$, т.е. $a_7=1$;

$240-2^{7}=112$ и $2^{6}\le 112<2^{7}$, т.е. $a_6=1$;

$112-2^{6}=48$ и $2^{5}\le 48<2^{6}$, т.е. $a_5=1$;

$48-2^{5}=16$ и $2^{4}\le 16<2^{5}$, т.е. $a_4=1$;

$16-2^{4}=0$ и $a_3=a_2=a_1=a_0=0$.

В результате $3056=2^{11}+2^{9}+2^{8}+2^{7}+2^{6}+2^{5}+2^{4}$.

Задумаемся над доказанным утверждением. Достаточно элементарные объекты — степени одного числа — представляют столь простым образом все натуральные числа. Это удивительный, поразительный факт!

В разделе математики "Теория чисел" доказывается, что и любую правильную дробь можно представить в виде конечной или бесконечной суммы отрицательных степеней любого натурального числа $P>1$.

Исходя из изложенных выше фактов, можно сформулировать следующее утверждение: в P-ичной системе любое неотрицательное действительное число можно записать в виде:

$$a = a_nP^{n} + a_{n-1}P^{n-1} + \ldots + a_1P + a_0 + a_{-1}P^{-1} + a_{-2}P^{-2} + \ldots =$$
$$= \sum_{i=-\infty}^{n} a_iP^{i},\quad 0\le a_i<P,\quad n\ge 0 \qquad (1.5)$$

— где $P>1$ является ***основанием*** позиционной системы счисления, а a_i — ***цифрами*** числа a в P-ичной системе счисления.

Величина целого неотрицательного числа n зависит от количества значащих цифр в целой части числа a. Здесь $a_n>0$ — первая (старшая) ***значащая*** цифра (в этом случае если среди цифр $a_0, \ldots, a_{n-1}$ есть нулевые, то они также являются значащими). Если разложение дробной части числа a по отрицательным степеням P является конечным, то цифры a_i при $i=-\infty, \ldots, k-1$ ($k\ge 0$) равны нулю и являются ***незначащими***, а $a_{-k}\ne 0$ — младшая ***значащая*** цифра.

В этом случае если $k>1$ и среди цифр $a_{-1}, \ldots, a_{-k+1}$ есть нулевые, то они также являются значащими, а если число a целое, т.е. $k=0$, то младшей значащей цифрой является a_0, даже если $a_0=0$. Исключением является число ноль, у которого в любой P-ичной системе счисления в представлении (1.5) $a_0=0$ — единственная значащая цифра и $n=k=0$.

Итак, системой счисления с минимальным алфавитом является двоичная система, все числа в которой записываются с помощью 0 и 1. Эта система получила наибольшее распространение при представлении чисел в компьютере из-за простоты выполнения в ней различных арифметических действий и надежности получения двух устойчивых физических состояний.

Если основание системы счисления P меньше десяти, то для символьного представления цифр в ней естественно использовать первые P десятичных цифр (от 0 до $P-1$). Например, в пятеричной системе счисления будут использоваться пять цифр: 0, 1, 2, 3, 4.

Для $10<P<37$ в качестве первых десяти цифр также обычно используют их десятичное представление, а в качестве остальных цифр служат буквы латинского алфавита. Из класса P-ичных систем в вычислительной технике наибольшее применение нашла шестнадцатеричная система, алфавит которой составляют цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Для систем счисления с основаниями, большими 36, единых правил для формы записи цифр не существует.

Для представления информации в бинарных файлах персональных компьютеров фактически используется 256-ричная система счисления, когда содержимое каждого байта представляется всего лишь одной цифрой.

Цифра эта изображается с помощью символа из кодовой таблицы, текущее состояние которой, вообще говоря, может быть различное (так, русские буквы в кодовой таблице могут как присутствовать, так и отсутствовать).

В этой своеобразной системе счисления символ “0” будет обозначать сорок девятую, а не первую цифру, так как код такого символа обычно равен 48, а символ “A” (первая заглавная латинская буква) с кодом 65 описывает 66-ю по счету цифру в 256-ричной системе счисления. Число ноль, т.е. первая цифра в алфавите системы, тогда будет записываться с помощью символа с нулевым кодом.

В дальнейшем если при описании произвольной P-ичной системы счисления вид ее цифр указан не будет, то мы будем считать, что первые десять цифр совпадают с десятичными, а следующие 26 — с латинскими буквами. Остальные цифры будем записывать в виде соответствующего числа в десятичной системе, заключенного в квадратные скобки. Так, [50] в системах счисления с основанием больше 50 будет обозначать 51-ю по счету от нуля цифру. Аналогично для записи максимальной цифры в произвольной P-ичной системе счисления мы можем использовать обозначение $[P-1]$.

1.4. Единственность представления чисел в позиционных системах

Целью данного параграфа является доказательство того, что в любой позиционной системе можно записать любое натуральное число и притом единственным образом. Этот факт мы будем доказывать только для P-ичных систем. Доказательство этого факта для нетрадиционных систем счисления приведено в уже упоминавшейся книге В.Н. Касаткина “Новое о системах счисления”.

Существует доказательство того, что любое действительное число можно представить в виде суммы целых степеней (неотрицательных для целой части и отрицательных для дробной) произвольного натурального числа $P>1$.

Если к тому же потребовать, чтобы количество одинаковых степеней P (т.е. коэффициент при соответствующей степени) не превосходило $P-1$, то для натуральных чисел такое представление окажется еще и единственным.

Докажем сформулированное выше утверждение только для натуральных чисел как теорему существования и единственности представления натурального числа в виде суммы степеней натурального числа P.

Теорема 1. Пусть P — произвольное натуральное число, большее единицы. Существует, и притом единственное, представление любого натурального числа X в виде

$$X = a_n \cdot P^n + a_{n-1} \cdot P^{n-1} + \ldots + a_1 \cdot P + a_0, \qquad (1.1)$$

где $0 \le a_i < P$, $0 \le i \le n$, $a_n \ne 0$.

Доказательство

Существование. Доказательство основано на методе построения, т.е. для произвольного натурального числа мы просто построим представление вида (1.1).

Так как числа P^0, P^1, P^2, P^3, ... образуют монотонно возрастающую числовую последовательность, то существует такое натуральное число n, что

$$P^n \le X < P^{n+1} \qquad (1.2)$$

Разделим интервал $[P^n; P^{n+1}]$ на $P-1$ равную часть. Тогда границами полученных интервалов окажутся числа $x_1 = 1 \cdot P^n$, $x_2 = 2 \cdot P^n$, ..., $x_p = P \cdot P^n = P^{n+1}$. Длина каждого нового промежутка $[x_k; x_{k+1}]$, где $k = 1, \ldots, P-1$, равна P^n.

Из (1.2) и проведенного построения следует, что число X попадет в один из интервалов $[x_k, x_{k+1}]$, то есть существует такое натуральное $1 \le k < P$, что

$$k \cdot P^n \le X < (k+1) \cdot P^n \qquad (1.3a)$$

Положим $a_n = k < P$. Очевидно, что $a_n \ne 0$.

Обозначим разницу между числом X и левой границей x_k соответствующего интервала как $Y = X - a_n \cdot P^n$. При этом $0 \le Y < P^n$ по построению.

Если $Y = 0$, то построение закончено, в противном случае опять, сравнивая уже величину Y с членами возрастающей последовательности 1, P, P^2, P^3, ..., P^n, найдем целое число m такое, что $P^m \le Y < P^{m+1}$, $0 \le m < n$.

Для номеров $m+1 \le i \le n-1$ положим $a_i = 0$ (таких номеров может и не оказаться, если $m+1 = n$) и вычислим a_m. Для этого, как и ранее, разделим интервал $[P^m; P^{m+1}]$ на $P-1$ равные части длиной P^m и определим, в какую из них попадает число Y. Т.е. среди целых значений b: $0 \le b < P$ найдем такое, что

$$b \cdot P^m \le Y < (b+1) \cdot P^m \qquad (1.3b)$$

Положим $a_m = b < P$ и обозначим $Z = Y - a_m \cdot P_m$. И так далее.

Процесс обязательно завершится, т.к. на каждом шаге мы сравниваем оставшееся число все с меньшим количеством различных неотрицательных степеней числа P (иными словами, $m < n$ и так далее). А как только результат очередного вычитания окажется меньше, чем P, мы положим a_0 равным ему, и построение закончится.

“...Письменность, это величайшее изобретение руки и ума человека, постепенно возникла из счета”.

(Дж. Бернал, английский ученый и философ)

ЧАСТЬ II. Системы счисления

Глава 1
Позиционные системы счисления

История систем счисления восходит к тому далекому прошлому, когда человек, записывая числа, пользовался насечками на палке или ссыпал камешки в мешочек. Эта система записи назвается ***единичной, “палочной”,*** или ***унарной***.

Совершенствуя искусство счета, человечество проделало огромный путь — от засечек на дереве до современного компьютера. Но все достижения в области вычислений базируются именно на единичной системе. Это и формирование абстрактного, оторванного от объекта понятия числительного, и изобретение соответствующих символов, и фактически первые математические модели.

Имеются достаточно обоснованные предположения о том, что сначала человек изобрел числа, а лишь затем — другие письменные знаки. Эволюция именно единичной системы постепенно привела к идее пересчитывания группами, к возникновению цифр и чисел, а в конечном итоге — к современным позиционным системам счисления.

Используется ли унарная система в наше время?

Ответ для многих может оказаться неожиданным: да, используется. Малыши используют при счете пальцы рук, первоклассники осваивают арифметические операции при помощи счетных палочек. Кроме того, с числами, записанными в единичной системе счисления, приходится иметь дело в теории алгоритмов — при изучении и конструировании алгоритмов в ***системе Поста (машина Поста)***.

Наиболее совершенными системами счисления являются позиционные системы. В первой части книги “Представление информации. Базовый курс” можно прочитать об истории систем счисления, о самых разнообразных видах непозиционных систем. В этой же части мы будем рассматривать только позиционные системы счисления.

В этой главе подробно рассматриваются принципы построения позиционных систем, правило перечисления чисел в P-ичных системах счисления. Доказывается теорема о единственности представления любого числа в произвольной позиционной системе счисления в виде степенного ряда. Приводится современная классификация позиционных систем счисления.

1.1. Некоторые определения из теории чисел

Приведем некоторые общематематические определения, которые мы будем использовать при дальнейшем изложении материала.

Определение. Числа 1, 2, 3, 4, 5, ..., использующиеся для счета предметов или для указания порядкового номера того или иного предмета среди однородных, называются ***натуральными***.

Определение. ***Обыкновенной (рациональной) дробью*** называется число вида $\frac{m}{n}$, где m — целое, а n — натуральное число. Число m называется числителем дроби, n — ее знаменателем.

Обыкновенные дроби делятся на правильные и неправильные.

Определение. Обыкновенная дробь $\frac{m}{n}$ называется ***правильной***, если ее числитель меньше знаменателя, и ***неправильной***, если ее числитель больше знаменателя или равен ему.

Например, следующие дроби являются правильными: $\frac{3}{7}, \frac{17}{34}, \frac{100}{101}$.

Дроби $\frac{3}{3}, \frac{7}{4}, \frac{203}{51}$ являются примером неправильных дробей.

Определение. Числа, которые можно представить в виде обыкновенной дроби, называются ***рациональными***, а остальные действительные числа называются ***иррациональными***.

Например, числа π, $\sqrt{2}$, $\sqrt{5}$ являются иррациональными.

Произвольное положительное действительное число можно записать в следующем виде:

$$a = N,n$$

— где N — это ***целая часть числа***, а запятой от нее отделена его ***дробная часть***. N и n — это ряд цифр, который в случае n может быть бесконечным.

Обычно n записывают в виде $n_1\, n_2 \ldots\, n_k \ldots$, где n_i — цифры конкретной системы счисления.

Ниже мы рассмотрим, что обозначает такая форма записи в различных системах счисления. В десятичной

системе соответствующее число называют ***десятичной дробью***, в системе счисления с основанием P — ***P-ичной дробью***, но если система счисления не указана, то мы просто будем называть такое число дробью (в отличие от обыкновенной или рациональной дроби).

Заметим, что во многих странах в этой форме записи чисел целая часть от дробной отделяется не запятой, а *точкой*, что порой и определяет для калькуляторов и компьютеров формат ввода и вывода числовой информации.

Определение. Если в дроби после запятой содержится бесконечное количество цифр, то эта дробь называется ***бесконечной***, если количество цифр после запятой в дроби конечно, то дробь называется ***конечной***.

Примеры конечных дробей: 3,14; 4,12120089; 3456543,00000001.

Примерами бесконечных дробей являются числа $\frac{1}{3}$=0,3333... или π=3,14159265358979323846264 3... (отношение длины окружности к ее диаметру).

Определение. Бесконечная дробь называется ***периодической***, если в ее записи после запятой с некоторого места начинает циклически бесконечно повторяться одна и та же группа цифр. Минимальная последовательно повторяющаяся группа цифр в записи дроби после запятой называется ее ***периодом***.

Для краткости записи период принято записывать один раз, заключая его в круглые скобки.

Например:

0,3333333333…=0,(3);

0,2341783417834178...=0,2(34178).

Более строго вышеприведенное определение можно записать так:

Определение*. Бесконечная дробь $N, n_1 n_2 ... n_k ...$ называется ***периодической***, если существуют такие натуральные числа p, q, что $n_{k+p}=n_k$ для всех $k>q$. Совокупность цифр $n_{q+1} n_{q+2} ... n_{q+p}$ называется периодом дроби, а число p — длиной периода.

Конечные и периодические дроби представляют собой ***рациональные*** числа, а действительные числа, запись которых является бесконечной непериодической дробью, — ***иррациональные***. То есть при делении одного натурального числа на другое мы можем получить либо конечную, либо периодическую дробь. Данный факт доказывается в курсе математики.

Идея довольно проста: при стандартной процедуре деления на любое натуральное число N мы имеем всего N различных остатков от деления, включая нулевой. А цифры десятичной дроби как раз и получаются в процессе деления числителя правильной обыкновенной дроби на знаменатель. Если остаток окажется равен нулю, то дробь будет конечной, в противном случае по крайней мере через $N-1$ шаг остатки повторятся и дробь будет периодической.

Определение. Последовательность с первым членом b_1, каждый член которой начиная со второго равен предыдущему, умноженному на одно и то же число q, называется ***геометрической прогрессией со знаменателем*** q.

Сумма первых n членов геометрической прогрессии равна, как известно,

$$S_n = \frac{b_1(1-q^n)}{1-q}.$$

Определение. Произведение натуральных чисел $1 \cdot 2 \cdot 3 \cdot ... \cdot n$ называется ***факториалом*** числа n и обозначается $n!$ Считается также, что $0!=1!=1$.

Определение. Числа, образующие последовательность по следующему закону: $f_0=f_1=1$; $f_i=f_{i-1}+f_{i-2}$, $i=2, 3, ...$ — называются **числами Фибоначчи.**

1.2. Базис систем счисления. Принцип позиционности

Определение. ***Системой счисления*** называется способ записи (нумерации) чисел. Символы, при помощи которых записывается число, называются ***цифрами***.

Надо различать понятия вид цифры и значение цифры. Например, в римской системе счисления числа записываются при помощи цифр I, V, X, L, C, D, M, которые имеют следующие значения: I — 1, V — 5, X — 10, L — 50, C — 100, D — 500 и M — 1000.

Определение. Системы счисления, в которых вклад каждой цифры в величину числа зависит от ее положения (позиции) в последовательности цифр, изображающей число, называются ***позиционными***.

Примером позиционной системы является наиболее привычная нам десятичная система счисления. Классификация позиционных систем счисления была проведена в теме 3 первой части.

Определение. Системы счисления, в которых каждой цифре соответствует величина, не зависящая от местонахождения этой цифры в записи числа, называются ***непозиционными***.

Примером непозиционной системы счисления является римская система.

Определение. Совокупность различных цифр, используемых в системе счисления для записи чисел, называется ***алфавитом*** системы счисления.

При рассмотрении позиционных систем счисления чрезвычайно важно понятие ***базиса*** системы счисления.

Определение. ***Базис*** позиционной системы счисления — это последовательность чисел, каждое из которых задает значение цифры по ее месту в записи числа, то есть "вес" каждого разряда.

Пример 1.1. Выпишем базисы некоторых систем счисления.

Базис десятичной системы счисления: 1, 10, 10^2, 10^3, 10^4, ..., 10^n, ...

Базис двоичной системы счисления: 1, 2, 2^2, 2^3, 2^4, ..., 2^n, ...

Базис восьмеричной системы счисления: 1, 8, 8^2, 8^3, 8^4, ..., 8^n, ...

Базисы приведенных систем счисления образуют геометрические прогрессии со знаменателями 10, 2 и 8 соответственно.

Договоримся называть такие системы счисления ***традиционными.*** В более общем виде для традиционных позиционных систем счисления базис можно записать в виде последовательных членов геометрической прогрессии

1, P, P^2, P^3, ..., P^n, ...

Определение. Знаменатель P геометрической прогрессии, члены которой образуют базис традиционной системы счисления, называется ***основанием*** системы.

Определение. Традиционные позиционные системы счисления с основанием P будем называть ***P-ичными***.

Зная базис P-ичной системы счисления, мы можем сказать, сколько "весит" единица каждого разряда в позиционной системе.

Наряду с широко известными (*традиционными*) системами счисления, базис которых образуют члены геометрических прогрессий, а значения цифр есть натуральные числа, существуют системы счисления, базисы которых построены на иных принципах. Такие системы счисления будем называть ***нетрадиционными***.

Нетрадиционными являются такие позиционные системы, как факториальная, фибоначчиева, уравновешенная, и другие.

Пример 1.2. Выпишем базисы некоторых нетрадиционных систем счисления.

Факториальная система: 1!, 2!, 3!, 4!, ..., $(n-1)!$, $n!$, ...

Фибоначчиева система: 1, 2, 3, 5, 8, 13, 21, ...

Выделение последовательности чисел в качестве базиса, которые задают вес или значение каждой цифры в записи числа в соответствии с ее местоположением в этой записи, и лежит в основе принципа построения позиционных систем счисления. Этот принцип был назван ***принципом позиционности.***

Среди позиционных систем счисления принято выделять смешанные P-Q-ичные системы счисления. Подробнее о смешанных и нетрадиционных системах счисления можно прочитать в параграфе 1.6 данной главы, в главе 4, а также в книге В.Н. Касаткина "Новое о системах счисления", Киев: Вища школа, 1982.

1.3. Алфавит и основание позиционной системы счисления

Определение. Совокупность различных цифр, используемых в системе счисления для записи чисел, называется ***алфавитом*** системы счисления. Количество этих цифр в P-ичных системах (размерность алфавита) равно ***основанию*** системы счисления.

Относительно приведенного определения сделаем следующее замечание. Обозначим числа, составляющие базис позиционной системы, через P_0, P_1, P_2,... Тогда легко доказать, что для количества цифр N_{d_k}, употребимых в k-м разряде числа, верно соотношение $N_{d_k} \le \frac{P_{k+1}}{P_k}$. Для P-ичных систем отношение $\frac{P_{k+1}}{P_k}$ постоянно для всех k и равно основанию P (именно это и записано в определении). Для нетрадиционных систем относительно размерности алфавита в общем случае сказать ничего нельзя. Это замечание о размерности алфавита будем называть ***следствием из принципа позиционности.***

Так, например, алфавит десятичной системы составляют цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Алфавитом же произвольной системы счисления с основанием P служат числа 0, 1, ..., $P-1$, каждое из которых может быть записано с помощью одного уникального (то есть отличного от других) символа, младшей же цифрой всегда является 0. Совокупность таких символов и образует множество цифр (алфавит) в P-ичной системе счисления.

Система счисления	Основание	Количество цифр	Цифры
Двоичная	2	2	0, 1
Троичная	3	3	0, 1, 2
Восьмеричная	8	8	0, 1, 2, 3, 4, 5, 6, 7
Шестнадцатеричная	16	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Основанием P-ичной системы счисления может быть любое натуральное число, большее единицы.

Покажем невозможность построения позиционной системы счисления с алфавитом, состоящим из одной цифры (скажем, цифры 1). Из цифры 1 состоит "палочная" система счисления. Отсутствие цифры 0 в нашей гипотетической позиционной системе счисления с одной цифрой не позволяет записывать в ней произвольные числа. Если же ввести цифру 0 в дополнение к "палочке", то это приведет к нарушению следствия из принципа позиционности о количестве цифр, используемых в каждом разряде. Действительно, базисом единичной системы в предположении ее позиционности является последовательность 1, 1, 1, 1, ... Отношение $\frac{P_{k+1}}{P_k}$ для любого k равно 1. Следовательно, количество цифр не может быть больше 1. Цифра 0 в одиночку тем более не может основать систему счисления.

Практикум по информатике в среде LogoWriter

Продолжение. Начало на с. 4

§ 34. Опыты с моделью свободного падения тела

При проведении опытов было бы полезно иметь возможность измерять время. Принятая нами условная единица времени — промежуток от одного вызова рекурсивной процедуры до другого. Значит, если ввести в программу счетчик вызовов, то он будет играть роль таймера. Перед самым первым вызовом необходимо установить начальное значение счетчика, равное нулю (так сбрасывают хронометр на соревнованиях в момент старта):

```
MAKE  "счетчик  0
```

Затем в процедуре после продвижения Черепашки значение счетчика должно увеличиваться на единицу:

```
MAKE  "счетчик  :счетчик  +  1
```

Наконец, чтобы воспользоваться показаниями таймера, надо иметь возможность прервать опыт в какой-то заранее заданный момент. Для этого можно в процедуру ввести еще одно условие остановки, например:

```
IF  :счетчик  =  100 [STOP]
```

Тогда движение остановится либо когда шарик достигнет границы полигона, либо на сто первом вызове процедуры, т.е. через сто единиц времени.

Задания

34.1. Введите в процедуру, которая моделирует равноускоренное движение, таймер и условие остановки по заданному значению таймера.

34.2. Напишите процедуру для установки шарика на "потолок" — на уровне последнего деления мерной линейки.

34.3. Проведите серию опытов, моделирующих свободное падение.

Начальное положение шарика задается процедурой из предыдущего задания, начальное значение таймера устанавливается равным нулю, начальное значение скорости тоже равно нулю. Изменяться в опытах может ускорение, а также время (в условии остановки по таймеру). Измеряемая величина — путь, который успеет пролететь шарик.

34.4. Сформулируйте (на основании опытных данных), как зависит путь, пройденный телом, от времени (при неизменном ускорении).

34.5. Сформулируйте, как зависит путь, пройденный телом, от ускорения (если на опыты отводится одно и то же время).

34.6. Пользуясь процедурой, которая ставит на экране точку с заданными координатами, и процедурой рисования осей, постройте "вручную" (без построителя графиков) точечные графики по полученным опытным данным, выбрав подходящий масштаб.

34.7. Подберите согласующуюся с опытными данными формулу зависимости пути, пройденного телом в свободном падении, от времени и ускорения.

§ 35. Сложение равномерных движений по координатным осям

Один из важных принципов механики — принцип суперпозиции. Любое сложное движение тела можно представить как сумму или наложение двух или нескольких простых. И наоборот, движение можно разложить на составляющие и считать, что каждое из них происходит независимо.

Изменим процедуру моделирования равномерного прямолинейного движения так, чтобы Черепашка могла двигаться только по горизонтали. Для такого перемещения есть команда `SETX`.

Черепашка при каждом вызове рекурсивной процедуры должна отодвинуться от своей предыдущей абсциссы на одно и то же число шагов.

Для определения абсциссы имеется датчик `X`. В результате получим следующую процедуру:

```
ЭТО ДВИЖЕНИЕ_4  :Vx
SETX  X + :Vx
IF COLORUNDER = 5 [STOP]
ДВИЖЕНИЕ_4  :Vx
END
```

Точно так же можно написать процедуру равномерного движения Черепашки по вертикали:

```
ЭТО ДВИЖЕНИЕ_5  :Vy
SETY  Y + :Vy
IF COLORUNDER = 5 [STOP]
ДВИЖЕНИЕ_5  :Vy
END
```

А теперь попробуем согласно принципу суперпозиции сложить эти движения. (Например, катер пересекает реку. Он двигается с некоторой скоростью поперек течения и в то же время перемещается вместе с речной водой по течению.)

```
ЭТО ДВИЖЕНИЕ_6  :Vx  :Vy
SETX  X + :Vx
SETY  Y + :Vy
IF COLORUNDER  = 5 [STOP]
ДВИЖЕНИЕ_6  :Vx  :Vy
END
```

У процедуры уже два параметра: горизонтальная и вертикальная скорости. Ведь Черепашка одновременно (и независимо) участвует в двух равномерных движениях по взаимно перпендикулярным направлениям.

Задания

35.1. Напишите процедуру, моделирующую сложение двух равномерных прямолинейных движений по осям. Подберите значения скоростей, при которых Черепашка попадает из нижнего левого угла в правый верхний. Является ли эта пара чисел единственной?

35.2. Подберите значения скоростей для перемещения Черепашки из каждого угла в противоположный.

35.3. Добавьте в сложное движение еще одну составляющую — равномерное вращение Черепашки вокруг своей оси. Изменятся ли при этом результаты предыдущих опытов?

§ 36. Сложение равнозамедленного движения по вертикали и равномерного по горизонтали

Равномерное движение Черепашки по вертикали можно, как это сделано в § 33, трансформировать в равноускоренное (или равнозамедленное).

```
ЭТО ДВИЖЕНИЕ_7  :A  :Vy
SETY  Y + :Vy
IF COLORUNDER = 5 [STOP]
ДВИЖЕНИЕ_7  :A  :Vy + :A
END
```

Черепашка будет двигаться точно так же, как в процедуре `ДВИЖЕНИЕ_2`.

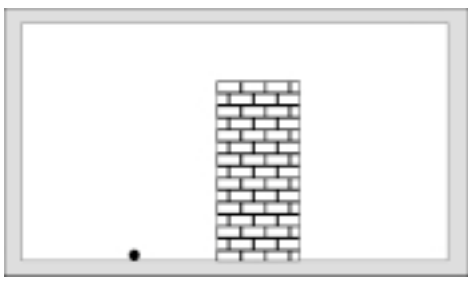
Но если в эту процедуру добавить еще и равномерное движение по горизонтали, то поведение Черепашки изменится.

Задания

36.1. Напишите процедуру с тремя параметрами для моделирования сложения равнозамедленного движения по вертикали и равномерного по горизонтали.
Параметры: горизонтальная скорость, вертикальная начальная скорость и ускорение по вертикали. Ускорение должно быть отрицательным.

36.2. Подберите такие значения параметров, чтобы Черепашка перелетала из левого нижнего угла "полигона" в правый нижний и обратно.

36.3. Напишите процедуру для рисования "кирпичной стенки" такого размера, чтобы оставалось небольшое пространство до верхней границы "полигона" (см. рисунок). Подберите при нескольких заданных значениях ускорения такие вертикальную и горизонтальную скорости, чтобы Черепашка перелетала через стенку.



Для заполнения области внутри замкнутого контура изображениями форм, например, фрагментами кирпичной кладки, служит команда `SHADE`. Правила ее применения такие же, как и для команды `FILL`.

§ 37. Упругий удар. Отражение мячика от стен

Когда мяч подлетает к стене под некоторым углом, можно считать, что он независимо участвует в двух движениях: параллельном стене и перпендикулярном стене. При ударе о стену скорость первого из этих движений остается неизменной, а скорость второго (перпендикулярного) меняется на противоположную (если считать удар идеальным, абсолютно упругим). Значит, при ударе Черепашки о левую или правую границы "полигона" должна быть выполнена команда изменения горизонтальной составляющей скорости:

```
MAKE "Vx   -1 * :Vx
```

А при ударе о нижнюю или верхнюю границы должно измениться на противоположное значение вертикальной составляющей скорости:

```
MAKE "Vy   -1 * :Vy
```

Чтобы получить движение шарика с отражением от границ "полигона", нужно описать условие достижения Черепашкой вертикальных и горизонтальных границ.

Пусть, например, абсциссы вертикальных границ равны 260 и —260, а горизонтальных — 160 и —160. Тогда условие достижения вертикальной границы

```
OR X > 260 X < -260
```

`OR` (или) — логическая связка для двух простых условий.

А условие для горизонтальной границы

```
OR Y > 160 Y < -160
```

Задания

37.1. Напишите процедуру для моделирования движения мячика с отражением от стен при условии, что и горизонтальное, и вертикальное движения — равномерные.

Окончание на с. 14

Практикум по информатике в среде LogoWriter

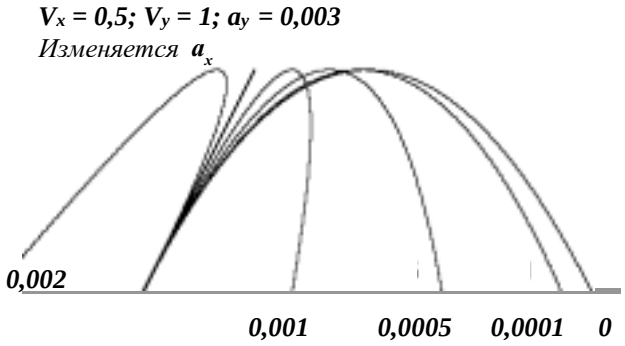
Окончание. См. с. 4, 13

Методические рекомендации и тексты программ к заданиям

Как показывают тесты, представления большинства людей о движении не соответствуют законам ньютоновской механики, а основываются на некоторой интуитивной теории.

Студентам университетов предлагались вопросы типа: как будет двигаться камень, свалившийся с плеча быстро идущего человека; по какой траектории будет двигаться раскручиваемое по окружности тело, если его отпустить, и т.п. Более 50 процентов ответов не соответствовало ньютоновской механике: многие полагали, что камень будет падать вертикально и стукнется о землю в той точке, над которой его отпустили; некоторые были даже убеждены, что падающий камень завернет назад и коснется земли, пройдя эту точку. Траекторию оторвавшегося от веревки мяча рисовали искривленной, считая, что некоторое время он будет продолжать двигаться по кривой. Очевидно, интуитивные представления о движении оказывают влияние не только на решение подобных задач, но и на выполнение действий в реальных ситуациях.

Как развеять неверные представления о движении? Казалось бы, ответ прост — путем обучения механике Ньютона. Однако интуитивные представления изменить трудно: школьники и студенты неправильно интерпретируют материал или "подгоняют" его под закрепившиеся внутренние схемы, которые сформировались в ежедневной практической деятельности. Самый прямой путь — опыты, лабораторные работы, в которых должны скорректироваться внутренние, интуитивные схемы. Однако в сегодняшних условиях, к сожалению, возможность такой деятельности у школьников почти отсутствует.

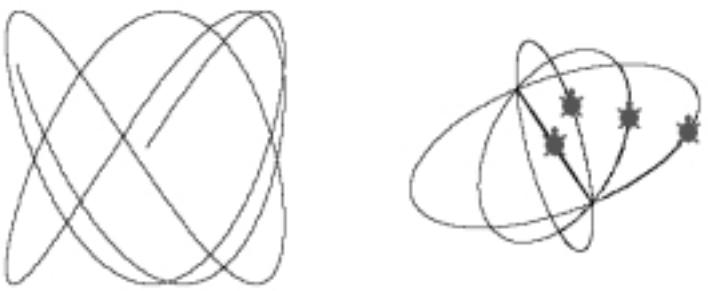


Работая над заданиями этой главы, школьники самостоятельно подготовят и проведут ряд компьютерных экспериментов и небольших учебных исследований. Причем предлагаемый способ моделирования отличается математической простотой. При создании модели не используется "готовое" уравнение движения (скажем, $H = gt^2/2$). Модели скорее физические, чем математические; необходимо описать физический закон и задать начальные условия.

Таким образом, появляется возможность моделировать и исследовать такие виды движения, аналитические уравнения которых слишком сложны и отсутствуют в школьной программе. Например, суперпозицию равнозамедленных движений по осям.

Дальнейшее развитие материала этой главы — моделирование и изучение колебаний, как простых, так и сложных. Например, сложение колебаний по осям.

В моделях можно учесть сопротивление среды, потерю энергии при ударе. Все это может послужить в качестве тем для индивидуальных проектов или факультативных занятий.



Работа над оформлением индивидуальных проектов (создание и форматирование текста, подготовка и вставка в текст иллюстраций, формул, таблиц; создание презентации или web-странички) — хорошая основа для интеграции разных направлений школьной информатики: "пользовательского" и "программистского".

§ 32. Прямолинейное равномерное движение

32.1
```
это прям :a :b
repeat 2 [fd :a rt 90 fd :b rt 90]
end

это поле
rg pu setpos [-290 -160] pd
прям 320 580
     pu lt 90 fd 10 pd setc 5 fill bk 10 rt 90 setc 1
      end
```

33.2
```
это старт
pu seth 0 setpos [-270 -160] setc 14 setsh 31
end
```

§ 33. Прямолинейное равноускоренное движение

33.3
```
это разм
pu setpos [-290 -160] pd
цикл  10
pu setpos [-290 -160] pd
цикл  100
end

это цикл :шаг
if y > 130 [stop]
sety y + :шаг
setx x + 3 + :шаг / 10 setx x - 3 - :шаг / 10
цикл :шаг
end
```

§ 34. Опыты с моделью свободного падения тела

34.1
```
это движение2 :a :v :T :предел
fd :v
if colorunder = 5 [stop]
if  :T = :предел [stop]
движение2 :a    :v + :a  :T + 1  :предел
end
```

§ 35. Сложение равномерных движений по координатным осям

35.3
```
это движение6  :Vx :Vy
setx x + :Vx
sety y + :Vy
rt 5
if colorunder = 5 [stop]
движение7  :Vx :Vy
end
```

§ 36. Сложение равнозамедленного движения по вертикали и равномерного по горизонтали

36.1
```
это движение8  :Vx :Vy :a
setx x + :Vx
sety y + :Vy
if colorunder = 5 [stop]
движение8  :Vx  :Vy + :a   :a
end
```

36.3
```
это прям :a :b
repeat 2 [fd :a rt 90 fd :b rt 90]
end

это стена
pu setx 50 sety -160 pd
прям 260 50
rt 45 pu fd 10 pd
setc 4 fill setc 1 setsh 29 shade
setc 14 setsh 0
end
```

§ 37. Упругий удар. Отражение мячика от стен

37.1
```
это движение9  :Vx :Vy
setx x + :Vx
sety y + :Vy
rt 5
if or x > 290 x < -290 [make "Vx 0 - :Vx]
if or y > 160 y < -160 [make "Vy 0 - :Vy]
движение9  :Vx :Vy
end
```

37.2
```
это движение10  :Vx :Vy :a
setx x + :Vx
sety y + :Vy
if or x > 290 x < -290 [make "Vx 0 - :Vx]
if or y > 160 y < -160 [make "Vy 0 - :Vy]
движение10  :Vx  :Vy + :a   :a
end
```

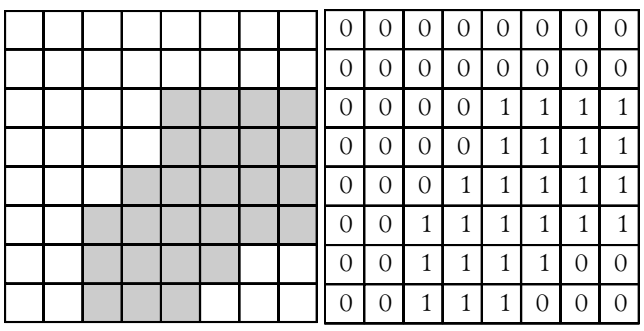
Черно-белые деревья

С.М. ОКУЛОВ

В компьютерной графике для обработки черно-белых изображений часто используется преобразование изображения в строку из десятичных чисел.

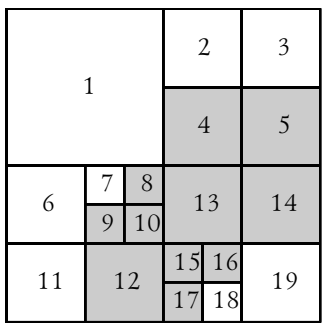
Предположим для простоты, что изображение имеет форму квадрата. Изображение или его часть (квадрат меньшего размера) последовательно делится на 4 равных квадрата. Если все точки в квадрате одного цвета (черного или белого), то процесс деления этой части изображения заканчивается. Квадраты, содержащие как черные, так и белые точки, опять делятся. Процесс продолжается до тех пор, пока каждый из квадратов не будет содержать точки только одного цвета.

Например, если использовать 0 для обозначения точек белого цвета и 1 для обозначения точек черного цвета, то изображение на *рис.* 1*а* будет записано следующей матрицей из нулей и единиц (*рис.* 1*б*). Матрица будет разделена на квадраты, как показано на *рис.* 1*в*. Серым выделены квадраты, содержащие только черные точки.



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	0	0
0	0	1	1	1	0	0	0

а *б*



в

***Рис.* 1**

Преобразование изображения в дерево выполняется следующим образом. Корень дерева представляет все изображение. Каждая вершина дерева описывает некий квадрат. Она может иметь 4 исходящих ребра к другим вершинам, представляющим квадраты, на которые делится исходный. Вершины без исходящих ребер соответствуют квадратам, состоящим из точек одного цвета. Например, изображению на *рис.* 1*а* с его разбивкой на квадраты соответствует дерево (*рис.* 2). Серые вершины обозначают квадраты, содержащие 2 цвета, а белые и черные — квадраты, состоящие из точек одного цвета (такого же, как цвет квадрата).

В дереве вершины пронумерованы в соответствии с нумерацией квадратов на рисунке. Квадраты обходятся слева направо и сверху вниз, начиная с верхнего левого угла (верхний левый, верхний правый, нижний левый, нижний правый). Каждая черная вершина дерева кодируется следующим образом: поднимаемся от вершины по ребрам к корню, записывая при этом подряд цифры от 1 до 4 в соответствии с тем, где находится текущий квадрат по отношению к вышестоящему (верхний левый — 1, верхний правый — 2, нижний левый — 3, нижний правый — 4). Получается число, записанное в системе счисления по основанию 5. Переводим его в десятичную систему счисления.

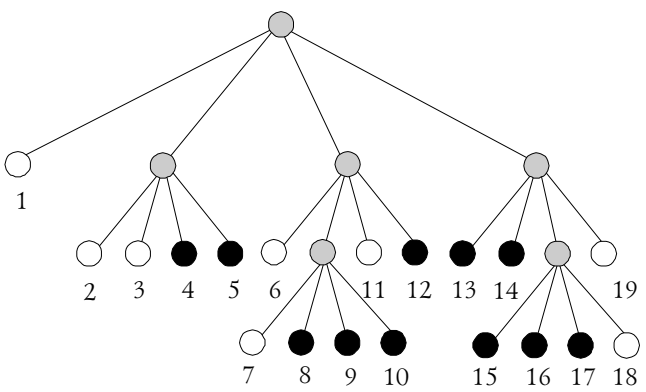


***Рис.* 2**

Например, вершина с номером 4 имеет путь Н—Л, В—П. Получается число 32_5 (по основанию 5), или 17_{10} (по основанию 10). Вершина с номером 12 имеет путь Н—П, Н—Л, или 43_5 и 23_{10} соответственно. Вершина с номером 15 имеет путь В—Л, Н—Л, Н—П, или $134_5=44_{10}$. Все дерево (и, соответственно, изображение) кодируется строкой чисел: 9 14 17 22 23 44 63 69 88 94 113.

Требуется написать программу перевода изображения в строку чисел и обратного преобразования — строки чисел в изображение.

Входные данные

Файл содержит описание одного или нескольких изображений. Все изображения — это квадратные рисунки, длины сторон квадратов — целые числа, являющиеся степенями двойки. Входной файл начинается с целого числа n, где $|n|$ — длина стороны квадрата ($|n| < 64$). Если число n больше 0, то затем следует $|n|$ строк по $|n|$ знаков в строке, заполненных 0 и 1. При этом 1 соответствует черному цвету. Если n меньше 0, то затем следует описание изображения в виде строки из десятичных чисел, оканчивающейся —1. Полностью черному квадрату соответствует строка из одного 0. Белый квадрат кодируется пустой строкой (ничего не вводится). Признаком конца входного файла является значение n, равное 0.

Выходные данные

Для каждого изображения из входного файла выводится его номер. В том случае, когда изображение задается с помощью 0 и 1, в выходной файл записывается его представление в виде строки десятичных чисел. Числа в строке сортируются в порядке возрастания. Для изображений, содержащих больше 12 черных областей, после каждых 12 чисел вывод начинается с новой строки. Количество черных областей выводится после строки из десятичных чисел. В том случае, когда изображение задается строкой из десятичных чисел, в выходной файл записывается его представление в виде квадрата, в котором символ '.' соответствует 0, а символ '*' — 1. Пример входного и выходного файлов приведен в таблице.

Входной файл	Выходной файл
8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0	Изображение 1 9 14 17 22 23 44 63 69 88 94 113 Общее число черных областей 11
−8 9 14 17 22 23 44 63 69 88 94 113 −1	Изображение 2 . . . . * * * * . . . . * * * * . . . * * * * * . . * * * * * * . . * * * * . . . . * * * . . .
2 0 0 0 0	Изображение 3 Общее число черных областей 0
−4 0 −1	Изображение 4 * * * * * * * * * * * * * * * *

Решение

Основная программа практически не требует пояснений.

```
Begin
 Assign(Input,'Input.txt');Reset(Input);
 Assign(Output,'Output.txt');Rewrite(Output);
 Readln(N);{вводим размерность изображения}
 NumTest:=0;{переменная для хранения номера
            теста}
 While N<>0 do begin
   Inc(NumTest);
   Writeln('Изображение',NumTest);
   If N>0 Then SolveA else SolveB;
   {первая процедура преобразует изображение
    в виде матрицы в строку десятичных
    чисел, вторая процедура выполняет
    обратное преобразование}
 end;
 Close(Input);
 Close(Output);
End.
```

Дальнейшее уточнение требует осознания того, каким образом в нашей задаче описывается очередной квадрат. Для описания квадрата надо знать координаты левого верхнего угла, длину стороны и тот путь по изображению, который предшествовал попаданию в этот квадрат. Путь по изображению — строка из цифр от 1 до 4. Назовем функцию обработки квадрата `RecA`:

```
Function RecA(i,j,d:integer;
              var Way:string):boolean.
```

Функция возвращает значение `true`, если квадрат черный. Введем массив для хранения результата — десятичных чисел, описывающих изображение. Назовем его `Cp` (`Cp:Array[1..MaxCn] Of LongInt`, где `MaxCn` — константа, равная 64^2). Счетчиком числа записей в `Cp` является значение переменной `Cnt`. Исходное изображение записано в массиве `A` (`A:Array[1..MaxN,1..MaxN] of boolean`, где `MaxN` — константа, равная 64). Значение элемента массива `A[i,j]`, равное `true`, соответствует черной клетке с координатами `(i,j)`, значение `false` — белой. Приведем процедуру `SolveA`.

```
Procedure SolveA;
 Var i:integer;
 Begin
  ReadA;
  {вводим описание изображения из входного
файла, формируем массив A}
  Cnt:=0;
  If RecA(1,1,N,") Then Begin Cnt:=1;
     Cp[Cnt]:=0; End;
  {если все изображение состоит из точек
   черного цвета, то записываем в первый
   элемент массива Cp ноль и заканчиваем
   обработку этого теста, во всех остальных
   случаях массив Cp сформирован рекурсивной
   функцией RecA}
  Sort;{сортируем массив Cp}
  PrintA;{выводим результат в выходной
          файл OutPut}
 End;
```

Функция `RecA` имеет вид:

```
Function
RecA(i,j,d:integer;Way:string):boolean;
 Var k:Integer; c:boolean;
 Begin
  If d=1 Then c:=A[i,j]
  {дошли до квадрата единичной длины,
   значение функции определяется цветом
   квадрата}
  Else Begin k:= d div 2;
     c:=RecA(i,j,k,'1'+Way) And
        RecA(i,j+k,k,'2'+Way) And
        RecA(i+k,j,d,'3'+Way) And
        RecA(i+k,j+k,d,'4'+Way);
    {значение функции на данном уровне
     рекурсии определяется значениями
     функции на квадратах меньшего размера}
     If c Then Dec(Cnt,4);
     {если все составляющие квадраты черные,
      то и данный квадрат черный}
  End;
  If c Then Begin Inc(Cnt);
  {квадрат черный, преобразуем путь (строку,
   описывающую число в пятеричной системе
   счисления) в десятичное число и записываем
   результат}
  Cp[Cnt]:=<преобразовать строку символов,
           описывающую число в пятеричной
           системе счисления, в число в
           десятичной системе счисления>;
   End;
   RecA:=c;
  End;
```

Осталось уточнить функцию преобразования пути по квадрату в десятичное число. Процедуры `Sort` и `PrintA`, мы вызываем их из процедуры `SolveA`, они "прозрачны".

```
Function Fr5To10(S:String):LongInt;
 Var Res:LongInt;
     i:integer;
 Begin
  Res:=0;
  For i:=1 to Length(S) do
     Res:=Res*5+Ord(S[i])-Ord('0');
  Fr5To10:=Res;
 End;
```

Окончание на с. 16

ЗАДАЧИ

Черно-белые деревья

Окончание. Начало на с. 15

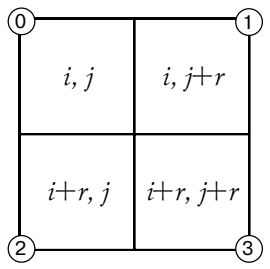
Рассмотрим теперь процедуру получения матрицы, описывающей изображение, из строки десятичных чисел.

```
Procedure SolveB;
 Var i:LongInt;
 Begin
  FillChar(A,SizeOf(A),False);
  {начальное заполнение A, считаем, что весь квадрат белый}
  Read(i);{считываем число}
  While i<>-1 Do Begin
   RecB(1,1,-N,Fr10To5(i));
   {преобразуем число в пятеричную систему счисления,
    находим квадрат, соответствующий данному числу, и
     записываем в элементы A значение true}
   Read(i);{читаем очередное число}
  End;
  PrintB;{вывод элементов матрицы A в выходной файл}
 End;
```

Функция перевода числа из одной системы счисления в другую достаточно прозрачна, но для полноты картины приведем ее.

```
Function Fr10To5(S:LongInt):LongInt;
 Var d, Res:LongInt;{рабочие переменные}
 Begin
  Res:=0;d:=1;
  While S<>0 Do Begin
   Res:=Res+(S Mod 5)*d;{остатки в пятеричной системе счисления}
   d:=d*10;
   S:=S div 5;
  End;
  Fr10To5:=Res;
End;
```

С процедурой RecB чуть сложнее. Квадрат описывается координатами верхнего левого угла и длиной стороны. Кроме того, у нас есть путь, представленный пятеричным числом. Вопрос: **как из очередной цифры этого числа сделать переадресацию по квадрату, т.е. перейти к одному из четырех квадратов меньшего размера**? Ответив на него, мы поймем суть процедуры RecB. Поясним ситуацию рисунком. Пусть координаты левого верхнего угла квадрата задаются значениями переменных i и j, а длина стороны нового квадрата — значение r. Цифры в кружках — это значения переменной k, получаемой из очередной цифры пятеричного числа с помощью операции k:=Way Mod 5-1. В зависимости от значения переменной k с помощью операций i+r*(k Div 2) и j+r*(k Mod 2) мы переходим к левым верхним углам любого из четырех квадратов меньшего размера. Проиллюстрируем это примером.



Номер вызова процедуры RecB	i	j	d	Way	k	r
1	1	1	8	32_5 или 17_{10}	1	4
2	1	5	4	3	2	2
3	3	5	2	0	—	—

Таким образом, черным цветом будет закрашен квадрат, отмеченный цифрой 4 на рисунке в формулировке задачи. Итак, процедура RecB.

```
Procedure RecB(i,j,d:Integer;Way:LongInt);
 Var k,r:Integer;
 Begin
  If Way=0 Then
    For k:=i To i+d-1 Do
      For r:=j To j+d-1 Do A[k,r]:=true
      {квадрат черный, известны его координаты и длина
       стороны, осталось заполнить соответствующую часть
        матрицы A}
  Else Begin
    k:=Way Mod 5-1;
    {определяем номер квадрата меньшего размера,
      0 — левый верхний, 1 — правый верхний,
      2 — левый нижний, 3 — правый нижний}
    r:=d div 2;
    {уменьшаем длину стороны квадрата в два раза}
    RecB(i+r*(k Div 2),j+r*(k Mod 2),r,Way Div 10);
    End;
 End;
```

К XX ГОДОВЩИНЕ ПЕРЕВОРОТА

Окончание. Начало на с. 1

рассчитывая продать не более 100 000 персональных компьютеров IBM PC различных модификаций.

Три кита, на спинах которых персональный компьютер стремительно выплыл на пик мирового прогресса и стал его символом, по очереди празднуют свое двадцатилетие.

В 1978 году был разработан один из самых популярных текстовых редакторов — "WordStar", в 1979-м — первая в мире электронная таблица "VisiCalk", а в следующем году наступит черед базы данных "dBase II". Именно эти программные средства убедили простых тружеников умственного труда не просто в пользе персонального компьютера, но в его необходимости. Из оборонно-научной дорогостоящей диковинки электронно-вычислительная машина превратилась в средство резкого увеличения производительности труда не сотен и даже не тысяч — миллионов человек.

Несомненно, свою роль сыграл и хорошо известный "феномен персонального компьютера", в котором информация начала приобретать человеческие черты цветной графики и звука. Но не будем забывать, что тот же легендарный "Apple-2", ставший хрестоматийным примером дотоле небывалого коммерческого успеха информационных технологий, по справедливости должен разделить этот успех с принципиально новым программным продуктом, существование которого было просто немыслимо вне рамок именно персонального компьютера. Заметим, что этот продукт вовсе не использовал все те новые возможности, которыми персональный компьютер отличался от выпускавшихся параллельно микро-ЭВМ. Ключевой, пожалуй, была лишь цена, делавшая тот же "Apple-2" доступным достаточно большому кругу пользователей.

Как вы, наверное, уже догадались, речь идет об электронной таблице "Визикалк", которую один из крупнейших финансистов США назвал главной осью, вокруг которой вращается весь персональный компьютер. В отличие от текстового редактора и базы данных, "Визикалк" в то время была пакетом, который уже сам по себе оправдывал приобретение достаточно дорогой компьютерной системы.

Идея электронных таблиц до гениальности проста и вполне может относиться к разряду величайших озарений, суть которых доступна любому мало-мальски грамотному человеку.

Она пришла в голову инженеру-программисту Дэниэлу Бриклину, выпускнику МТИ, когда он посещал курсы в Гарвардской школе бизнеса. Именно там он столкнулся с необходимостью утомительного заполнения экономических таблиц, рассчитывая изменение суммы налога на прибыль в зависимости от роста процентной ставки на издержки, увеличения цены на готовую продукцию и т.п. Дэниэлу приходилось пересчитывать уйму чисел в случае изменения всего лишь одной величины. Применение электронного калькулятора, облегчая задачу, вовсе не уменьшало риска мелкой ошибки, грозящей испортить всю работу.

И именно в этот момент Бриклина осенило. А вот было бы здорово, если бы под натужно заполняемой таблицей существовала другая таблица, содержащая уже не данные, а формулы, по которым эти данные нужно получать!

За программное воплощение этой идеи Дэниэла Бриклина взялся еще один программист — Роберт Фрэнкстон, и за полгода-год — с осени 1978-го по весну 1979-го — довел ее до готового программного продукта "Визикалк", означавшего, по замыслу авторов, "Визуальный калькулятор".

С тех пор были написаны сотни программ, основанных на "двухэтажных" расчетных таблицах, Их возможности просто несравнимы с "Визуальным калькулятором". Но никакой 50-мегабайтный Microsoft Excel никогда бы не родился, если бы ему не предшествовала эта первая реализация блестящего озарения Дэниэла Бриклина.

А.И. СЕНОКОСОВ

Гл. редактор
Е.Б.Докшицкая
Зам. гл. редактора
С.Л.Островский
Редакция:
Л.Н.Картвелишвили,
Ю.А.Соколинский,
Н.Л.Беленькая,
Н.П.Медведева
Дизайн и компьютерная верстка:
Н.И.Пронская
Корректоры:
Е.Л.Володина,
С.М.Подберезина

Отпечатано с готовых диапозитивов редакции в ОАО ПО "ПРЕССА-1", 125865, ГСП, Москва, ул. Правды, 24

Тираж 7000 экз.
Заказ №

121165, Москва, Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
для предприятий и организаций 32591
комплекта приложений 32744

Internet: inf@1september.ru
Fidonet: 2:5020/69.32
WWW: http://www.1september.ru