ИНФОРМАЦИОННОЕ ПРОСТРАНСТВО

Романовский Иосиф Владимирович

ШРИФТ БРАЙЛЯ ДЛЯ СЛЕПЫХ С ТОЧКИ ЗРЕНИЯ ПРОГРАММИСТА

Внезапно по непонятной причине в конце 2000-го года я стал интересоваться так называемым кодом Брайля и узнал много нового и про сам код и про его изобретателя Луи Брайля. Мне хотелось рассказать об этом шрифте, но недавно в русском переводе вышла интересная книга [1], в которой много написано о коде Брайля, и я решил, что опоздал. Но потом понял, что эта книга дорогая, не очень распространена, в ней написано не все, что хотелось бы, так что стоит о коде Брайля поговорить еще.

Код Брайля предназначен для слепых, буквы в нем изображаются комбинациями выпуклых точек, которые распознаются на ощупь. Каждая такая точка, точнее, место, где эта точка может располагаться – это своеобразная двоичная единица, и поэтому интересно взглянуть на код Брайля с точки зрения «компьютерщиков».

Сначала самая общая информация. Тексты в коде Брайля пишутся с помощью шаблона – металлической доски с прямоугольными прорезями для знаков (рисунок 1). Знак представляет собой прямоугольную таблицу (матрицу) из трех строк и двух столбцов. В каждой позиции может быть продавлена точка – и на обороте бумаги она будет выпуклой – или не продавлена – место останется гладким. Поэтому пишется текст в одном направлении, а читается в другом, зеркальном. Обычно знаки изображаются так, как они видны читающему (рисунок 2).

На рисунке 2 показан латинский алфавит в исходной кодировке Брайля. Поглядите на верхнюю строчку – первые 10 символов. В них используются только 4 верхних точки. Эта последовательность кодов многократно применяется для кодирования подряд идущего десятка зна-

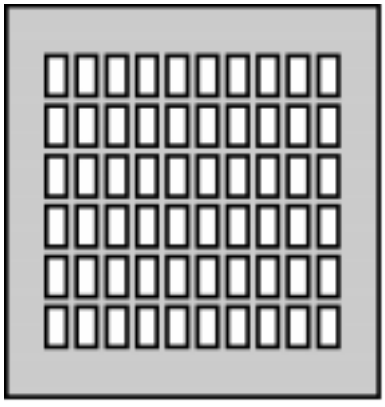


Рисунок 1

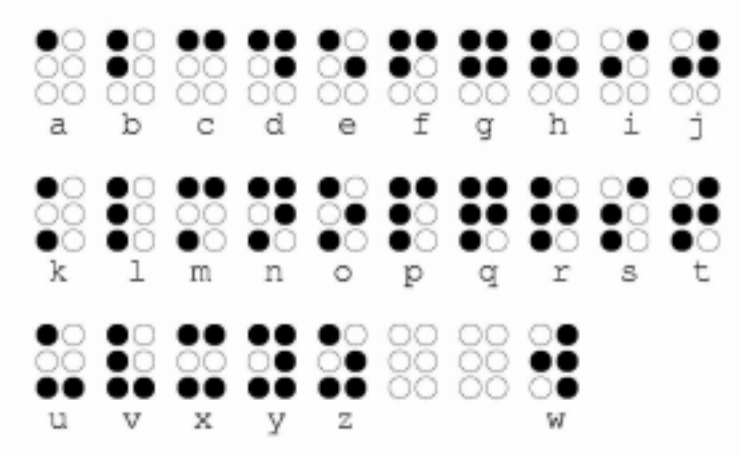


Рисунок 2

ков. Первый пример у нас уже есть. Второй – это идущие дальше десять букв, от K до T. У них только в нижней строчке стоит точка в левой позиции. А почти все остальные буквы имеют точку в нижнем ряду в правой позиции. Как будто код буквы состоит из двух цифр – нижние позиции задают старшую цифру, а верхние четыре – младшую.

Исключение составляет буква W. Почему? Для французского языка эта буква «не родная», она пришла в язык с иностранными словами, и Брайль вначале решил обойтись без нее. Когда ее добавили, место уже было занято, и буква получила код из «четвертого десятка».

Хочется, чтобы вы обратили внимание на выбор десяти рабочих сочетаний для младшей цифры. Всего, как мы знаем, возможных сочетаний 16. *Брайль не использовал тех комбинаций, у которых нет точек в верхней строчке или в левом столбце*, и это очень разумно – их труднее распознавать из-за возможного сдвига позиции пальца.

Давно уже коды Брайля используются по всему миру. Русский вариант кода Брайля был разработан Анной Александровной Адлер в 1889 году.

Теперь для тех, кто читал в этом журнале мою лекцию по ПостСкрипту [2], мы можем сделать маленькое упражнение в этом языке, разработав шрифт Брайля.

Мне нравится способ изображения, использованный в рисунках, – позиции изображаются кружками, точке соответствует черный кружок. Нумеруем, как это принято всюду, позиции сверху вниз, сначала первый столбик, а затем второй. В этой нумерации каждая раскраска задается шестью числами, принимающими значения 1 для точки и 0 для пустого места. Таким образом, буква M будет задаваться строчкой

```
1 0 1 1 0 0
```

Итак, шрифт. Сначала мы приведем полный текст программы, которая делает рисунок 2, а затем объясним его.

Программа:

```
%!   % Первая строка программы

% Описание шрифта

/braille 10 dict def
braille begin
/FontType 3 def  % required elements
/FontMatrix [.01 0 0 .01 0 0] def
/FontBBox [0 0 60 100] def
/Encoding 256 array def
% encoding vector
  0 1 255 {Encoding exch /.notdef
  put} for
  Encoding 32 /space put
% Encoding 36 /dollar put
% Encoding 37 /percent put
% Encoding 42 /asterix put
  Encoding 43 /plus put
  Encoding 44 /comma put
  Encoding 45 /hyphen put
  Encoding 46 /period put
% Encoding 47 /slash put
  Encoding 48 /0 put
  Encoding 49 /1 put
  Encoding 50 /2 put
  Encoding 51 /3 put
  Encoding 52 /4 put
  Encoding 53 /5 put
  Encoding 54 /6 put
  Encoding 55 /7 put
  Encoding 56 /8 put
  Encoding 57 /9 put
  Encoding 58 /colon put
  Encoding 59 /semicolon put
  Encoding 63 /question put
% Encoding 65 /A put
% Encoding 66 /B put
  Encoding 67 /C put  % math comma
% Encoding 68 /D put
% Encoding 69 /E put
  Encoding 70 /F put
  Encoding 71 /G put
  Encoding 72 /H put
% Encoding 73 /I put
% Encoding 74 /J put
% Encoding 75 /K put
% Encoding 76 /L put
% Encoding 77 /M put
% Encoding 78 /N put
% Encoding 79 /O put
% Encoding 80 /P put
  Encoding 81 /Q put
% Encoding 82 /R put
```

```
% Encoding 83 /S put
% Encoding 84 /T put
% Encoding 85 /U put
% Encoding 86 /V put
% Encoding 87 /W put
% Encoding 88 /X put
  Encoding 89 /Y put
  Encoding 90 /Z put
  Encoding 97 /a put % ascii "A"=65
  Encoding 98 /b put
  Encoding 99 /c put
  Encoding 100 /d put
  Encoding 101 /e put
  Encoding 102 /f put
  Encoding 103 /g put
  Encoding 104 /h put
  Encoding 105 /i put
  Encoding 106 /j put
  Encoding 107 /k put
  Encoding 108 /l put
  Encoding 109 /m put
  Encoding 110 /n put
  Encoding 111 /o put
  Encoding 112 /p put
  Encoding 113 /q put
  Encoding 114 /r put
  Encoding 115 /s put
  Encoding 116 /t put
  Encoding 117 /u put
  Encoding 118 /v put
  Encoding 119 /w put
  Encoding 120 /x put
  Encoding 121 /y put
  Encoding 122 /z put

/CharProcs 3 dict def
CharProcs begin
   /.notdef {} def
   /space    {0 0 0 0 0 0 bc } def
   /comma    {0 1 0 0 0 0 bc } def
   /plus     {0 0 1 1 0 1 bc } def
   /hyphen   {0 0 1 0 0 1 bc } def
   /period   {0 1 0 0 1 1 bc } def
   /colon    {0 1 0 0 1 0 bc }def
   /semicolon{0 1 1 0 0 0 bc }def
   /question {0 1 1 0 0 1 bc }def
   /0 {0 0 1 0 1 1 bc } def
   /1 {0 1 0 0 0 0 bc } def
   /2 {0 1 1 0 0 0 bc } def
   /3 {0 1 0 0 1 0 bc } def
   /4 {0 0 0 0 1 0 bc } def
   /5 {0 1 0 0 1 1 bc } def
   /6 {0 1 0 0 0 1 bc } def
   /7 {0 1 1 0 1 0 bc } def
   /8 {0 1 1 0 1 1 bc } def
   /9 {0 1 1 0 0 1 bc } def

   /C {0 0 0 0 0 1 bc } def
      % math comma
   /F {1 0 0 1 1 1 bc } def
      % to open fraction
   /G {0 0 0 1 0 1 bc } def
      % for a greek letter
   /H {0 0 1 1 0 0 bc } def
      % horizontal fraction line
   /Q {0 0 1 1 1 0 bc } def
      % open square root
   /Y {1 1 0 1 1 1 bc } def
      % close square root
   /Z {0 0 1 1 1 1 bc } def
      % nominator, end of fraction
   /a {1 0 0 0 0 0 bc } def
      % ascii "A"=65
   /b {1 1 0 0 0 0 bc } def
   /c {1 0 0 1 0 0 bc } def
   /d {1 0 0 1 1 0 bc } def
   /e {1 0 0 0 1 0 bc } def
   /f {1 1 0 1 0 0 bc } def
   /g {1 1 0 1 1 0 bc } def
   /h {1 1 0 0 1 0 bc } def
   /i {0 1 0 1 0 0 bc } def
   /j {0 1 0 1 1 0 bc } def
   /k {1 0 1 0 0 0 bc } def
   /l {1 1 1 0 0 0 bc } def
   /m {1 0 1 1 0 0 bc } def
   /n {1 0 1 1 1 0 bc } def
   /o {1 0 1 0 1 0 bc } def
   /p {1 1 1 1 0 0 bc } def
   /q {1 1 1 1 1 0 bc } def
   /r {1 1 1 0 1 0 bc } def
   /s {0 1 1 1 0 0 bc } def
   /t {0 1 1 1 1 0 bc } def
   /u {1 0 1 0 0 1 bc } def
   /v {1 1 1 0 0 1 bc } def
   /w {0 1 0 1 1 1 bc } def
   /x {1 0 1 1 0 1 bc } def
   /y {1 0 1 1 1 1 bc } def
   /z {1 0 1 0 1 1 bc } def
end

/BuildChar {    % stack has font char

  /col { gsave 0 setlinewidth
    3 {gsave
       currentpoint translate wr 0
       rmoveto wr wr wr -90 270 arc
       1 eq {fill} {stroke} ifelse
      grestore 0 w rmoveto} repeat
    grestore
  } def
```

```
  /bc { 6 3 roll
    /w 30 def       % size of a box
     /wr 12 def
     0 0 moveto
     gsave
       col w 0 rmoveto
       col
     grestore
  } def             % braille routine

  80 0                        % width
  0 0 60 100          % bounding box
  setcachedevice
  exch begin
       % font begin
  Encoding exch get
       % index by char in Encoding
  CharProcs exch get
       % lookup name in CharProcs
  end
  exec
       % execute char procdure
} def
end                       % of braille

% Подключение шрифта

/Braille braille definefont pop

% Использование шрифта


100 100 translate
/Braille findfont 36 scalefont setfont
0 162 moveto (abcdefghij) show
0 102 moveto (klmnopqrst) show
0  42 moveto (uvxyz  w) show
/Courier findfont 14 scalefont setfont
5 150 moveto  (abcdefghij) [ 30 30
28 30 30 26 30 30 26 ] xshow
5 90 moveto  (klmnopqrst) [ 32 26 30
28 29 30 30 28 30 ] xshow
5 30 moveto (uvxyzw) [ 30 29 30 28
86 ] xshow
showpage
```

Я разделил программу на три части, очень различные по размеру, – определение шрифта, его подключение и использование.

Использование – это самое простое, начнем с него. Для удобства начало координат смещается на 100 пунктов вправо и вверх. Затем устанавливается нужный шрифт (**braille**) размера 36 пунктов, Затем обычным образом печатаются три строчки. После этого устанавливается для подписей «нормальный» шрифт (**Courier**) и им печатаются подписи. Печатающяя команда **xshow** отличается тем, что берет пробелы между буквами из массива, который нужно задавать дополнительно. Мне было удобно использовать его, чтобы буквы попадали точно, куда нужно. Обратите особое внимание на то, как сделано смещение для буквы w в последней строчке.

Подключение совсем простое. Слово **definefont** очень близко по своему характеру к слову **def**. И то и другое требуют ключа и описания. К описанию мы сейчас перейдем, ключ мы уже использовали, и не было разницы в использовании нового ключа **/Braille** и стандартного ключа **/Courier**.

Описание состоит из задания для создаваемого шрифта специального информационного объекта – словаря и его заполнения. Словарь вводится так

```
/braille 10 dict def
```

Слово **dict** создает словарь, а 10 – это его первоначальная емкость. В каких единицах? В «элементах словаря», а как они определяются, будет видно при заполнении. Заполнение происходит внутри «операторных скобок»

```
braille begin
. . . . .
end
```

В основном заполнение состоит из обычных описаний обязательных параметров шрифта

```
/FontType 3 def
/FontMatrix [.01 0 0 .01 0 0] def
/FontBBox [0 0 60 100] def
/Encoding 256 array def
/CharProcs 3 dict def
/BuildChar
```

/FontType задает тип шрифта (3 для шрифтов, определенных пользователем).

/FontMatrix – это таблица пересчета изображаемого знака в шрифт размера 1000 пунктов. При выполнении

scalefont эта матрица просто заменяется другой, соответствующей требуемому размеру.

/FontBBox – это задание прямоугольника, объемлющего все символы данного шрифта.

/Encoding – массив, который будет связывать коды символов с ключами слов, изображающих эти символы. Сами слова находятся в словаре **/CharProcs**, который входит в словарь шрифта в качестве еще одного элемента.

/BuildChar – процедура, исполняемая при печати каждого символа данного шрифта.

Наряду с этими описаниями (процедура **/BuildChar** описана полностью, но мы этим описанием займемся позднее), в заполнение словаря входит заполнение его «сложных объектов» – массива перекодировки **/Encoding** и словаря исполняемых процедур **/CharProcs**.

Массив заполняется в два приема. Сначала во все его клеточки пишется невозможный символ **.notdef** (это делается обычным циклом),

```
0 1 255 {Encoding exch /.notdef put} for
```

а затем каждый нужный символ заполняется отдельно, например,

```
Encoding 98 /b put
Encoding 44 /comma put
```

(так в элемент 98, а это код ASCII буквы **b**, помещается ключ **/b**, а в элемент 44 помещается ключ **/comma**, чтобы рисовать запятую).

Словарь **/CharProcs** заполняется внутри операторных скобок. Мы уже знаем про эти скобки и можем кое-что записать внутрь.

```
CharProcs begin
   /b {1 1 0 0 0 0 bc } def
   /comma     {0 1 0 0 0 0 bc } def
   /space     {0 0 0 0 0 0 bc } def
   /.notdef {} def
end
```

Я выбрал для примера четыре символа. Символы шрифта будут печататься так: мы положим в стек шесть символов, единиц для черных точек и нулей для белых, а затем вызовем специальную рисующую процедуру **bc** (где же она? Еще появится). Вы можете убедиться в правильном задании буквы и поверить, что запятая определяется именно так. Важно, что пробел задается как буква – набором из шести белых точек (см. рисунок 2). В описания входит и определение пустого символа **.notdef**.

Процедура, рисующая символ, получает на стеке имя словаря шрифта и код символа. Она открывает словарь (словом **begin**) и получает возможность работать с его элементами. После этого по символу она достает словом **get** нужный ключ из массива **Encoding**, а по ключу тем же словом **get** получает из словаря **CharProcs** исполняемый текст. После этого словом **end** закрывается словарь и словом **exec** полученный текст исполняется. Мы опустим подробности «системной» процедуры **setcachedevice**, передающей интерпретатору метрические параметры символа – его ширину по обеим координатам и ограничивающий прямоугольник. Нас больше интересует рисование символа, так что обратимся к описанию команды **bc**.

Она рисует сначала левый столбец, а затем правый. Поэтому, совершенно естественно, мы начинаем с перестановки данных об этих столбцах. И вводим обозначения для шага между «точками» и для размера точки. Встаем в начало координат (интерпретатор обеспечивает перенос начала в опорную точку очередного символа), рисуем левый столбец, сдвигаемся вправо и рисуем правый столбец. Столбцы рисуются одинаково, рационально рисование столбца сделать специальной процедурой.

Процедура **col** рисует один столбец. Основная ее часть – троекратное повторение одного и того же действия, которое само состоит из трех шагов: нарисовать кружок, в зависимости от числа в стеке закрасить его или обвести, сдвинуться вверх на нужный шаг. Первый шаг делается в строке

```
currentpoint translate wr 0 rmoveto
wr wr wr -90 270 arc
```

второй – в строке

```
1 eq {fill} {stroke} ifelse
```

(понятно ли? Число в стеке сравнивается с 1, и булевский результат сравнения остается в стеке. К нему приписываются два действия – заливка и обрисовка, а слово **ifelse** выполняет то из них, которое нужно), третий шаг – в действиях

```
0 w rmoveto
```

Вот и все.

Упражнения.

1. Дополните код русскими буквами.
2. Разработайте шрифт для «пляшущих человечков» из рассказа А. Конан-Дойля.

Варианты кода Брайля.

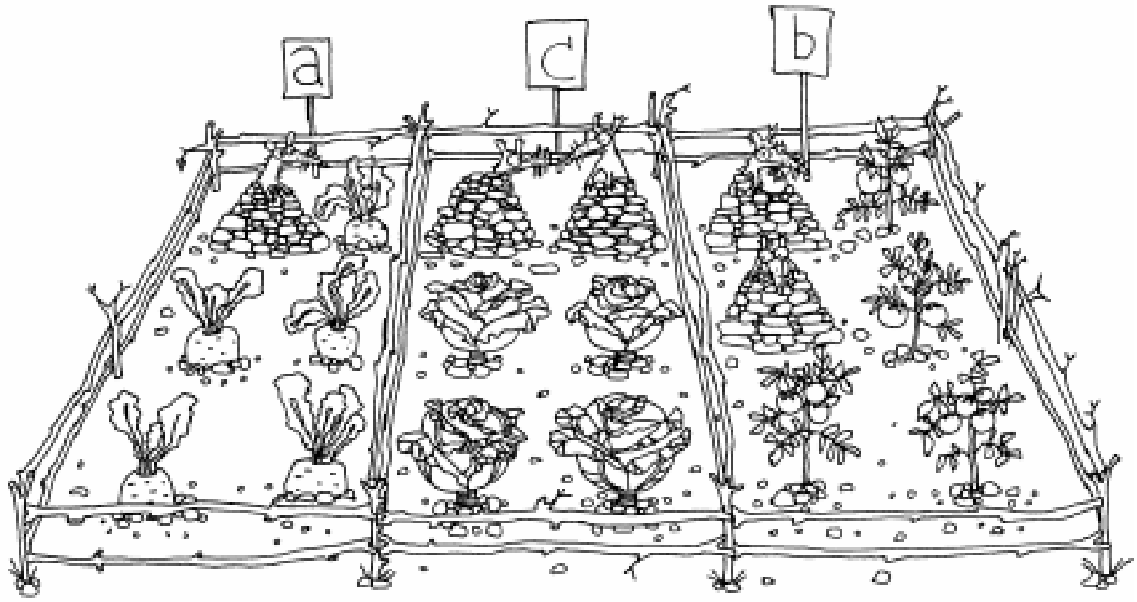
Мы рассмотрели только самый простой вариант кода Брайля – минимальный. Было разработано много его вариантов, предназначенных для различных профессиональных использований. Назовем некоторые из них

1. *Литературный Брайль*. В нем комбинации точек, не использованные для букв и знаков препинания, кодируют часто встречающиеся сочетания букв и небольшие слова.
2. *Музыкальный Брайль*. Это система записи нот.
3. *Брайль ASCII* и *Компьютерный Брайль*. Недавно появившиеся системы кодирования для наборов символов, используемых в программах. Эти две системы отличаются друг от друга и имеют разное назначение.
4. *Код Немета* для набора математических формул.

С нашей точки зрения интереснее всего этот четвертый вариант, к нему мы обратимся ниже. А сейчас отметим, что в настоящее время происходит пересмотр кодировок, делаются попытки создать унифицированный вариант кода Брайля, приспособленный для всех нужд сразу, создаются устройства вывода информации из компьютера, предлагающие незрячему пользователю строчку текста, «набранную по Брайлю», делаются попытки перехода на восьмиточечные символы.

Код Немета для набора математических формул.

Этот код был предложен в 1991 г. американским ученым А. Неметом (Dr. Abraham Nemeth) для записи в системе Брайля математических формул (мне кажется, что эта система была разработана под сильным влиянием системы TeX, но проверить это я не смог).



Покажем кодировку Немета на небольшом примере. Пусть требуется набрать выражение $\frac{1}{\sqrt{2\pi}}$.

Отметим, что в TeXe (точнее, в ЛаTeXe) эта формула запишется так:

```
$\frac{1}{\sqrt{2\pi}}$.
```

Запись этого выражения в кодировке Немета показана на рисунке 3, что значит (сравните с записью в TeXe):

```
Дробь-открывающая скобка
        Цифра 1
Дробь-черта
        Корень-открывающая скобка
        Цифра 2
        Метка греческой буквы
        Буква p
        Корень-закрывающая скобка
Дробь-закрывающая скобка
```

Рисунок 3

(проектируя алфавит для кода Брайля, я благоразумно добавил в шрифт цифры и некоторые символы, кодирующие знаки этого примера, так что мне понадобилось напечатать строку

```
F1HQ2GpYZ
```

Найдите эти символы в тексте программы).

В Интернете лежит очень симпатичный компьютеризованный учебник по системе Немета – пример для многих. Как бы хотелось, чтобы у нас нашлись люди, которые бы русифицировали учебник по системе Немета и внедрили бы эту систему в русский Брайль.

Литература.

1. Петцольд, Чарльз. Код. М., Микрософт Пресс, 2001, 495 с.
2. Романовский И. В. Язык программирования Постскрипт. Компьютерные инструменты в образовании, № 3-4, 1999 г., стр. 96-108.
3. Computorized Nemeth Code tutor http://www.upshowinst.org/downloads/nemeth_tutor.

Примечание. В Санкт-Петербурге находится уникальная библиотека – Санкт-Петербургская государственная библиотека для слепых. Она была открыта еще до революции 1917 г. на базе благотворительного общества слепых.

Романовский Иосиф Владимирович,
доктор физ.-мат. наук,
профессор СПбГУ.