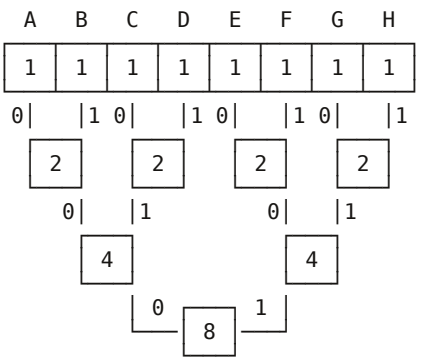
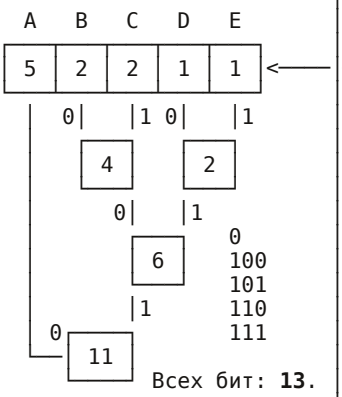
[**Дерево Хаффмана**]



Слева Направо и Снизу Вверх:
000, 001, 010, 011,
100, 101, 110, 111.

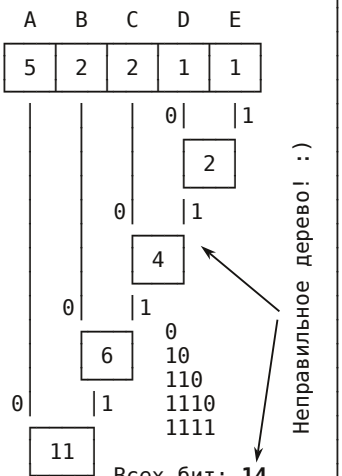
Стас!
Посмотри как числа "ложатся"!

[**Дерево Хаффмана**]



Всех бит: **13**.

Одно и тоже дерево,
только чуть иначе
(см. ниже):



Неправильное дерево! :)

Всех бит: **14**.

Высота дерева должна
быть как можно меньше! :)

[**Дополнение 1**]

0 - 0000	8 - 1000
1 - 0001	9 - 1001
2 - 0010	10 - 1010
3 - 0011	11 - 1011
4 - 0100	12 - 1100
5 - 0101	13 - 1101
6 - 0110	14 - 1110
7 - 0111	15 - 1111

Найти наибольший элемент массива

```
// JavaScript.

var y, tb, big_number;
var ArrayMas_1 = [5, 2, 2, 1, 1];

big_number = 1;

// В tb длина массива.
tb = ArrayMas_1.length;

// Найти наибольший элемент массива
// (найте в массиве ArrayMas_1[]).
for(y = 0; y < tb; y++)
  {
  // Если нужно, пропускаем 0.
  // Если в массиве присутствует 0, пропускаем его!
  if (ArrayMas_1[y] != 0)
     {
     if (ArrayMas_1[y] > big_number)
        {
        // В big_number наибольший элемент массива.
        big_number = ArrayMas_1[y];
        }
     }
  }
// ВЫХОД: В big_number наибольший элемент массива!
```

Найти наименьший элемент массива!
Извлечь! А на место записать 0!
После этого, повторить поиск и
извлечь другой наименьший элемент массива!

```
// На входе в big_number число к которому
// нужно получить указатель.
var big_number, index_big_number, y, sw;
var ArrayMas_1 = [5, 2, 2, 1, 1];
sw = 0;
for(y = 0; y < ArrayMas_1.length; y++)
   {
   if (big_number == ArrayMas_1[y])
      {
      // Указатель.
      index_big_number = y;
      // Найдено.
      sw = 1;
      break;
      }
   }
if (sw == 0) {alert("Не найдено!")};
// ВЫХОД: В index_big_number указатель.
```

Если строка кодируется 3-х битным кодом, то всех
бит будет - 3 бита * 36 символов = 108 бит.
Сжатый: 102 бита (108 бит - 102 бита = 6 бит),
на 6-ть бит меньше!

Кодирование на: https://asecuritysite.com/calculators/huff

Закодировать строку кодом Хаффмана

Строка: aaaaaaaabbbbbbbccccccddddddeeeeefffggh
(длина: 36 символов).

```
a - всех символов 8 => 01
b - всех символов 7 => 00
c - всех символов 6 => 110
d - всех символов 5 => 101
e - всех символов 4 => 100
f - всех символов 3 => 1110
g - всех символов 2 => 11111
h - всех символов 1 => 11110
```

Найти наименьший элемент массива

```
// JavaScript.

var z, small_number, tb;
var ArrayMas_2 = [5, 2, 2, 1, 1];

// В tb длина массива.
tb = ArrayMas_2.length;
// Число с конца массива.
small_number = ArrayMas_2[tb - 1];

// Ищем наименьший элемент массива, ищем в ArrayMas_2[].
// НАЧИНАЕМ ИСКАТЬ С КОНЦА МАССИВА!
// tb - 2 - Стать на предпоследний элемент массива.
// z >= 0 - На выходе будет: z = -1.
// В small_number наименьший элемент массива.
for(z = tb - 2; z >= 0; z--)
   {
   // Если нужно, пропускаем 0.
   // Если в массиве присутствует 0, пропускаем его!
   if (ArrayMas_2[z] != 0)
      {
      if (ArrayMas_2[z] < small_number)
         {
         small_number = ArrayMas_2[z];
         }
      }
   // Для проверки.
   // alert(ArrayMas_2[z] + " " + small_number);
   }
// ВЫХОД: В small_number наименьший элемент массива!
```

Продолжение на стр. 5.

[**Дополнение 1**]
Колонка чисел слева - десятичные числа.
Колонка чисел справа - двоичные числа.

Отсортировать числовой массив по убыванию

```
//
// JavaScript.
// Отсортировать числовой массив по убыванию.
//
var y, z, tb, big_number;  // Отсортировать именно так,
                           // а не через sort()!
// Входной массив.
var ArrayMas_1 = [2, 2, 5, 1, 1];

// Выходной массив, отсортированный по убыванию.
var ArrayMas_Sort = [];

// В tb длина массива.
tb = ArrayMas_1.length;

for(y = 0; y < tb; y++)
  {
  big_number = 1;

  // Найти наибольший элемент массива
  // (найти в массиве ArrayMas_1[]).
  for(z = 0; z < tb; z++)
     {
     // Если нужно, пропускаем 0.
     // Если в массиве присутствует 0, пропускаем его!
     if (ArrayMas_1[z] != 0)
        {
        if (ArrayMas_1[z] > big_number)
           {
           // В big_number наибольший элемент массива.
           big_number = ArrayMas_1[z];
           }
        }
     }

  // Там где был наибольший элемент массива, записать 0.
  for(z = 0; z < tb; z++)
     {
     if (big_number == ArrayMas_1[z])
        {
        // Записать 0.
        ArrayMas_1[z] = 0;
        // Прервать цикл.
        break;
        }
     }

  // Создаём отсортированный массив по убыванию.
  ArrayMas_Sort[y] = big_number;
  }

// ВЫХОД: Отсортированный массив ArrayMas_Sort.
// При этом входной массив ArrayMas_1 будет испорчен!

// Для проверки.
// alert(ArrayMas_Sort);
```

Префиксный код с переменными блоками

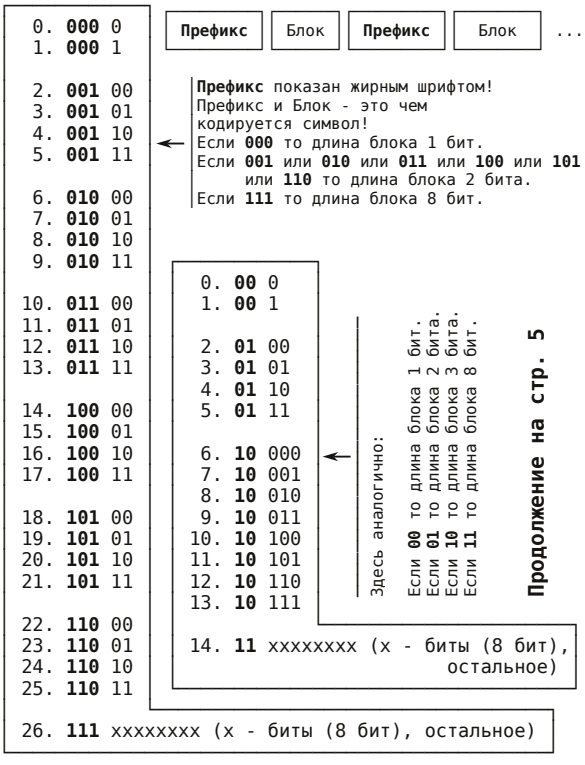
Символ	Количество	Префиксный код с переменными блоками, бит
ПРОБЕЛ	18	**00**0
Р	12	**00**1
К	11	**01**00
Е	11	**01**01
У	9	**01**10
А	8	**01**11
Г	4	**10**000
В	3	**10**001
Ч	2	**10**010
Л	2	**10**011
И	2	**10**100
З	2	**10**101
Д	1	**10**110
Х	1	**10**111
С	1	**11**000
Т	1	**11**001
Ц	1	**11**010
Н	1	**11**011
П	1	**11**100

Где **00** - блок в 1 бит, **01** - блок в 2 бита, **10** и **11** - блоки в 3 бита.
Префикс показан жирным шрифтом!
Остальной код - это блок.
Префикс и блок - это чем кодируется символ!

Кодирование для чайников, ч.1
https://habr.com/ru/articles/535862/
Спасибо Автору!

На **habr.com** было написано:
Вы написали дилетанский текст... Улыбнитесь! :)

А если алфавит не 19 символов, а 256 символов. Как табличку составить (пример см. ниже)?



Продолжение на стр. 5

Создаём таблицу 1

Сколько раз встречаются символы (байты) в файле, плюс какие символы (байты) используются

```
//
// JavaScript.
// Вставить в страницу Таблицу 1.
//
function CreateInsertTable(id_insert_table, id_c, id_s)
{
   var x, y, code_h, hex, id_code, id_sym, temp;
   code_h = 0;

   // 0123456789ABCDEF по горизонтали.
   // --------------------------------

   temp = "<table cellspacing=\"0\" cellpadding=\"0\"
          border=\"0\">";
   temp += "<tr>";

   temp += "<td> </td>";
   temp += "<td>  </td>";

   temp += "<td> 0</td>";
   temp += "<td> </td>";

   temp += "<td> 1</td>";
   temp += "<td> </td>";

   temp += "<td> 2</td>";
   temp += "<td> </td>";

   temp += "<td> 3</td>";
   temp += "<td> </td>";

   temp += "<td> 4</td>";
   temp += "<td> </td>";
```

Для заметок:

```
   temp += "<td> 5</td>";
   temp += "<td> </td>";

   temp += "<td> 6</td>";
   temp += "<td> </td>";

   temp += "<td> 7</td>";
   temp += "<td> </td>";

   temp += "<td> 8</td>";
   temp += "<td> </td>";

   temp += "<td> 9</td>";
   temp += "<td> </td>";

   temp += "<td> A</td>";
   temp += "<td> </td>";

   temp += "<td> B</td>";
   temp += "<td> </td>";

   temp += "<td> C</td>";
   temp += "<td> </td>";

   temp += "<td> D</td>";
   temp += "<td> </td>";

   temp += "<td> E</td>";
   temp += "<td> </td>";

   temp += "<td> F</td>";
   temp += "<td>  </td>";

   temp += "<td>0</td><td>1</td><td>2</td><td>3</td>";
   temp += "<td>4</td><td>5</td><td>6</td><td>7</td>";
   temp += "<td>8</td><td>9</td><td>A</td><td>B</td>";
   temp += "<td>C</td><td>D</td><td>E</td><td>F</td>";

   temp += "</tr>";
   temp += "</table>";

   temp += "<table cellspacing=\"0\" cellpadding=\"0\"
border=\"0\">";

   id_code = 0; id_sym = 0;

   for (x = 0; x < 16; x++)
      {
      temp += "<tr>";

      // 0123456789ABCDEF по вертикали.
      // ------------------------------

      temp += "<td>";

      if (code_h >= 10)
         {
         // В Hex.
         hex = code_h.toString(16);

         // Символ в верхний регистр.
         temp += hex.toUpperCase();
         }
         else
         {
         temp += code_h;
         }

      temp += "</td>";

      // Пробел.
      temp += "<td>  </td>";

      // Hex-Код.
      // --------

      for (y = 0; y < 16; y++)
         {
         temp += "<td id=" + id_c + id_code + ">";
         temp += "--";
         temp += "</td>";

         // Если не конец кода, то один пробел.
         if (y != 15)
            {
            // Пробел.
          temp += "<td> </td>";
            }
            else
            {
            // В конце два пробела.
            temp += "<td>  </td>";
            }

         id_code++;
         }

      // Символы.
      // --------

      for (y = 0; y < 16; y++)
         {
         temp += "<td id=" + id_s + id_sym + ">";
         temp += ".";
         temp += "</td>";

         id_sym++;
         }

      temp += "</tr>";
      code_h++;
      }

   temp += "</table>";

   // ВСТАВИТЬ В СТРАНИЦУ ТАБЛИЦУ.
   document.getElementById(id_insert_table).innerHTML =
temp;
}
```

Для заметок:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Создаём два массива

```
// --- JavaScript ---

// Здесь Hex-код (какие байты присутствуют в файле).
// "--" - Hex-код (байт) отсутсвует.
var ArrayDataHex = Array(256).fill('--'); // Заполнить
массив "--".

// Здесь результаты подсчётов (сколько раз байты
// (символы) встречаются в файле).
// 0 - Подсчёт отсутствует.
var ArrayTotalByte = Array(256).fill(0); // Заполнить
массив 0.

// ---

var x, y, sw_1, sw_2, hex, dec, bl, count;

sw_1 = 0;
bl = buffer.length;

for (x = 0; x < bl; x++)
   {

   if (sw_1 == 0)
      {
      // Для первого раза (новый байт).
      // ---

      for (y = 0; y < 256; y++)
         {
         if (buffer[x] == y)
            {
            // В Hex.
            hex = buffer[x].toString(16);

            // В верхний регистр.
            hex = hex.toUpperCase();

            // Если нужно, дополнить нулём.
            if (hex.length == 1)
               { hex = "0" + hex; }

            // Записать hex в массив.
            ArrayDataHex[y] = hex;
            // Попутно считаем сколько таких байт всего
            // (всего 1 байт).
            // Информацию о этом храним
            // в массиве ArrayTotalByte.
            ArrayTotalByte[y] = 1;

            sw_1 = 1;
            break;
            }
         }
      }
      else
      {
      // Остальные разы.
      // ---

      sw_2 = 0;
```

В buffer загруженный файл!

```
      for (y = 0; y < 256; y++)
         {

         // Получить Hex или "--" из массива.
         hex = ArrayDataHex[y];

         // Если не "--", значит Hex.
         if (hex != '--')
            {
            // Hex в Dec.
            dec = parseInt(hex, 16);

            if (dec == buffer[x])
               {
               // Есть байт в массиве.

               count = ArrayTotalByte[y];
               count++;
               ArrayTotalByte[y] = count;

               sw_2 = 1;
               break;
               }
            }

         }

      // Записываем байт (новый байт) в массив.
      if (sw_2 == 0)
         {

         for (y = 0; y < 256; y++)
            {
            if (buffer[x] == y)
               {
               // В Hex.
               hex = y.toString(16);

               // В верхний регистр.
               hex = hex.toUpperCase();

               // Если нужно, дополнить нулём.
               if (hex.length == 1)
                  { hex = "0" + hex; }

               // Записать hex в массив.
               ArrayDataHex[y] = hex;
               // Попутно считаем сколько таких байт
               // всего (всего 1 байт).
               // Информацию о этом храним
               // в массиве ArrayTotalByte.
               ArrayTotalByte[y] = 1;

               break;
               }
            }

         }

      }

   }

// !
// alert(ArrayDataHex);
// alert(ArrayTotalByte);

// Проверка. Self Control.
// Результат должен быть - размер файла.

// count = 0;

// for (y = 0; y < 256; y++)
//    {
//    if (ArrayTotalByte[y] != 0)
//       {
//       count = count + ArrayTotalByte[y];
//       }
//    }

// !
// alert("Размер файла: " + count + " байт(а)");
```

Наполнить созданную таблицу 1

```
//
// JavaScript.
// Наполнить Таблицу 1.
//
function InsertToTable(buffer, id_c, id_s, ArrayDataHex,
ArrayTotalByte)
{
   var y, id_code, id_sym, title, hex, dec, sym;

   // Вставить в страницу (в Таблицу 1) Hex-код.
   // Дополнительно вставить title и class.
   for (y = 0; y < 256; y++)
      {
      // Если не "--", значит Hex.
      if (ArrayDataHex[y] != '--')
         {
         title = "Байт: " + ArrayDataHex[y] +
         " (Dec: " + y + ") встречается в файле " +
         ArrayTotalByte[y] + " раз(а)";
         id_code = id_c + y;

         // Вставить title.
         document.getElementById(id_code).title = title;
         // Вставить class (class="bold").
         document.getElementById(id_code).className =
         "bold";
         // Вставить в страницу (в Таблицу 1) Hex-код.
         document.getElementById(id_code).innerHTML =
         ArrayDataHex[y];
         }
      }

   // Вставить в страницу (в Таблицу 1) Символы.
   for (y = 0; y < 256; y++)
      {
      // Получить Hex или "--" из таблицы.
      hex = ArrayDataHex[y];

      // Если не "--", значит Hex.
      if (hex != '--')
         {
         // Hex в Dec.
         dec = parseInt(hex, 16);

         // Буквы: ABCDEFGHIJKLMNOPQRSTUVWXYZ.
         // Буквы: abcdefghijklmnopqrstuvwxyz.
         if ( (dec >= 0x41 && dec <= 0x5a) ||
              (dec >= 0x61 && dec <= 0x7a) )
            {
            // Получаем символ по коду.
            sym = String.fromCharCode(dec);
            id_sym = id_s + y;
            // Вставить в страницу (в Таблицу 1) символ.
            document.getElementById(id_sym).innerHTML =
            sym;
            }

         else

         // Цифры: 0123456789.
         if (dec >= 0x30 && dec <= 0x39)
            {
            // Получаем символ по коду.
            sym = String.fromCharCode(dec);
            id_sym = id_s + y;
            // Вставить в страницу (в Таблицу 1) символ.
            document.getElementById(id_sym).innerHTML =
            sym;
            }

         else

         // Остальное: ~!@#$%^&*()_+=[]{},.?/\"'
         // плюс пробел
         if (
            dec == 0x7e || dec == 0x21 || dec == 0x40 ||
            dec == 0x23 || dec == 0x24 || dec == 0x25 ||
            dec == 0x5e || dec == 0x26 || dec == 0x2a ||
            dec == 0x28 || dec == 0x29 || dec == 0x5f ||
            dec == 0x2b || dec == 0X3d || dec == 0x5b ||
            dec == 0x5d || dec == 0x7b || dec == 0x7d ||
            dec == 0x2c || dec == 0x2e || dec == 0x3f ||
            dec == 0x2f || dec == 0x5c || dec == 0x22 ||
            dec == 0x27 || dec == 0x20 )
            {
            // Получаем символ по коду.
            sym = String.fromCharCode(dec);
            id_sym = id_s + y;
            // Вставить в страницу (в Таблицу 1) символ.
            document.getElementById(id_sym).innerHTML =
            sym;
            }
         }
      }
}
```

Префиксный код с переменными блоками

```
 0. 000 0
 1. 000 1

 2. 001 00
 3. 001 01
 4. 001 10
 5. 001 11

 6. 010 00
 7. 010 01
 8. 010 10
 9. 010 11

10. 011 00
11. 011 01
12. 011 10
13. 011 11

14. 100 00
15. 100 01
16. 100 10
17. 100 11

18. 101 000
19. 101 001
20. 101 010
21. 101 011
22. 101 100
23. 101 101
24. 101 110
25. 101 111

26. 110 000
27. 110 001
28. 110 010
29. 110 011
30. 110 100
31. 110 101
32. 110 110
33. 110 111

34. 111 xxxxxxxx (x - биты (8 бит), остальное)
```

Ещё одна табличка.

Префикс	Блок	Префикс	Блок	...

Префикс показан жирным шрифтом!
Префикс и Блок - это чем кодируется символ!
Если **000** то длина блока 1 бит.
Если **001** или **010** или **011** или **100** то длина блока 2 бита.
Если **101** или **110** то длина блока 3 бита.
Если **111** то длина блока 8 бит.

Закодировать строку кодом Хаффмана (пример 2)

Кодирование на: https://asecuritysite.com/calculators/huff

Строка:

```
AAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCC
CCCCCCCCCCDDDDDDDDDDDDDDDDDDDDDDDDEEEEEEEEEEEEEEEEEEEEEEFFFFFFFFFF
FFFFFFFFFFGGGGGGGGGGGGGGGGGGGGGGHHHHHHHHHHHHHHHHHHHHHIIIIIIIIIIIIII
IIIJJJJJJJJJJJJJJJJJKKKKKKKKKKKKKKKKLLLLLLLLLLLLLLLLMMMMMMMMMMMMMM
NNNNNNNNNNNNNNOOOOOOOOOOOOOPPPPPPPPPPPQQQQQQQQQQQRRRRRRRRRSSSSSSSSTT
TTTTTUUUUUUVVVVVWWWWXXXYYZ (длина: 351 символ, 26 разных симв.).
```

Если строка кодируется 5 битным кодом, то всех бит будет - 5 бит * 351 символ = 1755 бит.
Сжатый: 1534 бита (1755 бит - 1534 бита = 221 бит), на 221 бит меньше (221 / 5 = 44.2, меньше на 44 символа)!

```
A - символов 26 => 1101
B - символов 25 => 1100
C - символов 24 => 1010
D - символов 23 => 1001
E - символов 22 => 1000
F - символов 21 => 0110
G - символов 20 => 0101
H - символов 19 => 0100
I - символов 18 => 0010
J - символов 17 => 0001
K - символов 16 => 0000
L - символов 15 => 11110
M - символов 14 => 11101
N - символов 13 => 11100
O - символов 12 => 10110
P - символов 11 => 01111
Q - символов 10 => 01110
R - символов  9 => 00110
S - символов  8 => 111111
T - символов  7 => 111110
U - символов  6 => 101110
V - символов  5 => 001111
W - символов  4 => 001110
X - символов  3 => 1011110
Y - символов  2 => 10111111
Z - символов  1 => 10111110
```

Таблица 1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0	00	--	--	--	--	--	--	--	--	09	0A	--	0C	0D	--	--	....
1	--	--	--	--	--	--	--	--	--	--	--	1B	--	--	--	--	....
2	20	21	22	--	--	25	--	--	28	29	--	2B	2C	2D	2E	2F	!"..%..().+,../
3	30	31	32	33	34	35	36	37	38	39	3A	3B	--	3D	--	--	0123456789...=..
4	--	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	--	.ABCDEFGHIJKLMN.
5	50	51	52	53	54	55	56	57	58	59	--	--	5C	--	--	5F	PQRSTUVWXY..\.._
6	--	61	--	63	--	--	66	--	68	--	--	--	6C	6D	--	--	.a.c..f.h...lm..
7	--	--	--	--	74	--	--	--	--	--	--	--	--	--	--	--	....t..........
8	80	81	82	83	84	85	--	87	88	--	8A	8B	8C	8D	8E	8F	....
9	90	91	92	--	94	--	--	97	98	--	--	--	--	9D	--	9F	....
A	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	....
B	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	....
C	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	....
D	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	....
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	--	EB	EC	ED	EE	EF	....
F	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	....

"--" - Байт отсутствует в файле!
Наведите курсор мыши на hex-код (будет подсказка)!

1534 / 8 = 191.75 байт.

1. Сначала создаём Таблицу 1 (см. код на JavaScript).
2. Затем создаём два массива (см. код на JavaScript).
3. Потом наполняем Таблицу 1 (см. код на JavaScript).

Результат см. Таблицу 1.

Коды Голомба (примеры)

```
Унарный код
01
001
0001
00001
000001
0000001
00000001
000000001
0000000001
...
```

```
10
11
010   Код постоянной длины
011
0010
0011
00010
00011
000010
000011
0000010 ]
0000011 ]
00000010 \
00000011 m=2
...
```

```
100
101
110   Код постоянной длины
111
0100
0101
0110
0111
00100 ]
00101 ]
00110 ]
00111 ]
001000 \
001001 m=4
001010
001011
```

Унарный код

Унарный код

Взято с: https://studbooks.net/2259718/informatika/unarnyy

Я себе попробовал сделать так:

```
 1. 000 1
 2. 001 1
 3. 010 1
 4. 011 1
 5. 100 1
 6. 101 1
 7. 110 1

 8. 000 01
 9. 001 01
10. 010 01
11. 011 01
12. 100 01
13. 101 01
14. 110 01

15. 000 001
16. 001 001
17. 010 001
18. 011 001
19. 100 001
20. 101 001
21. 110 001

22. 000 0001
23. 001 0001
24. 010 0001
25. 011 0001
26. 100 0001
27. 101 0001
28. 110 0001

29. 000 0000
30. 001 0000
31. 010 0000
32. 011 0000
33. 100 0000
34. 101 0000
35. 110 0000

36. 111 xxxxxxxx (x - биты (8 бит), остальное).
```

Коды Голомба!
Упоминание было в книге: Ватолин Д., Ратушняк А., ... Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео (2002г.), **стр. 25.**

Префиксный код с переменными блоками

[**Дополнение**]

```
 0. 111 000 00 или 0. 111 000 0
 1. 111 000 01     1. 111 000 1
 2. 111 000 10     2. 111 001 0
 3. 111 000 11     3. 111 001 1
                   4. 111 010 0
 4. 111 001 00     5. 111 010 1
 5. 111 001 01     6. 111 011 0
 6. 111 001 10     7. 111 011 1
 7. 111 001 11     8. 111 100 0
                   9. 111 100 1
 8. 111 010 00    10. 111 101 0
 9. 111 010 01    11. 111 101 1
10. 111 010 10    12. 111 110 0
11. 111 010 11    13. 111 110 1
                  14. 111 111 xxxxxxxx (x - биты (8 бит),
12. 111 011 00                        остальное)
12. 111 011 00
13. 111 011 01    Плохо!
14. 111 011 10    Всё это плохо!
15. 111 011 11

16. 111 100 00    Для заметок: _____
17. 111 100 01
18. 111 100 10    _____
19. 111 100 11    _____

20. 111 101 00    _____
21. 111 101 01    _____
22. 111 101 10    _____
23. 111 101 11    _____

24. 111 110 00    _____
25. 111 110 01    _____
26. 111 110 10    _____
27. 111 110 11

28. 111 111 xxxxxxxx (x - биты (8 бит), остальное)
```

Вёрстка материала: Демидов С.В. Написано на коленке! Могут быть ошибки! ЧЕРНОВИК! Украина. (С) Демидов С.В.