Код Грея

Материал из Википедии — свободной энциклопедии [править]

Код Грея — система счисления, в которой два соседних значения различаются только в одном разряде. Наиболее часто на практике применяется **рефлексивный двоичный код Грея**, хотя в общем случае существует бесконечное множество кодов Грея для систем счисления с любым основанием. В большинстве случаев, под термином «код Грея» понимают именно рефлексивный бинарный код Грея.

Изначально предназначался для защиты от ложного срабатывания электромеханических переключателей. Сегодня коды Грея широко используются для упрощения выявления и исправления ошибок в системах связи, а также в формировании сигналов обратной связи в системах управления.

2-битный код Грея

```
00
01
11
10
```

3-битный код Грея

```
000
001
011
010
110
111
101
100
```

4-битный код Грея

```
0000
0001
0011
0010
0110
```

Содержание [убрать]

Название [править]

Название *рефлексный (отражённый) двоичный код* происходит от факта, что вторая половина значений в коде Грея эквивалентна первой половине, только в обратном порядке, за исключением

English
Español
Eesti
فارسی
Suomi
Français
עברית
हिन्दी
Magyar
Italiano
日本語
한국어
Nederlands
Polski
Português
Română
Slovenčina
Svenska
ไทย
Українська
Tiếng Việt
中文
Править ссылки

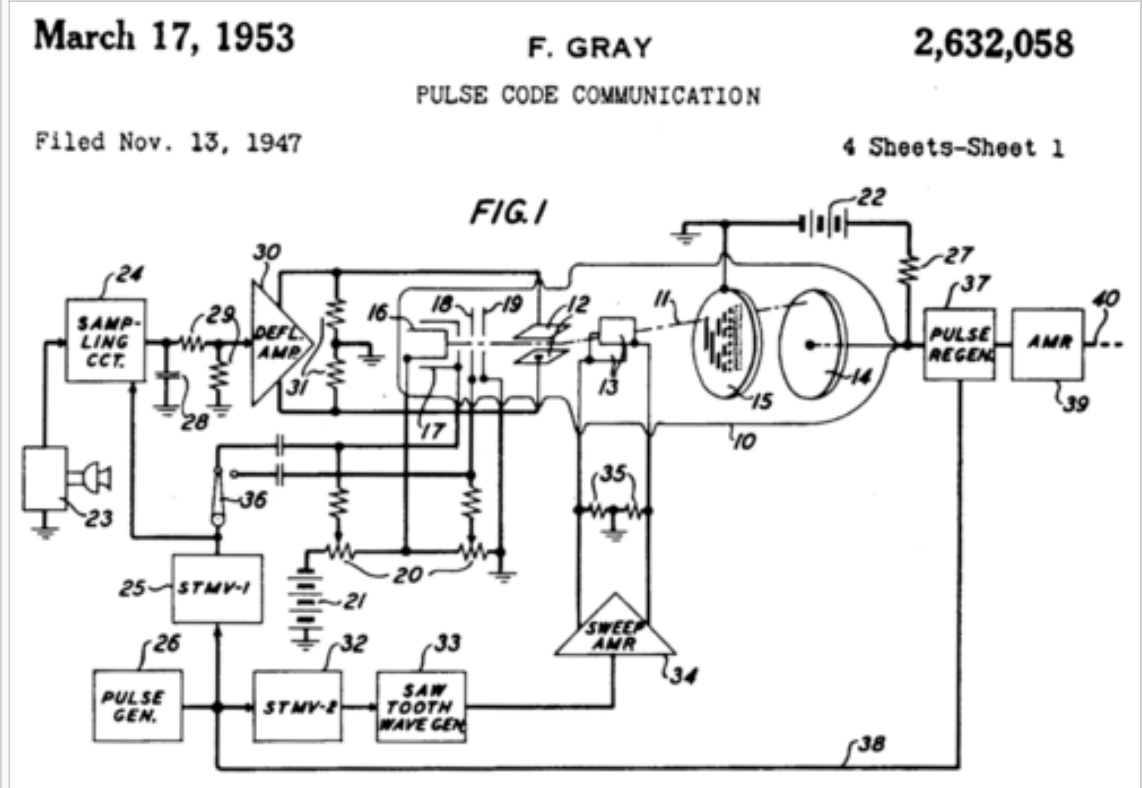
старшего бита, который просто инвертируется. Если же разделить каждую половину ещё раз пополам, свойство будет сохраняться для каждой из половин половины и т. д.

```
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000
```

Код получил имя исследователя лабораторий Bell Labs Фрэнка Грея. Он запатентовал и использовал этот код в своей импульсной системе связи (патент № 2632058).

Применения [править]

Использование кодов Грея основано прежде всего на том, что он минимизирует эффект ошибок при преобразовании аналоговых сигналов в цифровые (например, во многих видах датчиков).


March 17, 1953 F. GRAY 2,632,058

PULSE CODE COMMUNICATION

Filed Nov. 13, 1947 4 Sheets-Sheet 1

FIG. 1

Фрагмент главной страницы патента Грея

Коды Грея часто используются в датчиках-энкодерах. Их использование удобно тем, что два соседних значения шкалы сигнала отличаются только в одном разряде. Также они используются для кодирования номера дорожек в жёстких дисках.

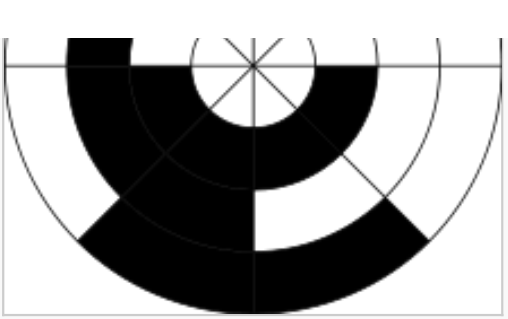
Код Грея можно использовать также и для решения задачи о Ханойских башнях .

Широко применяются коды Грея и в теории генетических алгоритмов для

кодирования генетических признаков, представленных целыми числами.

Код Грея используется для генерации сочетаний методом вращающейся двери[1]

В некоторых компьютерных играх (например, Duke Nukem 3D) для успешного прохождения уровня требуется подобрать нужную комбинацию положений нескольких переключателей. Для минимизации числа переключений при переборе вариантов следует использовать код Грея.



Круговой энкодер с трёхбитным кодом Грея

Алгоритмы преобразования [править]

Преобразование двоичного кода в код Грея [править]

Коды Грея легко получаются из двоичных чисел путём побитовой операции «Исключающее ИЛИ» с тем же числом, сдвинутым вправо на один бит. Следовательно, *i*-й бит кода Грея G_i выражается через биты двоичного кода B_i следующим образом:

$$G_i = B_i \oplus B_{i+1},$$

где $\oplus$ – операция «исключающее ИЛИ»; биты нумеруются справа налево, начиная с младшего.

Ниже приведён алгоритм преобразования из двоичной системы счисления в код Грея, записанный на языке C:

```c
unsigned int grayencode(unsigned int g)
{
    return g ^ (g >> 1);
}
```

Однако, необходимо помнить, что данный алгоритм будет работать правильно, если компилятор реализует НЕ циклический логический сдвиг (стандарт языка C не уточняет тип сдвига). Тот же самый алгоритм, записанный на языке Паскаль:

```pascal
function BinToGray(b: integer): integer;
begin
  BinToGray := b xor (b shr 1)
end;
```

Пример: преобразовать двоичное число 10110 в код Грея.

```
10110
01011
-----
```

```
11101
```

Преобразование кода Грея в двоичный код [править]

Обратный алгоритм – преобразование кода Грея в двоичный код – можно выразить рекуррентной формулой

$$B_i = B_{i+1} \oplus G_i,$$

причём преобразование осуществляется побитно, начиная со старших разрядов, и значение B_{i+1}, используемое в формуле, вычисляется на предыдущем шаге алгоритма. Действительно, если подставить в эту формулу вышеприведённое выражение для *i*-го бита кода Грея, получим

$$B_i = B_{i+1} \oplus G_i = B_{i+1} \oplus (B_i \oplus B_{i+1}) = B_i \oplus (B_{i+1} \oplus B_{i+1}) = B_i \oplus 0 = B_i.$$

Однако приведённый алгоритм, связанный с манипуляцией отдельными битами, неудобен для программной реализации, поэтому на практике используют видоизменённый алгоритм:

$$B_k = \bigoplus_{i=k}^{N} G_i,$$

где *N* – число битов в коде Грея (для увеличения быстродействия алгоритма в качестве *N* можно взять номер старшего ненулевого бита кода Грея); знак $\bigoplus$ означает суммирование при помощи операции «исключающее ИЛИ», то есть

$$\bigoplus_{i=k}^{N} G_i = G_k \oplus G_{k+1} \oplus \ldots \oplus G_{N-1} \oplus G_N.$$

Действительно, подставив в формулу выражение для *i*-го бита кода Грея, получим

$$B_k = \bigoplus_{i=k}^{N} G_i = \bigoplus_{i=k}^{N} (B_i \oplus B_{i+1}) = (B_k \oplus B_{k+1}) \oplus (B_{k+1} \oplus B_{k+2}) \oplus \ldots \oplus (B_{N-1} \oplus B_N) \oplus (B_N \oplus B_{N+1}) =$$
$$= B_k \oplus (B_{k+1} \oplus B_{k+1}) \oplus \ldots \oplus (B_N \oplus B_N) \oplus B_{N+1} = B_k \oplus B_{N+1} = B_k$$

Здесь предполагается, что бит, выходящий за рамки разрядной сетки (B_{N+1}), равен нулю.

Ниже приведена функция на языке C, реализующая данный алгоритм. Она осуществляет последовательный сдвиг вправо и суммирование исходного двоичного числа, до тех пор, пока очередной сдвиг не обнулит слагаемое.

```
unsigned int graydecode(unsigned int gray)
{
    unsigned int bin;
    for (bin = 0; gray; gray >>= 1) {
```

```
        bin ^= gray;
    }
    return bin;
}
```

Тот же самый алгоритм, записанный на языке Паскаль:

```
function GrayToBin(b: integer): integer;
 var g: integer;
begin
  g := 0;
  while b > 0 do begin
    g := g xor b;
    b := b shr 1;
  end;
  GrayToBin := g;
end;
```

Пример: преобразовать код Грея 11101 в двоичный код.

```
11101
01110
00111
00011
00001
-----
10110
```

Быстрое преобразование 8/16/24/32-разрядного значения кода Грея в двоичный код на языке BlitzBasic:

```
Function GRAY_2_BIN%(X%)
Return X Xor ((X And $88888888) Shr 3) Xor ((X And $CCCCCCCC) Shr 2) Xor ((X And $EEEEEEEE) S
End Function
```

Простой способ преобразования двоичного числа в код Грея выполняется по правилу: старший разряд записывается без изменения, каждый следующий символ кода Грея нужно инвертировать, если в натуральном коде перед этим была получена «1», и оставить без изменения, если в натуральном коде был получен «0».

Генерация кодов Грея [править]

Код Грея для n бит может быть рекурсивно построен на основе кода для n–1 бит путём переворачивания списка бит (то есть записыванием кодов в обратном порядке), конкатенации исходного и перевёрнутого списков, дописывания нулей в начало каждого кода в исходном списке и единиц — в начало кодов в перевёрнутом списке. Так, для генерации списка для n = 3 бит на основании кодов для двух бит необходимо выполнить следующие шаги:

Коды для n = 2 бит:	00, 01, 11, 10	
Перевёрнутый список кодов:		10, 11, 01, 00
Объединённый список:	00, 01, 11, 10	10, 11, 01, 00
К начальному списку дописаны нули:	000, 001, 011, 010	10, 11, 01, 00
К перевёрнутому списку дописаны единицы:	000, 001, 011, 010	110, 111, 101, 100

Ниже представлен один из алгоритмов создания последовательности кода Грея заданной глубины, записанный на языке Perl:

```perl
  my $depth = 16; # generate 16 Gray codes, 4 bits wide each
  my @gray_codes = ( '0', '1' );
  while(scalar(@gray_codes)<$depth)
     {
     my @forward_half=map{'0'.$_} @gray_codes;
     my @reverse_half=map{'1'.$_} reverse(@gray_codes);
     @gray_codes=(@forward_half,@reverse_half);
     }
```

Рекурсивная функция построение кода Грея на языке C:

```c
//n -- требуемая длина кода,
//m -- указатель на массив, способный хранить
// все коды Грея, длиной до n
// (должен быть выделен до вызова функции)
//depth -- параметр рекурсии

int gray (int n, int* m, int depth)

{
        int i, t = (1 << (depth - 1));

        if (depth == 0)
                m[0] = 0;

        else {
        //массив хранит десятичные записи двоичных слов
                for (i = 0; i < t; i++)
```

```
                        m[t + i] = m[t - i - 1] + (1 << (depth - 1));
        }
        if (depth != n)
                gray(n, m, depth + 1);

        return 0;
}
```

Быстрое преобразование 8/16/24/32-разрядного бинарного кода в код Грея на языке BlitzBasic:

```
Function BIN_2_GRAY%(X%)
Return X Xor ((X And $EEEEEEEE) Shr 1)
End Function
```

См. также [править]

- Код Хемминга
- Код Джонсона

Примечания [править]

1. ↑ Кнут, Дональд, Э. Искусство программирования, том 4, выпуск 3: генерация всех сочетаний и разбиений (раздел 7.2.1.3): Пер. с англ. - М.: ООО "И.Д. Вильямс", 2007. - 208 с. : ил.]

Библиография [править]

- Black, Paul E. *Gray code*. 25 февраля 2004. NIST. [1] (англ.).

Ссылки [править]

- NIST Dictionary of Algorithms and Data Structures: Gray code (англ.)
- Коды Грея

Категории: Электроника | Комбинаторика | Теория кодирования