RLE кодирование своими руками (VRLE8)

Структура:

Было: **0**, **0**, **0**, **0**, **0**, **1**, **2**, **3**, **0**, **0** - **До кодирования** (длина **10** байт).
Стало: **5**, **0**, **131**, **1**, **2**, **3**, **2**, **0**, **0** - **После кодирования** (длина **9** байт).

То, что стало в **hex** виде (закодированные байты):

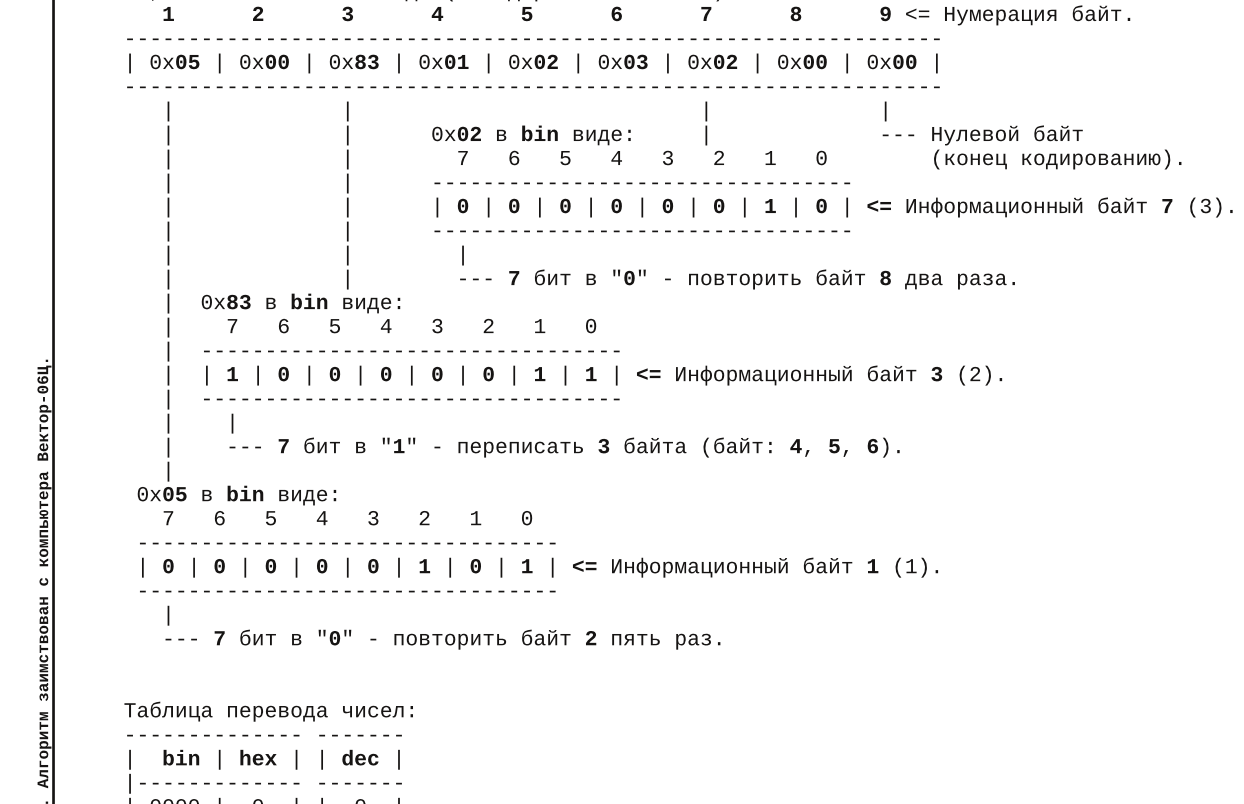
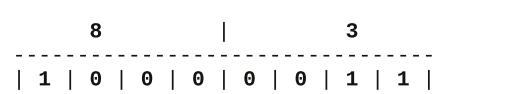


Таблица перевода чисел:

bin	hex	dec
0000	0	0
0001	1	1
0010	2	2
0011	**3**	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	**8**	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Пример перевода числа 0x**83** (байт) в двоичную систему.



83 hex = **10000011 bin**.
Смотрите таблицу (для **8-ки** и для **3-ки**).

bin - Двоичная система счисления.
hex - Шестнадцатеричная система счисления.
dec - Десятичная система счисления.

Реализация (на Си):

```
//
// VRLE8!
// Для кодирования используется один байт.
// Этот же байт используется и для декодирования.
// Это информационный байт!
//
// Что содержит информационный байт:
//
//    Значение 7-го бита "1" в этом байте означает о необходимости
//    переписать цепочку неповторяющихся байт, которые идут после этого байта,
//    количество переписываемых байт определяется битами: 6, 5, 4, 3, 2, 1, 0.
//
//    Значение 7-го бита "0" - означает о необходимости повторить следующий байт,
//    количество повторов определяется битами 6, 5, 4, 3, 2, 1, 0.
//
```

Вёрстка материала и Авторское право: Демидов С.В. RLE кодирование своими руками (VRLE8). Лицензия: Free. Алгоритм заимствован с компьютера Вектор-06Ц.

Страничка автора на https://drilnet.github.io. Украина, 64250, Харьковская обл., Балаклейский р-н, пос. Донец, Демидов С.В. E-mail: stdls[yxo]mail.ru, bbarsinov[yxo]gmail.com.

Вёрстка материала и Авторское право: Демидов С.В. RLE кодирование своими руками (VRLE8). Лицензия: Free. Алгоритм заимствован с компьютера Вектор-06Ц.

Страничка автора на https://drilnet.github.io. Украина, 64250, Харьковская обл., Балаклейский р-н, пос. Донец, Демидов С.В. E-mail: stdls[yxo]mail.ru, bbarsinov[yxo]gmail.com.

```
// Массивы:
//    ---
//    | array_in[10] - входной массив с байтами;
//    | array_out[20] - выходной массив с байтами (сюда кодируем);
//    | array_out_2[10] - выходной массив с байтами 2 (сюда декодируем).
//    ---
//
// Переменная long int:
//    ---
//    | array_in_length - здесь должна быть указана длина входного массива array_in.
//    ---
//
// Алгоритм заимствован с компьютера Вектор-06Ц.
// Украина. (C) Демидов С.В.
//

#include <stdio.h>

void main()
{

// -

        // Массив с входными байтами.
        unsigned char array_in[10] = {

//              0, 0, 0, 0, 0, 0, 0, 0, 0, 0 //
//              1, 2, 3, 4, 5, 6, 7, 8, 9, 0 // Некоторые варианты (для проверки).
//              0, 0, 0, 0, 0, 1, 2, 3, 4, 5 //
//              1, 2, 3, 4, 5, 0, 0, 0, 0, 0 //

                0, 0, 0, 0, 0, 1, 2, 3, 0, 0

                };

        // Массив с выходными байтами.
        // Сюда мы кодируем.
        unsigned char array_out[20];

        // Массив с выходными байтами 2.
        // Сюда мы декодируем.
        unsigned char array_out_2[10];

// -

        // Адрес в массиве array_in.
        unsigned long int addr_in;

        // Длина массива array_in.
        unsigned long int array_in_length;

        // Адрес в массиве array_out.
        unsigned long int addr_out;

        // Адрес в массиве array_out_2.
        unsigned long int addr_out_2;

        // Начальный адрес участка с неповторяющимися байтами.
        unsigned long int addr_in_start_0;

        // Конечный адрес участка с неповторяющимися байтами.
        unsigned long int addr_in_end_0;

        // Начальный адрес участка с повторяющимися байтами.
        unsigned long int addr_in_start_1;

        // Конечный адрес участка с повторяющимися байтами.
        unsigned long int addr_in_end_1;

        // Переключатель указывающий на неповторяющиеся байты, и на повторяющиеся.
        // sw = 0 - неповторяющиеся.
        // sw = 1 - повторяющиеся.
        unsigned char sw;

        // Для запоминания позиции в array_out.
        unsigned long int addr_out_pos;

        // Сколько неповторяющихся байт и сколько повторяющихся байт.
        unsigned char C;

        // Сколько всего закодированных данных (в байтах).
```

```
        unsigned long int sizerle8;

        // При декодировании, какие неповторяющиеся и какие повторяющиеся
        // bit_s = 128 - неповторяющиеся, bit_s = 0 - повторяющиеся.
        unsigned char bit_s;

        // Счётчик.
        unsigned char cb;

// -

        //
        // -----------------
        // | Закодировать. |
        // -----------------
        //

        // Длина массива array_in.
        array_in_length = 10;

        // Адрес в массиве array_in.
        addr_in = 0;

        // Адрес в массиве array_out.
        addr_out = 0;

        // Сколько всего закодированных данных.
        sizerle8 = 0;

        if (array_in_length == 1)
                {

                //
                // Массив с одним байтом.
                //

                C = 1;

                // Установить значение 7-го бита в "1", остальные оставить без изменений.
                array_out[addr_out++] = C | 0x80;

                // Записать один байт.
                array_out[addr_out++] = array_in[addr_in];

                // Записать 0 (конец упаковки).
                array_out[addr_out++] = 0;

                // Считаем закодированные данные (всего три байта).
                sizerle8++;
                sizerle8++;
                sizerle8++;

                }
                else
                {

                //
                // Массив больше одного байта.
                // Минимальная длина массива 2 байт!
                //

                // Определяем с чего начнём кодирование.
                if (array_in[addr_in] == array_in[addr_in + 1])
                        {
                        sw = 1; // Начнём с повторяющихся.
                        }
                        else
                        {
                        sw = 0; // Начнём с неповторяющихся.
                        }

                // Поехали!
                while (addr_in < array_in_length)
                {

                ////////////////////////////
                // Неповторяющиеся байты. //
                ////////////////////////////

                if (sw == 0)
                        {
```

Вёрстка материала и Авторское право: Демидов С.В. RLE кодирование своими руками (VRLE8). Лицензия: Free. Алгоритм заимствован с компьютера Вектор-06Ц.

Страничка автора на https://drilnet.github.io. Украина, 64250, Харьковская обл., Балаклейский р-н, пос. Донец, Демидов С.В. E-mail: stdls[yxo]mail.ru, bbarsinov[yxo]gmail.com.

```
                        // Начальный адрес участка с неповторяющимися байтами.
                        addr_in_start_0 = addr_in;

                        // Находим конечный адрес неповторяющихся байтов (находим на одном участке).
                        for (addr_in = addr_in; addr_in < array_in_length; addr_in++)
                                {
                                // +1 не выходит за границу массива array_in.
                                if (addr_in != array_in_length - 1)
                                        {
                                        if (array_in[addr_in] == array_in[addr_in + 1]) // +1.
                                                {
                                                // Для проверки.
//                                              printf("%s\n", "Прервано, начались повторяющиеся.");

                                                addr_in--;

                                                // Конечный адрес участка с неповторяющимися байтами.
                                                addr_in_end_0 = addr_in;

                                                //////////
                                                sw = 1; // Переключиться.
                                                //////////

                                                // Выйти.
                                                break;
                                                }
                                        }
                                        else
                                        {
                                        // Конечный адрес участка с неповторяющимися байтами.
                                        addr_in_end_0 = addr_in;

                                        // Выйти.
                                        break;
                                        }
                                } // Конец for.

                        C = 0;

                        // Запомнить позицию.
                        addr_out_pos = addr_out;

                        // Шаг вперёд.
                        addr_out++;

                        // Считаем закодированные данные (плюс один байт).
                        sizerle8++;

                        //
                        // Кодируем неповторяющиеся байты!
                        //
                        for (addr_in_start_0 = addr_in_start_0;
                                addr_in_start_0 < addr_in_end_0 + 1; addr_in_start_0++)
                                {

                                // Переписываем неповторяющиеся байты.
                                array_out[addr_out++] = array_in[addr_in_start_0];

                                C++;

                                // Считаем закодированные данные.
                                sizerle8++;

                                // Для проверки.
//                              printf("%s %u\n", "Размер \"порции\" (неповторяющиеся байты) >", C);

                                // Максимальная "порция" 128 байт.
                                if (C == 127)
                                        {
                                        // Записываем сколько переписанных байтов.
                                        // Установить значение 7-го бита в "1", остальные оставить без
изменений.
                                        array_out[addr_out_pos] = C | 0x80;

                                        // Не запоминать позицию и не делать шаг вперёд,
                                        // если все неповторяющиеся байты были переписаны.
                                        if (addr_in_start_0 != addr_in_end_0)
                                                {
                                                // Запомнить позицию.
                                                addr_out_pos = addr_out;
```

Вёрстка материала и Авторское право: Демидов С.В. RLE кодирование своими руками (VRLE8). Лицензия: Free. Алгоритм заимствован с компьютера Вектор-06Ц.

Страничка автора на https://drilnet.github.io. Украина, 64250, Харьковская обл., Балаклейский р-н, пос. Донец, Демидов С.В. E-mail: stdls[yxo]mail.ru, bbarsinov[yxo]gmail.com.

```
                                        // Шаг вперёд.
                                        addr_out++;

                                        // Считаем закодированные данные (плюс один байт).
                                        sizerle8++;
                                        }

                                C = 0;
                                }
                        } // Конец for.

                // Если C == 0, значит не записывать (уже было записано).
                if (C != 0)
                        {
                        // Записываем сколько переписанных байтов.
                        // Установить значение 7-го бита в "1", остальные оставить без изменений.
                        array_out[addr_out_pos] = C | 0x80;
                        }

                // Для проверки.
//              printf("%s %u\n", "Конечный адрес участка с неповторяющимися байтами (позиция): ",
addr_in_end_0);

                }
                else
                {

                //////////////////////////
                // Повторяющиеся байты. //
                //////////////////////////

                // Начальный адрес участка с повторяющимися байтами.
                addr_in_start_1 = addr_in;

                // Находим конечный адрес повторяющихся байтов (находим на одном участке).
                for (addr_in = addr_in; addr_in < array_in_length; addr_in++)
                        {
                        // +1 не выходит за границу массива array_in.
                        if (addr_in != array_in_length - 1)
                                {
                                if (array_in[addr_in] != array_in[addr_in + 1]) // +1.
                                        {
                                        // Для проверки.
//                                      printf("%s\n", "Прервано, начались неповторяющиеся.");

                                        // Конечный адрес участка с повторяющимися байтами.
                                        addr_in_end_1 = addr_in;

                                        //////////
                                        sw = 0; // Переключиться.
                                        //////////

                                        // Выйти.
                                        break;
                                        }
                                }
                                else
                                {
                                // Конечный адрес участка с повторяющимися байтами.
                                addr_in_end_1 = addr_in;

                                // Выйти.
                                break;
                                }
                        } // Конец for.

                C = 0;

                //
                // Кодируем повторяющиеся байты!
                //
                for (addr_in_start_1 = addr_in_start_1;
                        addr_in_start_1 < addr_in_end_1 + 1; addr_in_start_1++)
                        {

                        C++;

                        // Для проверки.
//                      printf("%s %u\n", "Размер \"порции\" (повторяющиеся байты) >", C);

                        // Максимальная "порция" 128 байт.
```

Вёрстка материала и Авторское право: Демидов С.В. RLE кодирование своими руками (VRLE8). Лицензия: Free. Алгоритм заимствован с компьютера Вектор-06Ц.

Страничка автора на https://drilnet.github.io. Украина, 64250, Харьковская обл., Балаклейский р-н, пос. Донец, Демидов С.В. E-mail: stdls[yxo]mail.ru, bbarsinov[yxo]gmail.com.

```
                                if (C == 127)
                                        {
                                        // Записываем сколько раз повторить.
                                        array_out[addr_out++] = C;
                                        // Записываем что повторить.
                                        array_out[addr_out++] = array_in[addr_in];

                                        // Считаем закодированные данные (плюс один байт).
                                        sizerle8++;
                                        // Считаем закодированные данные (плюс один байт).
                                        sizerle8++;

                                        C = 0;
                                        }

                                } // Конец for.

                        // Если C == 0, значит не записывать (уже было записано).
                        if (C != 0)
                                {
                                // Записываем сколько раз повторить.
                                array_out[addr_out++] = C;
                                // Записываем что повторить.
                                array_out[addr_out++] = array_in[addr_in];

                                // Считаем закодированные данные (плюс один байт).
                                sizerle8++;
                                // Считаем закодированные данные (плюс один байт).
                                sizerle8++;
                                }

                        // Для проверки.
//                      printf("%s %u\n", "Конечный адрес участка с повторяющимися байтами (позиция): ",
addr_in_end_1);

                        }

                // Сделать шаг вперёд.
                addr_in++;

                } // Конец while.

                // Записать 0 (конец упаковки).
                array_out[addr_out++] = 0;

                // Считаем закодированные данные (считать и 0).
                sizerle8++;

                }

// -

        //
        // ---------------------------------------------
        // | Показать что получилось после кодирования. |
        // ---------------------------------------------
        //

        // Было.
        printf("\n%s" , "Было: ");
        for ( addr_in = 0; addr_in < array_in_length; addr_in++)
                {
                printf("%u ", array_in[addr_in]);
                }
        printf("%s%u%s\n", "- ", array_in_length, " байт(а).");

        // Стало.
        printf("\n%s" , "Кодирование (стало): ");
        for (cb = 0; cb < addr_out; cb++)
                {
                printf("%u ", array_out[cb]);
                }
        printf("%s%u%s\n", "- ", cb, " байт(а).");

        // Для проверки.
//      printf("\n%s %u %s\n", "Размер закодированных данных:", sizerle8, "байт(а).");

// -

        //
        // ------------------
```


Вёрстка материала и Авторское право: Демидов С.В. RLE кодирование своими руками (VRLE8). Лицензия: Free. Алгоритм заимствован с компьютера Вектор-06Ц.

Страничка автора на https://drilnet.github.io. Украина, 64250, Харьковская обл., Балаклейский р-н, пос. Донец, Демидов С.В. E-mail: stdls[yxo]mail.ru, bbarsinov[yxo]gmail.com.

Вёрстка материала и Авторское право: Демидов С.В. RLE кодирование своими руками (VRLE8). Лицензия: Free. Алгоритм заимствован с компьютера Вектор-06Ц.

Страничка автора на https://drilnet.github.io. Украина, 64250, Харьковская обл., Балаклейский р-н, пос. Донец, Демидов С.В. E-mail: stdls[yxo]mail.ru, bbarsinov[yxo]gmail.com.

```
        // | Декодирование. |
        // ------------------
        //

        // Адрес в массиве array_out_2.
        addr_out_2 = 0;

        // Распаковка.
        for (addr_out = 0; addr_out < sizerle8; addr_out++)
                {

                // Если встретится 0, значит прервать цикл (конец распаковки).
                if (array_out[addr_out] == 0)
                        {
                        break;
                        }

                // Сбросить в "0" значение битов 6, 5, 4, 3, 2, 1, 0.
                // 7 бит оставить без изменений.
                bit_s = 128 & array_out[addr_out];

                if (bit_s == 128)
                        {
                        // Неповторяющиеся байты.
                        // ---

                        C = array_out[addr_out];
                        C = 127 & C; // 7F и C.

                        for (cb = 0; cb < C; cb++)
                                {
                                array_out_2[addr_out_2++] = array_out[++addr_out];
                                }
                        }
                        else
                        {
                        // Повторяющиеся байты.
                        // ---

                        C = array_out[addr_out++];

                        for (cb = 0; cb < C; cb++)
                                {
                                array_out_2[addr_out_2++] = array_out[addr_out];
                                }
                        }

                }

        printf("\n%s", "Декодирование: ");

        for (cb = 0; cb < addr_out_2; cb++)
                {
                printf("%u ", array_out_2[cb]);
                }

        printf("%s%u%s\n\n", "- ", cb, " байт(а).");

}
```

Этот материал доступен на **GitHub**: https://github.com/drilnet/rle

---

Логотип для алгоритма VRLE8 (в gray-цветах):

Версия логотипа: 07b