Лекция 17. Побитовые операции (bitwise operation)

Ревун Артем Леонидович
ст. преп. кафедры вычислительных систем

Курс «Программирование», весенний семестр, 2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)

Содержание

- Операции: ~, &, |, ^, >> , <<, &=, |=, ^=, >>=, <<=
- Обзор двоичной, восьмеричной и шестнадцатеричной систем счисления
- Два средства языка C для обработки отдельных битов значения: побитовые операции и битовые поля
- Ключевые слова: _Alignas, _Alignof

Машинный счёт (1)

Пример: 1234

$$1000 + 200 + 30 + 4$$

$$\Rightarrow 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

Число 1234 *по основанию 10*

Пример: 11011_2

$$10000 + 1000 + 10 + 1$$

$$\Rightarrow 1 \times 2^4 + 1 \times 2^3 + \cancel{0 \times 2^2} + 1 \times 2^1 + 1 \times 2^0$$

Число 11011 по основанию 2 (двоичное число)

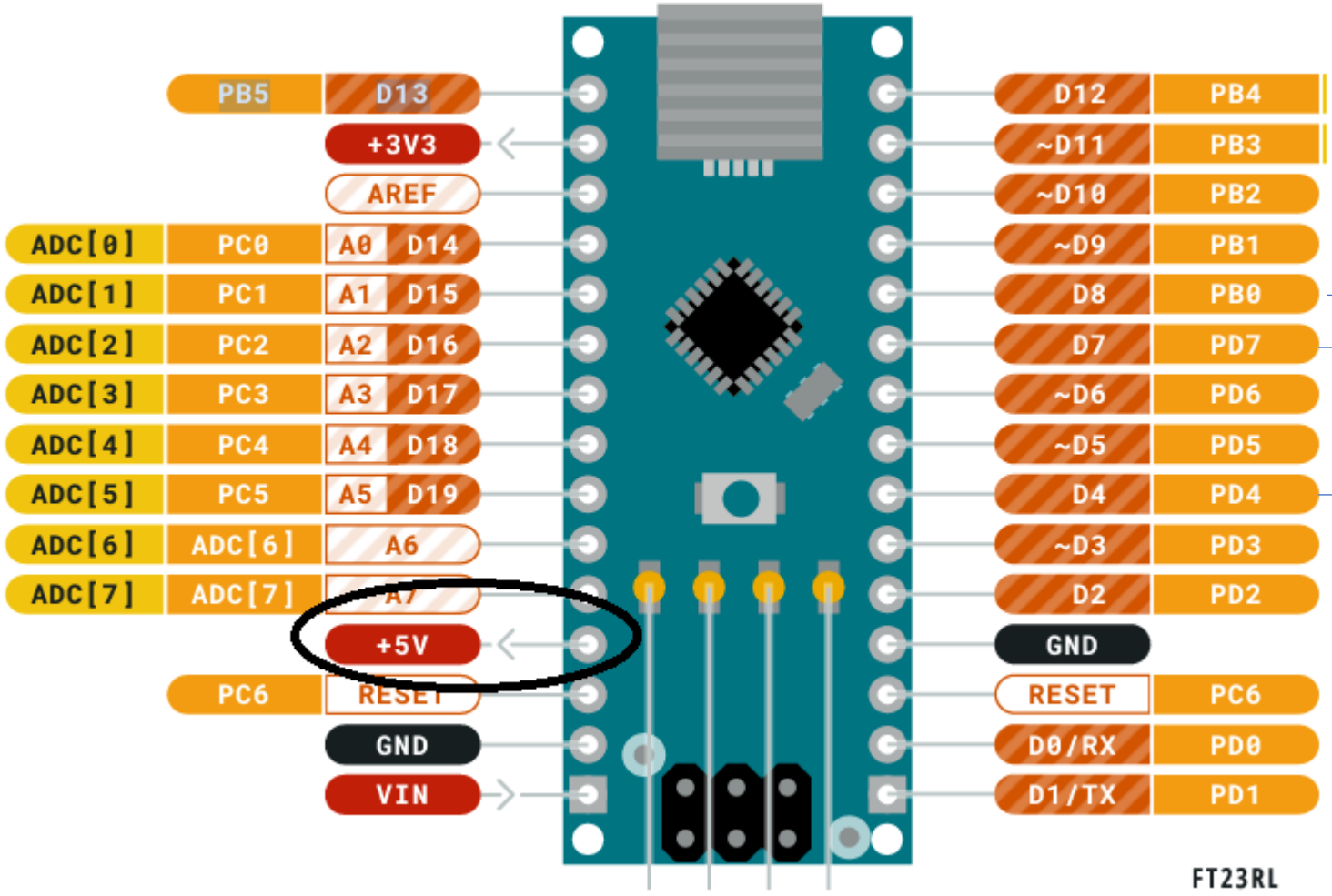
Машинный счёт (2)

В двоичной системе можно представить любое целое число (при достаточном количестве битов) в форме комбинации нулей и единиц. Эта система очень удобна для цифровых вычислительных систем, у которых информация выражается в виде комбинаций включенных (**1**) и выключенных (**0**) состояний, что можно интерпретировать как единицы нули.

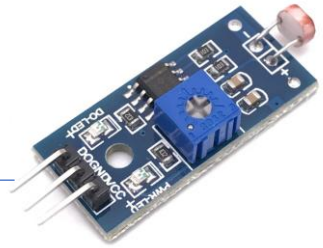
Обычно в электронике **логическая единица** 1-2,5 Вольт, **логический ноль** 0,5-0,7 Вольт.

Машинный счёт (3)

Arduino Nano

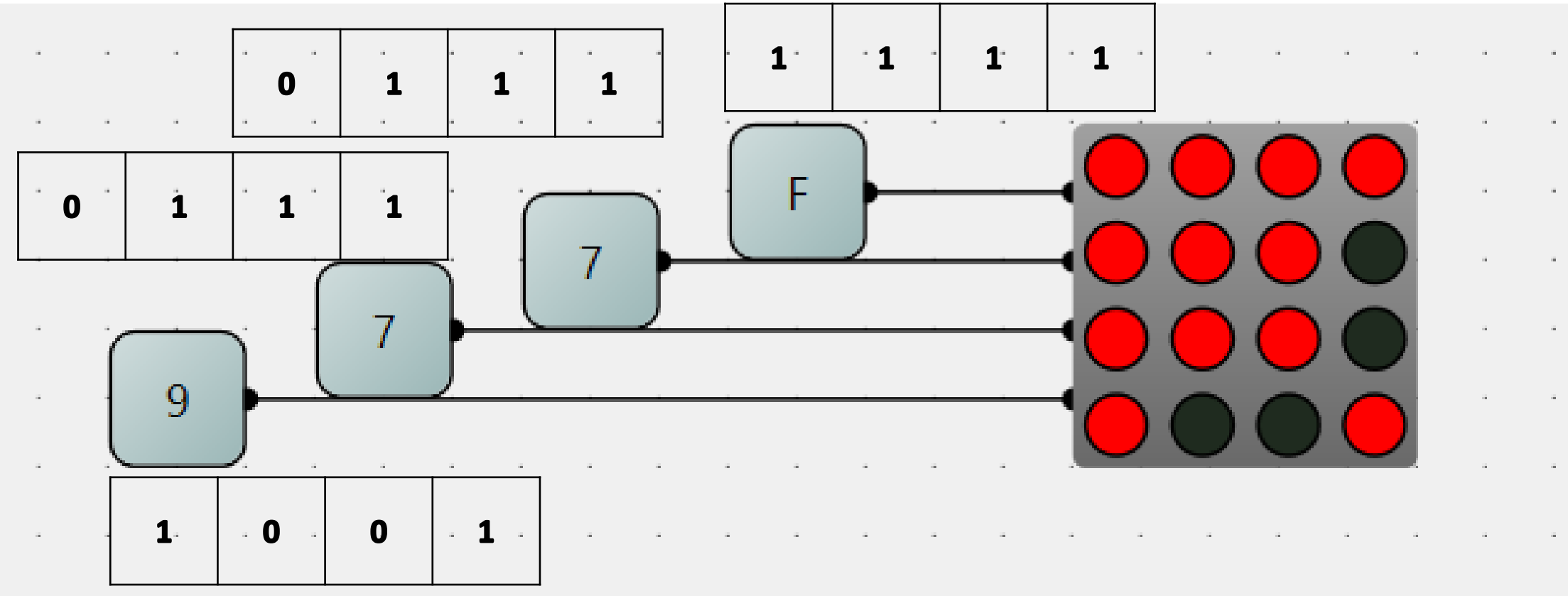


~3-5 Вольт

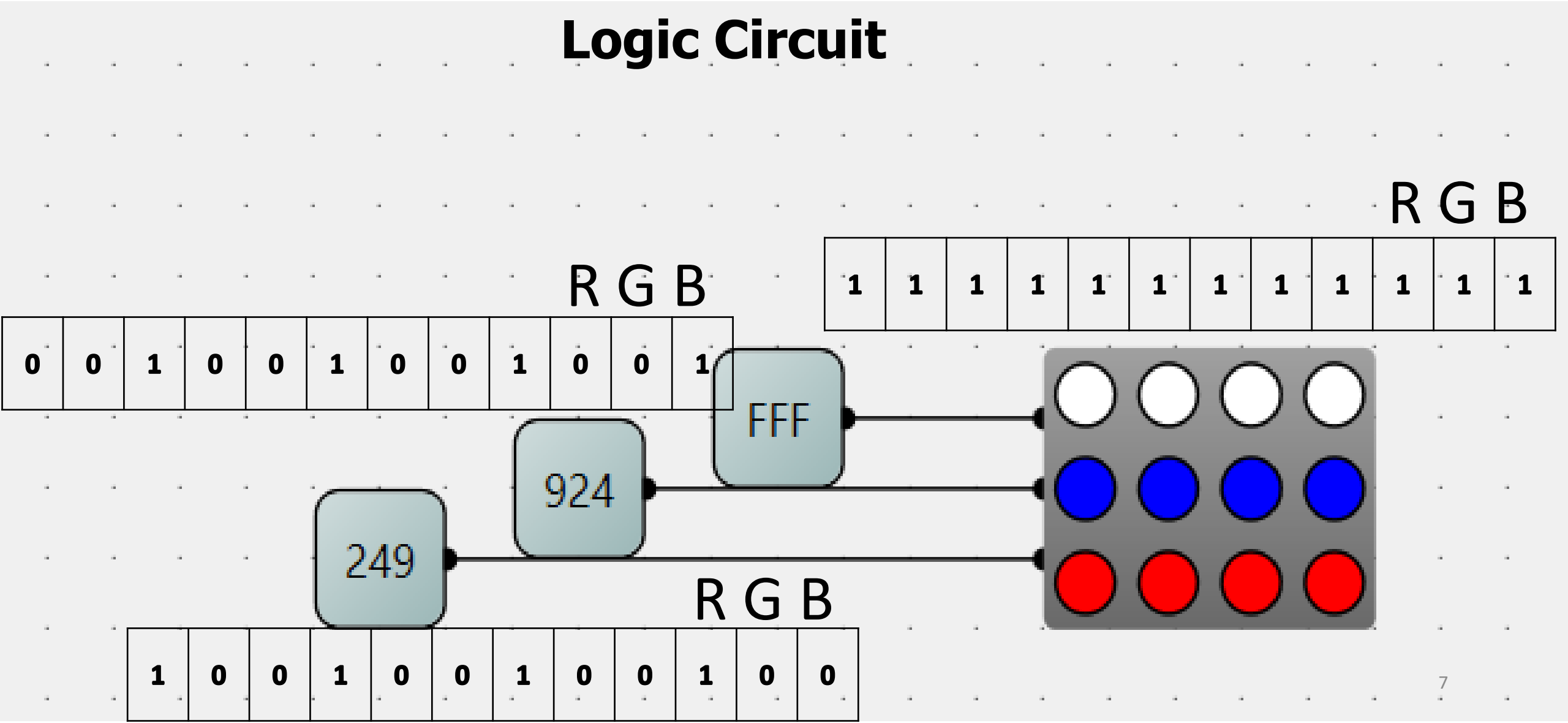


Машинный счёт (4)

Logic Circuit



Машинный счёт (5)



Двоичные целые числа (1)

В языке Си термин ***байт*** обозначает размер памяти, используемой для хранения *набора символов системы*, поэтому в Си байт может содержать 8, 9, 16 и др. кол-во битов.

Однако в характеристиках модулей памяти и систем передачи данных предполагается, что байт содержит 8 битов. Для ясности в мире вычислений 8-битовый байт часто обозначается ***октет***. Можно считать, что биты в байте пронумерованы справа налево от 0 до 7. Седьмой бит называется старшим, а нулевой бит — младший .

Двоичные целые числа (2)

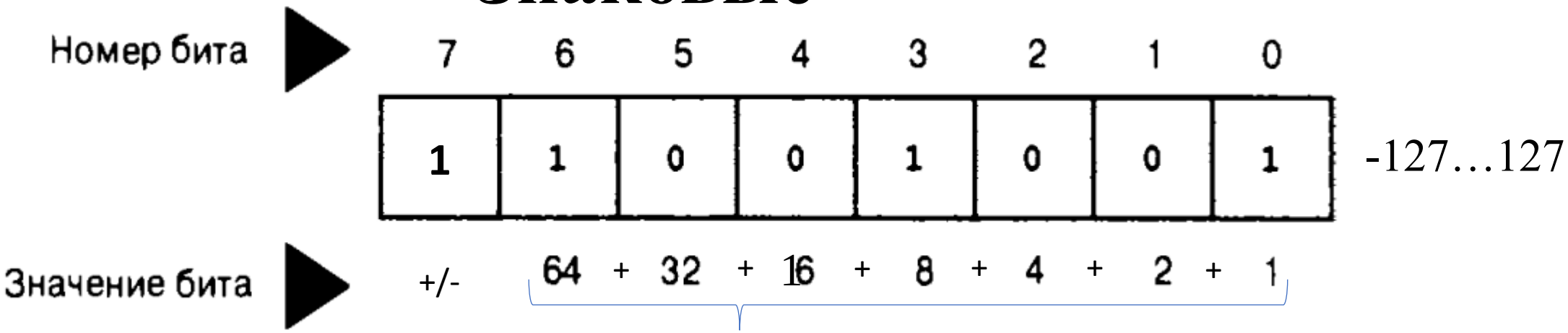
Без знаковые

В этом примере биты 6, 3 и 0 установлены в 1.

Значением этого байта является 64 + 8 + 1, или 73.

Двоичные целые числа (3)

Знаковые



119

0	1	1	1	0	1	1	1

-119

1	1	1	1	0	1	1	1

-0

1	0	0	0	0	0	0	0

0

0	0	0	0	0	0	0	0

Двоичные целые числа (4)

Метод *дополнения до двух* нивелирует эту неоднозначность. Пример: ***на однобайтовом числе.***

1	0	0	0	0	0	0	0	0	256
0	1	0	0	0	0	0	0	0	- 128
0	1	0	0	0	0	0	0	0	= -128

-127

1	0	0	0	0	0	0	1

-1

1	1	1	1	1	1	1	1

Двоичные целые числа (5)

Метод *дополнения до двух* нивелирует эту неоднозначность. Пример: ***на однобайтовом числе.***

По сути своей метод инвертирует число по битово и прибавляет к нему единицу.

0	**0**	**0**	**0**	**0**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**	**0**
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Число 1 записанное двоичной СС

Побитовый NOT

+1

-1

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Двоичные числа с плавающей запятой (1)

0.527 = 5/10 + 2/100 + 7/1000 – десятичная

Двоичные числа хранятся при помощи двоичной дроби и двоичной экспоненты. Знаменатели в двоичной дроби могут быть представлены только степенями двойки.

1/2 + 0/4 + 1/8 = 0.50 + 0.00 + 0.125 = 0.625

Точнее всего, можно составить комбинации кратные степеням 1/2, а менее точно например 1/3 и 2/5.

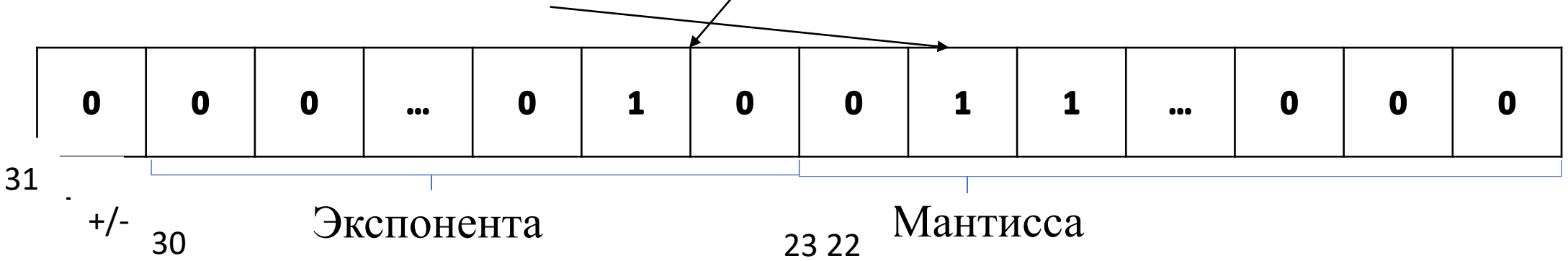
Двоичные числа с плавающей запятой (2)

Допустим возьмем число 5.5:

Целая часть $5_{10} = 101_2$

Дробная часть $0.5 = 0.1_2$

$5.5 = 101.1 \rightarrow 1.011 * 2^2$



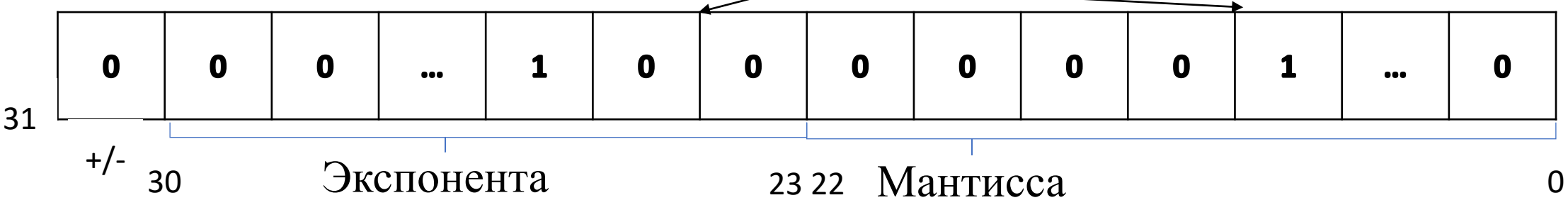
Двоичные числа с плавающей запятой (3)

Допустим возьмем число 5.5 * 3 = 16,5:

Целая часть $16_{10} = 10000_2$

Дробная часть $0.5 = 0.1_2$

$16.5 = 10000.1 \rightarrow 1.00001 * 2^4$



Восьмеричная СС – ОСТ (1)

Восьмеричная цифра	Двоичный эквивалент
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Шестнадцатеричная СС - HEX (1)

Десятичное число	Шестнадцатеричная цифра	Двоичный эквивалент
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Битовые поля в структурах (1)

Один из методов *манипулирования битами*, предполагает создание в структуре *битовых полей*. Это набор соседствующих битов внутри значения типа `signed int` или `unsigned int` (C99 и C11 добавляют такую возможность для `_Bool`).

Битовое поле создаётся в структуре при помощи **оператора :** при этом должно быть размечено каждое поле и определен его размер.

Битовые поля в структурах (2)

```
struct {
    unsigned int autfd : 1;
    unsigned int bldfc : 1;
    unsigned int undln : 1;
    unsigned int itals : 1;
} prnt;
prnt.itals = 0;
prnt.undln = 1 ;
```

Битовые поля в структурах (3)

```
struct {
unsigned int code1 : 2;
unsigned int code2 : 2;
unsigned int code3 : 8;
} prcode;
prcode.code1 = 0; //2^2 = (0..3)
prcode.code2 = 3; //2^2 = (0..3)
prcode.code3 = 102; //2^8 = (0..255)
```

Битовые поля в структурах (4)

```
struct {
    unsigned int autfd : 31;
    unsigned int bldfc : 1;
    unsigned int undln : 16;
    unsigned int itals : 16;
} prnt;
prnt.bldfc = 0;
prnt.undln = 65535 ;
```

Битовые поля в структурах (4)

```
struct {
    unsigned int autfd : 31;
    unsigned int bldfc : 1;
    unsigned int undln : 16;
    unsigned int itals : 16;
} prnt;
prnt.bldfc = 0;
prnt.undln = 65535 ;
```

Отдельное поле не должно перекрывать границу между двумя смежными областями *unsigned int*. Иначе компилятор сдвинет такое поле в новый *unsigned int*.

Битовые поля в структурах (5)

```
struct {
    unsigned int field1 : 1;
    unsigned int : 2;
    unsigned int field2 : 1;
    unsigned int : 0;
    unsigned int field3 : 1;
} stuff;
```

Примеры структур (1)

Представить свойства выводимого на экран окна.

- Окно может быть прозрачным или непрозрачным.
- Цвет фона выбирается из следующей палитры: черный, красный, зеленый, желтый, синий, пурпурный, голубой и белый.
- Рамка может быть скрыта или отображена.
- Цвет рамки выбирается из той же палитры, что и цвет фона.
- Для рамки применяются три стиля линии: сплошная, пунктирная и штриховая.

Примеры структур (2)

Представить свойства выводимого на экран окна.

- Окно может быть прозрачным или непрозрачным. (1)
- Цвет фона выбирается из следующей палитры: черный, красный, зеленый, желтый, синий, пурпурный, голубой и белый. (3)
- Рамка может быть скрыта или отображена. (1)
- Цвет рамки выбирается из той же палитры, что и цвет фона. (3)
- Для рамки применяются три стиля линии: сплошная, пунктирная и штриховая. (2)

Примеры структур (3)

```
struct box_props {
    bool opaque : 1;
    unsigned int fill_color : 3;
    unsigned int : 4;
    bool show_border : 1;
    unsigned int border_color : 3;
    unsigned int border_style : 2;
    unsigned int : 2;
};
```

Примеры структур (4)

```
struct box_props {
    …
    unsigned int fill_color : 3;
    …
};
```

Комбинация битов	Десятичный эквивалент	Цвет
000	0	Черный
001	1	Красный
010	2	Зеленый
011	3	Желтый
100	4	Синий
101	5	Пурпурный
110	6	Голубой
111	7	Белый

Примеры структур (5)

```
#define SOLID 0
#define DOTTED 1
#define DASHED 2
enum colors = {black, red, green, yellow, blue,
magenta, cyan, white};
struct box_props {
    …
};
struct box_props box = (true, yellow , true,
green, DASHED);
```

Примеры структур (6)

Исходные настройки окна:

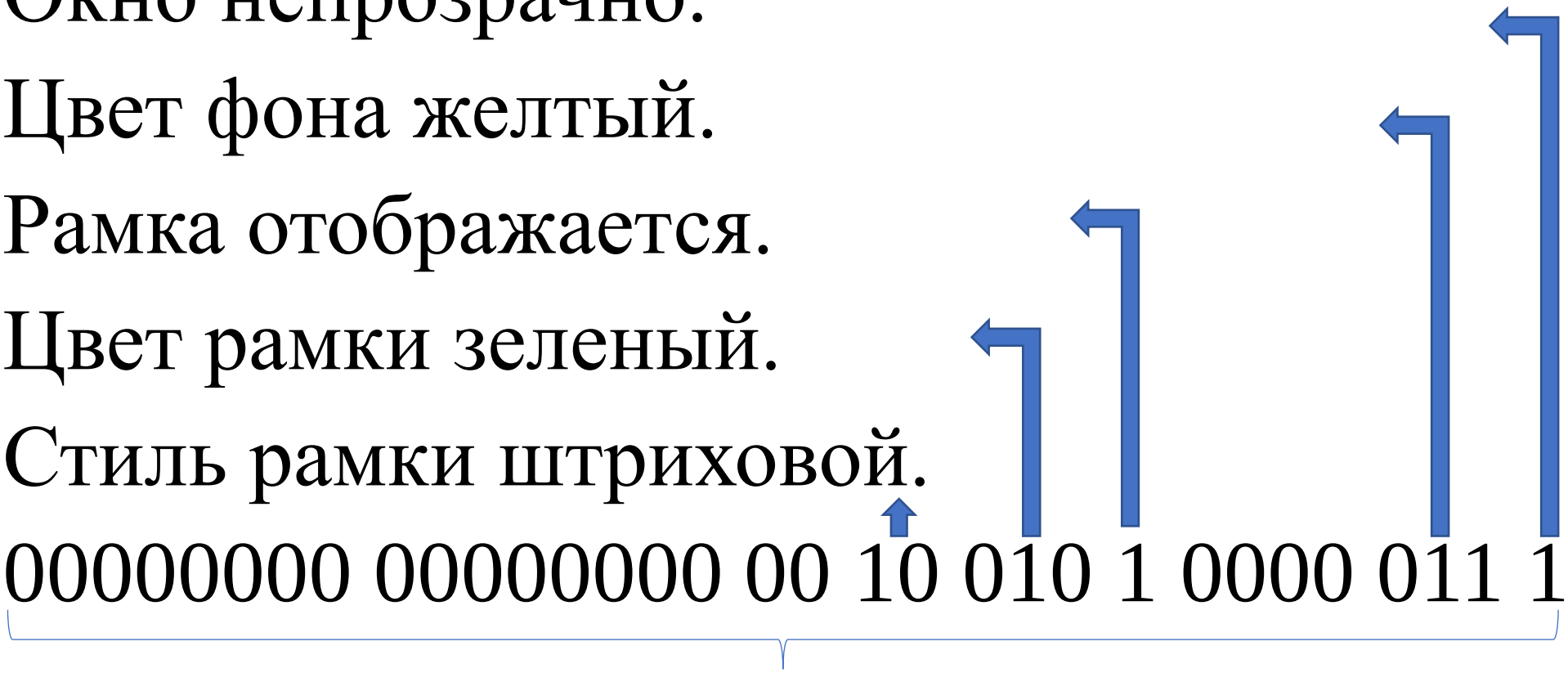
Окно непрозрачно.

Цвет фона желтый.

Рамка отображается.

Цвет рамки зеленый.

Стиль рамки штриховой.



Примеры структур (6)

Исходные настройки окна:

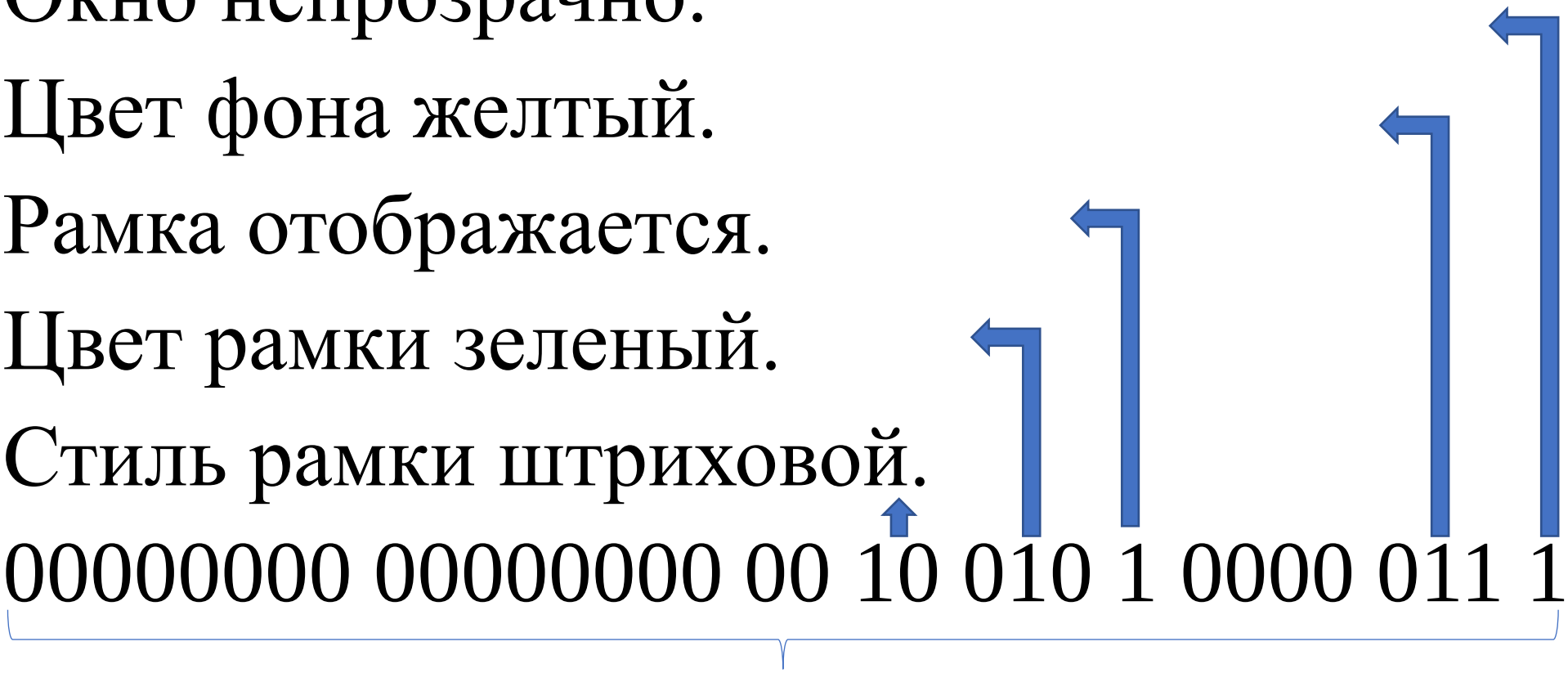
Окно непрозрачно.

Цвет фона желтый.

Рамка отображается.

Цвет рамки зеленый.

Стиль рамки штриховой.



Примеры другой структур (1)

```
typedef struct bitlaptop{
    unsigned int cpuType: 2;
    unsigned int ramV: 3;
    unsigned int hasGPU: 1;
    unsigned int osSet: 2;
    unsigned int storT: 2;
    unsigned int storV: 4;
    unsigned int screenS: 2;
} BITLAPTOP; //2byte struct
```

Примеры другой структур (2)

```
static char *cpu[] = {"Intel", "AMD", "Apple
Silicon"};
static const int ramVol[] = {8, 16, 32, 64};
static const char *os[] = {"Linux", "macOS",
"Windows"};
static const char *storageT[] = {"HDD", "SSD",
"SSD+NVMe"};
static const int storageV[] = {128, 256, 512,
1024, 2048};
static const char *screen[] = {"<14''",">14'' &&
<=15''",">15'' && <=16''",">=17''"};
```

Примеры другой структур (3)

```
void displayLaptopSpecs(const BITLAPTOP * bl) {
    printf("Тип процессора: %s\n", bl->cpuType < 3 ? cpu[bl->cpuType] : "Неизвестно");
    printf("Объем оперативной памяти: %d ГБ\n", bl->ramV < 5 ? ramVol[bl->ramV] : 0);
    printf("Наличие дискретного графического ускорителя: %s\n", bl->hasGPU ? "Есть" : "Нет");
    printf("Семейство ОС: %s\n", bl->osSet < 3 ? os[bl->osSet] : "Неизвестно");
    printf("Тип накопителя: %s\n", bl->storT < 3 ? storageT[bl->storT] : "Неизвестно");
    printf("Размер накопителя: %d ГБ\n", bl->storV < 5 ? storageV[bl->storV] : 0);
    printf("Размер экрана: %s дюймов\n", screen[bl->screenS]);}
```

Продолжение следует…
Спасибо за внимание

Ревун Артем Леонидович
ст. преп. кафедры вычислительных систем

Курс «Программирование», весенний семестр, 2024
Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)