

**Курс «Алгоритмы и алгоритмические языки»
1 семестр 2014/2015**

Лекция 10

Поразрядные операции

- ◇ & (поразрядное И)
- ◇ | (поразрядное включающее ИЛИ)
- ◇ ^ (поразрядное исключаящее ИЛИ)
- ◇ << (сдвиг влево)
- ◇ >> (сдвиг вправо)
 - ◆ Беззнаковое число – заполнение нулями
 - ◆ Знаковое число – заполнение значением знакового разряда («арифметический сдвиг») или нулями («логический сдвиг»)
- ◇ ~ (дополнение до 1, или *инверсия*)

Тип «степень множества» (булеан)

- ◇ Пусть U – множество. Множество всех подмножеств множества U называется *степенью множества U* .
- ◇ Пусть B – степень конечного множества U . Тогда $|B| = 2^{|U|}$.
- ◇ *Характеристической функцией χ_C* подмножества C множества U называется функция, принимающая значение 1 на элементах U , входящих в состав C , и значение 0 на остальных элементах U .
- ◇ Множества удобно задавать через их характеристические функции. При этом в зависимости от количества элементов базового множества U его характеристическая функция, кодированная битами целого типа, может иметь тип
 - `unsigned char` (если $|U| \leq 8$)
 - `unsigned int` (если $|U| \leq 32$)
 - `unsigned long long` (если $|U| \leq 64$)

Тип «степень множества»

◇ **Пример.** Пусть $U = \{r, o, y, g, c, b, v, w\}$. Тогда его подмножества задаются переменными типа `unsigned char`: $|B| = 2^{|U|}$

$\{\}$ задается значением 00000000,

$\{r, y, g\}$ – значением 10110000

$\{r, o, y, g, c, b, v, w\}$ – значением 11111111

буквы r, o, y, g, c, b, v, w являются первыми буквами семи цветов спектра и белого цвета:

r – *red*

c – *cyan*

o – *orange*

b – *blue*

y – *yellow*

v – *violet*

g – *green*

w – *white*

Реализация операций над множествами с помощью поразрядных операций

◇ $\&$ (поразрядное И) соответствует пересечению множеств:

$$B = \{r, y, g, b, w\}, C = \{r, o, y, g, v\}$$

$$\chi_B = 10110101, \chi_C = 11110010$$

$$\chi_B \& \chi_C = 10110101 \& 11110010 = 10110000$$

$$B \cap C = \{r, y, g\}$$

Реализация операций над множествами с помощью поразрядных операций

◇ \mid (поразрядное включающее ИЛИ) соответствует объединению множеств:

$$B = \{r, y, g, b, w\}, C = \{r, o, y, g, v\}$$

$$\chi_B = 10110101, \chi_C = 11110010$$

$$\chi_B \mid \chi_C = 10110101 \mid 11110010 = 11110111$$

$$B \cup C = \{r, o, y, g, b, v, w\}$$

Реализация операций над множествами с помощью поразрядных операций

◇ \sim (инверсия) соответствует дополнению до множества U :

$$B = \{r, y, g, b, w\}$$

$$\chi_B = 10110101$$

$$\chi_{\sim B} = 01001010$$

$$\sim B = \{o, c, v\}$$

Структуры

- ◆ Структура – это совокупность нескольких переменных, часто разных типов, сгруппированных под одним именем для удобства
 - ◆ Переменные, перечисленные в объявлении структуры, называются ее *полями*, *элементами*, или *членами*.

◆ **Объявление структуры:**

```
struct метка структуры { поля структуры } ;
```

```
struct point
```

```
{
```

```
    int x;
```

```
    int y;
```

```
} f, g;
```

```
struct point h, center = {32, 32};
```


Структуры

- ◆ Поля структуры могут иметь любой тип, например, тип массива или тип другой структуры

```
struct rect
{
    struct point pt1;
    struct point pt2;
};
```

- ◆ *Инициализация структуры:*

```
struct rect r = {.pt1 = {4, 4},
                .pt2 = {7, 6}};

/* Остальные элементы - нулевые */
struct rect r2 = {.pt2.x = 5};
```

- ◆ Размер структуры в общем случае **не равен** сумме размеров ее элементов (**выравнивание**)

Структуры

◇ Доступ к полям структуры: операция точка "."

◆ `f.x, g.y, r.pt1.x`

◇ Присваивание структур целиком: `f = g;`

◇ Массивы структур

```
#define NRECT 15
```

```
/* Первый прямоугольник вокруг 0, 0 */
```

```
struct rect rectangles[NRECT]
```

```
= {{-1, -1, 1, 1}};
```

```
/* Последний прямоугольник - большой */
```

```
#define BOUND 1024
```

```
struct rect bounded_rectangles[NRECT]
```

```
= {[NRECT-1] = {-BOUND, -BOUND,  
                BOUND,  BOUND}};
```

Указатели на структуры

◇ `struct rect r = {.pt1 = {4, 4},
 .pt2 = {7, 6}};`

`struct rect *pr = &r;`

◇ Доступ к полям структуры через указатель:

`pr->pt1 (= (*pr).pt1), pr->pt2.x`

◇ Адресная арифметика:

`struct rect *pr = &bounded_rectangles[0];`

`while (pr->pt1.x != -BOUND)`

`pr++;`