Sem 1.02.1 Основы логики

Презентация к семинару в рамках курса
«Программирование на Си»
Подготовил Кострицкий А. С.

Кафедра ИУ-7
МГТУ имени Н. Э. Баумана
Москва, Россия

Москва — 2023 — TS2302162137

Булево множество

$$\mathbb{B} = \{TRUE, FALSE\}$$

в языке Си отсутствовало изначально.

Правдивым считается любое целое ненулевое выражение, ложным — равное нулю.

Сколько унарных булевых операторов может существовать? Перечислите их.

Рассмотрим унарный оператор отрицания. В языке Си - «!».

$!1 = 0,$

$!0 = 1,$

$!42 = 0,$

$!(-42) = 0.$

Можно «привести» любое целое к булеву множеству из единицы и нуля с помощью двойного отрицания.

Когда будем описывать числа в двоичном коде, например, в таблицах истинности, будем использовать одну из двух систем:

1. Перечисленные по порядку числа: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.
2. Код Грея, в котором при переходе к рядом стоящему числу можно изменять только один флажок: 0000, 0001, 0011, 0010, 0110, 0111, 0101,0100,1100,1101,1111,1110, 1010,1011,1001,1000.

Бинарный ***логический*** оператор И:

A	B	$A\ AND\ B$
0	0	0
0	1	0
1	0	0
1	1	1

А как в Си выглядит таблица истинности для &&?

Бинарный ***логический*** оператор ИЛИ:

A	B	$A\ OR\ B$
0	0	0
0	1	1
1	0	1
1	1	1

А как в Си выглядит таблица истинности для $||$?

Сколько ещё бинарных булевых операторов можно назвать? Если вызывает трудности, попробуйте перечислить их в виде правых столбцов таблиц истинности, помня, что &&=0001, ||=0111.

Правила де Моргана:

1. Отрицание ИЛИ есть И отрицаний.
2. Отрицание И есть ИЛИ отрицаний.

При упоминании нестрогой модели вычислений (*англ. non-strict evaluation*) обычно имеют в виду, что аргументы не вычисляются до тех пор, пока их значение не используется в теле функции.
В ряде языков, в том числе и в Си, булевы выражения имеют нестрогий порядок вычисления, называемый в русской литературе «вычислениями по короткой схеме» (short-circuit evaluation), где вычисления прекращаются, как только результат становится однозначно известен — например, значение «истина» в операции дизъюнкции, «ложь» в операции конъюнкции, и так далее.

Именно поэтому в Си возможна волшебная конструкция вида

```
if(scanf("%d", &n) == 1 && n > 0)
    return EXIT_FAILURE;
```

и многие другие.

Существуют также языки без ленивой обработки булевых вычислений или с возможностью отключения таковой. О том, в какой именно момент неудачливый программист узнаёт об отсутствии в языке ленивой логики, говорить не приходится.

Везде, где необходимо логическое, лучше приводить целое к элементу подмножества из единицы и нуля. Определение нечётности

```
odd = ((n % 2) == 1);
```

много лучше, чем

```
odd = (n % 2);
```

поскольку во флажке заведомо остаётся в первом случае либо единица, либо нуль.

1. Сколько булевых функций одного аргумента существует?
2. Сколько булевых функций двух аргументов существует?
3. Почему при существовании такого большого количества функций мы используем только три?
4. Сколько булевых функций n аргументов существует?
5. Что произойдёт...

```
#define TRUE 1
#define FALSE 0
...
n = 27
...
if(n == TRUE)
...
```