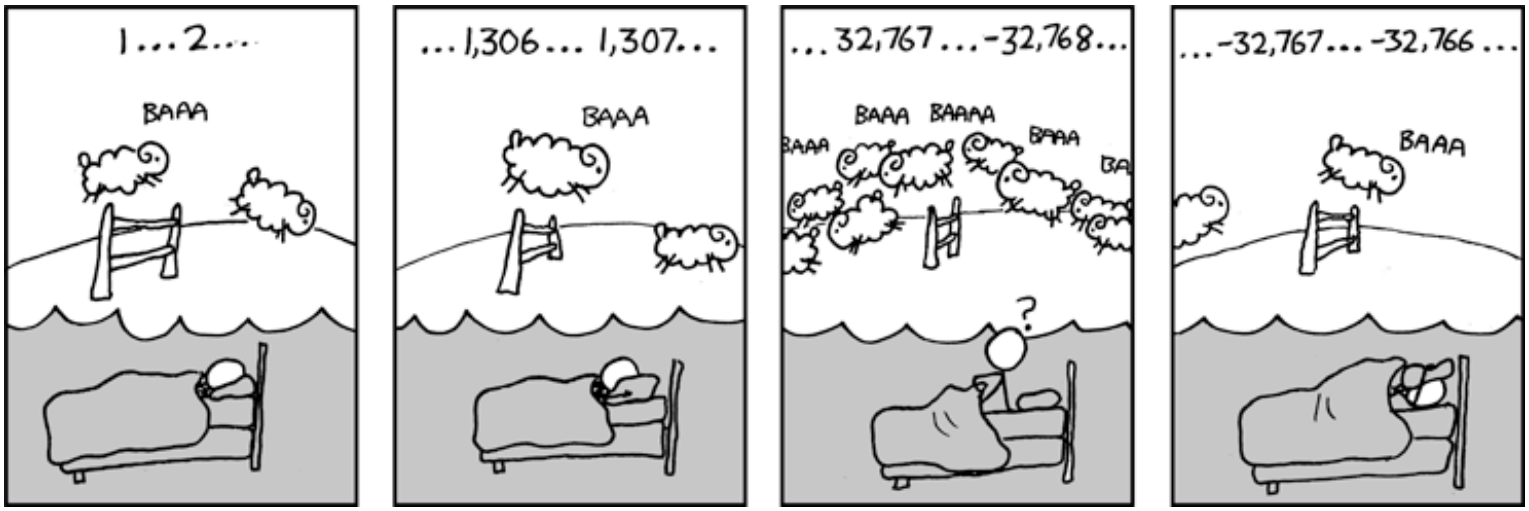
Основы программирования

Лекция № 2, 3 марта 2016 г.



http://xkcd.com/571

Знакомство

Я — Владимир Парфиненко,

- бакалавр физики (ФФ), магистр математики (ММФ),
- профессиональный программист (Excelsior),
- регулярно чему-то учу (ФФ, АФТИ, ЛШ ФМШ).

Контакт: vladimir.parfinenko@gmail.com

Представление целых чисел

Двоичная система счисления

Самый простой метод записи чисел, использующий только две цифры: 0 и 1.

С помощью *n позиций* можно записать 2^n чисел.

$$\texttt{000}_2 = 0, \qquad \texttt{100}_2 = 4,$$
$$\texttt{001}_2 = 1, \qquad \texttt{101}_2 = 5,$$
$$\texttt{010}_2 = 2, \qquad \texttt{110}_2 = 6,$$
$$\texttt{011}_2 = 3, \qquad \texttt{111}_2 = 7.$$

Перевод между двоичной и десятичной

Перевод из двоичной в десятичную:

$$11001_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 1 = 25.$$

Перевод из десятичной в двоичную:

$$\begin{aligned} 25 &= 2 \cdot 12 + 1, \\ 12 &= 2 \cdot 6 + 0, \\ 6 &= 2 \cdot 3 + 0, \\ 3 &= 2 \cdot 1 + 1, \\ 1 &= 2 \cdot 0 + 1. \end{aligned}$$

Числа в памяти компьютера

1 *байт* состоит из 8 *бит* и может кодировать $2^8 = 256$ различных чисел.

Например, число 337 кодируется минимум 2 байтами:

$$\underbrace{\texttt{00000001}}_{\text{биты 15...8}}\underbrace{\texttt{01010001}}_{\text{биты 7...0}}$$

Предельные значения:

- 8 бит: 0 . . . 255,
- 16 бит: 0 . . . 65 535,
- 32 бита: 0 . . . 4 294 967 295,
- 64 бита: 0 . . . 18 446 744 073 709 551 615.

Шестнадцатеричная система счисления

Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Перевод в двоичную и обратно через тетрады (по 4 бита):

$$101011100_2 = \{0001\}\{0101\}\{1100\} = 15C_{16}.$$

«Понятная» запись констант:

- 8 бит: диапазон чисел $0_{16} \ldots FF_{16}$,
- 32 бита: диапазон чисел $0_{16} \ldots FFFFFFFF_{16}$,
- 0xDEADBEEF, 0xCAFEBABE, 0xCDCDCDCD, ...

Знаковые числа

Рассмотрим сложение двух 8-битных чисел $\texttt{FF}_{16} + \texttt{01}_{16}$:

$$\begin{array}{r} \,11111111 \\ +\;\;00000001 \\ \hline 1\,\underbrace{00000000}_{\text{8 бит}} \end{array}$$

С точки зрения компьютера $(\texttt{FF}_{16} + \texttt{01}_{16}) \mod 2^8 = \texttt{00}_{16}$.

То есть $X + 1 = 0$. Чему равно X?

Двоичный дополнительный код

Для представления отрицательных чисел используется *двоичный дополнительный код*:

$$(-X) = 2^N - X, \text{ где } N - \text{битность чисел.}$$

8-битные знаковые числа:

$$\begin{aligned} 0 &= \texttt{00000000}_2, & -128 &= \texttt{10000000}_2, \\ 1 &= \texttt{00000001}_2, & -127 &= \texttt{10000001}_2, \\ &\dots, & &\dots, \\ 126 &= \texttt{01111110}_2, & -2 &= \texttt{11111110}_2, \\ 127 &= \texttt{01111111}_2, & -1 &= \texttt{11111111}_2. \end{aligned}$$

Переполнение

Беззнаковые 8-битные:

- $255 + 1 = 11111111_2 + 1 = 00000000_2 = 0$,
- $0 - 1 = 00000000_2 - 1 = 11111111_2 = 255$.

Знаковые 8-битные:

- $127 + 1 = 01111111_2 + 1 = 10000000_2 = -128$,
- $-128 - 1 = 10000000_2 - 1 = 01111111_2 = 127$.

Epic Fails

- 22 сентября 2009 г. — Twitter: порядковый номер твитов переполнил 32-битное беззнаковое целое.
- 9 февраля 2013 г. — OpenStreetMap: порядковый номер точек на карте переполнил 32-битное знаковое целое.
- 1 декабря 2014 г. — YouTube: количество просмотров одного видео переполнило 32-битное знаковое целое.

Целочисленные типы данных в C и операции над ними

Беззнаковые целочисленные типы в C

Тип	Размер, бит	Максимум
`unsigned char`	8	255
`unsigned short`	16	65 535
`unsigned int`	32	4 294 967 295
`unsigned long long`	64	$18{,}4 \cdot 10^{18}$

Знаковые целочисленные типы в C

Тип	Размер, бит	Минимум	Максимум
`signed char`	8	-128	127
`short`	16	$-32\,768$	$32\,767$
`int`	32	$-2\,147\,483\,648$	$2\,147\,483\,647$
`long long`	64	$-9{,}2 \cdot 10^{18}$	$9{,}2 \cdot 10^{18}$

Арифметические операции

Оператор	Операция
`a + b`	сложение
`a - b`	вычитание
`a * b`	умножение
`a / b`	деление нацело
`a % b`	остаток от деления
`-a`	отрицание
`a++`	инкремент
`a--`	декремент

Загадка: модуль числа неотрицательный?

Вычисление модуля введенного числа:

```
int x;
printf("Enter number: ");
scanf("%d", &x);
if (x < 0) {
    x = -x;
}
printf("Absolute value = %d\n", x);
```

Исполнение:

```
Enter number: -2147483648
Absolute value = -2147483648
```

Битовые логические операции

Бинарные:

AND (&):

```
  0101
  0011
------
  0001
```

OR (|):

```
  0101
  0011
------
  0111
```

XOR (^):

```
  0101
  0011
------
  0110
```

Унарные:

NOT (~):

```
  0101
------
  1010
```

Битовые сдвиги

Побитовый сдвиг беззнакового x на n бит...

- влево (x << n) эквивалентен умножению на 2^n,
- вправо (x >> n) эквивалентен делению на 2^n.

Пример:

<< 4:
```
0101110000110101
1100001101010000
```

>> 4:
```
0101110000110101
0000010111000011
```

Битовые операции на практике

Чтение `i`-го бита из `x`:

```
bit = (x >> i) & 1;
```

Выставление `i`-го бита в `x`:

```
x = x | (1 << i);
```

Сброс `i`-го бита в `x`:

```
x = x & ~(1 << i);
```

Вещественные числа

Представление вещественных чисел

Способы представления:

- строка (`"3,1415"`),
- рациональная дробь ($22/7 \approx 3{,}1429$),
- двоичная запись с фиксированной точкой ($11{,}001001_2 = 2^1 + 2^0 + 2^{-3} + 2^{-6} \approx 3{,}1406$),
- двоичная запись с плавающей точкой.

Числа с плавающей точкой

$$x = m \cdot b^e,$$

где:

- m — *мантисса* (значащая часть),
- b — *основание степени* (обычно 2 или 10),
- e — *экспонента* (порядок).

Стандарт IEEE 754

	binary32	binary64
тип в C	`float`	`double`
знак (бит)	1	1
экспонента (бит)	8	11
мантисса (бит)	23	52
точность (дес. знаки)	7	16
мин. абс. значение	$1{,}18 \cdot 10^{-38}$	$2{,}23 \cdot 10^{-308}$
макс. абс. значение	$3{,}40 \cdot 10^{38}$	$1{,}80 \cdot 10^{308}$

Особенные числа

Числа ± 0, $\pm\infty$

$$1/(+0) = +\infty,$$
$$1/(-0) = -\infty,$$
$$\log(0) = -\infty.$$

Не-числа (NaN, Not a Number):

$$NaN = \sqrt{-1}, 0/0, 0 \cdot \infty, \infty/\infty, \infty - \infty,$$
$$NaN = f(NaN),$$
$$NaN \neq NaN.$$

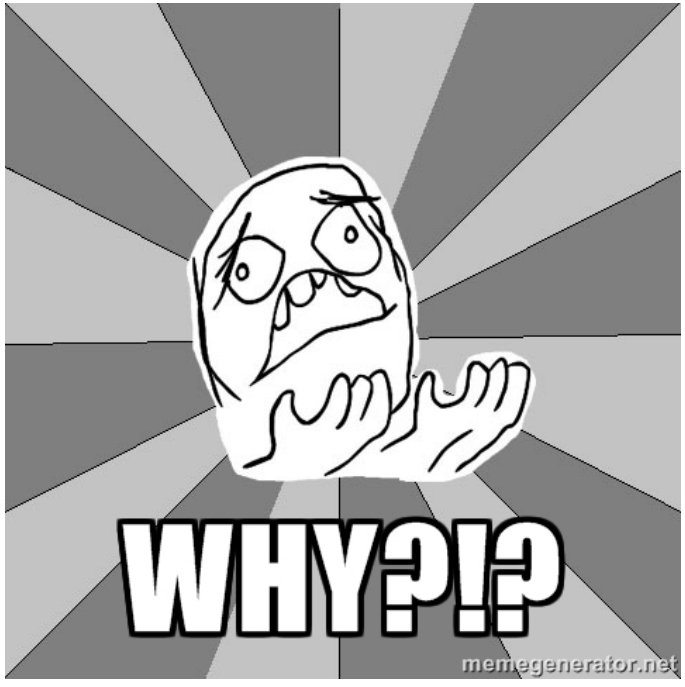
Проблемы с точностью

- Округление: $\mathrm{tg}(\pi) \neq 0, \mathrm{tg}(\pi/2) \neq \infty$.
- Накопление ошибок: $a + (b + c) \neq (a + b) + c$.
- Потеря точности: $a - b$, если $a \approx b$.
- Потеря точности: $a + b$, если $a \gg b$ или $a \ll b$.
- Сравнение чисел.

Неправильное сравнение чисел

```
float x = 0.1f;
float y1 = x * x;
float y2 = 0.01f;

y1 == y2; // => 0
```



Причина неправильного сравнения чисел

```
float x = 0.1f; // 0.100000001490
float y1 = x * x;
// 0.010000000708 =
//     0 01111000 01000111101011100001011
float y2 = 0.01f;
// 0.009999999776 =
//     0 01111000 01000111101011100001010

y1 == y2; // => 0
```

Правильное сравнение чисел

Идея:

```
fabs(y1 - y2) < 0.00001f; // => 1
```

Но нужно так же учесть:

- очень большие и очень маленькие числа,
- бесконечности,
- ...

Реализация: github.com/cypok/floats-comparison

Конец второй лекции