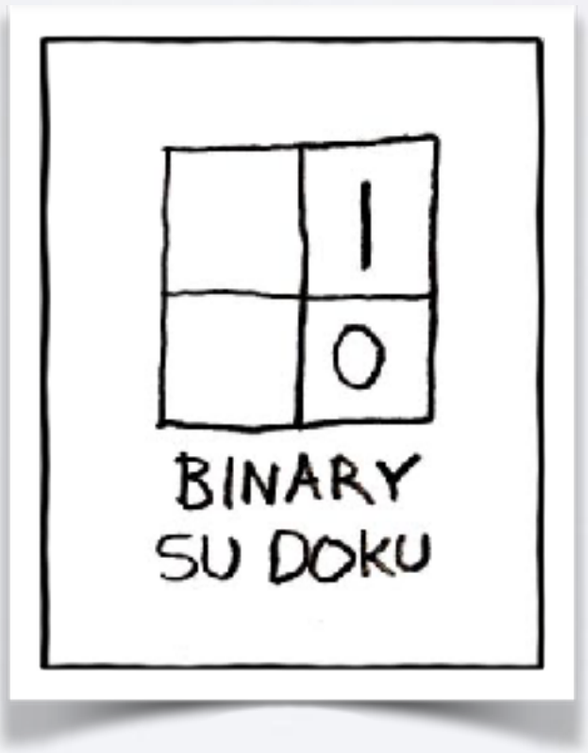
ОСНОВЫ ПРОГРАММНОГО КОНСТРУИРОВАНИЯ

Лекция № 2
11 сентября 2017



https://xkcd.com/74

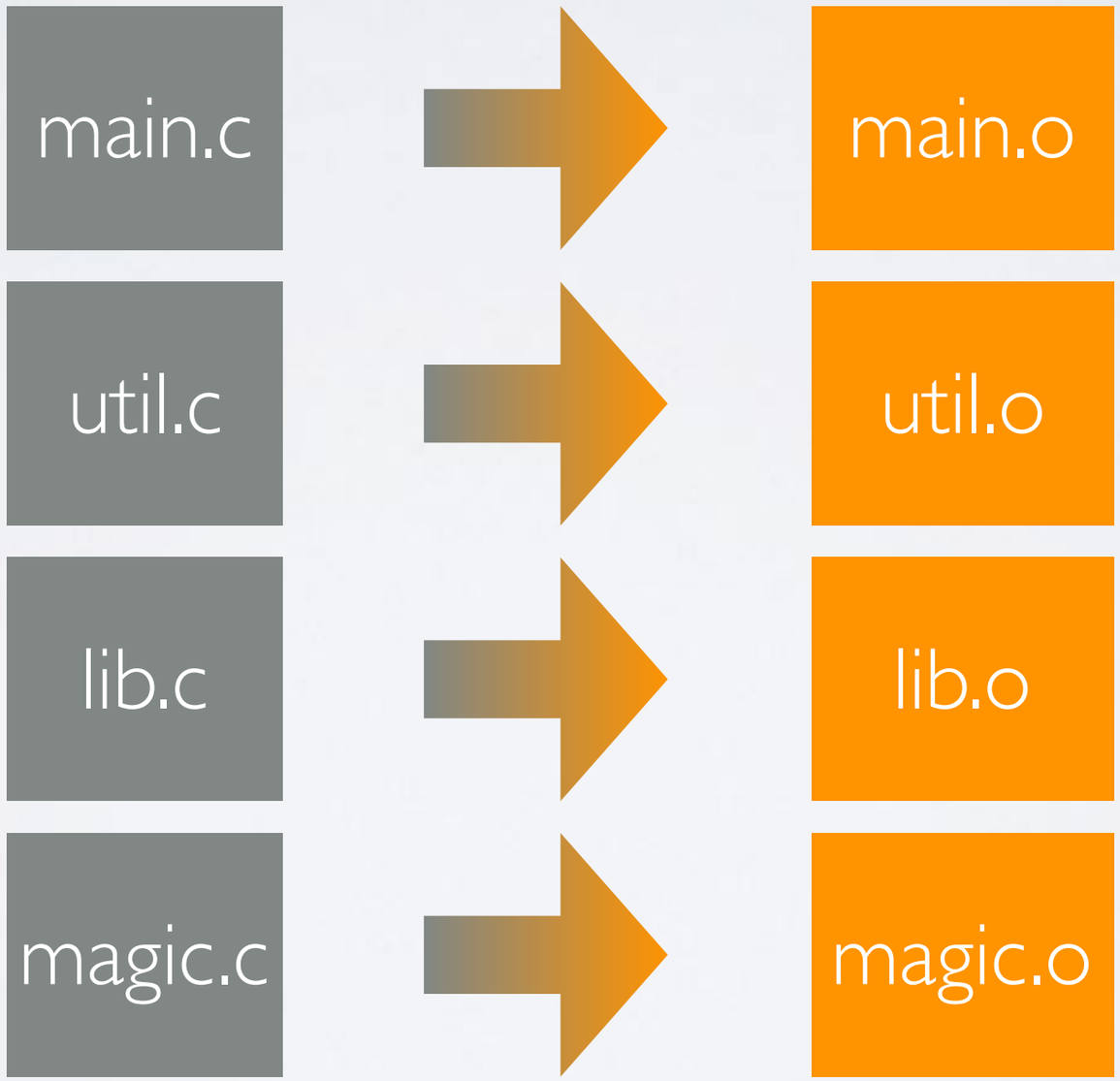
ПРОГРАММА И ЕЕ ПУТЬ

- Программа состоит из функций, в т. ч. функции `main()`.
- Функции расположены в файлах *.c. Например:
  - main.c
  - util.c
  - lib.c
  - magic.c

ЭТАП 1. КОМПИЛЯЦИЯ

Исходные файлы (модули)

Объектные файлы



ОБЪЕКТНЫЙ ФАЙЛ FILE.O

- Машинный код функций, объявленных в file.c.
- Память под объявленные глобальные переменные.
- Ссылки на внешние функции.
- Ссылки на глобальные переменные.

- hello_world.c:
  - Машинный код функции `main()`.
  - Ссылка на внешнюю функцию `printf()`.

ЭТАП 2. ЛИНКОВКА

РАБОТА ЛИНКЕРА

- Операционная единица: **имя**. Каждый объектный модуль (в т. ч. библиотечный):
  - **Предоставляет** какие-то имена (функции, переменные, …)
  - **Требует** какие-то имена.
- Линкер удовлетворяет зависимости (все начинается с имени `main`).

ОШИБКИ ЛИНКЕРА

- Ошибки:
  - Имя требуется одним из модулей, но никаким не предоставляется.
  - Одно и то же имя предоставляется более, чем одним модулем.

```
Undefined symbols for architecture x86_64:
  "_foo", referenced from:
      _fact in fact-xGnG1z.o
ld: symbols(s) not found for architecture x86_64

duplicate symbol _fact in:
    /var/.../foo-ogM19a.o
    /var/.../bar-X6PLBX.o
ld: 1 duplicate symbol for architecture x86_64
```

ИСПОЛНЯЕМЫЙ ФАЙЛ

- Содержит все нужные имена (и ничего лишнего). Все ссылки на имена в объектных файлах были разрешены.

- По построению зависит от объектных файлов и библиотек (те зависят от исходных файлов).

- Для выполнения не нужно больше ничего (за исключением динамических библиотек).

ИНЬ И ЯН ПРОГРАММИРОВАНИЯ

- Алгоритмы.
- Структуры данных.
- Алгоритмы + Структуры данных = Программы.

ДАННЫЕ (ИНФОРМАЦИЯ)

- Данные для человека:
  - Числа (целые, вещественные, комплексные)…
  - Наборы чисел (ряды, векторы, матрицы).
  - Текст (символы).
  - Изображения (видео).
  - Звук.
  - «Записи».
- Компьютер может: 00011110011110101010101

МАШИННОЕ ПРЕДСТАВЛЕНИЕ ДАННЫХ

- Целые числа – двоичная система счисления, дополнительный код.
- Вещественные числа – представление с плавающей точкой.
- Символы – числа (кодировка).
- Набор чисел, символов – массив.
- Картинка – массив пикселов с цветом (RGB).
- Звук – набор отсчетов (квантованная амплитуда).

АЛГОРИТМ

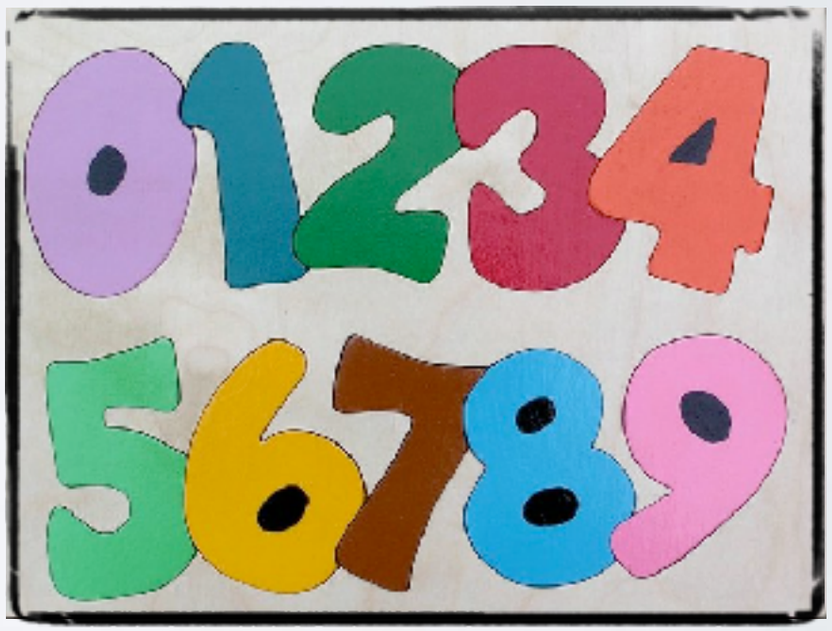
- Решает определенную задачу.
- Конечная упорядоченная последовательность действий.
- Обычно имеет входные параметры и выходные результаты.
- Пример: алгоритм Евклида поиска НОД двух чисел:
  - НОД(a, b) = НОД(a-b, b) если a > b, ...

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

- Структуры данных:
  - Типы данных (числа, символы, строки, …).
  - Переменные.
- Алгоритмы:
  - Операции над данными, в определенной последовательности – согласно синтаксису языка.

ЧИСЛА В МАШИНЕ

- Нет единственного представления.
- Каждое представление имеет свои достоинства и недостатки.
- "123.45" *(строка)* – тоже вариант.



ДВОИЧНАЯ СИСТЕМА

- Быстрее всех.
- Память можно представить как:
  - набор байтов, или 8-битных чисел *(char)*.
  - набор коротких слов, или 16-битных чисел *(short int)*.
  - набор слов, или 32-битных чисел *(int)*.
  - набор длинных слов, или 64-битных чисел *(long int)*.

ПЕРЕВОД 2 ⇔ 10

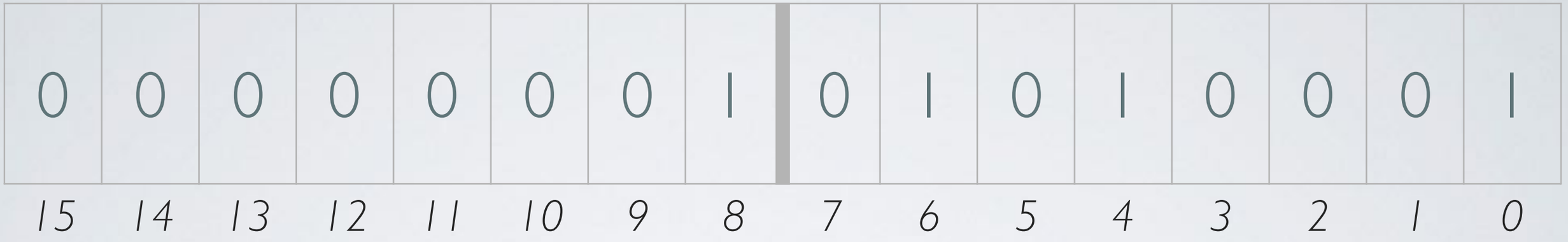
- Делим пополам, сохраняя остатки.

- 337=168*2+**1**; 168=84*2+**0**; 84=42*2+**0**; 42=21*2+**0**; 21=10*2+**1**; 10=5*2+**0**; 5=2*2+**1**; 2=1*2+**0**; 1=0*2+**1**.

- Записываем в обратном порядке: $101010001_2=337_{10}$.

- Обратно: $101010001_2=2^0+2^4+2^6+2^8=1+16+64+256=337$.

В ПАМЯТИ



- Для представления числа 337 нужно как минимум 2 байта.
  - 8 бит: 0..255
  - 16 бит: 0..65535.
  - 32 бита: 0..4294967295 (4,2 млрд.)
  - 64 бита: 0..18446744073709551615 ($\simeq 1{,}8*10^{19}$).

16-РИЧНАЯ СИСТЕМА

- Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- Число в двоичной записи делится на тетрады (по 4 бита):
  - $10101000{1}_2 = \{0001\}\{0101\}\{0001\} = 151_{16}$.
- «Понятная» запись констант:
  - 8 бит: диапазон чисел $0..FF_{16}$ ($0..255_{10}$).
  - 32 бита: диапазон чисел $0..FFFFFFFF_{16}$.
  - 0xDEADBEEF, 0xCAFEBABE, 0xCDCDCDCD…

КАК ХРАНИТЬ ЗНАК

- Отдельной памяти под знак нет, нужно втискивать его в те же 16 (8, 32, 64, …) бит.
- Кодируем знак: «плюс» ⇨ 0, «минус» ⇨ 1.
- Попробуем поместить знак в старший бит:
  - $+337 \Rightarrow 0000\ 0001\ 0101\ 0001_2$
  - $-337 \Rightarrow 1000\ 0001\ 0101\ 0001_2$

ПРОБЛЕМЫ СО ЗНАКОМ В СТАРШЕМ БИТЕ

- Простейшая арифметика (сложение, вычитание) отличается от беззнакового случая.

- Появляются случаи +0 и −0.

ДРУГОЙ ПОДХОД К ОТРИЦАТЕЛЬНЫМ ЧИСЛАМ

- Процессор выполняет операции сложения и вычитания по модулю, соответствующему размеру слова.
  - $(255 + 1) \bmod 2^8 = 0$.
- Забудем на секунду о модуле: X+1=0. Чему равно X?

	1	1	1	1	1	1	1	1
+								
	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0

ДВОИЧНЫЙ ДОПОЛНИТЕЛЬНЫЙ КОД

- $-X \Rightarrow 2^N - X$, где N – размер слова.
  $(X+(-X)) \bmod 2^N = (X+2^N-X) \bmod 2^N = 2^N \bmod 2^N = 0$.
  $(X-(-X)) \bmod 2^N = (X+X-2^N) \bmod 2^N = 2X$.

- Единица в старшем бите все-таки служит индикатором знака.
  Для N=8:

  - $0000\ 0000_2 \Rightarrow 0$ *(мин. положительное число)*.
    $0111\ 1111_2 \Rightarrow 127$ *($2^{N-1}-1$, макс. положительное число)*.
    $1000\ 0000_2 \Rightarrow -128$ *(-2^{N-1}, мин. отрицательное число)*.
    $1111\ 1111_2 \Rightarrow -1$ *(макс. отрицательное число)*.

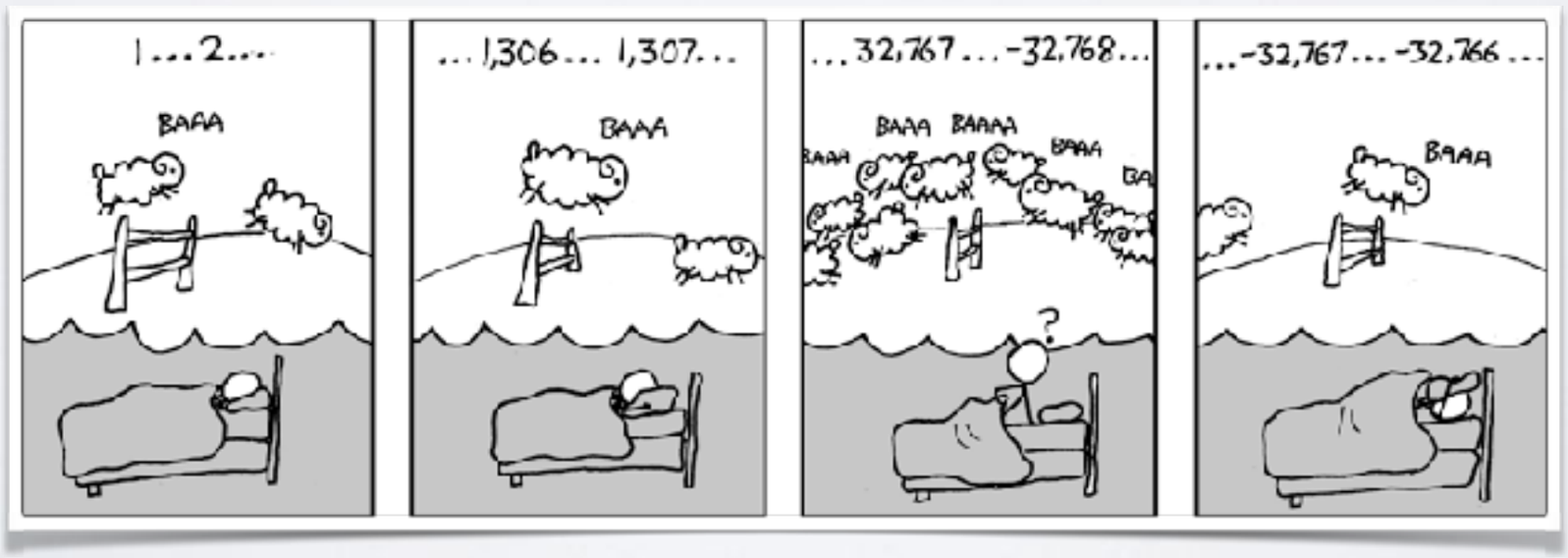
  - Беззнаковый ряд 0…255 соответствует ряду в доп. коде:
    0, 1, 2, 3, …, 126, 127, −128, −127, …, −2, −1

УМНОЖЕНИЕ И ДЕЛЕНИЕ

- Приходится учитывать знак. А вы как думали?
- Знаковое умножение и деление отличается от беззнакового.

ПЕРЕПОЛНЕНИЕ

- $(-128)-1 = 1000\ 0000_2-1 = 0111\ 1111_2 = 127$.
  Ого!
- $127+1 = 0111\ 1111_2+1 = 1000\ 0000_2 = -128$.
  Ой!



НЕДОСТАТКИ ДВОИЧНОЙ СИСТЕМЫ

- Нормальные люди (не программисты) хотят видеть числа в 10-тичной системе (а не 2, 8, 16, …)
- Преобразование в 10-тичную систему требует памяти и вычислений (последовательное деление на 10 с остатком).
- Для встроенных систем (калькуляторы, холодильники) это может быть критично.

ВАРИАНТЫ?

- Хранить число прямо в строке (123456 ⇨ "123456") слишком накладно по памяти.
- Промежуточный вариант: двоично-десятичное кодирование (Binary-Coded Decimal, BCD), т.е. битовое кодирование каждой десятичной цифры.
- Packed BCD: на каждую цифру тратится 4 бита (полубайт).

PACKED BCD

- 337_{10} ⇨ {3}{3}{7} ⇨ $\{0011\}_2\{0011\}_2\{0111\}_2 = 1100110111_2$.
- BCD-представление числа 337_{10} аналогично двоичному представлению 337_{16}! Для знака обычно выделяется младший полубайт:
  - «+» ⇨ $C_{16}=1100_2$
    «−» ⇨ $D_{16}=1101_2$.
- Со знаком: 337_{10} ⇨ $337C_{16}$= 0011 0011 0111 1100_2.
- В 32 битах можно представить числа ±9 999 999 (7 цифр+знак).

ОПЕРАЦИИ С PACKED BCD

- Можно складывать и вычитать битовые представления напрямую, если после этого выполнять коррекцию!

- $11_{16}+15_{16}=26_{16}$. Коррекция не требуется.

- $11_{16}+19_{16}=2A_{16}$. Поскольку был выход за границы 0..9, добавляем еще 6 к переполнившемуся разряду: $2A_{16}+6=30_{16}$.

- Вычитание: $21_{16} - 19_{16} = 8_{16}$. Поскольку был займ, вычитаем 6: $8_{16} - 6_{16} = 2_{16}$.

ПРЕИМУЩЕСТВА BCD

- Упрощенный ввод/вывод: не нужно переводить из одной системы исчисления в другую.
- Нет потерь точность при вводе/выводе вещественных чисел (stay tuned…).
- Легко умножать/делить на 10, округлять десятичных разрядов.

ЗАДАЧКА НА СЛОЖЕНИЕ

X = последние две ненулевые цифры номера вашего студенческого.

Например, 070509 → X = 59.

Вычислить 8 · X, используя Packed BCD. Ответа недостаточно, должны быть промежуточные выкладки.

СТРОКИ

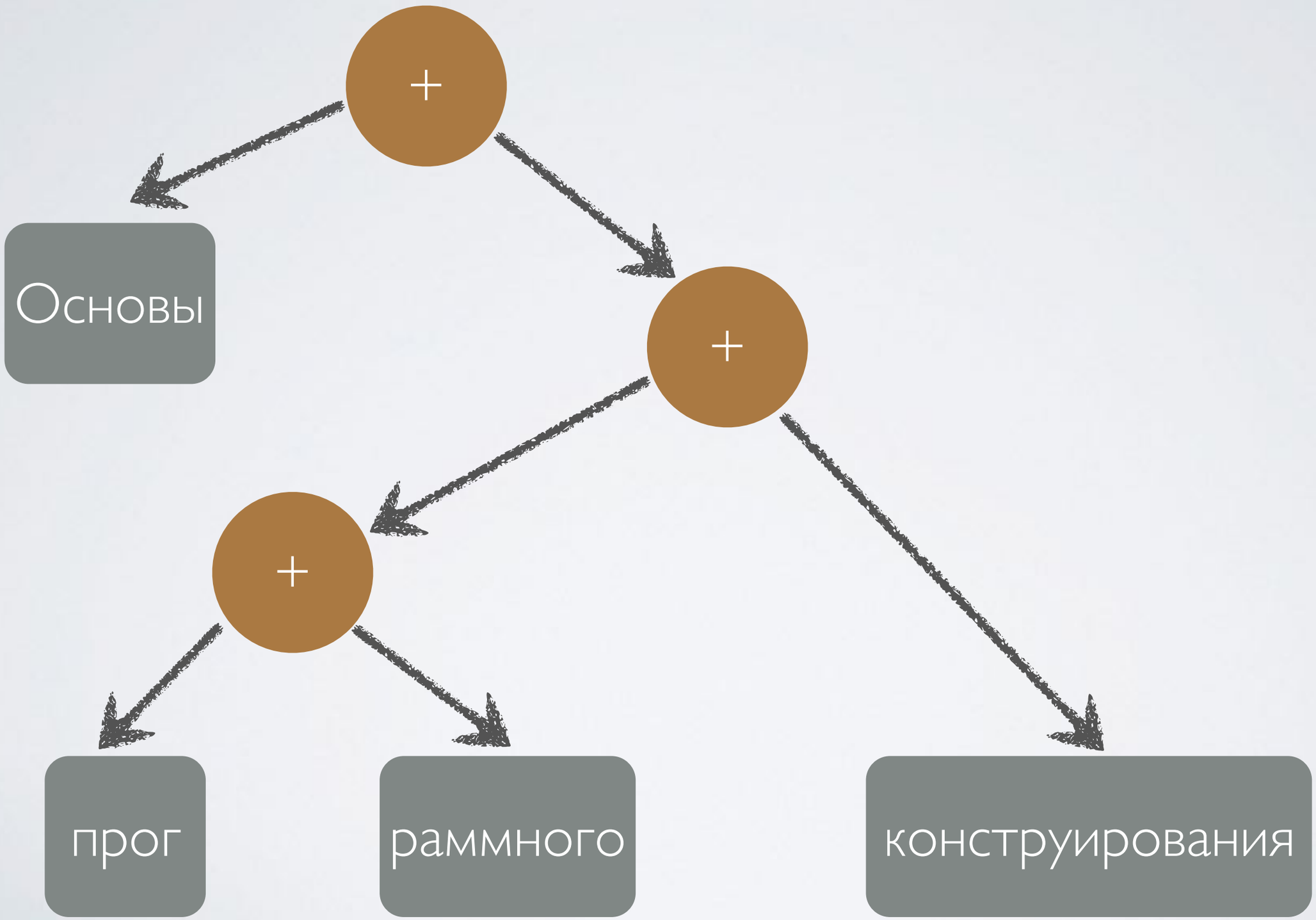
- Строка – просто массив символов, но нужно как-то хранить длину…
- Два стандартных приема:
- Хранение длины в начале строки (первые 1/2/4 байта).

6	s	t	r	i	n	g

- Специальный символ. В языке C этим символом является символ с кодом 0.

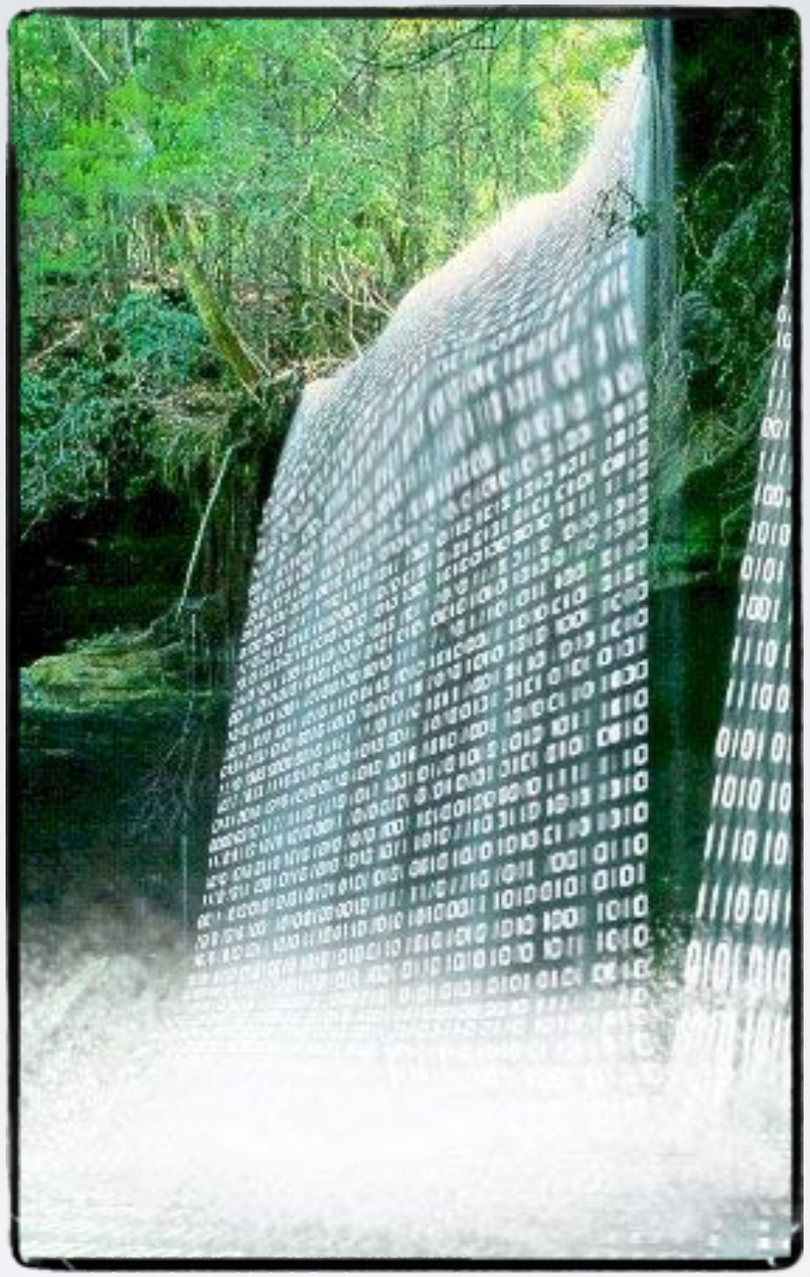
s	t	r	i	n	g	\0

«ВЕРЕВКИ»



ОПЕРАЦИИ СО СТРОКАМИ

- Конкатенация ("abc" + "def" ⇨ "abcdef").
- Вычисление длины.
- Выделение подстроки.
- Взятие символа по индексу *i*.
- Удаление/замена подстроки.
- Поиск символов.
- Поиск подстроки.
- Сравнение двух строк.
- Копирование.
- Разбиение на подстроки (например, на слова).
- Преобразование в число и наоборот.



ВЕЩЕСТВЕННЫЕ ЧИСЛА

Возможны ли точные вычисления с ними?

СПОСОБЫ ПРЕДСТАВЛЕНИЯ

Рациональные дроби с числителем и знаменателем неограниченного размера.

- 22/7 = 3,142857142857142 8
- 884279719003555 / 28147497671 0656 = = 3,14159265358979 31

ПРЕДСТАВЛЕНИЕ С ФИКСИРОВАННОЙ ТОЧКОЙ

0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1
7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8

$1.0101000{1}_2 = 2^0+2^{-2}+2^{-4}+2^{-8} = 1+1/4+1/16+1/256 =$
$= 1.31640625_{10} = 337/256.$

ФИКСИРОВАННАЯ ТОЧКА. ОСОБЕННОСТИ

- Больше точность — меньше диапазон.
  Больше диапазон — меньше точность.

- Округление при представлении чисел (отбрасывание дробной части).

- Умножение и деление могут привести к потере точности, но предсказуемо.

- Вычисления так же быстры, как вычисления с целыми числами.

  - Sony Playstation, Nintendo Gameboy.

  - Кодирование звука: GSM Full rate, Ogg Vorbis.

FIXED-POINT BCD

- $1 = 56,9966 р. *(курс ЦБ РФ с 11.09.2017)*.
- 56,9966 → 569966 → 569966_{16} → 66 99 56 00 (в памяти, little endian).
- Метод часто используется для хранения денежных величин (COBOL).

ПРЕДСТАВЛЕНИЕ С ПЛАВАЮЩЕЙ ТОЧКОЙ

- $x = m \cdot b^e$.
  - m — мантисса (значащая часть, *significand*).
  - b — основание степени (обычно 2 или 10).
  - e — экспонента (порядок).

- $3{,}1415 = 0{,}31415 \cdot 10^1$.
  $400 = 4 \cdot 10^2 = 0{,}4 \cdot 10^3$.
  - Нормальная форма: $|m|<1$.

СТАНДАРТ IEEE 754

binary32

± (S)	Экспонента (E)	Мантисса (M)
1	8 бит	23 бита

binary64

S	E	M
1	11 бит	52 бита

ВИДЫ ХРАНИМЫХ ЧИСЕЛ

Тип	Экспонента	Мантисса
±0	0	0
Денормализованные числа	0	0.mmmmmmm
Нормализованные числа	1…254 (1…2046)	1.mmmmmmm
±∞	255 (2047)	0
Не-числа (NaN)	255 (2047)	любая

НОРМАЛИЗОВАННЫЕ ЧИСЛА

- Для binary32:
- $N = (-1)^S \cdot 2^{E-127} \cdot 1.mm\ldots m$.
  - $1 \le M < 2$.
- $|N|_{min} = 2^{1-127} \cdot 1.00\ldots0 = 2^{-126} \approx 1{,}18 \cdot 10^{-38}$.
- $|N|_{max} = 2^{254-127} \cdot 1.11\ldots1 = 2^{127} \cdot (2-2^{-23}) \approx 3{,}4 \cdot 10^{38}$.

ДЕНОРМАЛИЗОВАННЫЕ ЧИСЛА

- Для binary32:
- $D = (-1)^S \cdot 2^{-126} \cdot 0.mm\ldots m$.
- $|D|_{min} = 2^{-126} \cdot 0.00\ldots01 = 2^{-126} \cdot 2^{-23} = 2^{-149} \approx 1{,}4 \cdot 10^{-45}$.
- $|D|_{max} = 2^{-126} \cdot 0.11\ldots11 = 2^{-126} \cdot (1 - 2^{-23}) \approx 1{,}18 \cdot 10^{-38}$.
- $|D|_{max} + |D|_{min} = |N|_{min}$.

ПАРАМЕТРЫ BINARY64

- $|D|_{min} \approx 5 \cdot 10^{-324}$.
- $|D|_{max} \approx 2{,}23 \cdot 10^{-308}$.
- $|N|_{min} \approx 2{,}23 \cdot 10^{-308}$.
- $|N|_{max} \approx 1{,}80 \cdot 10^{308}$.

ТОЧНОСТЬ

- Десятичных знаков:
  - $\log_{10}(2^{23+1}) \approx 7{,}225$ (binary32).
  - $\log_{10}(2^{52+1}) \approx 15{,}955$ (binary64).

ПРОЧИЕ ЧИСЛА

- $\pm\infty$
  - Деление на 0.
  - $\mathrm{tg}(\pi/2)$.
- NaN (Not a Number):
  - Корень из -1.
  - ∞/∞, $0/0$, $0\cdot\infty$.
  - $\infty-\infty$
  - func(NaN)
  - NaN $\neq$ NaN!

ПРИМЕР КОДИРОВАНИЯ

- −118.625:
  - Знак отрицательный: S = 1.
  - Двоичная запись 118.625: 1110110.101
  - Сдвиг влево: $1.1101101101 \times 2^{6}$.
  - Отбрасываем старшую 1; E = 127+6 = 133.

S	E, 8 bits	M, 23 bits
1	10000101	11011010100000000000000

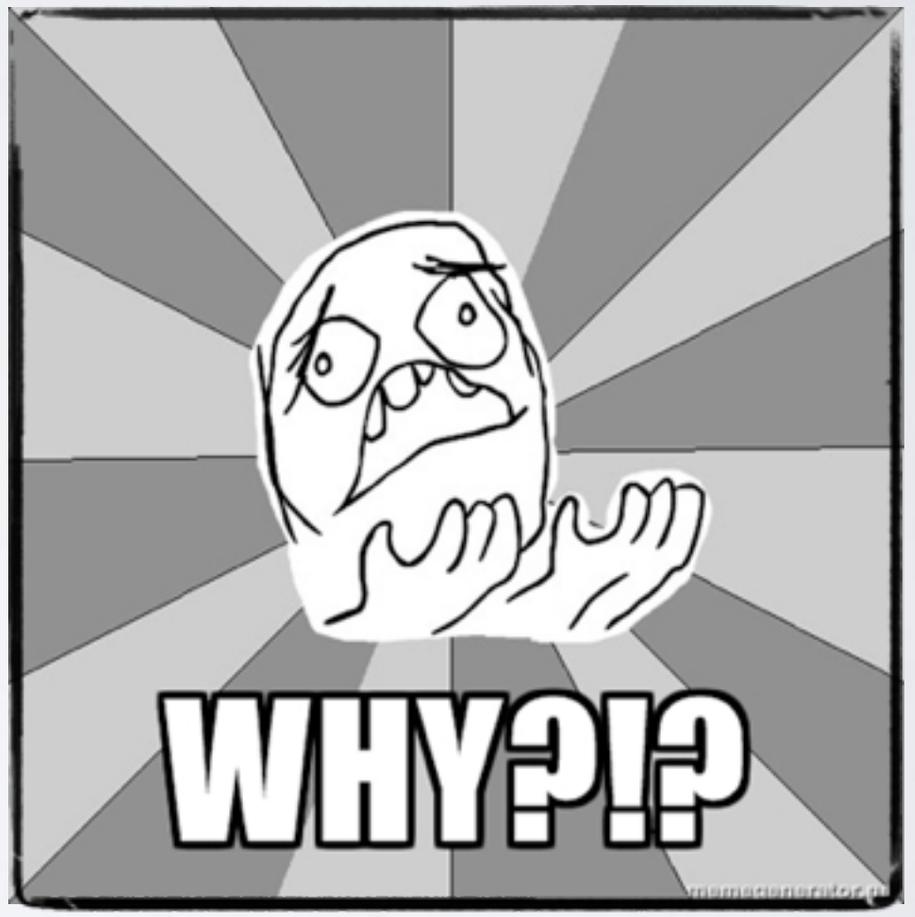
ПРОБЛЕМЫ

- Округление: $\mathrm{tg}\ \pi \neq 0$; $\mathrm{tg}(\pi/2) \neq \infty$.
- Накопление ошибок: $a \oplus (b \oplus c) \neq (a \oplus b) \oplus c$.
- Потеря точности: $a \ominus b$ при $a \approx b$, $a \oplus b$ при $a \ll b$.
- Сравнение чисел.

ПРОБЛЕМА СРАВНЕНИЯ

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

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



ПРОБЛЕМА СРАВНЕНИЯ

```
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
```

один бит
отличается

ПРОБЛЕМА СРАВНЕНИЯ

```
// float.h

/* Difference between 1.0 and
   the minimum float greater than 1.0 */
#define FLT_EPSILON 1.1920929e-07F

/* Difference between 1.0 and
   the minimum double greater than 1.0 */
#define DBL_EPSILON 2.2044604925031308e-16
```

ПРОБЛЕМА СРАВНЕНИЯ

```
#include <float.h>
#include <math.h>

int almostEqual(float a, float b, float epsilon) {
    if (a == b) {
        return 1;
    } else {
        float A = fabs(a);
        float B = fabs(b);
        float diff = fabs(a - b);
        if (a == 0 || b == 0 || diff < FLT_MIN) {
            return diff < epsilon * FLT_MIN;
        } else {
            return diff / (A + B) < epsilon;
        }
    }
}
```

ВЫЧИСЛЕНИЕ π

$$t_0 = \frac{1}{\sqrt{3}}$$

$\pi \approx 6 \cdot 2^i \cdot t_i$ при $i \to \infty$.

$$t_{i+1} = \frac{\sqrt{t_i^2 + 1} - 1}{t_i} \quad \text{или} \quad t_{i+1} = \frac{t_i}{\sqrt{t_i^2 + 1} + 1}$$

i	1-я формула	2-я формула
0	3.464101615137754386 3	3.464101615137754386 3
10	3.141592927873374074 8	3.141592927385097988 5
16	3.141592671741285869 3	3.141592653656639422 2
17	3.141581007579623330 2	3.141592653606506191 3
20	3.140543492400840630 5	3.141592653590056016 8
24	3.224515243534552544 3	3.141592653589796890 7

ПРИМИТИВНЫЕ ТИПЫ ДАННЫХ

- Числа (целые, дробные).
- Символы (строки?).
- Булевы переменные (True / False, Истина / Ложь).
  - C99: bool, true, false @ stdbool.h

Данные предметной области

Сложные типы данных

Списки
Деревья
Хеш-таблицы

Составные типы данных

Массивы
Записи
(структуры)

Примитивные типы данных

Числа, символы, перечисления

ПИРАМИДА ДАННЫХ

КОНЕЦ ВТОРОЙ ЛЕКЦИИ