Структура и архитектура программных систем

github.com/HowProgrammingWorks

Timur Shemsedinov

Chief Technology Architect at Metarhia
Lecturer at Kiev Polytechnic Institute

github.com/tshemsedinov

Код позволяет найти общий язык

Болтовня ничего не стоит.
Покажите мне код.

// Линус Торвальдс

Человекочитаемость

Программы должны писаться для людей,
которые будут их читать,
а машины, которые будут эти программы
исполнять — второстепенны

// Гарольд Абельсон

Проблема понимания

Терминология позволяет инженерам понимать друг друга и смежных специалистов.

Стандарты нужны для унификации процессов интеграции в инженерии.

Шаблоны проектирования нужны, чтобы распространять идеи, а не реализацию.

Сложность

Я всегда мечтал о том, чтобы моим компьютером можно было пользоваться так же легко, как телефоном; моя мечта сбылась: я уже не могу разобраться, как пользоваться моим телефоном

// Бьёрн Страуструп

Алгоритмы vs структуры данных

Плохие программисты беспокоятся о коде.
Хорошие программисты беспокоятся о структурах данных и связям между ними.

// Линус Торвальдс

Algorithm & Programm

Алгоритм —

это формальное описание порядка вычислений для класса задач.

Программа —

программный код и данные, объединенные в одно целое для вычисления и управления ЭВМ.

Что такое программы?

Алгоритмы + Структуры данных = Программы

// Никлаус Вирт

Что такое программы?

Программы становятся медленнее быстрее,
чем «железо» становится быстрее

// Никлаус Вирт

Programming Language

Язык программирования —
формальный синтаксис и связанная с ним семантика, который может исполняться на вычислительной технике или транслироваться в другой язык

Но мы пишем скорее на API, чем на языке

Что такое программное обеспечение?

Большинство программ на сегодняшний день подобны египетским пирамидам из миллиона кирпичиков друг на друге и без конструктивной целостности — они просто построены грубой силой и тысячами рабов

// Алан Кей

Module & Component

Модуль –

целостный, функционально полный, независимый компонент программной системы имеющий имя, интерфейс, реализацию.

Modularity

Модульность

- модульность повышает переиспользование кода
- упрощает интеграцию компонентов
- улучшает компоновку и тестирование программ по частям

Library

Библиотека (иногда синоним модуля) — набор программных объектов (например: функций, классов, монад, типов) подготовленный для повторного использования (опубликованный в виде репозитория).

Framework

Фреймворк —

платформа, диктующая структуру (иногда архитектуру) и правила построения программной системы.

Архитектура —

разделить задачу на части, дать названия частям и собрать их в единое целое.

Architecture

Архитектура

файл-серверная,
клиент-сервер (бекенд и фронтенд),
Многозвенная (трех-звенная),
многослойная, монолитная,
гексогональная, луковая,
микросервисная, облачная...

Процесс разработки

- Использование Open source в разработке
- Лицензии на код и безопасность
- Организация процесса разработки
- Надежность и качество
- Ревью кода, рефакторинг
- IDE, линтер, CI/CD, системы контроля версий
- Владение кодом и bus-factor

Engineering

Инженерия —

> извлечение практической пользы из имеющихся ресурсов при помощи науки, техники, различных методик, организационной структуры, а также приемов и знаний.

Software engineering

Инженерия программного обеспечения — приложение инженерии к индустрии программного обеспечения. Включает архитектуру, исследование, разработку, тестирование, развертывание и поддержку ПО.

Programming

Программирование —

это искусство и инженерия решения задач при помощи вычислительной техники.

Coding

Кодирование —

написание исходного кода программы
при помощи определенного синтаксиса
(языка), стиля и парадигмы
по готовому ТЗ.

Software development

Разработка программного обеспечения — это соединение программирования и кодирования на всех этапов жизненного цикла ПО: проектирования, разработки, тестирования, отладки, поддержки, сопровождения и модификации.

Что такое программирование?

Программирование сегодня — это гонка разработчиков программ, стремящихся писать программы с большей и лучшей идиотоустойчивостью, и вселенной, которая пытается создать больше отборных идиотов. Пока вселенная побеждает.

// Рик Кук

Programming Paradigm

Парадигма программирования —

Парадигма задает набор идей и понятий, допущений и ограничений, концепций, принципов, постулатов, приемов и техники программирования для решения задач на ЭВМ.

Programming Paradigm

Парадигма предлагает модель решения задач, определенный стиль, шаблоны (примеры хороших и плохих решений) применяемых для написания программного кода.

Как вообще все это может работать?

Большинство хороших программистов делают свою работу не потому, что ожидают оплаты или признания, а потому что получают удовольствие от программирования

// Линус Торвальдс

Debug

Отладка —

процесс обнаружения и устранения ошибок в ПО при помощи вывода сообщений или инструментов: отладчика, профилировщика, декомпилятора, систем мониторинга ресурсов и логирования, систем CI и тестирования.

Что такое отладка?

Отладка кода вдвое сложнее, чем его написание. Так что если вы пишете код настолько умно, насколько можете, то вы по определению недостаточно сообразительны, чтобы его отлаживать.

// Брайан Керниган

По типизации

Языки со статической типизацией

определяют типы идентификаторов во время компиляции

Языки с динамической типизацией

определяют типы идентификаторов на этапе выполнения

По строгости типизации

Языки с сильной типизацией

запрещено смешивать типы и нет автоматического неявного приведения

Языки со слабой типизацией

в выражениях можно смешивать типы, неявные преобразования типов происходят автоматически

Защита от дурака

Ограничение возможностей языка с целью предотвращения программистских ошибок в лучшем случае опасно

// Бьёрн Страуструп

По уровню

Тем ниже уровень языка

чем он ближе к машине и оборудованию, их конструкции и особенностям

Тем выше уровень языка

чем он ближе к человеку и его способу мышления, к естественным языкам

Системное программирование

Группа А (системное)

Производство средств производства

Группа Б (прикладное)

Производство товаров потребления

Сравнение asm и машинного кода

```
pr1:  mov  ax, 5          00: 66 B8 05 00
      mov  bx, 7          04: 66 BB 07 00
      add  ax, bx         08: 66 01 D8
      jmp  exit           0B: EB 05
pr2:  push ax             0D: 66 50
      xor  ax, ax         0F: 66 31 C0
exit: mov  ax, 4c00h      12: 66 B8 00 4C
      int  21h            16: CD 21
```

По сфере применения

- Системное программирование
- Для десктопных оконных приложений
- Для встраиваемых систем и автоматизация
- Для баз данных
- Для сетевых серверов
- Для вычислений

По сфере применения

- Для задач искусственного интеллекта
- Для параллельных вычислений
- Для веба
- Для скриптов: сборка, трансляция...
- Для тестирования
- Для мобильных платформ
- Для игр и графики

По модели работы с памятью

Ручная

C, C++

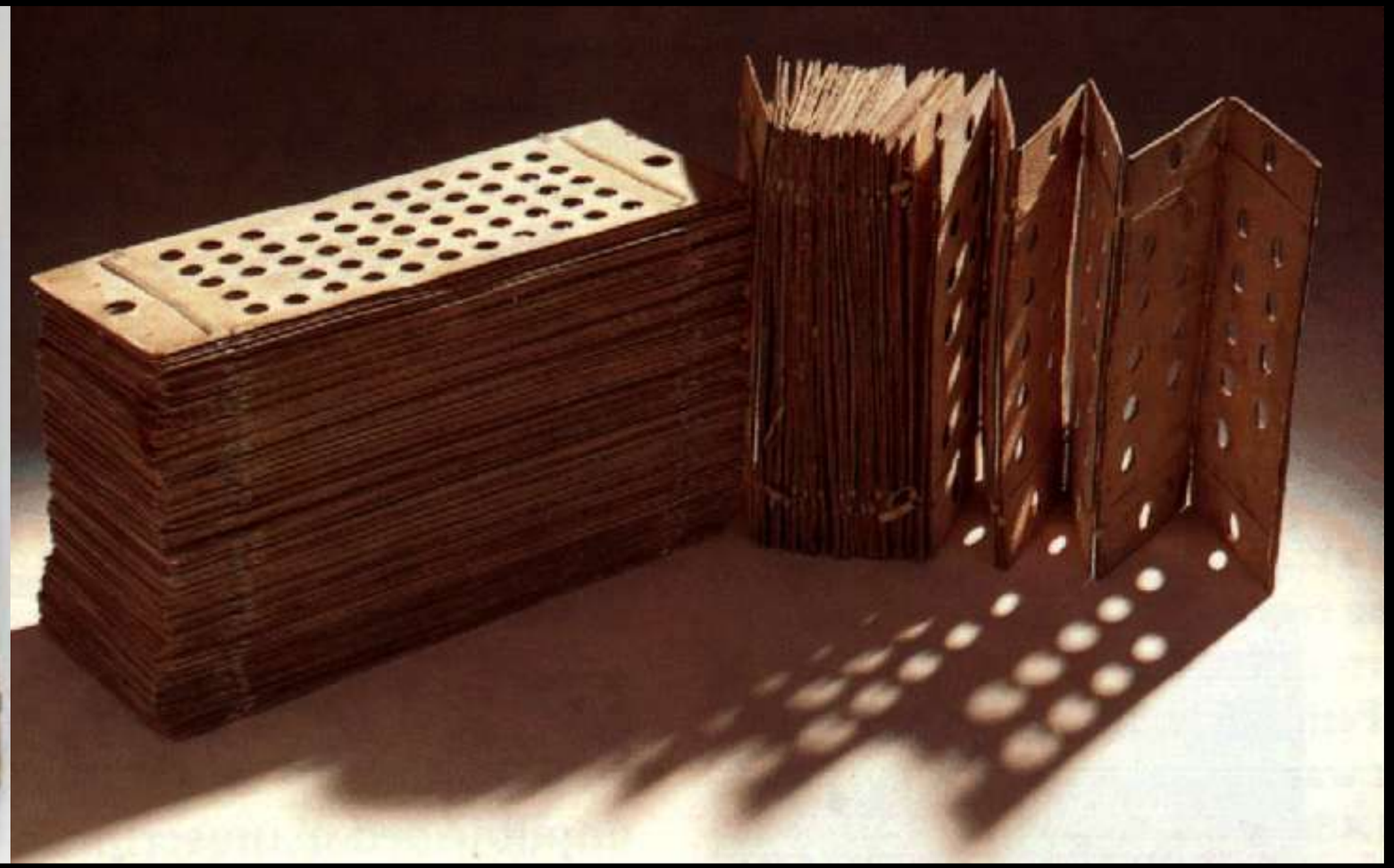
Автоматическая (сборщик мусора)

JS, Java, C#

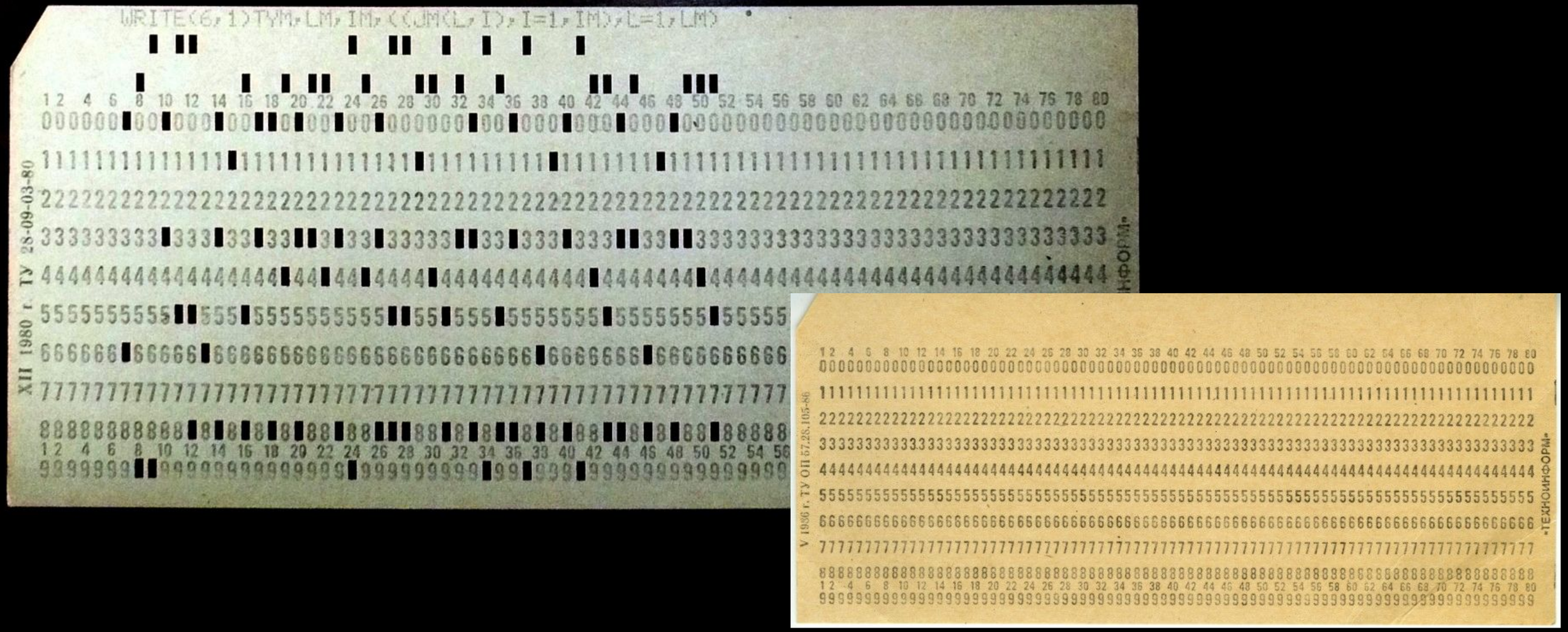
Автоматическая

Rust

Cards for Mechanical computers



Cards for Electronic computers



Machine code

```
00000000:  66 B8 05 00
00000004:  66 BB 07 00
00000008:  66 01 D8
0000000B:  EB 05
0000000D:  66 50
0000000F:  66 31 C0
00000012:  66 B8 00 4C
00000016:  CD 21
```

Assembly Language

```
pr1:    mov   ax, 5
        mov   bx, 7
        add   ax, bx
        jmp   exit
pr2:    push  ax
        xor   ax, ax
exit:   mov   ax, 4c00h
        int   21h
```

Fortran

```
      INTEGER N
      N = 1
10    IF (N .LE. 100) THEN
        WRITE (*,*) N
        N = 2*N
        GOTO 10
      ENDIF
      INTEGER I
      DO 20 I = 10, 1, -2
        WRITE (*,*) 'I =', i
20    CONTINUE
```

Basic

```
10 LET S = 0
20 MAT INPUT V
30 LET N = NUM
40 IF N = 0 THEN 100
50 FOR I = 1 TO N
65 LET S = S + V(I)
70 NEXT I
80 PRINT S/N
90 GO TO 10
100 END
```

LISP

```
(+ 1 2 3 4)

(defun factorial (n)
  (if (= n 0) 1
    (* n (factorial (- n 1)))))

(defun factorial (n)
  (loop for i from 1 to n
    for fac = 1 then (* fac i)
    finally (return fac)))
```

Algol 60

```
BEGIN
  INTEGER ARRAY candidates[0:1000];
  INTEGER i;
  FOR i := 0 STEP 1 UNTIL 1000 DO
  BEGIN
    candidates[i] := 1;
  END;
END;
```

C

```c
#include<stdio.h>

int f(int n) {
  if (n == 0 || n == 1) return n;
  else return (f(n - 1) + f(n - 2));
}

int main() {
  printf("%d\n", f(7));
  return 0;
}
```

C++

```cpp
#include <iostream>

using namespace std;

class Point {
  public:
    double x, y;
};

int main() {
  Point point1;
  point1.x = 10.0;
  point1.y = 20.0;
  cout << point1.x << "," << point1.y << endl;
  return 0;
}
```

Ada

```
function Get_Maximum (Of : My_Array_Type) return
Element_Type is
   Maximum : Element_Type := Of (Of'First);
begin
   for I in Of'First + 1 .. Of'Last loop
      if Of (I) > Maximum then
         Maximum := Of (I);
      end if;
   end loop;
   return Maximum;
end Get;
```

dBase, FoxBase, Clipper

```
@ 0,0
arr := { 1, 2, 3 }
str := "abc"
FOR EACH a, b IN arr, str
   ? a, b
   a *= 2
   IF a == 3
      ?? Upper( b )
      EXIT
   ENDIF
NEXT
```

Haskell

```haskell
fibonacci :: Integer -> Integer
fibonacci 0 = 1
fibonacci 1 = 1
fibonacci x = fibonacci (x-1) + fibonacci (x-2)

fibs = 1 : 1 : zipWith (+) fibs (tail fibs)
fib n = fibs !! n
```

Python

```python
def sumDigit(num):
    sum = 0
    while(num):
        sum += num % 10
        num = int(num / 10)
    return sum

num = [15, 300, 2700, 821, 52, 10, 6]
print('Maximum is:', max(num, key=sumDigit))
```

Java

```
public class Main {
  public static void main(String[] args) {
    int a[] = new int [100];
    Random rnd = new Random();
    int max = 0;
    for (int i = 0; i < a.length; i++) {
      a[i] = rnd.nextInt(100);
      if (a[j] > max) max = a[j];
    }
  }
}
```

Delphi

```
procedure TForm1.MyButtonClick(Sender: TObject);
var
  Button: TButton;
begin
  Button := Sender as TButton;
  ShowMessage(Button.Caption + ' clicked');
end;

AButton := TButton.Create;
AButton.Caption = 'Click me';
AButton.OnClick := MyButtonClick;
```

SQL

```
CREATE TABLE Department(
   Id INT PRIMARY KEY        NOT NULL,
   Name                CHAR(50) NOT NULL,
   EmpId               INT      NOT NULL
);

INSERT INTO DEPARTMENT (Id, Name, EmpId)
VALUES (1, 'Software', 7);

SELECT * FROM Department WHERE Id = 1;
```

PHP

```php
<?php  class UserLogin {
         private $logged_in = false;
         function __construct(sname) {
           session_name(sname);
           session_start();
           $session_id = session_id();
           if (isset($_SESSION['userid'])) {
             $this->logged_in = true;
           }
         }
       }
                                                    ?>
```

C#

```
[Serializable]
public class Point {
  public int x = 0;
  public int y = 0;
}

IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream("File.bin",
  FileMode.Create, FileAccess.Write, FileShare.None);
formatter.Serialize(stream, p1);
stream.Close();
```

```
Point p1 = new Point();
p1.x = 10;
p1.y = 20;
```

Go

```
package main

import "fmt"

func f(from string) {
    for i := 0; i < 3; i++ {
        fmt.Println(from, ":", i)
    }
}

func main() {
    f("direct")
    go f("goroutine")
}
```

Rust

```rust
fn main() {
    for n in 1..50 {
        println!("{}", n);
        if n % 3 == 0 {
            println!("if n % 3 == 0");
        } else if n % 5 == 0 {
            println!("n % 5 == 0");
        }
    }
    println!("Done");
}
```

TypeScript

```
interface Person {
  firstName: string;
  lastName: string;
}

function greeter(person: Person) {
  return 'Hello, ' + person.firstName + ' ' + person.lastName;
}

const user = { firstName: 'Marcus', lastName: 'Aurelius' };
console.log(greeter(user));
```

Swift

```
var square = 0
var diceRoll = 0
while square < finalSquare {
    diceRoll += 1
    if diceRoll == 7 {
      diceRoll = 1
    }
    square += diceRoll
    if square < board.count {
        square += board[square]
    }
}
print("Game over!")
```

Kotlin

```kotlin
class Animal(val name: String)

class Zoo(val animals: List<Animal>) {
    operator fun iterator(): Iterator<Animal> {
        return animals.iterator()
    }
}

fun main() {
    val zoo = Zoo(listOf(Animal("zebra"), Animal("lion")))
    for (animal in zoo) {
        println("Watch out, it's a ${animal.name}")
    }
}
```

Структура и архитектура программных систем

github.com/HowProgrammingWorks

Timur Shemsedinov

Chief Technology Architect at Metarhia
Lecturer at Kiev Polytechnic Institute

github.com/tshemsedinov