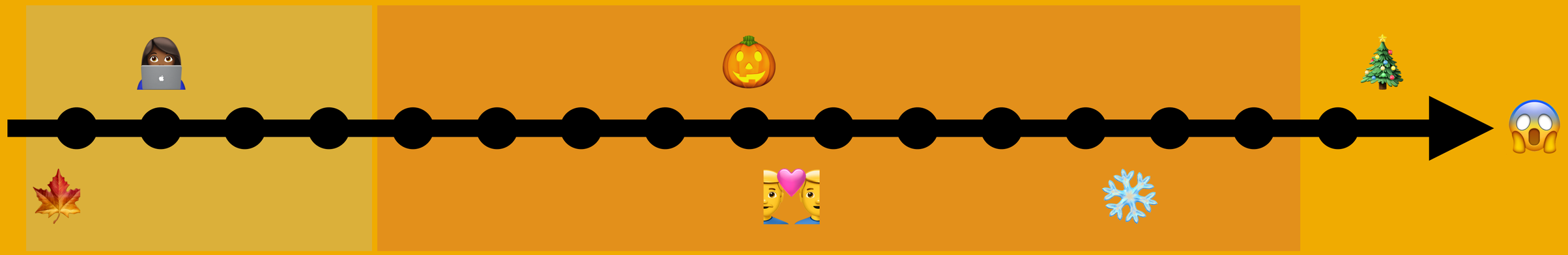
ИНФОРМАТИКА
ПОЛИТЕХ
2020

12. История IT и музыки 🎸

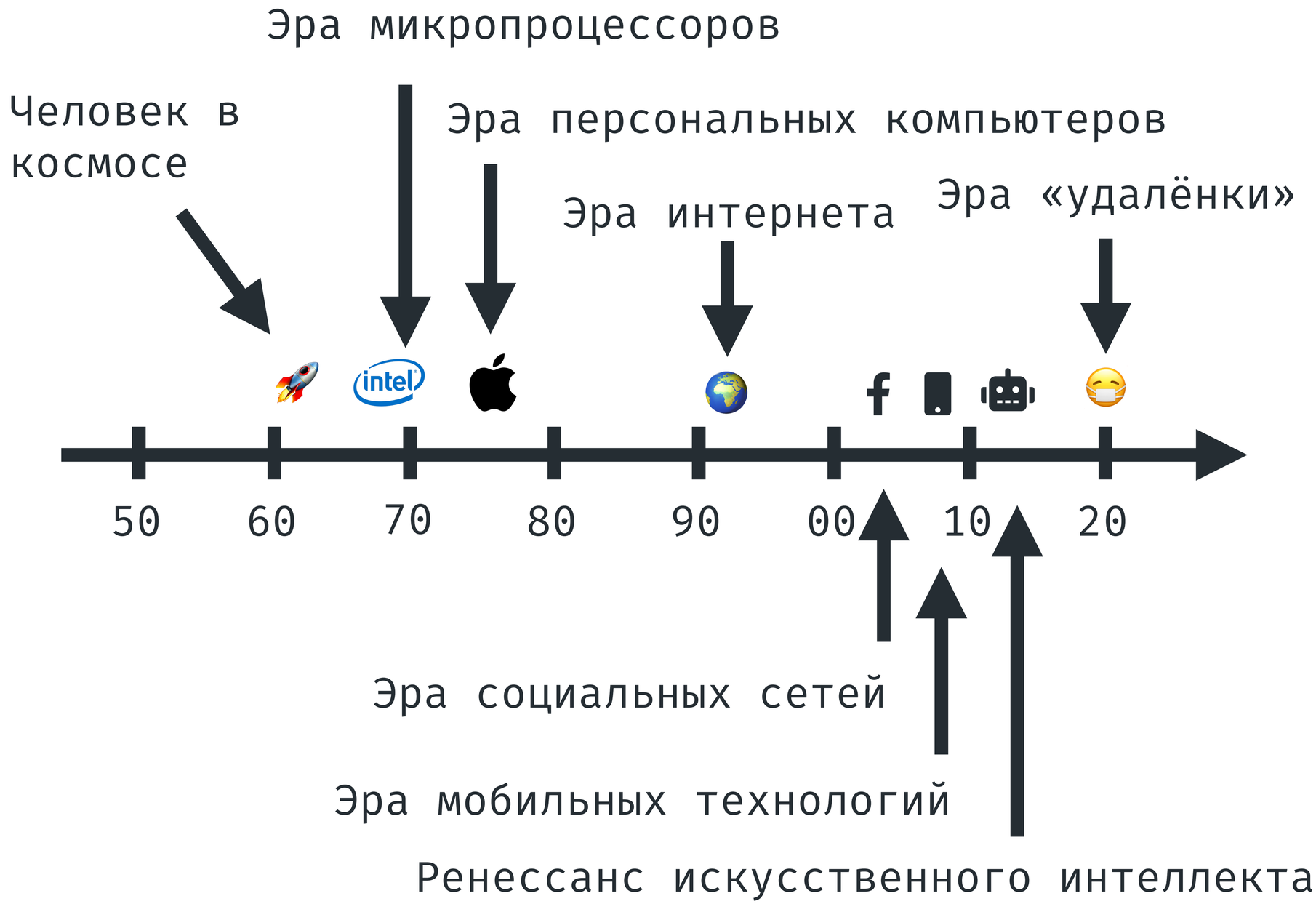
Лекция о параллельных историях развития информационных технологий и музыки

Константин Кориков 🤦 Елена Кононенко 🤦‍♀️



МЫ ТУТ

СЕМЕСТР


Эра микропроцессоров
Человек в космосе
Эра персональных компьютеров
Эра интернета
Эра «удалёнки»
intel
50
60
70
80
90
00
10
20
Эра социальных сетей
Эра мобильных технологий
Ренессанс искусственного интеллекта

Барабаны

Калимба

Духовые

Хронограф

Грамофон

Игорь Федорович Стравинский

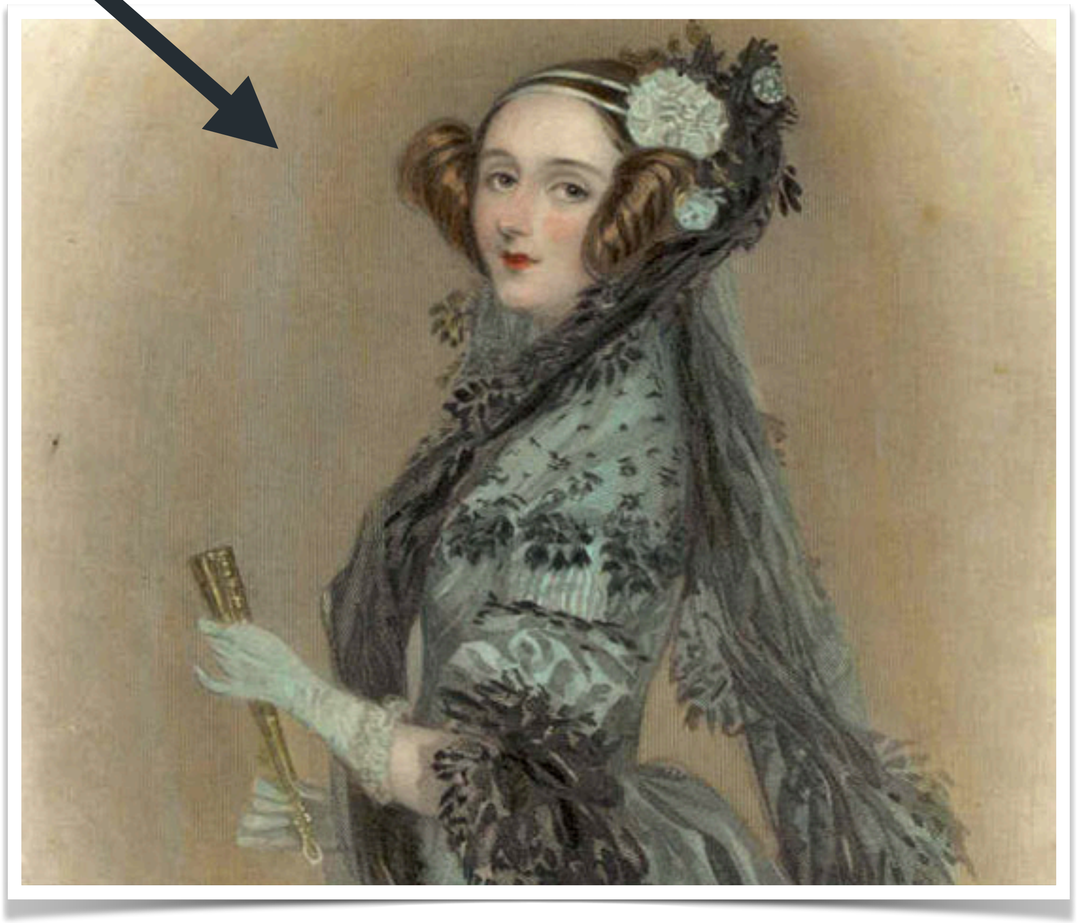
Глен Миллер

Джордж Гершвин

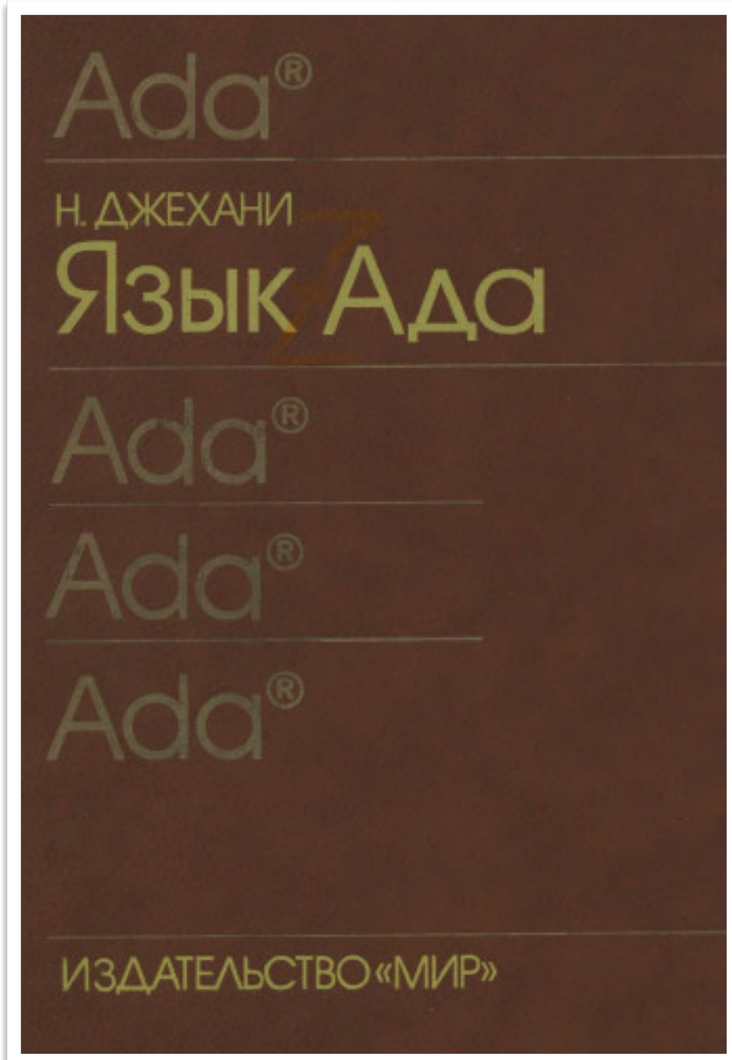
Джон Кейдж

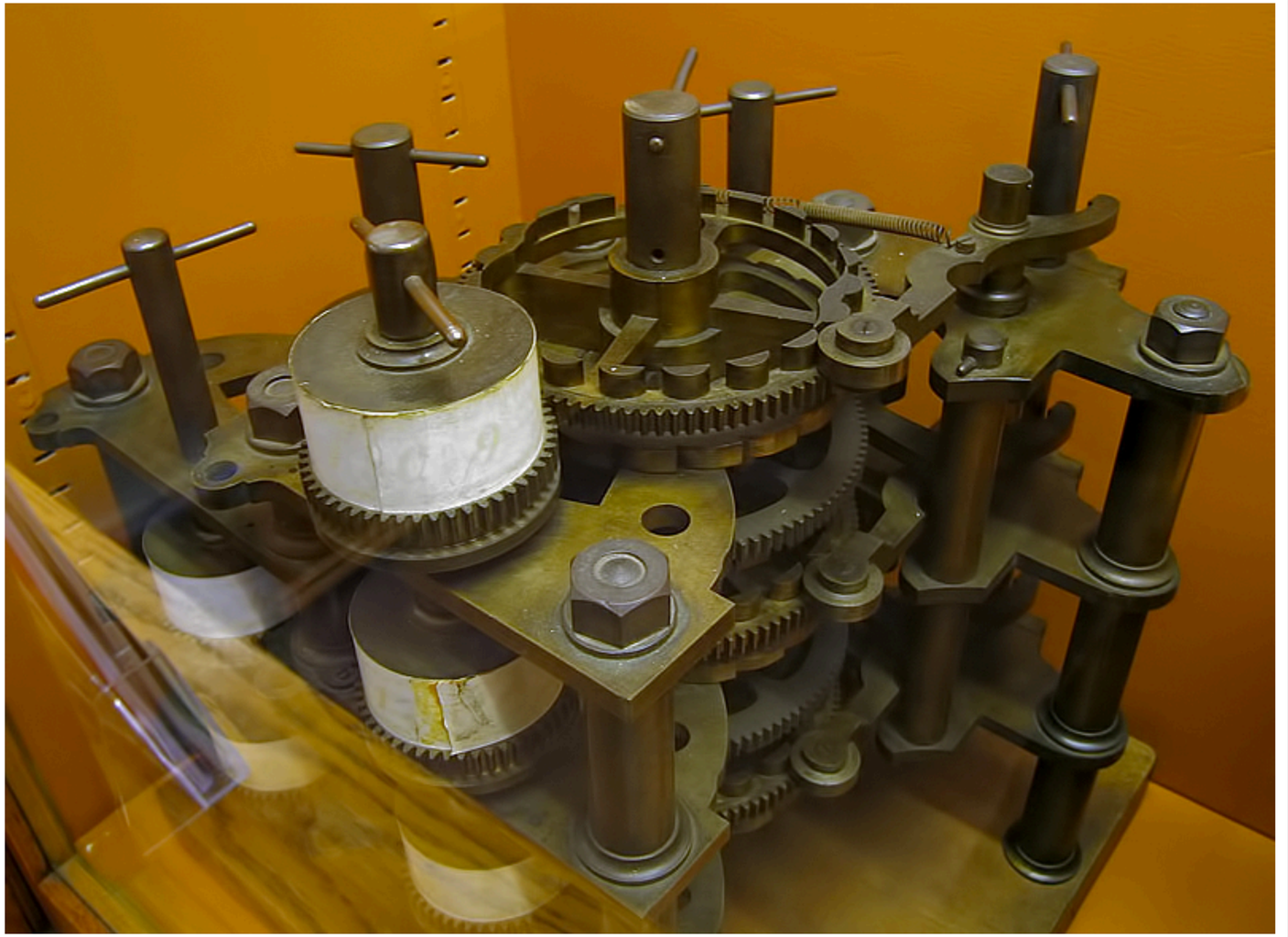
Билл Хейли

Первый программист

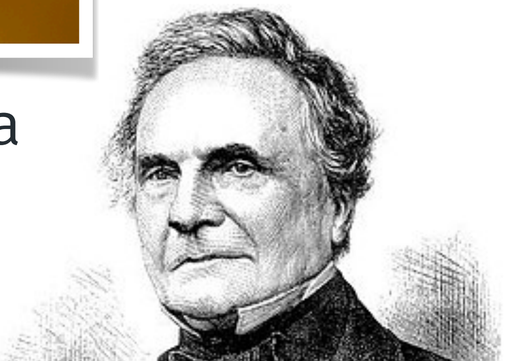


Ада Байрон (графиня Лавлейс)


Ada®
Н. ДЖЕХАНИ
Язык Ада
Ada®
Ada®
Ada®
ИЗДАТЕЛЬСТВО «МИР»

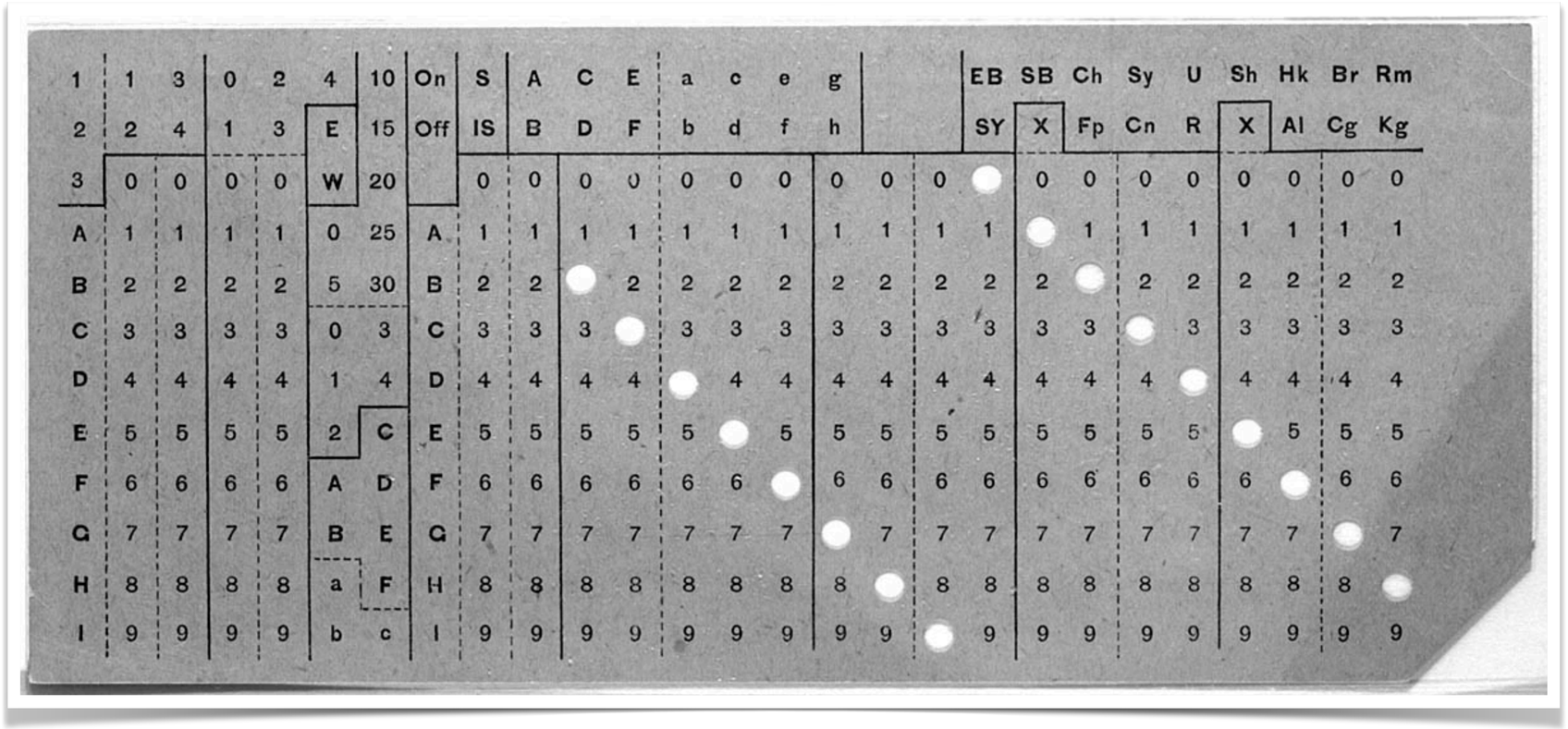


Часть паровой машины Чарлза Бэббиджа

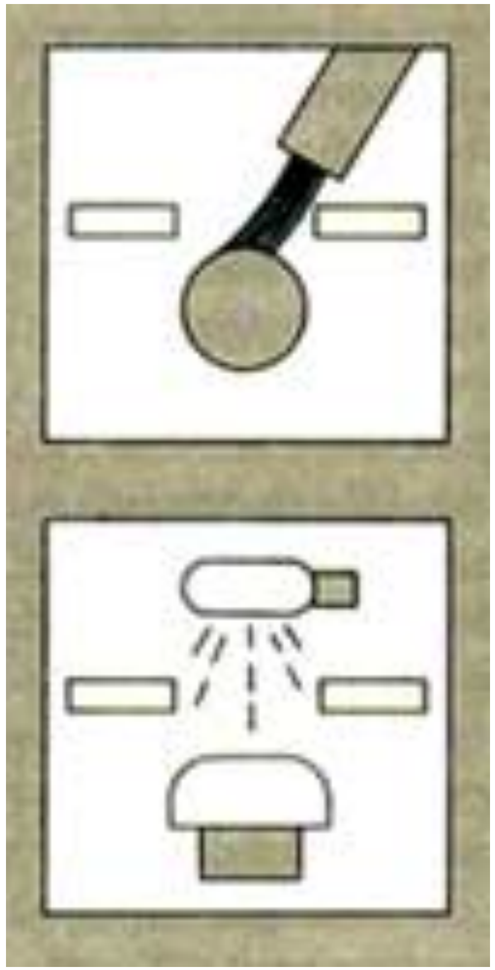


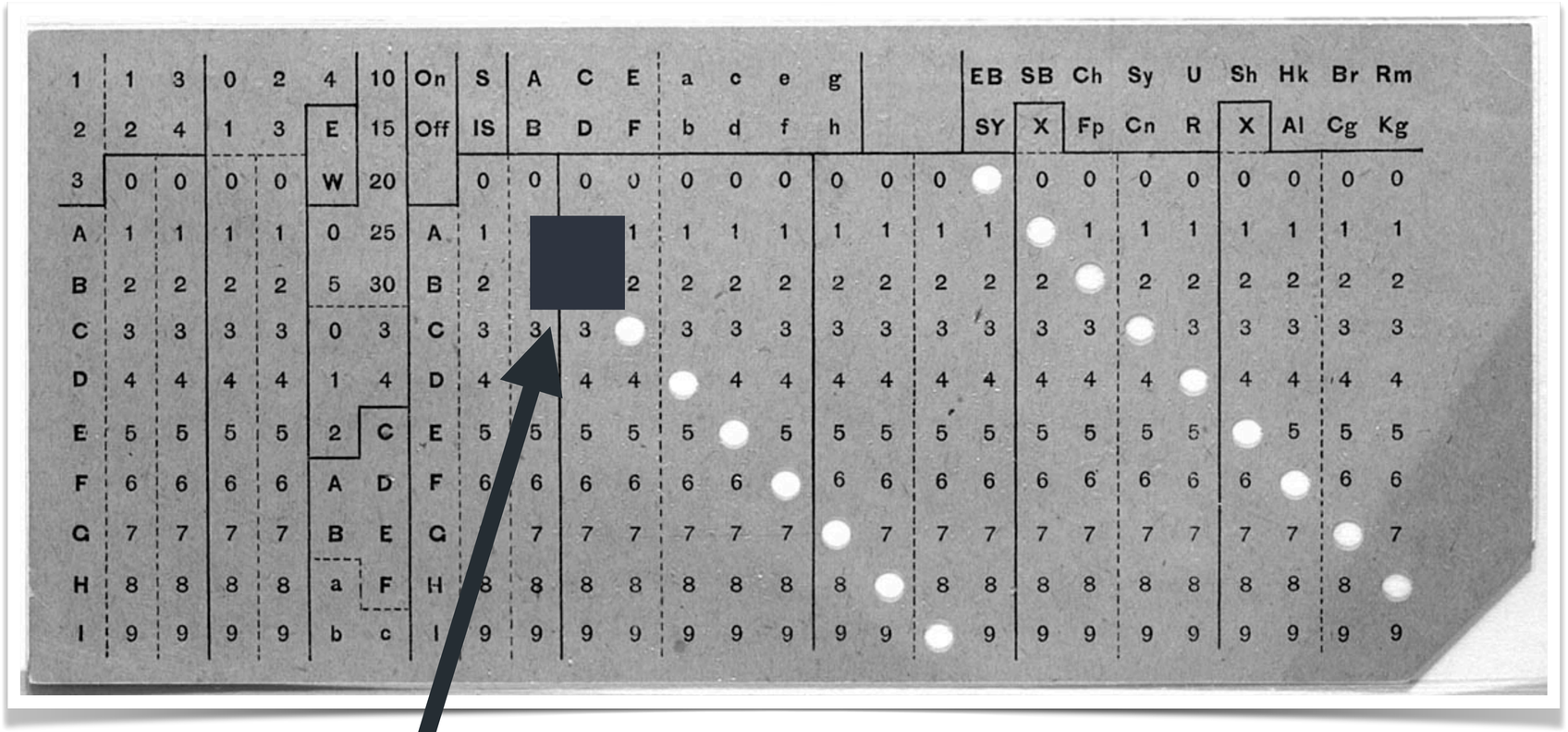
Перфокарты ткацкого станка

Табулятор Германа Холлерита (1890)


1 1 3 0 2 4 10 On S A C E a c e g EB SB Ch Sy U Sh Hk Br Rm
2 2 4 1 3 E 15 Off IS B D F b d f h SY X Fp Cn R X Al Cg Kg
3 0 0 0 0 W 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
A 1 1 1 1 0 25 A 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
B 2 2 2 2 5 30 B 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
C 3 3 3 3 0 3 C 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
D 4 4 4 4 1 4 D 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
E 5 5 5 5 2 C E 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
F 6 6 6 6 A D F 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
G 7 7 7 7 B E G 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
H 8 8 8 8 a F H 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
I 9 9 9 9 b c I 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

Перфокарта





Заплатка (англ. patch) = патч

#42 30.08.2004 в 16:00

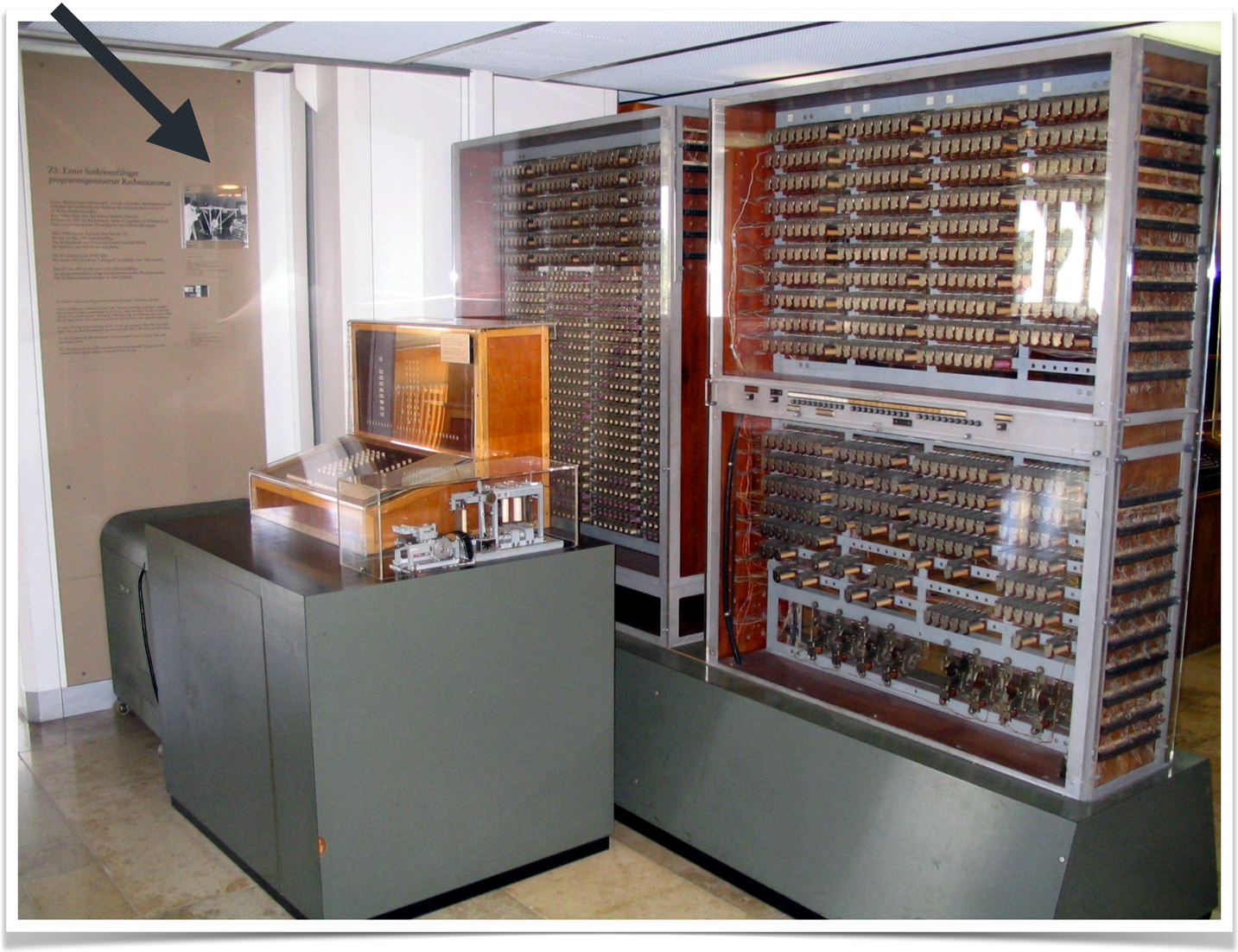
<Sashok> -- Здравствуйте, это канал об аниме? -- Да. -- Как мне пропатчить KDE2 под FreeBSD?

Комикс по мотивам цитаты

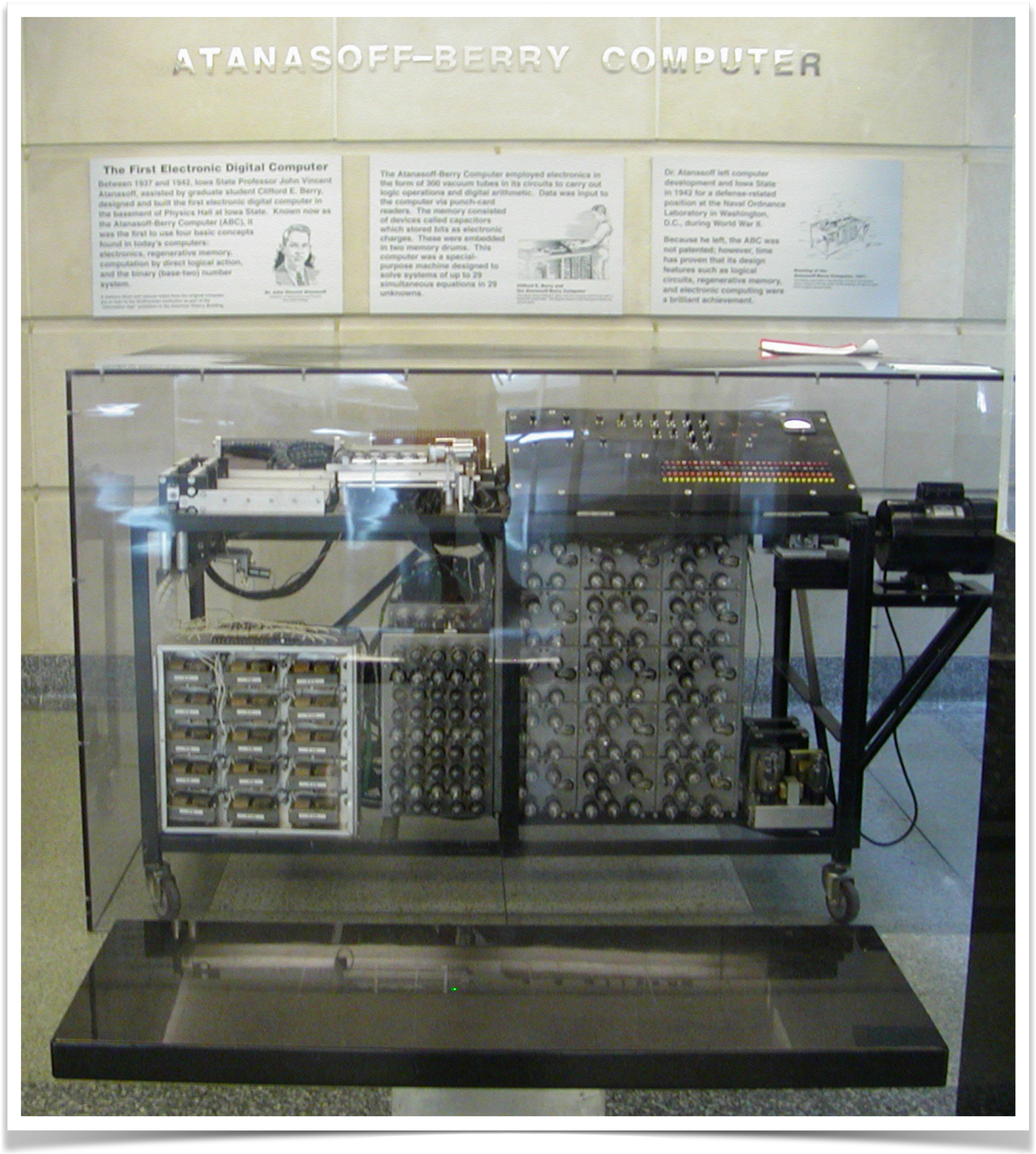
[:||:] [−] 125779 [+]


THE TABULATING MACHINE COMPANY
GENERAL OFFICES - 270 BROADWAY
NEW YORK, N. Y.

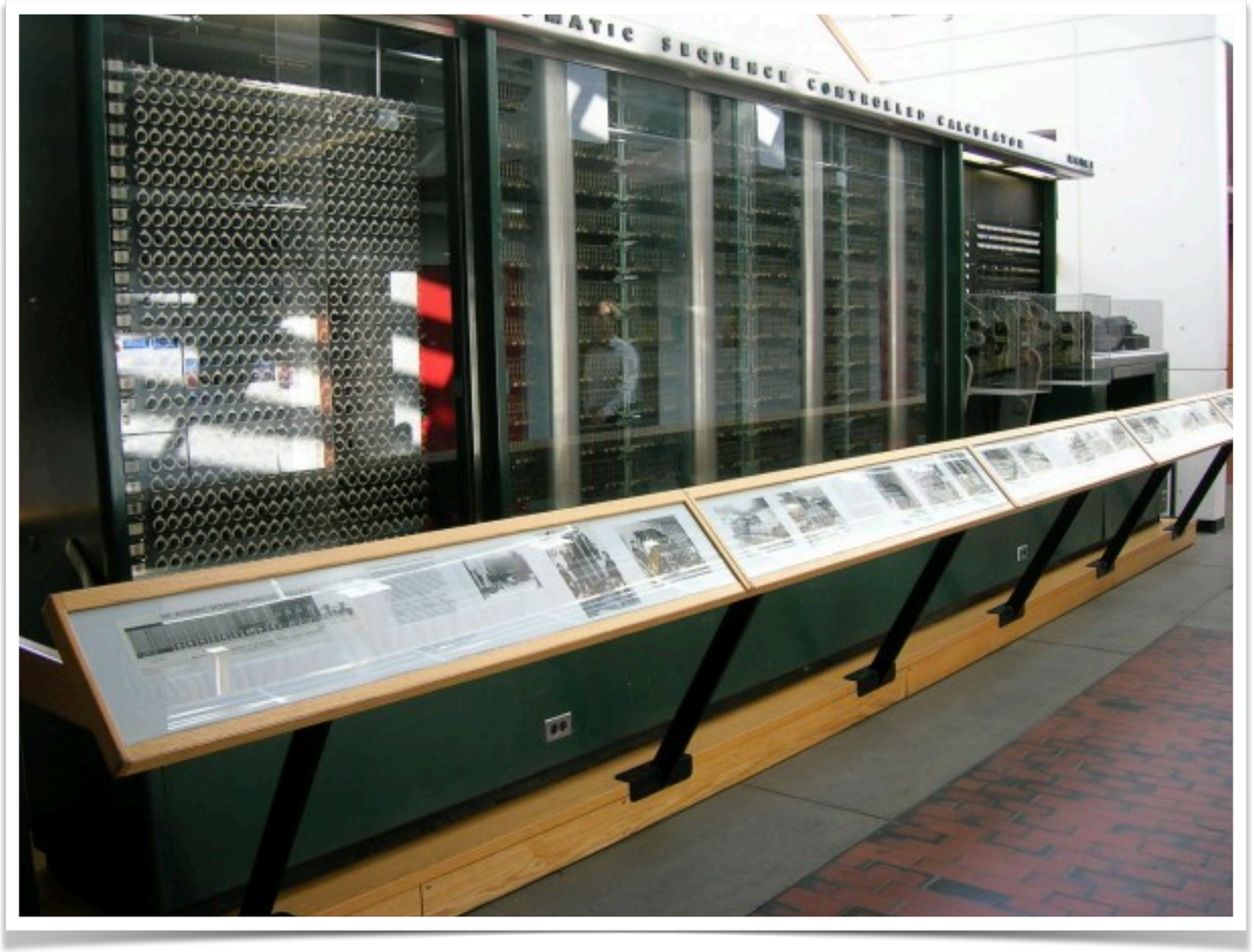
Автор: Конрад Цузе



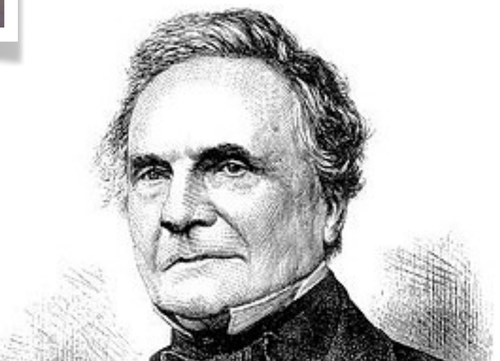
Z3 (1941)

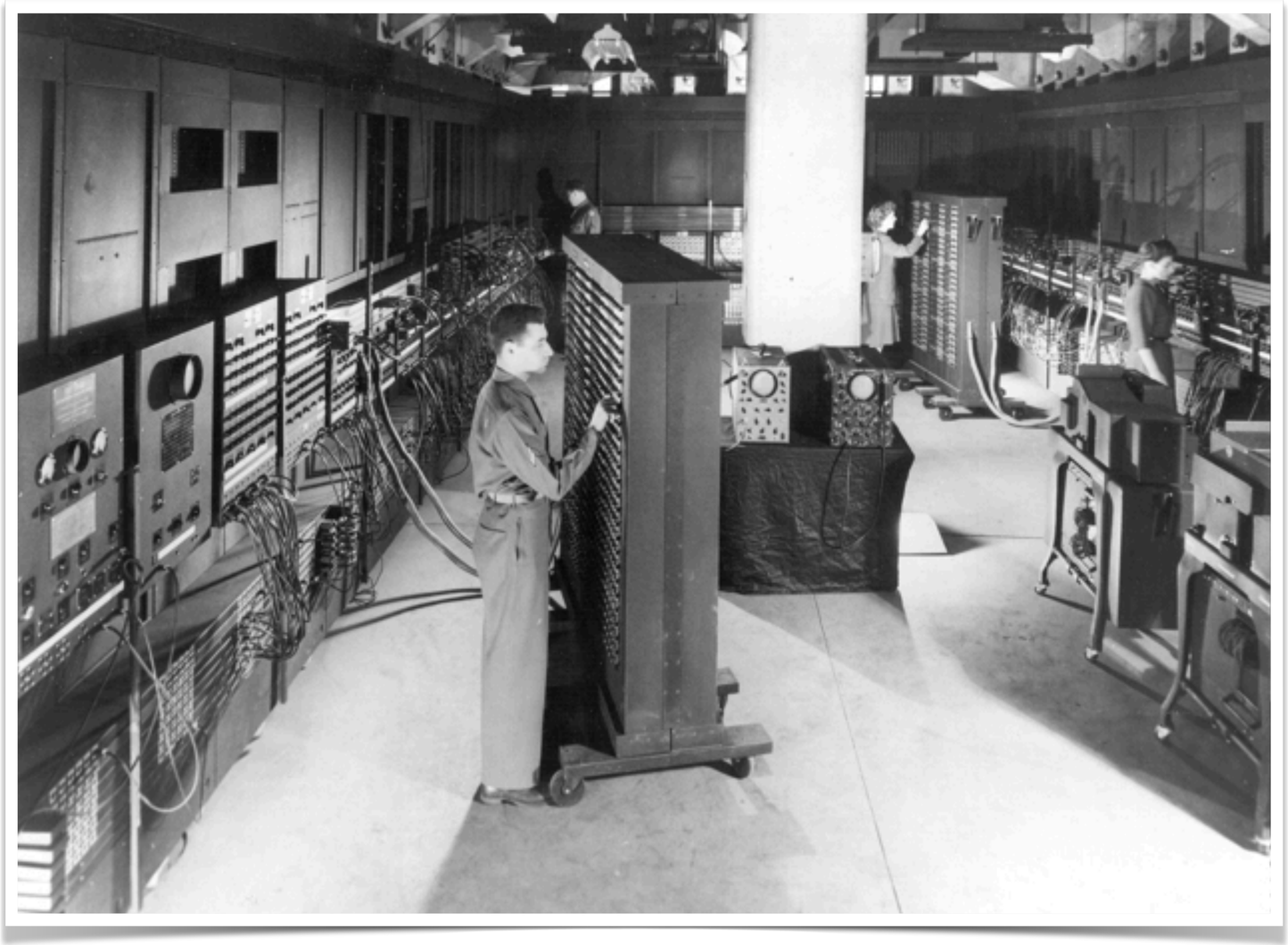

ATANASOFF-BERRY COMPUTER
The First Electronic Digital Computer
Between 1937 and 1942, Iowa State Professor John Vincent Atanasoff, assisted by graduate student Clifford E. Berry, designed and built the first electronic digital computer in the basement of Physics Hall at Iowa State. Known now as the Atanasoff-Berry Computer (ABC), it was the first to use four basic concepts found in today's computers: electronics, regenerative memory, computation by direct logical action, and the binary (base-two) number system.
The Atanasoff-Berry Computer employed electronics in the form of 300 vacuum tubes in its circuits to carry out logic operations and digital arithmetic. Data was input to the computer via punch-card readers. The memory consisted of devices called capacitors which stored bits as electronic charges. These were embedded in two memory drums. This computer was a special-purpose machine designed to solve systems of up to 29 simultaneous equations in 29 unknowns.
Dr. Atanasoff left computer development and Iowa State in 1942 for a defense-related position at the Naval Ordnance Laboratory in Washington, D.C., during World War II.
Because he left, the ABC was not patented; however, time has proven that its design features such as logical circuits, regenerative memory, and electronic computing were a brilliant achievement.

ABC（1944）



MARK I (1944)





ENIAC (1945)

Элвис Пресли

Чак Берри

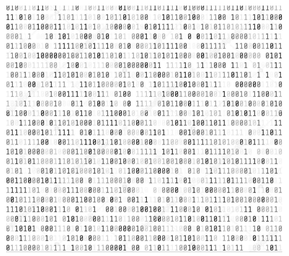
Мы тут

50 60 70 80 90 00 10 20

Мадди Уотерс

Джонни Кэш

Литтл Ричард

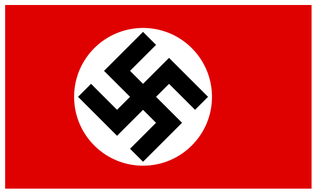


Программирование
в машинных кодах

Мы тут

50 60 70 80 90 00 10 20

Plankalkül
(~1945, Конрад Цузе)



Plankalkül
(~1945)

```
1 P1 max3 (V0[:8.0],V1[:8.0],V2[:8.0]) → R0[:8.0]
2 max(V0[:8.0],V1[:8.0]) → Z1[:8.0]
3 max(Z1[:8.0],V2[:8.0]) → R0[:8.0]
4 END
5 P2 max (V0[:8.0],V1[:8.0]) → R0[:8.0]
6 V0[:8.0] → Z1[:8.0]
7 (Z1[:8.0] < V1[:8.0]) → V1[:8.0] → Z1[:8.0]
8 Z1[:8.0] → R0[:8.0]
9 END
```

Парадигма программирования — способ программирования

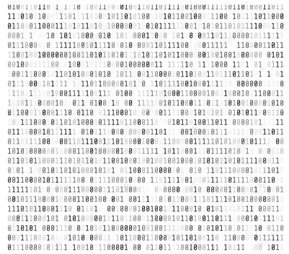
Обычно **языки программирования** поддерживают разные парадигмы программирования

Императивное программирование (парадигма)

программирование с помощью явно написанной последовательности команд, которые меняют состояние системы

NB: Парадигмы помогают контролировать сложность!

IBM 704 (1954)



Программирование
в машинных кодах

Мы тут

50 60 70 80 90 00 10 20

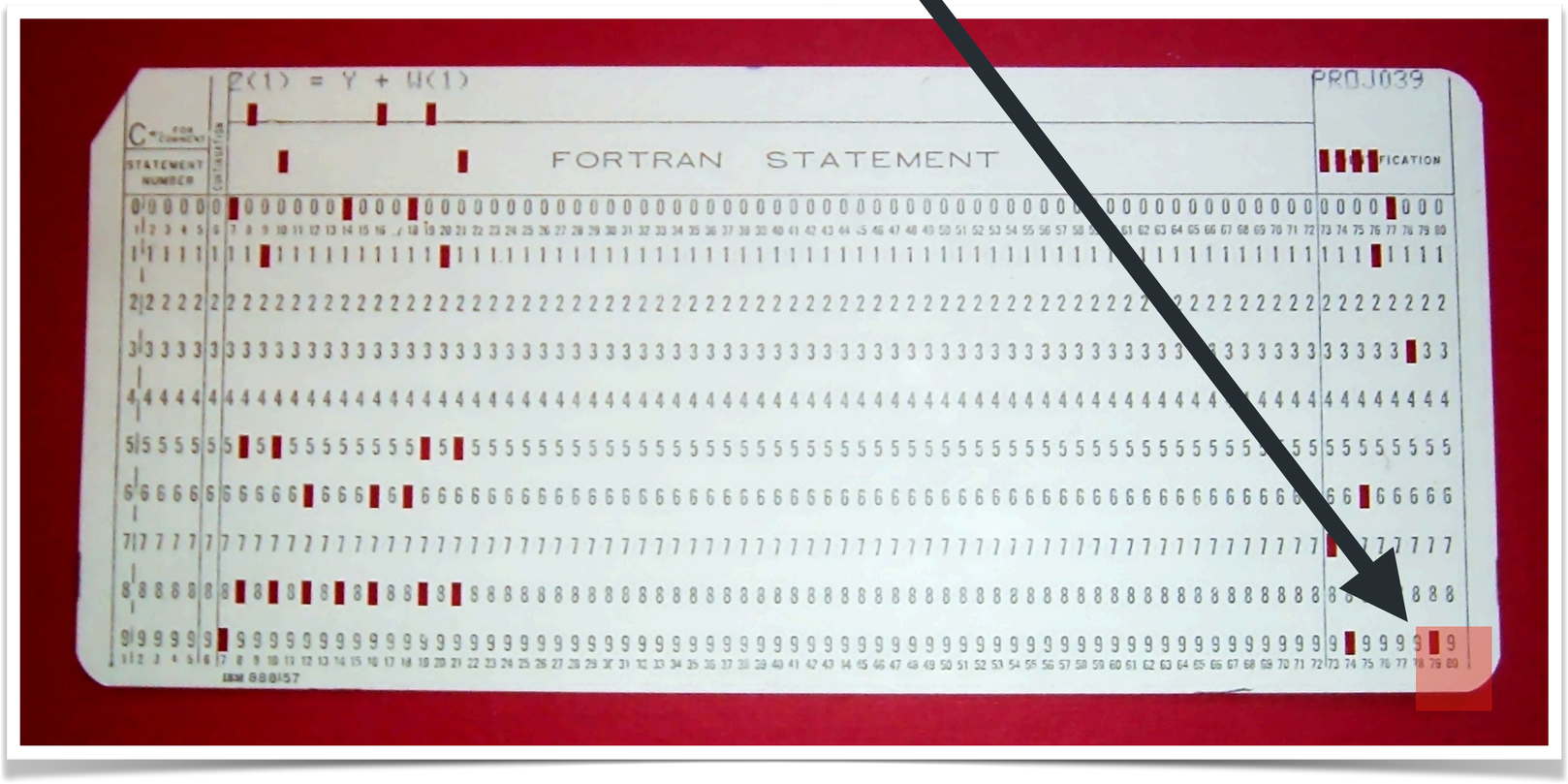
FORTRAN
(1957)

IBM

FORTRAN

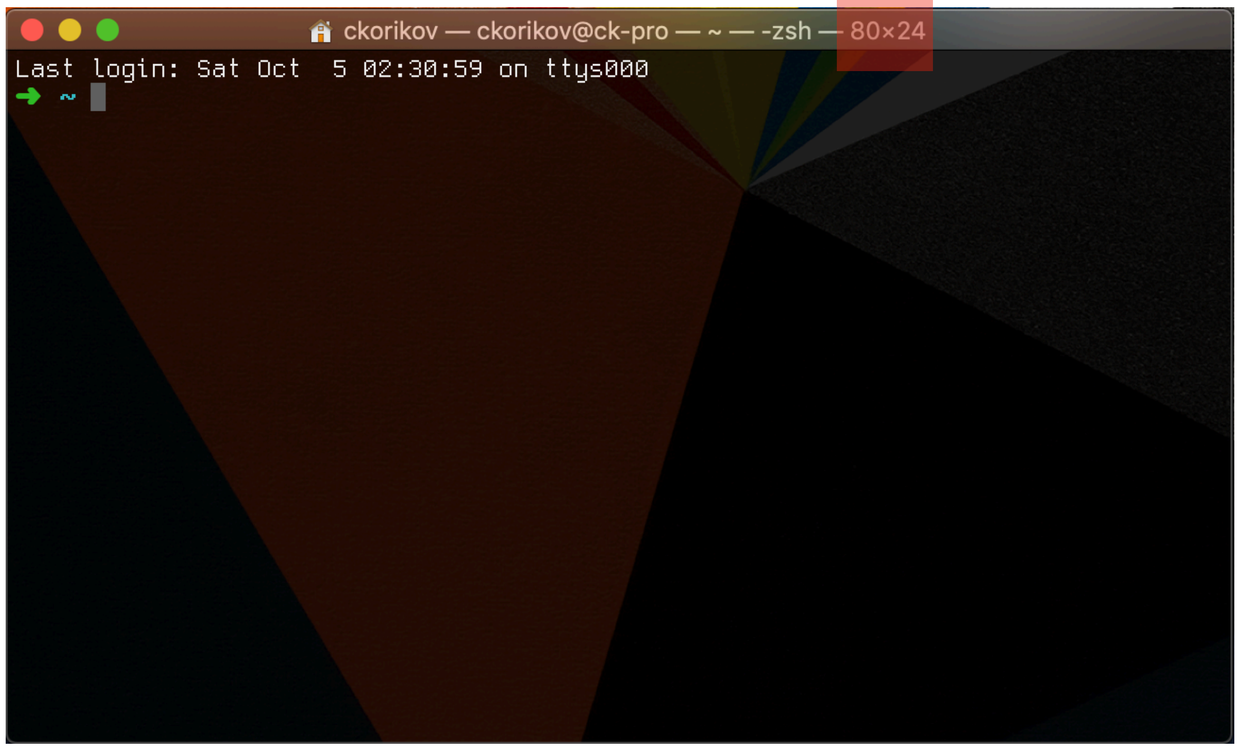
```
! AREA OF A TRIANGLE - HERON'S FORMULA
! INPUT - CARD READER UNIT 5, INTEGER INPUT
! OUTPUT -
! INTEGER VARIABLES START WITH I,J,K,L,M OR N
      READ(5,501) IA,IB,IC
  501 FORMAT(3I5)
      IF(IA.EQ.0 .OR. IB.EQ.0 .OR. IC.EQ.0) STOP 1
      S = (IA + IB + IC) / 2.0
      AREA = SQRT( S * (S - IA) * (S - IB) * (S - IC) )
      WRITE(6,601) IA,IB,IC,AREA
  601 FORMAT(4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,
     $13H SQUARE UNITS)
      STOP
      END
```

80 столбцов

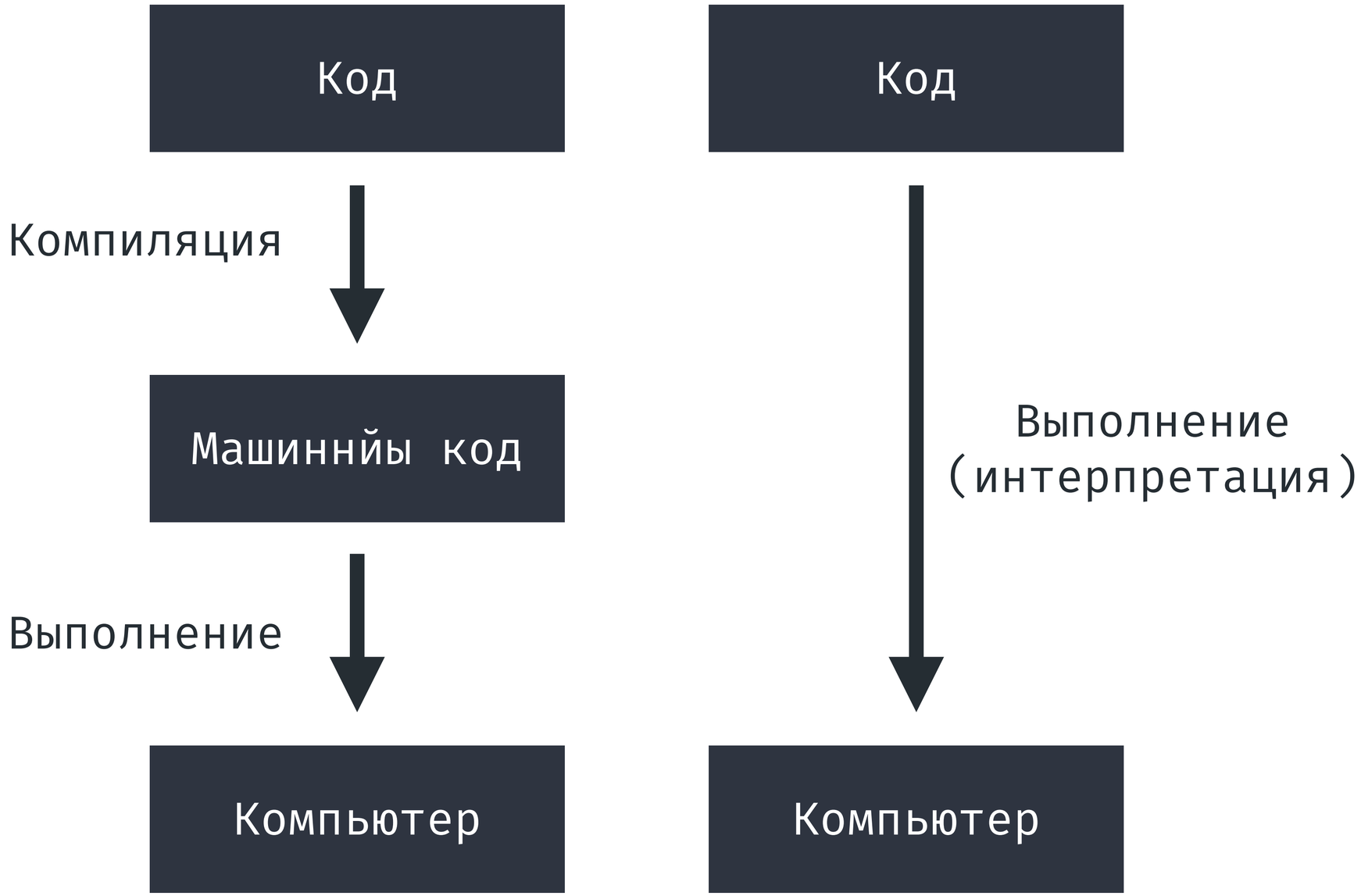


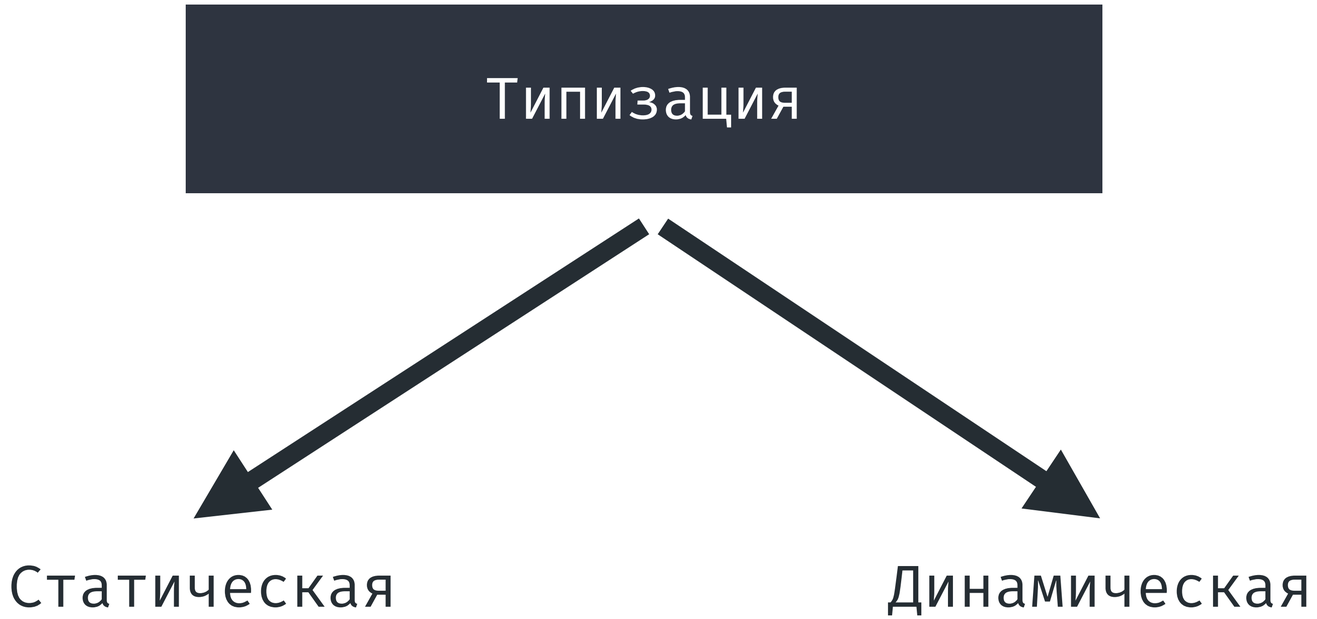
Строчка кода на Fortran на перфокарте

80 столбцов


ckorikov — ckorikov@ck-pro — ~ — -zsh — 80×24
Last login: Sat Oct 5 02:30:59 on ttys000
➜ ~

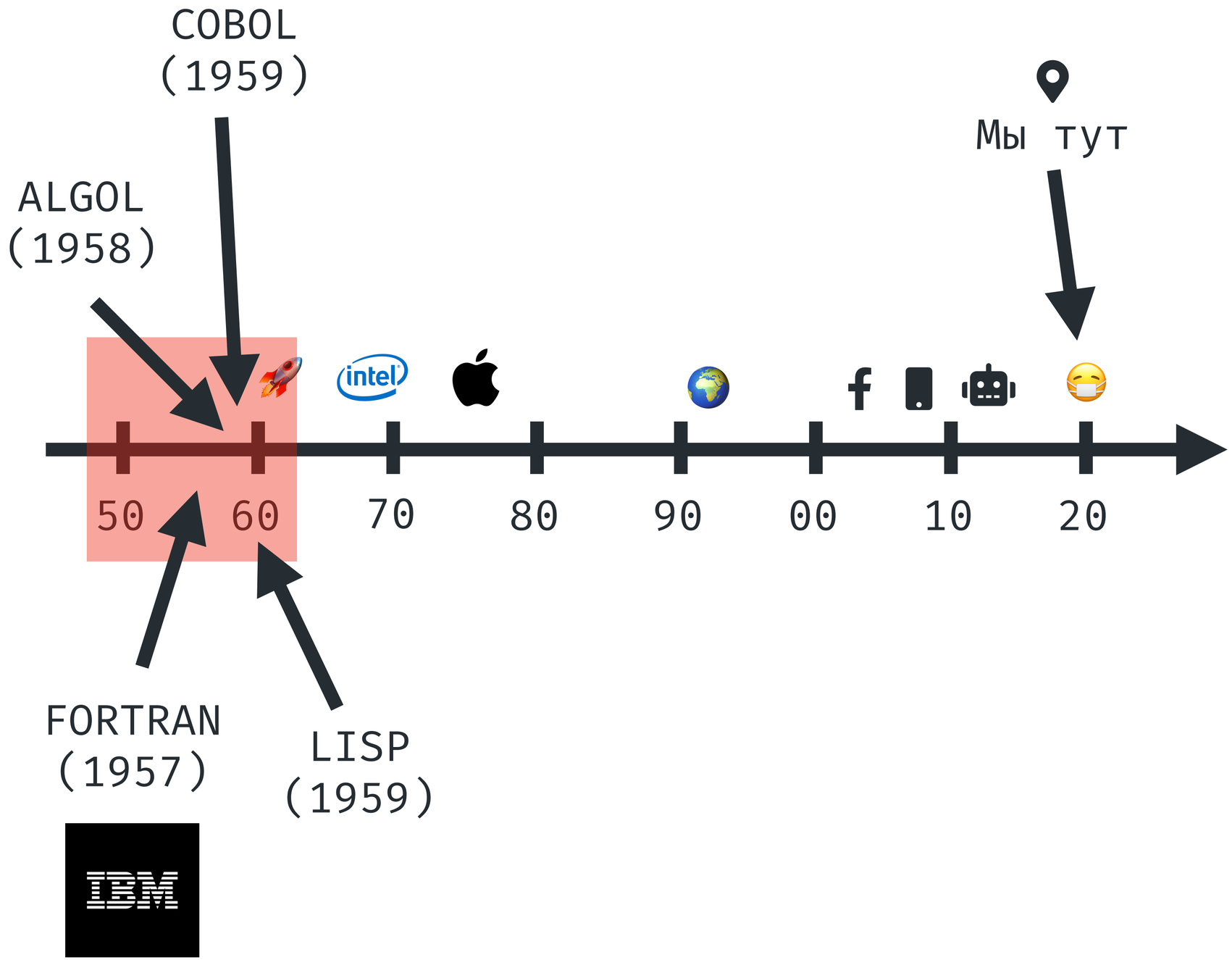
Компилятор Fortran


Код
Компиляция
Машиннйы код
Выполнение
Компьютер
Код
Выполнение
(интерпретация)
Компьютер


Типизация
Статическая
Динамическая

Типизация

Строгая  Слабая


COBOL
(1959)
ALGOL
(1958)
Мы тут
50
60
70
80
90
00
10
20
FORTRAN
(1957)
LISP
(1959)
IBM

LISP

```lisp
(defun comb (m list fn)
  (labels ((comb1 (l c m)
             (when (>= (length l) m)
               (if (zerop m) (return-from comb1 (funcall fn c)))
               (comb1 (cdr l) c m)
               (comb1 (cdr l) (cons (first l) c) (1- m)))))
    (comb1 list nil m)))

(comb 3 '(0 1 2 3 4 5) #'print)
```

Функциональное программирование (парадигма)

способ программирования с помощью функций в математическом смысле: нет побочных эффектов

ALGOL

```
BEGIN
FILE F (KIND=REMOTE);
EBCDIC ARRAY E [0:11];
REPLACE E BY "HELLO WORLD!";
WHILE TRUE DO
  BEGIN
  WRITE (F, *, E);
  END;
END.
```

COBOL

```cobol
       IDENTIFICATION DIVISION.
       PROGRAM-ID. A-Plus-B.

       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  A       PIC S9(5).
       01  B       PIC S9(5).

       01  A-B-Sum PIC S9(5).

       PROCEDURE DIVISION.
           ACCEPT A
           ACCEPT B

           ADD A TO B GIVING A-B-Sum

           DISPLAY A-B-Sum

           GOBACK
           .
```

The Beatles

Джими Хендрикс

Мы тут

50 60 70 80 90 00 10 20

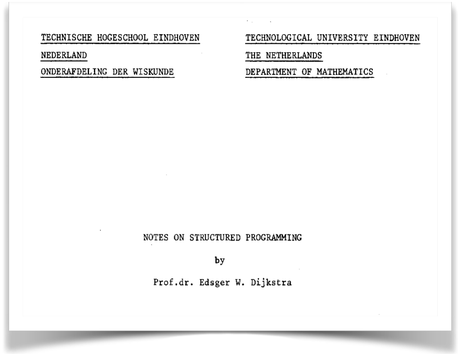
Led Zeppelin

The Rolling Stones

The Stooges

IBM System/360 (1964)

NCR 315 (1965)



TECHNISCHE HOGESCHOOL EINDHOVEN
NEDERLAND
ONDERAFDELING DER WISKUNDE

TECHNOLOGICAL UNIVERSITY EINDHOVEN
THE NETHERLANDS
DEPARTMENT OF MATHEMATICS

NOTES ON STRUCTURED PROGRAMMING

by

Prof.dr. Edsger W. Dijkstra

Структурное
программирование
(Дейкстра, 1968)

Мы тут

50 60 70 80 90 00 10 20

BASIC
(1964)

Basic

```
input "N = ",f
limit = 500                         # set upper limit - can be changed, removed
f = int(f)
if f > limit then f = limit
a = 0 : b = 1 : c = 0 : n = 0       # initial values


while n < f
    print n + chr(9) + c    # chr(9) = tab
    a = b
    b = c
    c = a + b
    n += 1

end while

print " "
print n + chr(9) + c
```

Структурное программирование (парадигма)

императивное программирование в виде иерархии структурных блоков на ветвлениях и циклах (без goto)

Процедурное программирование (парадигма)

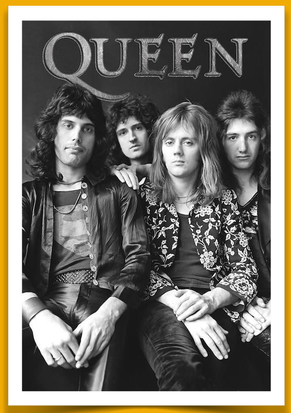
структурное программирование с разбиением программы на процедуры (функции)

Black Sabbath

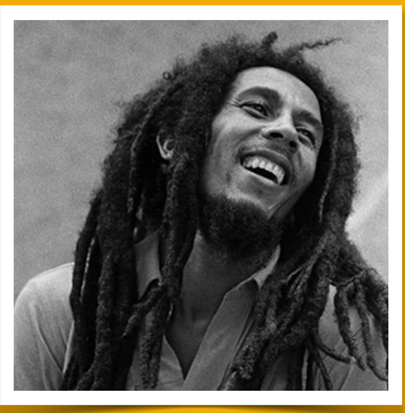
Мы тут

50 60 70 80 90 00 10 20

ABBA

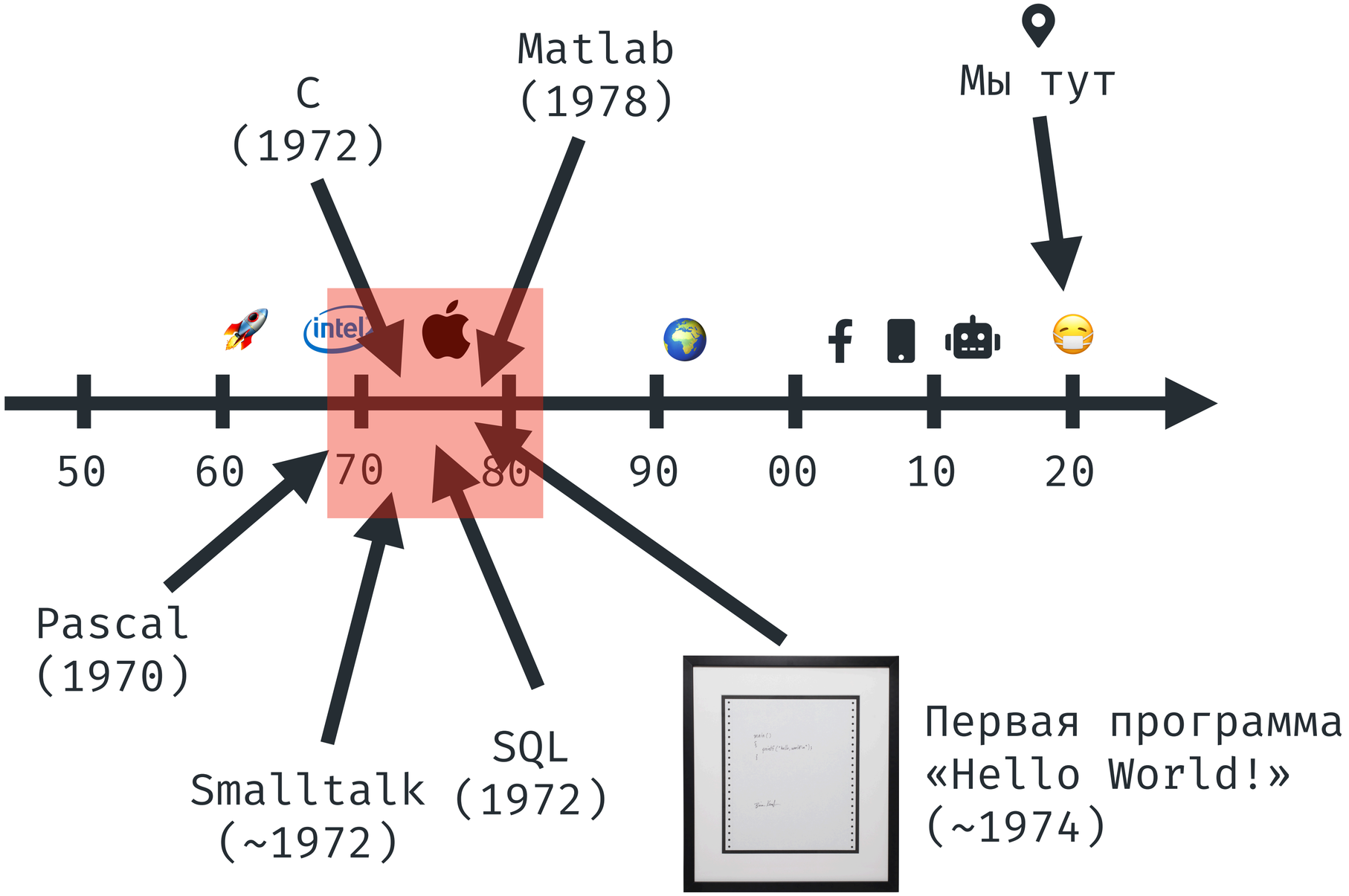


Queen



Боб Марли

IBM System/32 aka IBM 5320 (1975)


Мы тут
C
(1972)
Matlab
(1978)
50
60
70
80
90
00
10
20
Pascal
(1970)
Smalltalk
(~1972)
SQL
(1972)
Первая программа
«Hello World!»
(~1974)

Декларативное программирование (парадигма)

способ программирования, когда указывается результат работы без явного указания шагов решения

SQL

```sql
SELECT LAT_N, CITY, TEMP_F
FROM STATS, STATION
WHERE MONTH = 7
AND STATS.ID = STATION.ID
ORDER BY TEMP_F;
```

C

```c
long long int fibb(int n) {
    int fnow = 0, fnext = 1, tempf;
    while(--n>0){
        tempf = fnow + fnext;
        fnow = fnext;
        fnext = tempf;
        }
        return fnext;
}
```

Matlab

```matlab
function F = fibonacci(n)

    Fn = [1 0]; %Fn(1) is F_{n-2}, Fn(2) is F_{n-1}
    F = 0; %F is F_{n}

    for i = (1:abs(n))
        Fn(2) = F;
        F = sum(Fn);
        Fn(1) = Fn(2);
    end

    if n < 0
        F = F*((-1)^(n+1));
    end

end
```

Pascal

```pascal
function fib(n: integer): integer;
 begin
  if (n = 0) or (n = 1)
   then
    fib := n
   else
    fib := fib(n-1) + fib(n-2)
 end;
```

Smalltalk

```
 1 |fibo|
 2 fibo := [ :i |
 3    |ac t|
 4    ac := Array new: 2.
 5    ac at: 1 put: 0 ; at: 2 put: 1.
 6    ( i < 2 )
 7      ifTrue: [ ac at: (i+1) ]
 8      ifFalse: [
 9         2 to: i do: [ :l |
10           t := (ac at: 2).
11           ac at: 2 put: ( (ac at: 1) + (ac at: 2) ).
12           ac at: 1 put: t
13         ].
14         ac at: 2.
15      ]
16 ].
17
18 0 to: 10 do: [ :i |
19   (fibo value: i) displayNl
20 ]
```

Объектно-ориентированное программирование (парадигма)

способ программирования, в котором моделируется задача объектами

The Sugarhill Gang

Metallica

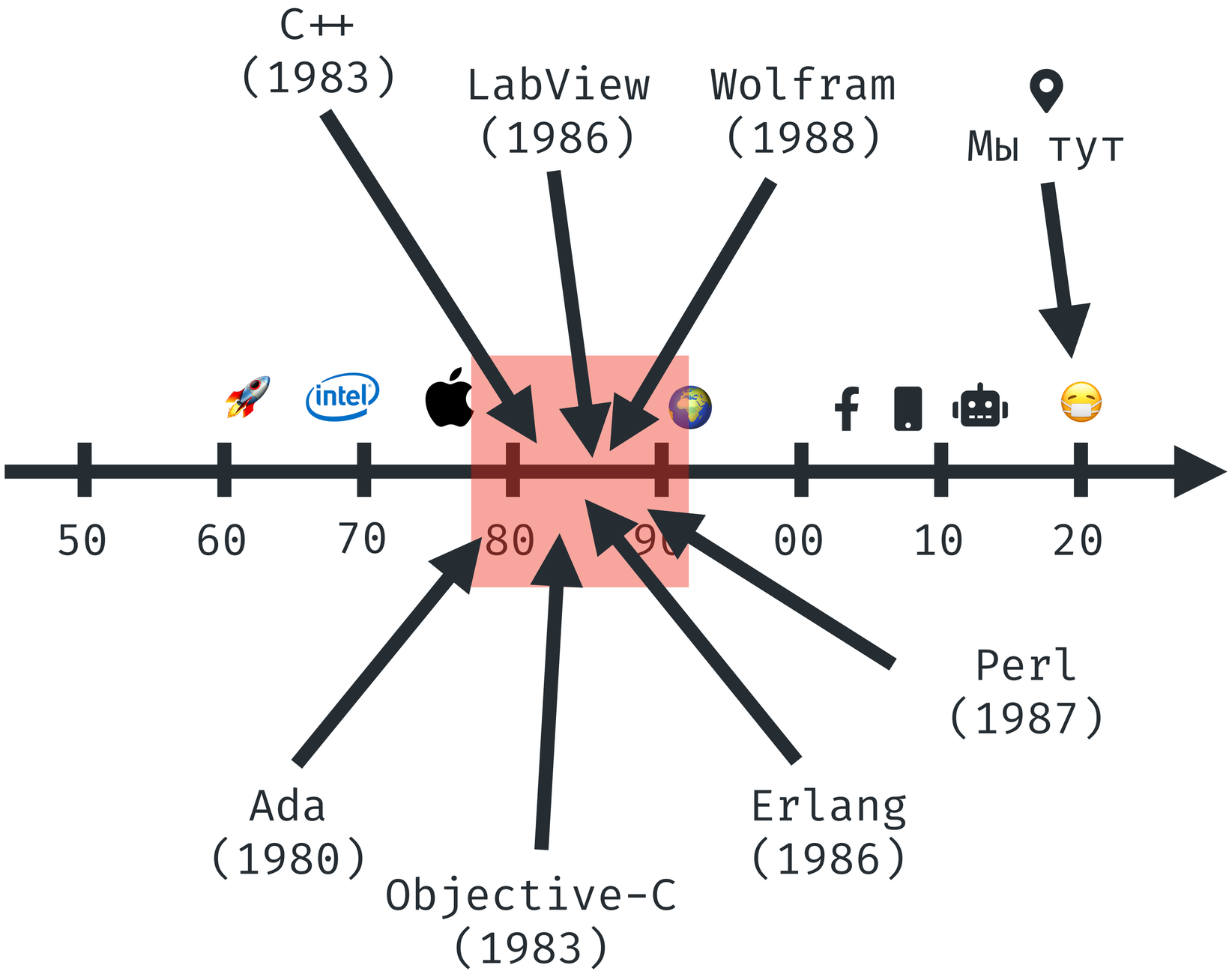
Мы тут

50 60 70 80 90 00 10 20

MTV

Хип-хоп сообщество

IBM 5150 PC (1981)


C++
(1983)
LabView
(1986)
Wolfram
(1988)
Мы тут
intel
50
60
70
80
90
00
10
20
Ada
(1980)
Objective-C
(1983)
Erlang
(1986)
Perl
(1987)

Ada

```ada
with Ada.Text_IO; use Ada.Text_IO;
procedure Hello is
begin
   Put_Line ("Hello, world!");
end Hello;
```

Objective-C

```objc
-(long)fibonacci:(int)position
{
    long result = 0;
    if (position < 2) {
        result = position;
    } else {
        result = [self fibonacci:(position -1)] + [self fibonacci:(position -2)];
    }
    return result;
}
```

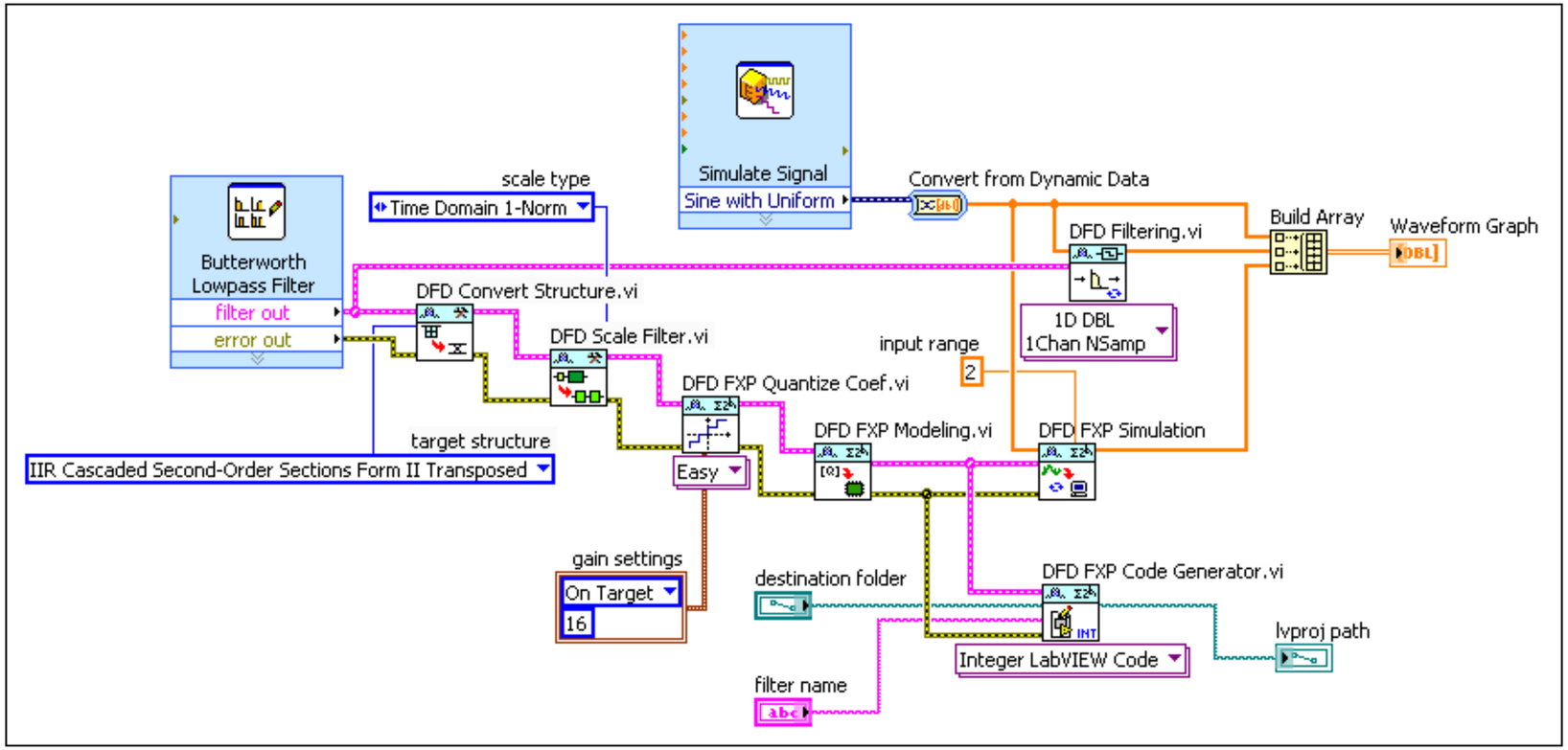
C++

```cpp
#include <iostream>

int main()
{
        unsigned int a = 1, b = 1;
        unsigned int target = 48;
        for(unsigned int n = 3; n <= target; ++n)
        {
                unsigned int fib = a + b;
                std::cout << "F("<< n << ") = " << fib << std::endl;
                a = b;
                b = fib;
        }

        return 0;
}
```

Erlang

```erlang
-module(comb).
-compile(export_all).

comb(0,_) ->
    [[]];
comb(_,[]) ->
    [];
comb(N,[H|T]) ->
    [[H|L] || L <- comb(N-1,T)]++comb(N,T).
```

LabView (1986)



Perl

```perl
sub fib (Int $n --> Int) {
    $n > 1 or return $n;
    my ($prev, $this) = 0, 1;
    ($prev, $this) = $this, $this + $prev for 1 ..^ $n;
    return $this;
}
```

Wolfram

```
BinarySearch[x_List, val_] := Module[{lo = 1, hi = Length@x, mid},
  While[lo <= hi,
   mid = lo + Round@((hi - lo)/2);
   Which[x[[mid]] > val, hi = mid - 1,
    x[[mid]] < val, lo = mid + 1,
    True, Return[mid]
    ];
   ];
  Return[-1];
  ]
```

Spice Girls

The Prodigy

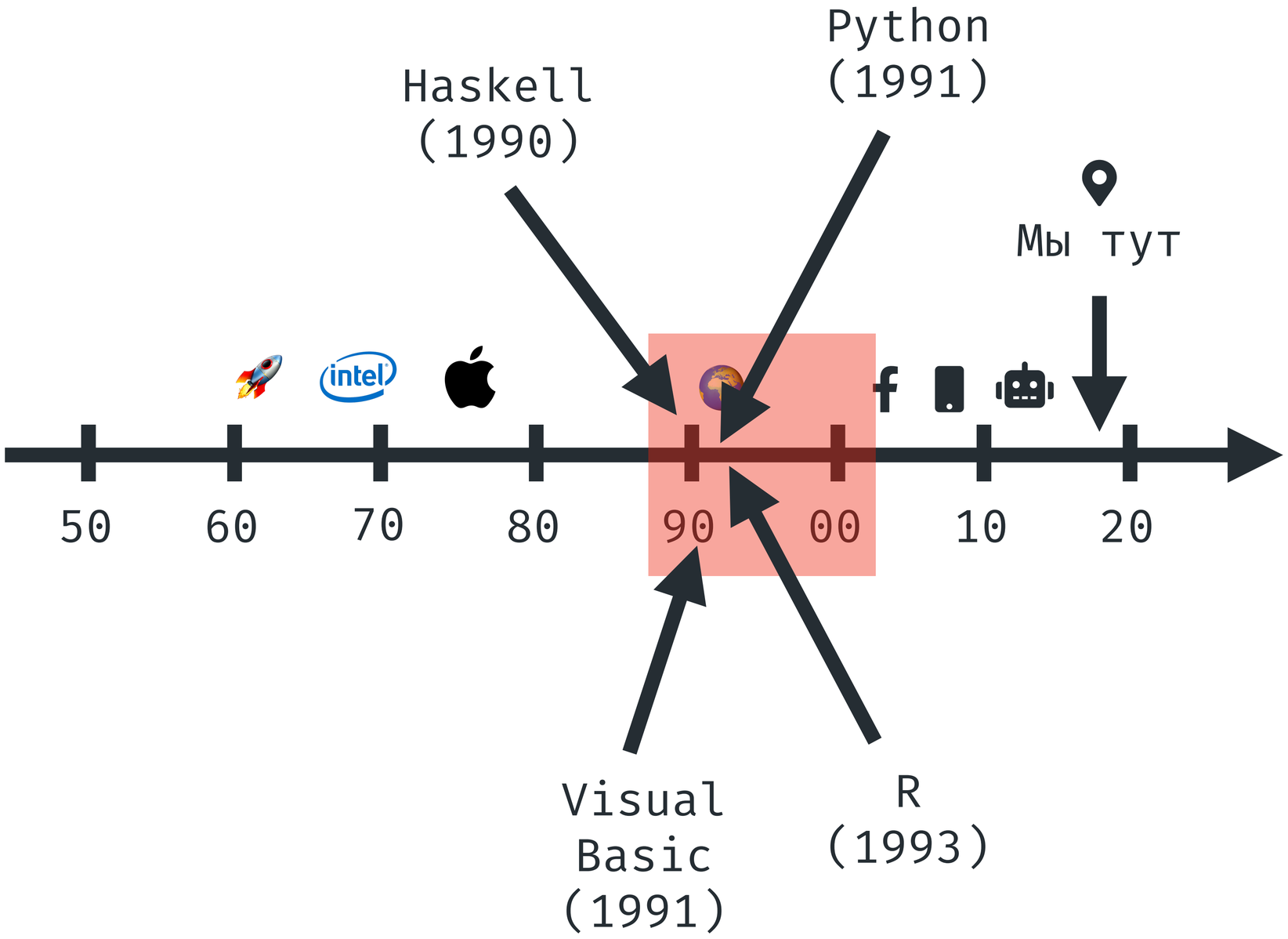
50 60 70 80 90 00 10 20

Nirvana

Backstreet Boys

Селин Дион

Apple The PowerBook 540c (1994)


Haskell
(1990)
Python
(1991)
Мы тут
50
60
70
80
90
00
10
20
Visual
Basic
(1991)
R
(1993)

Haskell

```haskell
fib x =
  if x < 1
    then 0
    else if x < 2
           then 1
           else fib (x - 1) + fib (x - 2)
```

Python

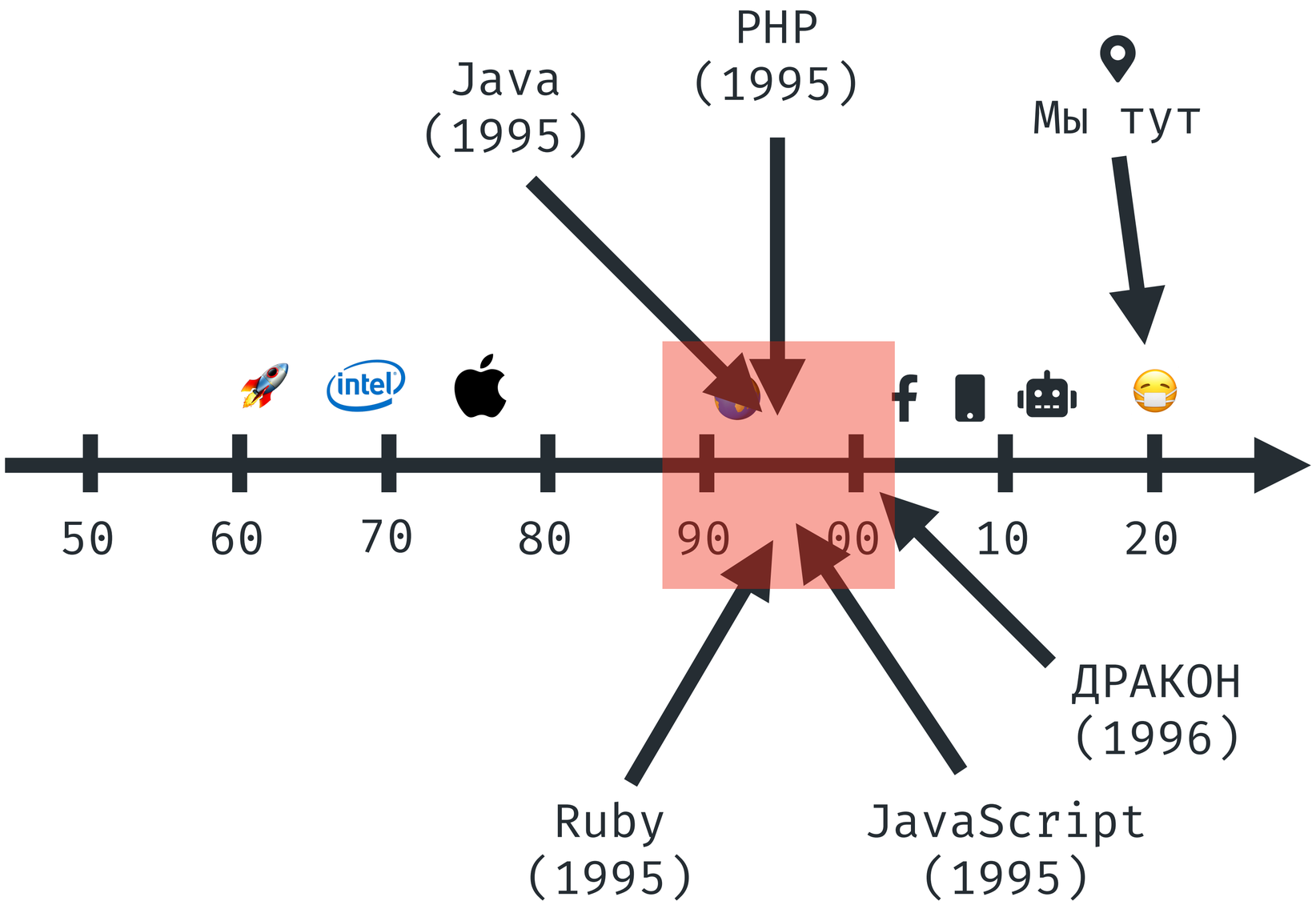
```python
def fibIter(n):
    if n < 2:
        return n
    fibPrev = 1
    fib = 1
    for num in xrange(2, n):
        fibPrev, fib = fib, fib + fibPrev
    return fib
```

VisualBasic

```
 1 Sub fibonacci()
 2     Const n = 139
 3     Dim i As Integer
 4     Dim f1 As Variant, f2 As Variant, f3 As Variant 'for Decimal
 5     f1 = CDec(0): f2 = CDec(1) 'for Decimal setting
 6     Debug.Print "fibo("; 0; ")="; f1
 7     Debug.Print "fibo("; 1; ")="; f2
 8     For i = 2 To n
 9         f3 = f1 + f2
10         Debug.Print "fibo("; i; ")="; f3
11         f1 = f2
12         f2 = f3
13     Next i
14 End Sub 'fibonacci
```

R

```r
fib=function(n,x=c(0,1)) {
   if (abs(n)>1) for (i in seq(abs(n)-1)) x=c(x[2],sum(x))
   if (n<0) return(x[2]*(-1)^(abs(n)-1)) else if (n) return(x[2]) else
 return(0)
}

sapply(seq(-31,31),fib)
```


Java
(1995)
PHP
(1995)
Мы тут
50
60
70
80
90
00
10
20
Ruby
(1995)
JavaScript
(1995)
ДРАКОН
(1996)

Java

```java
public static long itFibN(int n)
{
 if (n < 2)
  return n;
 long ans = 0;
 long n1 = 0;
 long n2 = 1;
 for(n--; n > 0; n--)
 {
  ans = n1 + n2;
  n1 = n2;
  n2 = ans;
 }
 return ans;
}
```

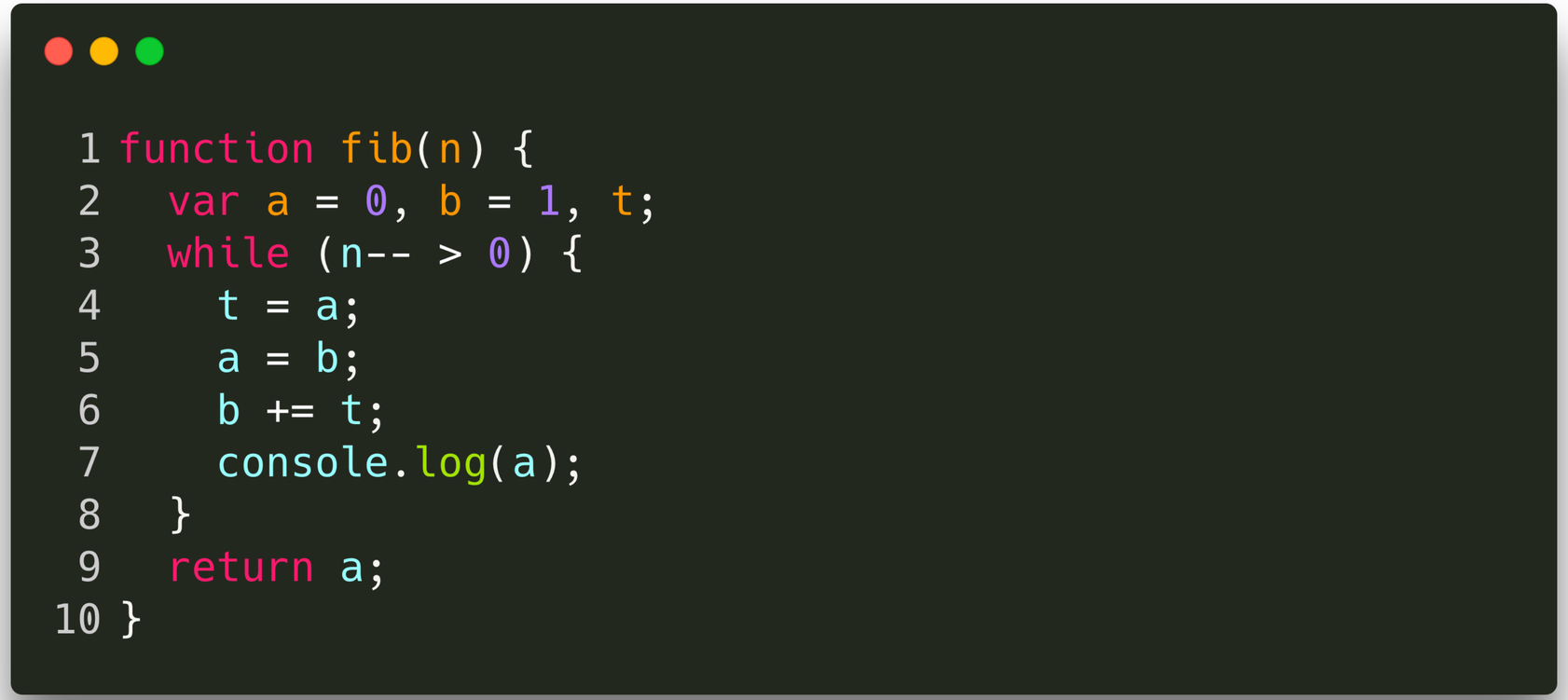
PHP

```php
function fibIter($n) {
    if ($n < 2) {
        return $n;
    }
    $fibPrev = 0;
    $fib = 1;
    foreach (range(1, $n-1) as $i) {
        list($fibPrev, $fib) = array($fib, $fib + $fibPrev);
    }
    return $fib;
}
```

Ruby

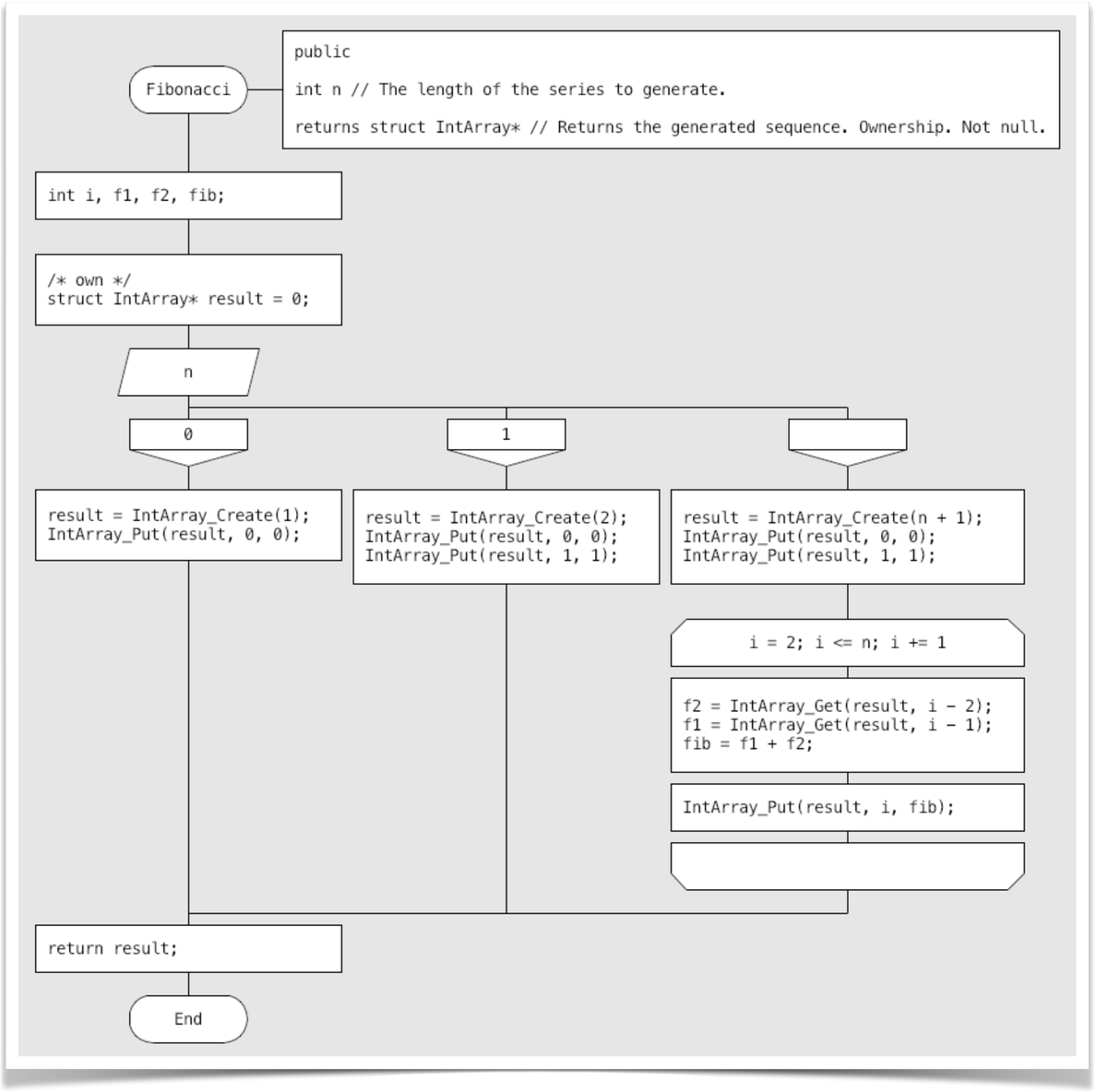
```ruby
def fib(n, sequence=[1])
  return sequence.last if n == 0

  current_number, last_number = sequence.last(2)
  sequence << current_number + (last_number or 0)

  fib(n-1, sequence)
end
```

JavaScript



```
 1 function fib(n) {
 2   var a = 0, b = 1, t;
 3   while (n-- > 0) {
 4     t = a;
 5     a = b;
 6     b += t;
 7     console.log(a);
 8   }
 9   return a;
10 }
```

ДРАКОН



Дружелюбный русский алгоритмический язык, который обеспечивает наглядность

Britney Spears

Arctic Monkeys

Мы тут

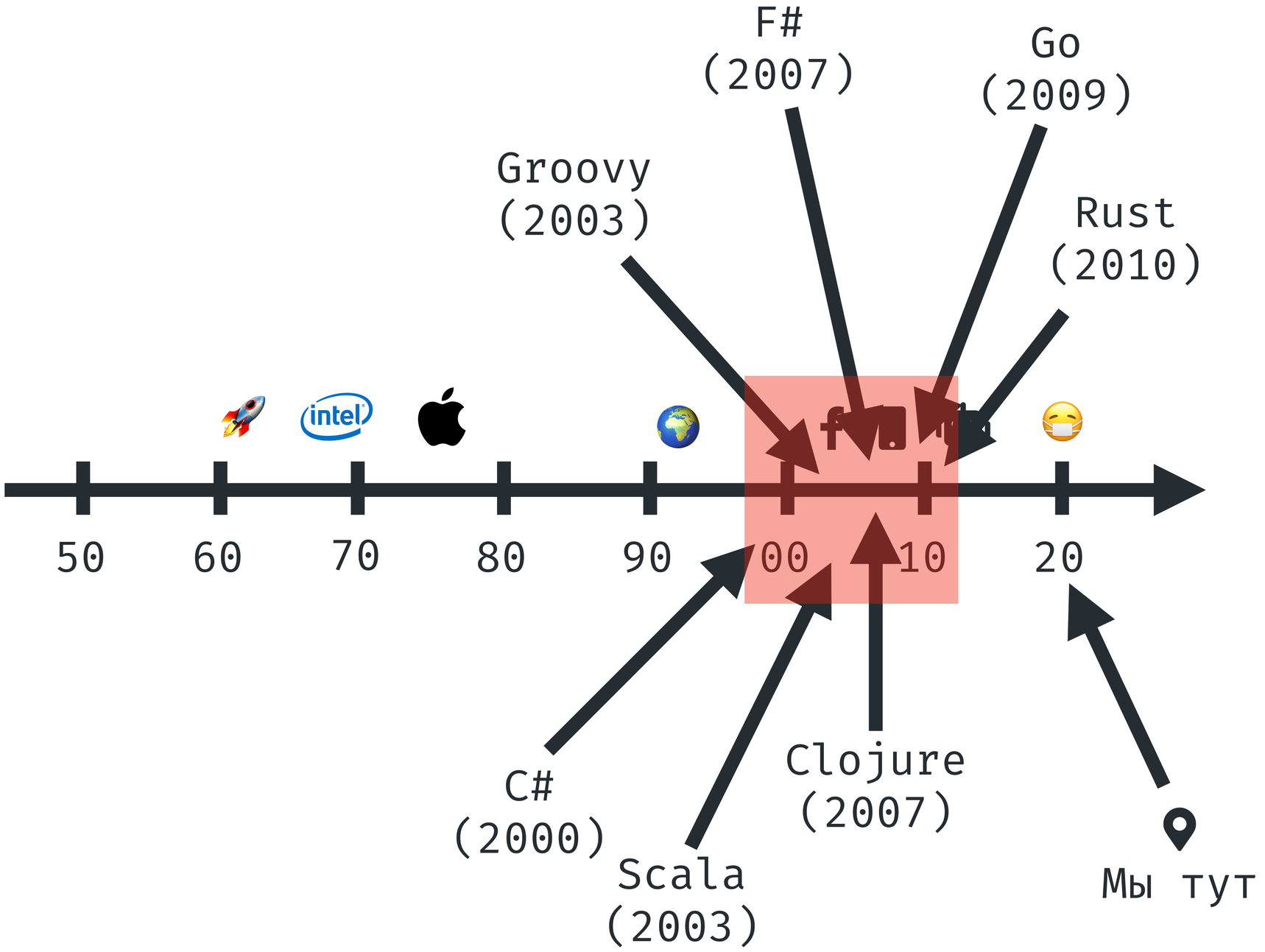
50 60 70 80 90 00 10 20

Eminem

Interpol

Tokio Hotel

Apple iPhone (2007)


F#
(2007)
Go
(2009)
Groovy
(2003)
Rust
(2010)
50
60
70
80
90
00
10
20
C#
(2000)
Scala
(2003)
Clojure
(2007)
Мы тут

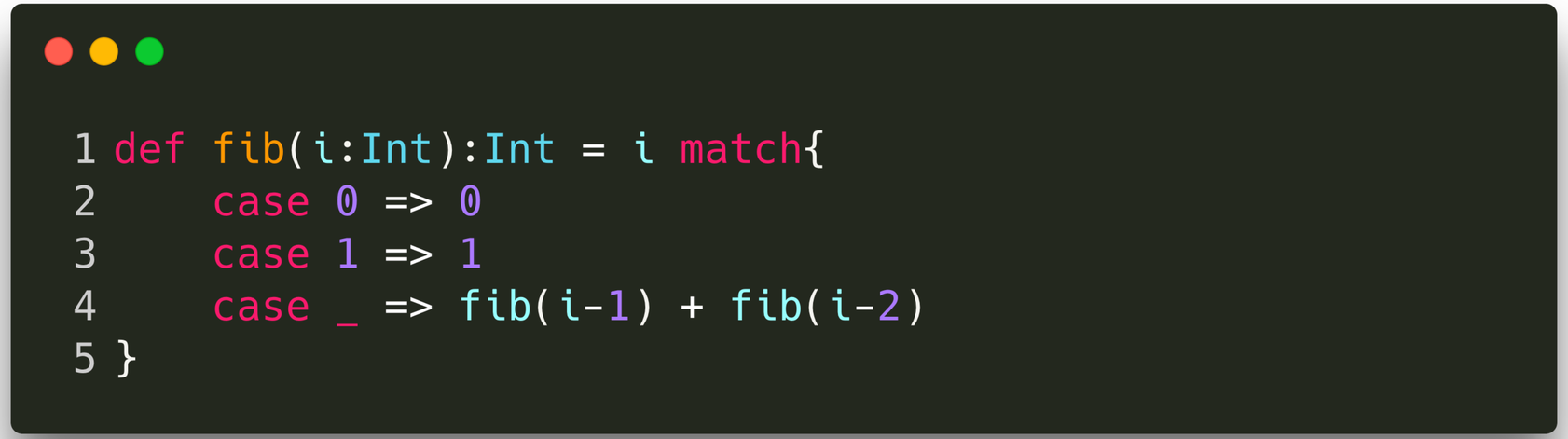
C#

```
 1 public static ulong Fib(uint x) {
 2     if (x == 0) return 0;
 3
 4     ulong prev = 0;
 5     ulong next = 1;
 6     for (int i = 1; i < x; i++)
 7     {
 8         ulong sum = prev + next;
 9         prev = next;
10         next = sum;
11     }
12     return next;
13 }
```

Groovy

```groovy
def rFib
rFib = {
    it == 0   ? 0
    : it == 1 ? 1
    : it > 1  ? rFib(it-1) + rFib(it-2)
    /*it < 0*/: rFib(it+2) - rFib(it+1)

}
```

Scala



```scala
def fib(i:Int):Int = i match{
    case 0 => 0
    case 1 => 1
    case _ => fib(i-1) + fib(i-2)
}
```

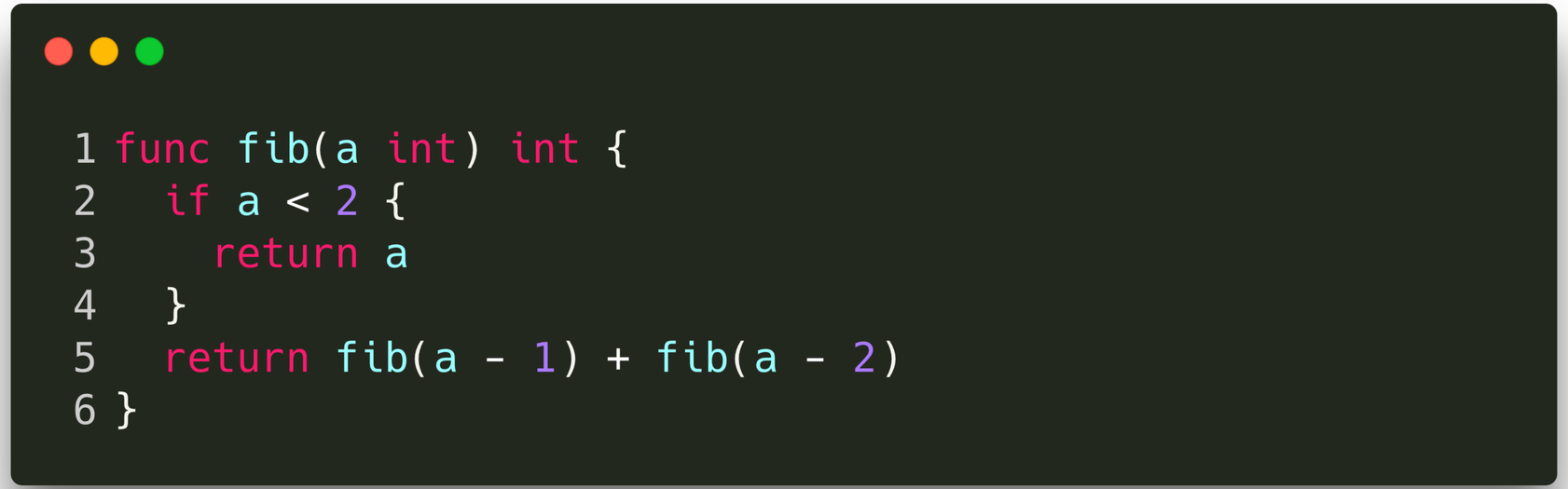
Clojure

```clojure
(def fib (lazy-cat [0 1] (map + fib (rest fib))))
```

F#

```fsharp
let fibonacci n : bigint =
  let rec f a b n =
    match n with
    | 0 -> a
    | 1 -> b
    | n -> (f b (a + b) (n - 1))
  f (bigint 0) (bigint 1) n
> fibonacci 100;;
val it : bigint = 354224848179261915075I
```

Go



```go
func fib(a int) int {
  if a < 2 {
    return a
  }
  return fib(a - 1) + fib(a - 2)
}
```

Rust

```rust
use std::mem;
fn main() {
    let mut prev = 0;
    // Rust needs this type hint for the checked_add method
    let mut curr = 1usize;

    while let Some(n) = curr.checked_add(prev) {
        prev = curr;
        curr = n;
        println!("{}", n);
    }
}
```

Rick and Morty

Kendrick Lamar

Adele

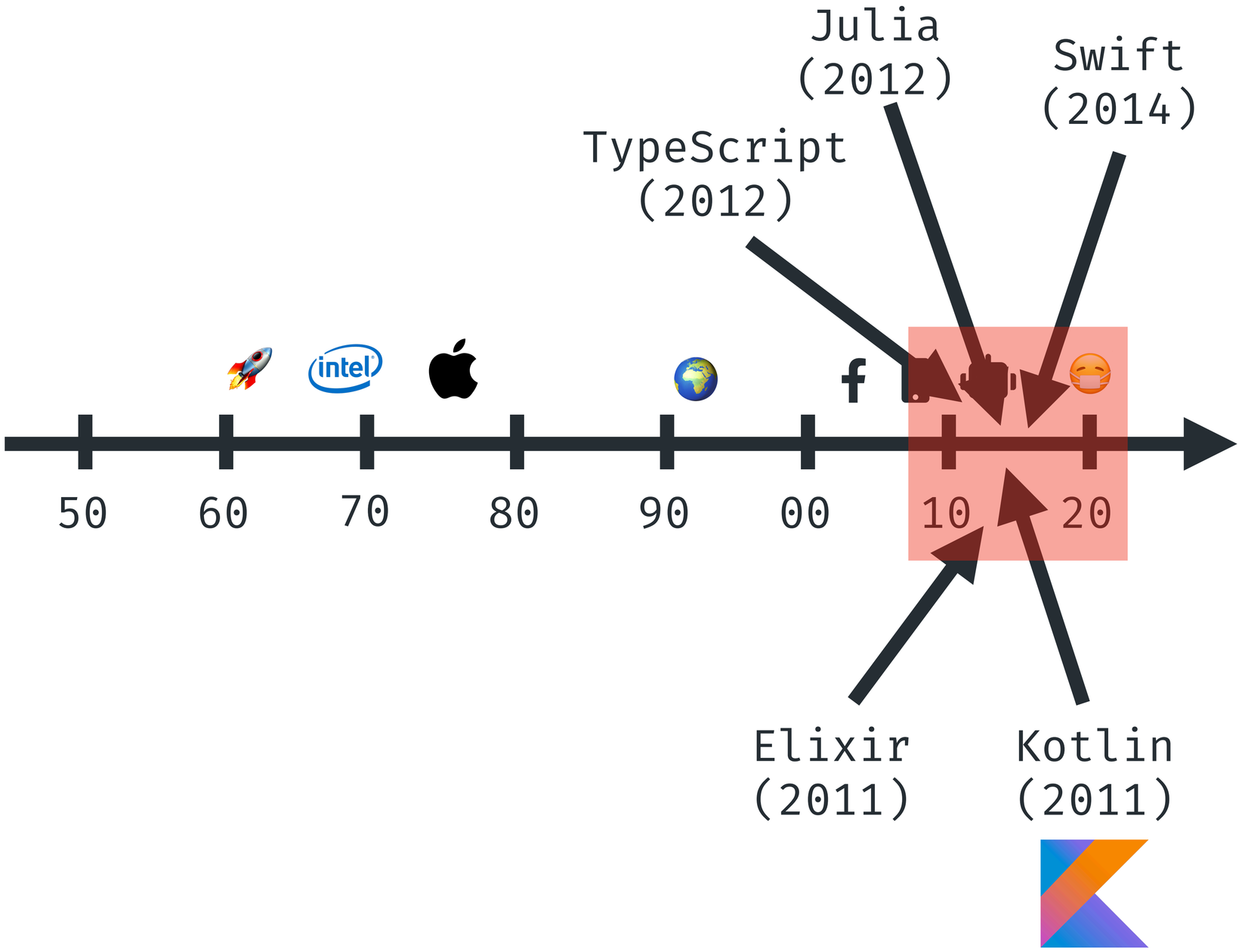
50 60 70 80 90 00 10 20

Drake

Ed Sheeran

Psy

Apple MacBook Air (2019)


Julia
(2012)
Swift
(2014)
TypeScript
(2012)
50
60
70
80
90
00
10
20
Elixir
(2011)
Kotlin
(2011)

Elixir

```elixir
defmodule Binary do
  def search(list, value), do: search(List.to_tuple(list), value, 0, length(list)-1)

  def search(_tuple, _value, low, high) when high < low, do: :not_found
  def search(tuple, value, low, high) do
    mid = div(low + high, 2)
    midval = elem(tuple, mid)
    cond do
      value <  midval -> search(tuple, value, low, mid-1)
      value >  midval -> search(tuple, value, mid+1, high)
      value == midval -> mid
    end
  end
end

list = [0,1,4,5,6,7,8,9,12,26,45,67,78,90,98,123,211,234,456,769,865,2345,3215,14345,24324]
Enum.each([0,42,45,24324,99999], fn val ->
  case Binary.search(list, val) do
    :not_found -> IO.puts "#{val} not found in list"
    index      -> IO.puts "found #{val} at index #{index}"
  end
end)
```

Kotlin

```kotlin
enum class Fibonacci {
    ITERATIVE {
        override fun invoke(n: Long) = if (n < 2) {
            n
        } else {
            var n1 = 0L
            var n2 = 1L
            var i = n
            do {
                val sum = n1 + n2
                n1 = n2
                n2 = sum
            } while (i-- > 1)
            n1
        }
    },
    RECURSIVE {
        override fun invoke(n: Long): Long = if (n < 2) n else this(n - 1) + this(n - 2)
    };

    abstract operator fun invoke(n: Long): Long
}

fun main(a: Array<String>) {
    val r = 0..30L
    Fibonacci.values().forEach {
        print("${it.name}: ")
        r.forEach { i -> print(" " + it(i)) }
        println()
    }
}
```

TypeScript

```typescript
function replace(input: string, key: number) : string {
    return input.replace(/([a-z])/g,
        ($1) => String.fromCharCode(($1.charCodeAt(0) + key + 26 - 97) % 26 + 97)
        ).replace(/([A-Z])/g,
        ($1) => String.fromCharCode(($1.charCodeAt(0) + key + 26 - 65) % 26 + 65));
}

// test
var str = 'The five boxing wizards jump quickly';
var encoded = replace(str, 3);
var decoded = replace(encoded, -3);

console.log('Enciphered: ' + encoded);
console.log('Deciphered: ' + decoded);
```

Julia

```julia
function binarysearch(lst::Vector{T}, val::T) where T
    low = 1
    high = length(lst)
    while low ≤ high
        mid = (low + high) ÷ 2
        if lst[mid] > val
            high = mid - 1
        elseif lst[mid] < val
            low = mid + 1
        else
            return mid
        end
    end
    return 0
end
```

Swift

```swift
func binarySearch<T: Comparable>(xs: [T], x: T) -> Int? {
  var recurse: ((Int, Int) -> Int?)!
  recurse = {(low, high) in switch (low + high) / 2 {
    case _ where high < low: return nil
    case let mid where xs[mid] > x: return recurse(low, mid - 1)
    case let mid where xs[mid] < x: return recurse(mid + 1, high)
    case let mid: return mid
  }}
  return recurse(0, xs.count - 1)
}
```

Mother Tongues

Tracing the roots of computer languages through the ages

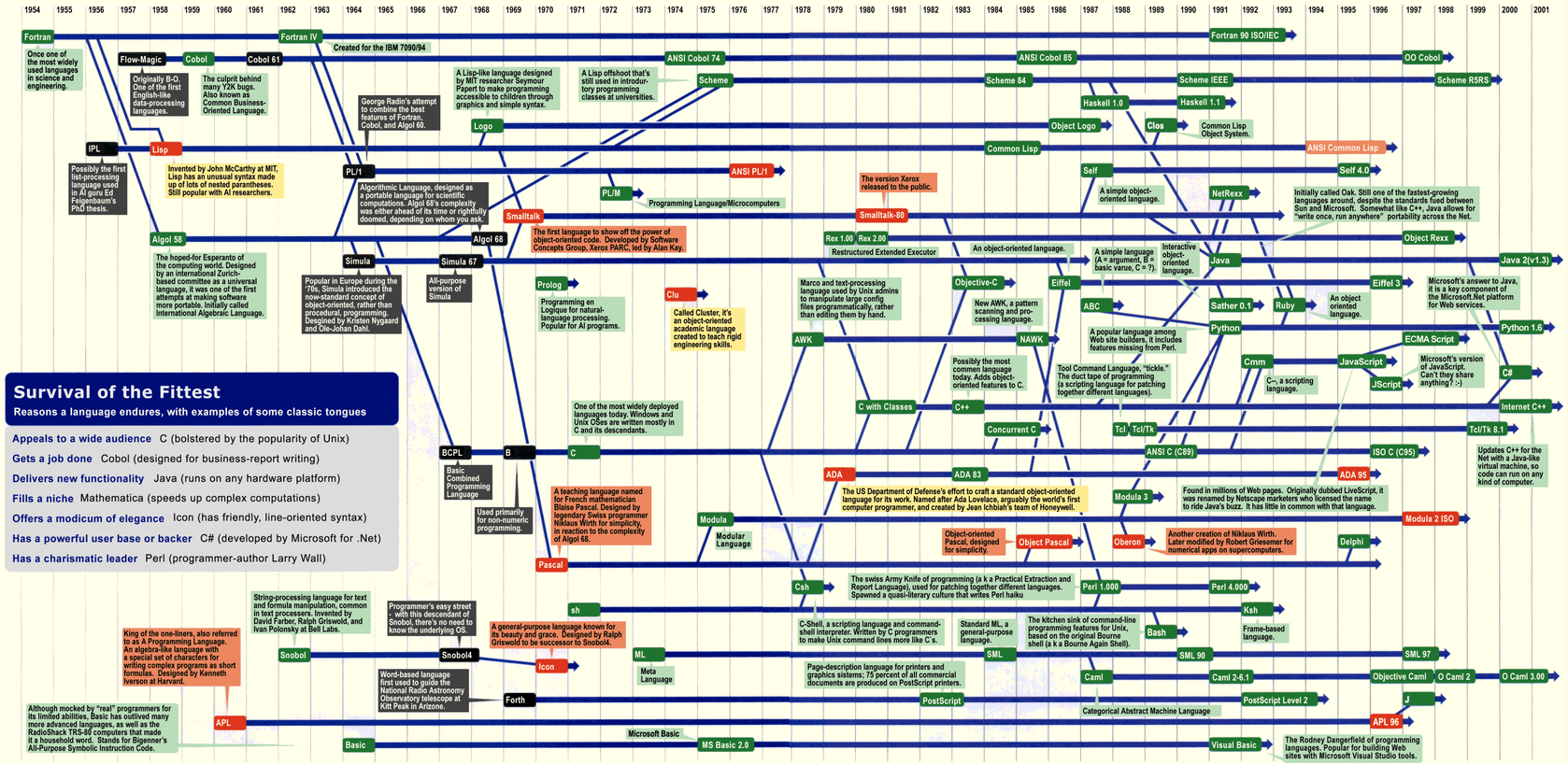
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html*. - Michael Mendeno

Key

- 1954 Year Introduced
- **Active:** thousands of users
- **Protected:** taught at universities; compilers available
- **Endangered:** usage dropping off
- **Extinct:** no known active users or up-to-date compilers
- **Lineage continues**



Survival of the Fittest

Reasons a language endures, with examples of some classic tongues

- **Appeals to a wide audience** C (bolstered by the popularity of Unix)
- **Gets a job done** Cobol (designed for business-report writing)
- **Delivers new functionality** Java (runs on any hardware platform)
- **Fills a niche** Mathematica (speeds up complex computations)
- **Offers a modicum of elegance** Icon (has friendly, line-oriented syntax)
- **Has a powerful user base or backer** C# (developed by Microsoft for .Net)
- **Has a charismatic leader** Perl (programmer-author Larry Wall)

Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University