Языки программирования. Лекция 1. История. Зачем и почему?

Алексей Недоря, ноябрь 2025

Профессия: Языки программирования и компиляторы

Краткая профессиональная биография:

- Первый компилятор: 1984
- Компиляторы для 9 языков программирования
- Участие в стандартизации языков программирования
  - Modula-2 ISO/IEC
  - Oberon-2 Oakwood Guidelines
- Системная архитектура
- Разработка 7 языков программирования

- t.me/vorchalki_o_prog
- http://digital-economy.ru/avtory/aleksei-nedoria-synergetic-lab-ru
- https://ontonet.org/gruppy/vorchalki-o-programmirovanii
- http://алексейнедоря.рф/

@VORCHALKI_O_PROG

Ликбез: Разработка языков программирования

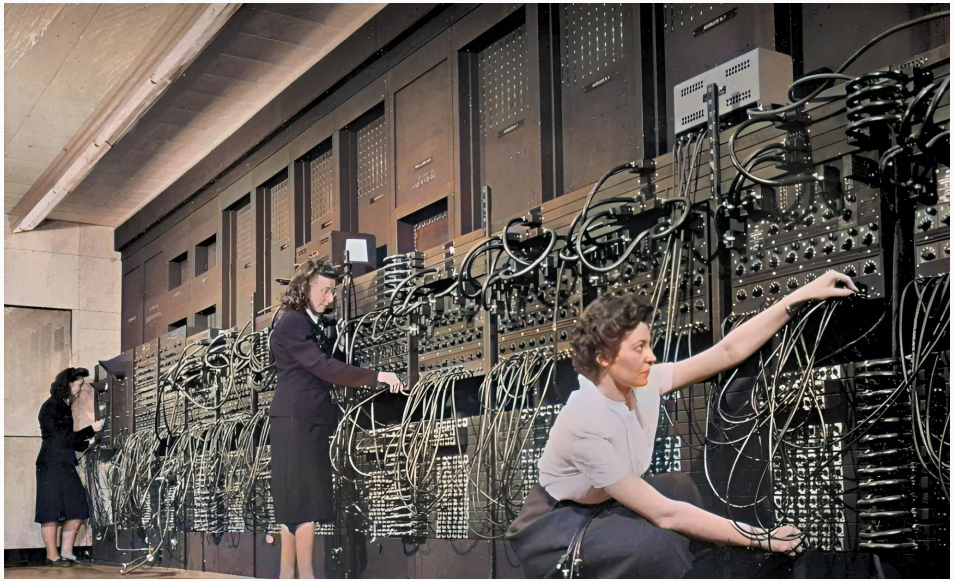
План лекций

1. История. Зачем и почему
2. Как. Проблемы и примеры
3. Несколько вариантов: зависит от интереса слушателей и вопросов
4. ?

Лекция 1

- Начало истории
- Языки XXI века
- Зачем делают языки
- Почему делают языки

Начало истории. ENIAC Short Code



1943–46	ENIAC coding system	John von Neumann, John Mauchly, J. Presper Eckert and Herman Goldstine after Alan Turing. The first programmers of ENIAC were Kay McNulty, Betty Jennings, Betty Snyder, Marlyn Meltzer, Fran Bilas, and Ruth Lichterman.
1946	ENIAC Short Code	Richard Clippinger and John von Neumann after Alan Turing

ENIAC originally had programs set up into the machine by a **combination of plugboard wiring**. The task of mapping it [program] onto the machine [...] usually took weeks.

ENIAC was demonstrated as a stored-program computer in **April 1948**. This modification

- **reduced the reprogramming time to hours instead of days**, but also
- reduced the speed of ENIAC by a factor of 6
- eliminated the ability of parallel computation.

FORTRAN

70 лет назад был разработан FORTRAN, первый широко используемый язык высокого уровня.

1954–55	FORTRAN (concept)	Team led by John W. Backus at IBM

1956–58	LISP (concept)	John McCarthy
1957	COMTRAN	Bob Bemer
1957	GEORGE	Charles Leonard Hamblin
1957	FORTRAN I (implementation)	John W. Backus at IBM

1952	Operator programming	Alexey Andreevich Lyapunov with the participation Kateryna Yushchenko

$[1 \to i]\,\{0 \to c_{i-1}\}^{\downarrow}\ A_i F\,(i)\,p\,(i=6)^{\uparrow}$ останов

https://en.wikipedia.org/wiki/Timeline_of_programming_languages

Лауреаты премии Тьюринга (1966 - 2024) [13/58 = 0.22]

Year	Recipient(s)	Rationale	Language(s)
1971	John McCarthy	research on artificial intelligence	LISP
1972	Edsger W. Dijkstra	principal contributor to the development of the ALGOL	Algol
1977	John Backus	FORTRAN, formalization of programming language specifications	FORTRAN
1978	Robert W. Floyd	the theory of parsing, the semantics of programming languages	Algol 60, Algol 68
1979	Kenneth E. Iverson	pioneering effort in programming languages and mathematical notation	APL
1980	Tony Hoare	fundamental contributions to the definition and design of programming languages	Algol-W, PL/I, Ada
1983	Dennis Ritchie (Ken Thompson)	generic operating systems theory and implementation of the UNIX	C
1984	Niklaus Wirth	developing a sequence of innovative computer languages	Eiler, Algol-W, Modula, Pascal
1991	Robin Milner	the first language to include polymorphic type inference, semantics	ML
2001	Ole-Johan Dahl, Kristen Nygaard	ideas fundamental to the emergence of object-oriented programming	Simula I, Simula 67
2003	Alan Kay	pioneering many of the ideas at the root of contemporary OOP languages	Smalltalk
2005	Peter Naur	fundamental contributions to programming language design and the definition	Algol 60
2008	Barbara Liskov	foundations of programming language and system design, data abstraction	CLU

XXI Век

Language	Company	v1.0 Development	Rating 2024
Go	Google	2007-2012 (5 years)	8 / 8
Rust	Mozilla	2009-2015 (6 years)	11 / 14
Typescript	Microsoft	2010-2014 (4 years)	5 / 41
Swift	Apple	2010-2014 (4 years)	21 / 20
Kotlin	JetBrains	2010-2016 (6 years)	17 / 18
Project Verona	Microsoft	2019-?	
Тривиль	А. Недоря	2022-2023 (9 months)	
Carbon	Google	2022-?	
ArkTS 1.2	Huawei	2022-?	

#	IEEE Spectrum	TIOBE Index
1	Python	Python
2	Java	C++
3	Javascript	Java
4	C++	C
5	Typescript	C#
6	SQL	Javascript
7	C#	Visual Basic
8	Go	Go
9	C	SQL
10	HTML	FORTRAN

Зачем делают языки?

Язык	**Зачем?**
FORTRAN	
Pascal	
Smalltalk	
Java	
C#	
Javascript	
Rust	
Kotlin	
Swift	

Зачем делают языки?

Язык	**Для чего, зачем и почему?**
FORTRAN	Надо считать, а не на чем
Pascal	Обучение студентов
Smalltalk	Обучение детей
Java	Переносимость: Write once, run anywhere
C#	Проигрыш в суде, минимизация зависимостей
Javascript	Новая область - динамический HTML
Rust	Новый способ управление памятью
Kotlin	Замена устаревшего языка (better Java)
Swift	Замена устаревшего языка (Objective-C)

- Algol-60
- LISP
- C
- CLU
- Forth
- Ada
- Modula-2
- C++
- Scratch
- Oberon
- Haskell
- Pony

- Альфа-6
- Рефал
- ЯРМО
- Эль-76
- Рапира

- RuC
- Ficus
- Тривиль

Легко ли разработать язык программирования? Сколько стоит разработка?

Легко ли разработать язык программирования?

- Увы, нет. Это больше искусство, чем наука (след. лекция)

Сколько стоит разработка?

- Очень по разному
- Для индустриального языка: тысячи человеко-лет (след. лекция)

Почему Google, Apple, Microsoft, Huawei, … разрабатывают языки?

Почему большие компании разрабатывают языки программирования?

Очевидно, разработка языка приносит компаниям деньги.

За счет чего?

- Контроль экосистемы
- Технологическое преимущество
- Контроль над языком и средствами разработки

Контроль экосистемы

Экосистема языка программирования - это инструмент

- удержания разработчиков
- и получения прибыли

в течении долгого времени.

Технологическое преимущество. А как же open source?

Работа над языком **вынуждает** смотреть на разработку ПО с разных сторон

- со стороны бизнеса
- со стороны программиста
- со стороны “железа”
- со стороны информационной безопасности
- со стороны взаимодействия с другими языками и инструментами
- ….

Приводит к появлению

- специалистов широкого профиля
- центров компетенции

Контроль языка и инструментов разработки

Не свой язык:

- Юридические проблемы и санкций (Microsoft C#, Huawei ArkTS)
- Язык может быть изменен непредсказуемым образом

Не свой компилятор и инструменты:

- Компилятор и инструменты могут быть изменены, затрудняя использование существующего кода
- Компилятор, инструменты (toolchain), библиотеки, IDE, плагины могут содержать backdoors, строить неверный код и т.д.

Свой язык и свои инструменты разработки

- Информационная безопасность
- Планирование, устойчивость

Вопросы

EOF