Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Казанский национальный исследовательский технологический университет»

(ФГБОУ ВО «КНИТУ»)

Реферат

по дисциплине

«Визуальное программирование»

на тему: Поколения языков программирования

Работу выполнила:

Студентка группы 4391-22

Еремина О.В.

Проверил преподаватель:

Зеленко О.В.

Казань, 2021

Оглавление

[Введение 3](#_Toc86591393)

[Первое поколение 4](#_Toc86591394)

[Второе поколение 6](#_Toc86591395)

[Третье поколение 9](#_Toc86591396)

[Четвертое поколение 12](#_Toc86591397)

[Пятое поколение 13](#_Toc86591398)

[Вывод 17](#_Toc86591399)

[Источники 18](#_Toc86591400)

## Введение

**Язык программирования** - это искусственный язык. Он отличается от естественного ограниченным, достаточно малым количеством числом слов, значение которых понятно компьютеру (транслятору), и очень строгими правилами записи команд (операторов).

**Программа** – это логически упорядоченная последовательность команд необходимая для управления компьютером.

Используя данное определение можно с уверенностью сказать, что двоичные коды для процессора, обладают ограниченным количеством слов (команд), они записываются по строгим правилам, а также понятны устройству, которое их получает.

Языками первого поколения были двоичные коды, на которых создавались первые программы для ЭВМ.  К сожалению, программы, написанные в двоичном коде, было сложно тестировать, так как ошибки могли приводить к полному отказу оборудования. Такой подход создания программ был трудоемок и требовал от специалиста особых навыков и знания всей архитектуры ЭВМ для которой разрабатывалась программа.

Языки программирования разделены на несколько поколений языков программирования. Исторически эта классификация использовалась для обозначения возрастающей силы стилей программирования.

## Первое поколение

К первому поколению (англ. *first-generation programming language*, 1GL) относят машинные языки — языки программирования на уровне команд процессора конкретной машины. Для программирования не использовался транслятор, команды программы вводились непосредственно в машинном коде переключателями на передней панели машины. Такие языки были хороши для детального понимания функционирования конкретной машины, но сложны для изучения и решения прикладных задач.

Термины «первое поколение» и «второе поколение» были введены одновременно с термином «третье поколение». Фактически, эти термины в то время не использовались. С появлением языков высокого уровня, языки ассемблера стали относить к первому поколению языков.

***Особенности языка первого поколения:***

1) Одна строка кода соответствовала одной команде конкретному устройству.

2) Каждое устройство обладало своим уникальным набором команд, которые не повторялись во всей спроектированной ЭВМ.

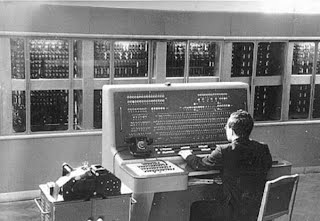
3) Каждая команда в ЭВМ была уникальна и не повторялась в рамках одной ЭВМ.

**Логика работы языка первого поколения.**

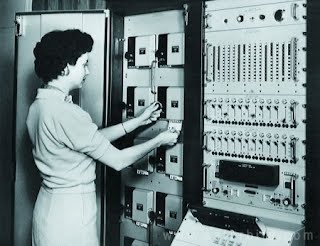
***1) этап.***

Оператор вводил в ЭВМ команду с помощью:

а) Панели ввода – Чаще всего это были либо тумблеры (кнопки переключения состояний).

[](https://sites.google.com/site/raznyeurokipoinformatiki/home/osnovy-programmirovania/pokolenia-azykov-programmirovania/66959bfd676d%5B1%5D.jpg?attredirects=0)

б) Устройств автоматического ввода – Это могла быть перфолента или перфокарты.

[](https://sites.google.com/site/raznyeurokipoinformatiki/home/osnovy-programmirovania/pokolenia-azykov-programmirovania/1365013563_perfocard2%5B1%5D.jpg?attredirects=0)

***2) Этап***

Команда распознавалась устройством для которого была предназначена и выполнялась им.

 Вывод результата мог происходить как  на перфоленту так и в виде световой индикации на пульте управления ЭВМ.

[](https://sites.google.com/site/raznyeurokipoinformatiki/home/osnovy-programmirovania/pokolenia-azykov-programmirovania/0_98e55_1b0f06ed_XL%5B1%5D.jpg?attredirects=0)

***3) Этап***

Оператор снимал результаты выполнения команд и расшифровывал их на нормальный язык.

## Второе поколение

Языки второго поколения (2GL) создавались для того, чтобы облегчить тяжёлую работу по программированию, перейдя в выражениях языка от низкоуровневых машинных понятий ближе к тому, как обычно мыслит программист. Эти языки появились в 1950-е годы, в частности, такие языки как Фортран и Алгол. Наиболее важной проблемой, с которыми столкнулись разработчики языков второго поколения, стала задача убедить клиентов в том, что созданный компилятором код выполняется достаточно хорошо, чтобы оправдать отказ от программирования на ассемблере. Скептицизм по поводу возможности создания эффективных программ с помощью автоматических компиляторов был довольно распространён, поэтому разработчикам таких систем должны были продемонстрировать, что они действительно могут генерировать почти такой же эффективный код, как и при ручном кодировании, причём практически для любой исходной задачи.

В 1950 году гениальный американский ученный венгерского происхождения Джон фон Нейман, он заложил основные принципы построения ЭВМ, которые используются и по сей день. По сути это был первый стандарт создания ЭВМ, который облегчал жизнь не только создателям, но и программистам.

Благодаря появлению контроллеров оборудования, программист мог больше не заботиться о знании всех команд этого устройства, а достаточно было знать набор команд отдельного контролера, а дальше управление устройством была забота платы, которая знала, что с ним делать.

Для упрощения создания программ в 1950 году было принято решение создать язык *assembly*это и послужило появлению языков второго поколения. Языки второго поколения позволили представить машинный код в более удобной для человека форме для обозначения команд и объектов, над которыми эти команды выполняются, вместо двоичных кодов использовались буквы или сокращенные слова, которые отражали суть команды.

Таким образом язык программирования второго поколения позволили создать логический слой между программой и оборудованием теперь можно было писать приложение, которое могло интерпретироваться (подстраиваться) для каждой модели ЭВМ в необходимый программный код.

**Появились такие понятия как:**

**Синтаксис языка** – совокупность требований для записи команд образуют,

**Семантика языка** –смысл каждой команды.

**Тестирование**– процесс поиска ошибок в программе

**Отладка**– процесс устранения ошибок

**Трансляция**– перевод программы с языка в машинный код

**Логика работы Языка Второго поколения**

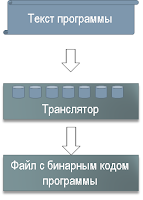
    Не важно был ли это *assembly*или другой язык, разрабатываемый в те годы, принцип работы у них был схож:

1) Программист подготавливал файл с инструкциями, описывающими нужные действия программы, основываясь на синтаксисе языка 1 поколения

2) Транслятор последовательно считывал команды, одну за другой и преобразовывал их на основе словаря в бинарные (двоичные) коды той ЭВМ, на которой нужно было выполнять команду.

***По сути словарь команд ЭВМ, выпускался разработчиками этой ЭВМ. Далее он подключался к транслятору, и с помощью него можно было преобразовывать команды в понятный устройству код.***

3) После преобразования команд получался бинарный файл, который можно было запускать, на тех ЭВМ для которых был транслирован.

[](https://sites.google.com/site/raznyeurokipoinformatiki/home/osnovy-programmirovania/pokolenia-azykov-programmirovania/3.png?attredirects=0)

    Принято считать двоичные коды команд языком первого поколения ПО, а языки использующие трансляторы для перевода команд в бинарный код языками второго поколения. Хотя есть множество ресурсов, которые опровергают данное высказывание, но это их дело, я же буду придерживаться такого мнения.

    В те годы был целый шквал языков программирования, каждый разработчик вкладывал в свой язык что-то свое, но до наших дней дошли не все. Самыми знаменитыми языками стали:

**Assembly (Ассемблер)** сегодня используется в основном для написания системного ПО.

**FORTRAN (Фортран)**  в основном для инженеров, математиков и ученых, которые имеют в основном дело с формулами и проблемами, ориентированными на вычисления.

**COBOL (Кобол)**был разработан в начале 60-х годов, для того чтобы было легче писать программы для бизнеса, которые могли использоваться в таких видах деятельности как обработка заказов, ведение бухгалтерии, планирование производства и т.д.

**Особенности языков второго поколения**

Языки второго поколения, заложили основные черты современных языков. И обладали следующими особенностями:

1) Представление инструкций, выполняемых программой описывались

2) Каждая инструкция в программе занимала одну строку.

3) Все представляемые данные (цифры) вводились в двоичной, восьмеричной или шестнадцатеричной системе счисления.

4) Написанные команды переводились из текста в машинный код и после этого процессор мог начать их выполнение.

5) От программиста требовалось знание особенностей работы тех ЭВМ, для которых он писал программу потому, а так же знание синтаксиса того языка на котором он писал код (текст) программы.

6) Для каждой модели ЭВМ (платформы) существовал свой словарь инструкций что позволяло переносить программу на все возможные платформы для которых существовал компилятор. До сих пор такую философию поддерживает ОС Linux в которой программы предоставляют в исходных кодах.

## Третье поколение

Под третьим поколением (3GL) первоначально понимались все языки более высокого уровня, чем ассемблер. Главной отличительной чертой языков третьего поколения стала независимость от аппаратного обеспечения. то есть выражение алгоритма в форме, не зависящей от конкретных характеристик машины, на которой он будет исполняться. Код, написанный на языке третьего поколения, перед исполнением транслируется либо непосредственно в машинные команды, либо в код на ассемблере и затем уже ассемблируется. При компиляции, в отличие от предыдущих поколений, уже нет соответствия один-к-одному между инструкциями программы и генерируемым кодом.

Стала широко использоваться интерпретация программ — при этом инструкции программы не преобразуются в машинный код, а исполняются непосредственно одна за другой. Независимость от «железа» достигается за счёт использования интерпретатора, скомпилированного под конкретную аппаратную платформу.

**Особенности языков третьего поколения**

    Языки 3 поколения унаследовали все достоинства языков 2 поколения и дополнили их своими достоинствами.

    1) Простота и понятность использования – привлекала пользователей к написанию небольших программ в основном это были экономически и научные системы облегчающие жизнь пользователей ЭВМ.

    2) Независимость от конкретного компьютера – это достигалось тем что теперь между пользователем и архитектурой ЭВМ была операционная система, которая позволила скрыть прямое управление устройствами и предоставить удобный механизмы взаимодействия с ними через собственные функции ОС.

    3) Возможность использования специальных синтаксических приемов – программы стали более сложными в них появились блоки команд, объединённые в процедуры и функции, что позволяло выполнять программу не по очереди с 1 строки и до последней, а переходить в нужное место по мере необходимости, или вызвать нужные блоки команд.

    5)Модульность программ – написание отдельных процедур и функций для решения отдельных маленьких задач позволило их повторно использовать в других проектах, что привело к созданию тысяч библиотек из готовых функций.

***Подавляющее большинство языков 3 поколения успешно применяется и сегодня. Для написания «быстрых программ» и операционных систем.***

    На данном этапе развития языков программирования были введены следующие понятия:

**Процедура** — это независимая именованная часть программы, которую после однократного описания можно многократно вызвать по имени из последующих частей программы для выполнения определенных действий. Процедуры не возвращают результата.

**Функция**— это подпрограмма специального вида, которая, кроме получения параметров, выполнения действий и передачи результатов работы через параметры имеет ещё одну возможность — она может возвращать результат.

    По сути развитие языков и становление третьего поколения идет до сих пор. На сегодня их сотни. Но среди них можно выделить самые популярные:

**BASIC (Beginner's All-purpose Symbolic Instruction Code)** - Язык, разработанный в 1963 году, который поначалу создавался для того, чтобы инженеры могли производить на компьютерах различные симуляции.

***PASCAL***- Язык программирования, созданный в 1970х годах и который создавался для обучения программированию.

**С** - создавался для написания операционных систем (многие UNIX совместимые операционные системы написаны на этом языке), уже долгое время является одним из самых популярных языков программирования.

**С++** следующий этап развития языка C с возможностью использовать не только процедуры и функции, но и объединять их в классы а классы в объекты

**JAVA**– платформа независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

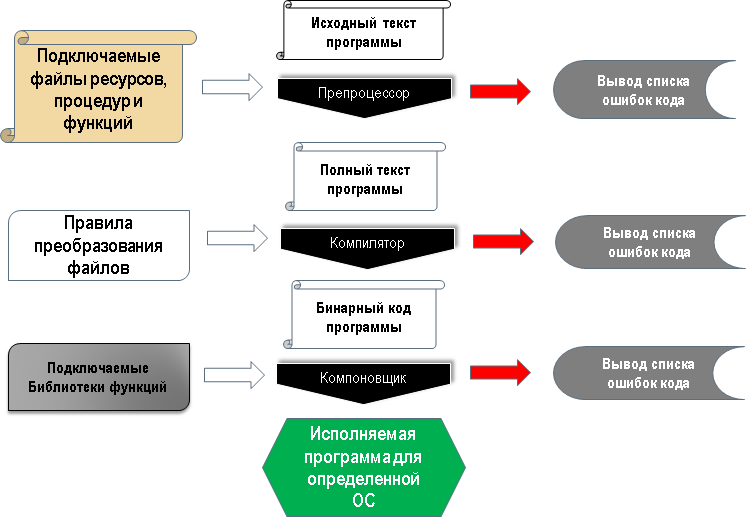
**Логика работы Языка третьего поколения**

1) Программист подготавливает файл инструкций на основе своих функций и внешних библиотек необходимых для решения задачи.

2) Во время сборки программы препроцессор дополняет текст программиста функциями, которые были вызваны из библиотек.

3) Полученный полный текст программы проверяется компилятором, который проверяет все ли на месте и правильность самого кода. Далее преобразует всю программу в бинарный файл.

4)  На основе выбранной ОС для которой нужно собрать программу, компоновщик связывает бинарный код программы с нужными библиотеками и создает код программы понятный конкретной ОС.

[](https://sites.google.com/site/raznyeurokipoinformatiki/home/osnovy-programmirovania/pokolenia-azykov-programmirovania/logik.png?attredirects=0)

## Четвертое поколение

Термин *языки программирования четвёртого поколения* (4GL) лучше представлять, как *среды разработки* четвёртого поколения. Они относятся к временному периоду с 1970-х по начало 1990-х.

Языки этого поколения предназначены для реализации крупных проектов, повышают их надежность и скорость создания, *ориентированы на специализированные области применения*, и используют не универсальные, а объектно-ориентированные языки, оперирующие конкретными понятиями узкой предметной области. В эти языки встраиваются мощные операторы, позволяющие одной строкой описать такую функциональность, для реализации которой на языках младших поколений потребовались бы тысячи строк исходного кода.

**К языкам четвертого поколения можно отнести следующие языки.**

**Язык 1С** - средой исполнения языка является программная платформа «1С:Предприятие». Визуальная среда разработки («Конфигуратор») является неотъемлемой частью пакета программ «1С:Предприятие»

**VBScript**– язык для написания скриптов (микропрограмм - макросов) как в MS Windows так и для приложений MS OFFICE так же распознается Internet Explorer позволяя генерировать WEB страницы.

**JavaScript**— прототипо-ориентированный сценарный язык программирования. Используется для создания WEB форм, и разработки кросс платформенных приложений так же можно применять в разработке скриптов для Windows и Unix платформ.

**ActionScript**— объектно-ориентированный язык программирования, который добавляет интерактивность, обработку данных и многое другое в содержимое Flash-приложений. ActionScript исполняется виртуальной машиной (ActionScript Virtual Machine), которая является составной частью Flash Player.

**SQL англ. Structured Query Language** — «Структурированный язык запросов») — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. Язык разного вида используется в базах данных таких как MS SQL, Postures SQL, MySQL и других базах.

## Пятое поколение

Рождение языков пятого поколения относится к середине 90-х годов. Довольно неожиданно вокруг самого названия 5GL разгорелись жаркие споры. Возникло несколько программистских "школ", представители каждой из которых имеют свое мнение о том, какие средства разработки считать языками пятого поколения, а какие  -  нет. Например, представители "английского" направления рассматривают средства разработки пятого поколения в более широком аспекте, чем это принято делать в отношении обычных языков программирования. Они считают, что к системам 5GL можно отнести не только новые мощные языки, но и системы создания программ, ориентированные на непрограммиста. Подобные системы отличаются стремлением предоставить конечному пользователю-неспециалисту богатые возможности создания прикладных программ с помощью визуальных средств разработки без знания программирования. Главная идея, которая закладывается в эти системы 5GL,  -  возможность компьютерного интерактивного или полностью автоматического преобразования инструкций, вводимых в компьютер, наиболее удобными человеку методами в максимально наглядном виде, в текст на универсальных языках программирования, описывающий готовую программу. Наличие подобного промежуточного этапа (получение не готового исполняемого модуля, а только исходных текстов, требующих дальнейшей обработки) объясняется низкой эффективностью автоматически генерируемого кода приложений, созданных с использованием подобных систем пятого поколения, что связано с внутренней сложностью последних и стремлением разрабатывать независимые от платформы продукты. Исходные тексты обычно генерируются на языках более низкого уровня, как правило, третьего поколения. Благодаря автоматическому процессу получения текстов программы результирующий код оказывается хоть и неэффективным, но высоконадежным и не содержащим ошибок. Правда, при этом возникает проблема совместимости с имеющимися на рынке компиляторами. После генерации кода созданного приложения необходимо перевести его в машинное представление. Для этого требуется тесная интеграция с имеющимися коммерческими компиляторами, легкая настройка, ориентированная на пользователя-непрограммиста и соответствие получаемого кода требованиям конкретных средств разработки. В большинстве случаев из-за острой конкурентной борьбы решить проблему совместимости в целом не удается, поэтому системы разработки 5GL ориентируются обычно на определенные версии компиляторов.

Создавая программы, решающие самые разные задачи, разработчики стараются использовать богатый многолетний опыт компьютерной индустрии. Системы пятого поколения имеют открытую архитектуру и поддерживают большое количество продуктов третьих фирм, предоставляя пользователю возможность интеграции с готовыми решениями для различных областей. Это могут быть всевозможные визуальные редакторы, генераторы отчетов, стандартные библиотеки, удобные Мастера (Wizards) быстрого создания типовых приложений, CASE-системы, средства интеграции с базами данных и т. п. Чем больше приложений удастся объединить в одном пакете, тем большими возможностями он будет обладать.

В большинстве подобных систем используются усовершенствованные технологии, воплощенные в средствах более низкого уровня. Например, возможность "мышиного" создания программ без ручного набора текстов, с использованием средства визуального проектирования ПО, взята из современных систем разработки 4GL. Другое перспективное направление, заимствованное из 4GL,  -  методы программной инженерии. Различные формальные нотации, поддерживаемые CASE-системами, позволяют с помощью мышки быстро создавать заготовки программ, SQL-"скрипты", описывающие структуру баз данных. Поскольку ряд современных языков 5GL создан на основе успешно реализованных продуктов 4GL, граница между этими поколениями сильно размыта. Системы разработки пятого поколения только появляются на свет, и нередко аббревиатура 5GL используется больше для рекламы.

Некоторые компьютерные эксперты считают продукты последнего поколения уже не языками, а средствами разработки, прикладными пакетами, не имеющими к процессу создания программ с помощью ЯП никакого отношения. Проектирование программы происходит в специализированном визуальном редакторе, и работа с исходными текстами отсутствует.

Однако значительно большая часть специалистов считает, что языки пятого поколения являются именно языками программирования, требующими от разработчика соответствующей квалификации и умения составлять программы вручную. Сторонники этого мнения под языками 5GL понимают специализированные языки, оперирующие не абстрактными переменными, а понятиями своей предметной области, например бухгалтерскими счетами или ферзями и пешками. Это, как правило, узкоспециализированные языки, предоставляющие программисту мощные высокоуровневые возможности обработки информации из конкретной области знаний. К языкам пятого поколения относят также интегрированные с базами знаний и экспертными системами программные комплексы с собственными языками программирования. Типичный пример  -  созданная в Австралии самообучающаяся нейронная сеть LISA со встроенным языком описания фактов, сущностей и взаимосвязей, на торговой марке которой красуется "лейбл" 5GL.

Несмотря на внешнюю противоположность языков 5GL (прикладные пакеты или языки программирования), их определяет то, что они предоставляют программисту средства разработки, использующие наиболее естественные для человеческого мышления понятия. При этом неважно, как подобные возможности будут реализованы  -  в виде усовершенствованных средств визуального проектирования или в виде новых мощных языков программирования, оперирующих привычными терминами. Наиболее актуальными для систем 5GL станут достижения в таких областях, как логическое программирование (Пролог-подобные языки и машины вывода), объектно-ориентированное программирование, исследовательское программирование (когда проект слишком сложен и неясен, а средства разработки позволяют быстро создать шаблон программы и включать в него работающие кусочки с постепенным приближением к конечному результату), использование для создания программ естественных языков, технологии управления базами знаний, методы обработки и анализа текстовой информации (энциклопедии, Web-страницы, документы) с возможностью смыслового поиска и т. д. Для создания достаточно сложных приложений планируется использовать различные экспертные системы и базы знаний со встроенными языками логического программирования, позволяющие автоматизировать многие рутинные процессы и помочь пользователю найти правильный путь решения тех или иных задач.

Пока сложно сказать, насколько успешной окажется тенденция к полной и недостижимой универсальности. На практике с помощью систем 5GL этого направления пока удавалось создать небольшие и логически простые приложения, которые при реализации на языках третьего поколения потребовали бы не более 10 000 исходных строк кода. При попытках разработки сложных программ возникает проблема, типичная для более старых языков,  -  необходимость отладки, что требует от пользователя высокой квалификации.

Языки 5GL, ориентированные на конкретные области применения, уже в ближайшее время могут завоевать самую широкую популярность. Это относится прежде всего к продуктам, позволяющим создавать приложения для работы с базами данных  -  области информатики, наиболее успешно поддающейся формализации. Наглядное подтверждение этому  -  тенденции развития практически всех известных СУБД корпоративного уровня. Вслед за встроенными языками СУБД появляются и другие проблемные ЯП. Кроме того, универсальные языки логического программирования пятого поколения наподобие Пролога, в основе которых лежат мощные математические аппараты, совсем не канули в Лету. Эти языки продолжают более чем успешно развиваться, другое дело, что для их грамотного применения требуется высокая культура программирования (и проектирования) в сравнении с Си++. И используются они для несколько других задач, чем создание коробочных бухгалтерий (хотя для этих целей они также подходят значительно лучше Си или Паскаля). Неудивительно, что в нашей стране мощными зарубежными средствами разработки, реализующими алгоритмы искусственного интеллекта или имеющими в своей основе уникальные математические теории, нередко наиболее активно интересуется ФАПСИ, что явствует хотя бы из открытых семинаров ведущих российских фирмдистрибьюторов соответствующих продуктов. Хочется надеяться, что рано или поздно и коммерческие фирмы поймут выгоду использования хоть и дорогих, но очень мощных по своим возможностям систем создания приложений сверхвысокого уровня.

## Вывод

\

Заключение

Выво

Изобретение языка программирования высшего уровня позволило нам общаться с машиной, понимать её (если конечно Вам знаком используемый язык), как понимает американец немного знакомый с русским языком древнюю азбуку Кириллицы. Проще говоря, мы в нашем развитии науки программирования пока что с ЭВМ на ВЫ. Только посмотрите, как развилась наука программирования с того времени, как появились языки программирования, а ведь язык программирования высшего уровня, судя по всему, ещё младенец. Но если мы обратим внимание на темпы роста и развития новейших технологий в области программирования, то можно предположить, что в ближайшем будущем, человеческие познания в этой сфере, помогут произвести на свет языки, умеющие принимать, обрабатывать и передавать информации в виде мысли, слова, звука или жеста. Так и хочется назвать это детище компьютеризированного будущего: «языки программирования "высочайшего" уровня». Возможно, концепция решения этого вопроса проста, а ближайшее будущее этого проекта уже не за горами, и в этот момент, где-нибудь в Запорожье, Амстердаме, Токио или Иерусалиме, перед стареньким 133MHz горбится молодой, никем не признанный специалист и разрабатывает новейшую систему искусственного интеллекта, которая наконец-то позволит человеку, с помощью своих машинных языков, вести диалог с машиной на ТЫ.

Размышляя над этим, хочется верить в прогресс науки и техники, в высоко компьютеризированное будущее человечества, как единственного существа на планете, пусть и не использующего один, определенный разговорный язык, но способного так быстро прогрессировать и развивать свой интеллект, что и перехода от многоязыковой системы к всеобщему пониманию долго ждать не придется.

## Источники

* <https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D0%BE%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F>
* <https://star-wiki.ru/wiki/Programming_language_generations>
* <https://sites.google.com/site/raznyeurokipoinformatiki/home/osnovy-programmirovania/pokolenia-azykov-programmirovania>
* <https://www.itweek.ru/themes/detail.php?ID=40584>