Колледж Автономной некоммерческой образовательной организации высшего образования «Научно-технологический университет „Сириус“

Реферат на тему

“История мёртвых языков программирования”

Выполнила: студентка К0709/24-2 Берг Жанна Олеговна

Проверила: Яковлева Софья Вячеславовна

Введение

История мёртвых языков программирования представляет собой увлекательную тему, которая проливает свет на эволюцию технологий, потребностей разработчиков и тенденций в области программирования. Языки программирования, которые в настоящее время считаются "мёртвыми" или устаревшими, имели огромное значение в своём время, но со временем были вытеснены более современными и эффективными альтернативами.

Программирование как дисциплина постоянно развивается, и в процессе изменения технологий некоторые языки уходят в историю. Причинами их исчезновения могут быть разные факторы: появление более удобных и мощных языков, неактуальность, неудобность в использовании, изменение парадигм программирования, требования рынка и другие.

Определение "мёртвого языка программирования"

Перед тем как перейти к конкретным языкам, важно понять, что подразумевается под понятием "мёртвого языка программирования". Обычно это языки, которые больше не имеют активного сообщества разработчиков, не получают обновлений и не используются в современных проектах. Тем не менее, такие языки могут сохранять историческую значимость и определённой аудитории, интересующейся их особенностями.

Примеры мёртвых языков программирования:

Cobol

Разработчик: Codasyl, 1960.

Cobol возник благодаря тому, что научные и бизнес-подразделения IBM использовали разные языки программирования. 70 лет назад высокоуровневые языки предназначались либо для инженерных вычислений, либо для управления данными. Если в научной среде стандартом был Fortran, среди бизнеса единства не было — компании использовали Comtran, Flow-Matic и другие.

В 1960 году Министерство обороны США организовало комитет по разработке единого универсального языка программирования для бизнес-задач — им стал Cobol.

Cobol был одним из четырех «материнских» языков, наряду с Algol, Fortran и LISP. Сегодня он почти забыт, но когда-то был самым популярным языком в мире и на нем по-прежнему работают многие устаревшие бизнес-системы.

Вклад: С точки зрения синтаксиса и семантики вклад Cobol в современные языки незначителен. Куда важнее его влияние на концепцию записи данных. В Fortran и Algol единственной структурой данных был статический массив. Cobol может читать структурированные файлы с иерархическими данными — он автоматически деструктурирует их в репрезентативные переменные. Это подход, который предшествовал современным способам записи данных.

Причина смерти: Здесь есть два фактора. Первый: Cobol не пересекается с другими PLT — и его синтаксис практически не пересекается с другими языками. Поэтому языки второго или третьего поколения, созданные на базе своих предков, почти не содержат в своем ДНК Cobol. Причина этого не столько во внутренних проблемах языка, сколько в пренебрежительном отношении к нему в академической среде. Codasyl создавала язык для решения конкретных бизнес-задач — поэтому научные круги почти не обращали на него внимание.

Второй фактор: Cobol был чрезвычайно сложным даже по меркам современных языков программирования. Это означает, что компиляторы Cobol требовали большей вычислительной мощности, чем могли предоставить ему микрокомпьютеры и миникомпьютеры.

Algol

Разработчик: Комитет Algol, 1960.

Примечание: Algol-58 выпущен двумя годами ранее, но от него быстро отказались. По этой причине я объединю два этих языка. Разработчики Algol хотели создать хороший язык для исследования алгоритмов. Поэтому язык, по сути, был формализованным «псевдокодом».

Из четырех материнских языков Algol — самый «мертвый». LISP и Cobol до сих пор на слуху, поскольку на них работает множество устаревших систем, а Fortran иногда используется в научных целях. Но я встречал множество программистов, которые ни разу не слышали об Algol — при этом по степени влияния на современные языки с ним может сравнится разве что LISP.

Вклад: Вот несколько примеров: лексическая область видимости, структурное программирование, вложенные функции, языковые спецификации, семантика вызова по имени, грамматики БНФ, блочные комментарии. Следы Algol видны в каждом современном языке программирования.

Причина смерти: Algol разработан для изучения алгоритмов, а не для коммерческого применения. В спецификации не было определено никаких операций ввода-вывода, что делало невозможным его использование для решения практических задач. Конечно, вы могли бы написать расширение компилятора, но тогда стоило бы добавить и другие вещи. Именно это сделал

группа из 70 разработчиков — в 1960 году они добавили в Algol возможности ввода-вывода и дополнительные структуры данных. Среди них, например, Jovial, Simula, CLU и CPL.

Именно эти расширения, а не оригинальный Algol, легли в основу более поздних языков программирования. Сейчас мы называем C «подобным Algol»-языком, но правильнее было бы говорить, что он похож на BCPL, который похож на CPL, а уже тот похож на Algol. Такими образом, язык похоронили собственные обновления.

Энтузиасты предпринимали попытки сделать Algol практичнее — в 1968 году группа разработчиков представила Algol-68, который радикально отличался от оригинала, но не имел того же влияния на IT. Стоит отметить, что принципы Algol получили продолжение в Pascal Никлауса Вирта.

APL

Разработчик: Кен Айверсон, 1962.

В оригинале APL — написанная от руки нотация для математических массивов, которую IBM взяла за основу для создания языка программирования. Язык использовался для обработки массивов — и позволял сравнительно короткими командами манипулировать большими блоками чисел.

Если вы раньше слышали об APL, то скорее всего знаете его как «этот странный язык символов». Один из самых известных фрагментов кода на нем — реализация игры «Жизнь»: *(рис. 1)*

APL использует собственные символы, поэтому для него нужна специальная клавиатура. Выглядела она вот так: *(рис. 2)*

Несмотря на это, язык стал популярным на мейнфреймах благодаря очень низким требованиям к памяти.

Вклад: Основная заслуга языка — инструменты для обработки массива. Раньше при добавлении двух списков разработчики использовали цикл или массив, APL позволил работать на весь массив сразу. Например, так: *(рис. 3)*

В научных кругах появление языка стало большим событием. В прикладной математике большинство задач сводится к крупномасштабным операциям с большими матрицами. Когда появился инструмент для их быстрой обработки, математики смогли работать эффективнее.

APL лег в основу R, Numpy, Pandas, Matlab и других языков и библиотек для программирования. У него есть и прямые потомки — J, Dyalog, K, Q, — которые оказались менее успешными, хотя до сих пор используются в финансовом секторе.

Причина смерти: Очевидная проблема – клавиатура с символами, которые не используются больше нигде, кроме APL. Кеннет Айверсон исправил этот недостаток с помощью J, который использует диграфы вместо APL-символов: вместо ~: в нем можно писать ≠. Сделано это было только в 1990 году — слишком поздно для популяризации радикально другого стиля программирования.

Существует и более тонкая проблема — APL и J умеют работать только с однородными данными. Языки не позволяют хранить строки и числа в одной и той же структуре данных.

Basic

Разработчик: Джон Кемени, 1964.

Basic — первый демократичный язык программирования. Он был создан как упрощенный аналог Fortran и предназначался для людей, которые не имели отношения к науке, но хотели научиться программировать.

Язык стал популярным в эпоху микрокомпьютеров — у первых устройств было слишком мало памяти для компиляции «настоящих» языков программирования. В то же время урезанному компилятору Basic требовалось всего 2 КБ. Basic стал лингва франка для начинающих программистов: если в 1970-х вы программировали дома, то, вероятно, писали именно на этом языке.

Вклад: Язык оказал сильное техническое влияние на интерпретацию вычислений — Basic был первым языком программирования с возможностью вычисления в реальном времени (Dartmouth Time Sharing System), опередив APL на год. Если APL был доступен только клиентам IBM, а Basic был доступен всем.

Кроме того, он имел большое социальное влияние: Basic сделал программирование доступным для неспециалистов — как для взрослых, так и для детей. Многие влиятельные программисты 80-х и 90-х годов учились

программировать на Basic. Многие корпоративные программы были написаны на Basic — вероятно, его популярность ускорила упадок Cobol.

Также на Basic написаны инструменты, которые входят в пакет Office. Со временем Microsoft превратила Basic в Visual Basic — на нем же написаны OpenOffice и LibreOffice. Недавно он уступил место JavaScript и теперь его используют для создания макросов.

Причина смерти: Большинство людей, которые учились писать код на BASIC, считали его «второстепенным» языком. Его можно использовать, если вы учитесь в школе или вам нужно написать простую программу для малого бизнеса — но настоящие программисты использовали «настоящие» языки. Как только компьютеры с оперативной памятью более 16 КБ стали доступны на массовом рынке, Basic начал терять популярность, а Pascal и C – приобретать.

Какое-то время Basic продолжал существовать как популярный язык программирования для детей и подростков, но похоже, умер и в этой нише.

PL/I

Разработчики: IBM, 1966.

IBM работала с двумя языками программирования: для научных исследований в компании использовался Fortran, а для бизнес-приложений — Comtran. В ответ на конкуренцию со стороны Cobol, IBM попыталась создать собственный унифицированный язык, которым могли пользоваться и научные-, и бизнес-подразделения. В результате получилась некая смесь двух языков с большим количеством дополнительных функций.

Вклад: Авторы Algol-68 насмешливо назвали PL/I устаревшим языком. Но все возможности Algol-68 появились в PL/I раньше и работали лучше. В то время

как Cobol первым научился читать структурированные данные, PL/I был первым языком, который реализовал их как тип.

В Cobol чтение имени пользователя даст вам две глобальные переменные — user и name. В PL/I вы получите одну переменную с полем user.name. PL/I был также первым высокоуровневым языком с указателями для прямого управления памятью, константами и перегрузкой функций.

Многие из идей, впервые использованные в этом языке, вошли в массовое программирование через C, который представлял собой смесь BCPL и PL/I. Например, C в точности копирует синтаксис комментариев PL/I.

Simula 67

Разработчики: Оле Даль и Кристен Найгаард, 1967.

Simula 67 — расширенная версия Algol для математического моделирования. Первая версия языка (Simula I) имела специальный синтаксис моделирования — разработчикам показалось, что он получился слишком специализированным, а в симуляциях содержалось слишком много дублирований кода. Даль и Найгаард хотели создать более универсальный язык, возможности которого выходили бы за пределы моделирования.

Их идея заключалась в том, чтобы дать пользователям возможность определять новые типы объектов — классы — с разрешением полиморфного определения функций. После этого пользователи могли создать функции моделирования как частные случаи объектно-ориентированной системы.

Вклад: Хотя Simula 67 не был первым объектно-ориентированным языком программирования, он впервые использовал правильные объекты и заложил фундамент для языков-последователей этой методологии. Речь идет о

разделении класса/объекта, создании подклассов, виртуальных методах и защищенных атрибутах.

Язык вдохновил подавляющее большинство академических исследований объектов в программировании, которые проводились после 1967 года. Создатели CLU и ML писали, что они вдохновлялись идеями Simula. Бьярн Страуструп защитил докторскую диссертацию по Simula и включил несколько идей из него в C ++.

Причина смерти: В своей докторской диссертации Страуструп утверждал, что Simula — слишком медленный язык для массового использования. «Желаю удачи в выполнении операции, если вы не на мейнфрейме», — комментировал он. Стоит отметить, что Smalltalk-80, который развивал те же идеи, и имел преимущество в 13 дополнительных лет закона Мура — и даже он считался слишком медленным. В Simula реализованы идеи, которые затем были интегрированы в быстрые и простые языки.

Причины исчезновения языков

Существуют несколько причин, почему языки программирования становятся "мёртвыми":

- *Появление новых технологий*

Одной из основных причин является появление более современных языков, которые предлагают улучшенные функции, простоту использования и большее сообщество. Например, языки как Python и JavaScript предлагают более высокую продуктивность и доступность в отличие от старых языков.

- *Изменение требований индустрии*

Суммарные требования к безопасности, производительности и функциональности программного обеспечения постоянно меняются. Языки, которые не могут соответствовать этим требованиям, незамедлительно теряют актуальность.

- *Сообщество и поддержка*

Отсутствие активного сообщества также влияет на жизнестоимость языка. Если разработчики не хотят использовать или поддерживать язык, он становится почти мёртвым.

Влияние мёртвых языков на современное программирование

Несмотря на то, что мёртвые языки программирования больше не используются, они оставили значительный след в развитии технологий. Многие концепции, введённые в этих языках, были заимствованы и адаптированы в современных языках. К примеру, многие структурные и объектно-ориентированные подходы из Pascal всё ещё активно используются в современных языках.

Заключение

История мёртвых языков программирования предлагает увлекательный взгляд на дух времени в сфере технологий. Она демонстрирует, как быстро меняется индустрия и как важны адаптация и развитие для успешного существования любого языка. Мёртвые языки, хотя и не используются, продолжают быть важной частью истории программирования и служат уроком для нынешних и будущих технологий.

Список литературы

1. Beck, K. (1999). Extreme Programming Explained: Embrace Change.

2. McIlroy, M. (1968). Mass Produced Software Components.

3. Wirth, N. (1976). Algorithms + Data Structures = Programs.

4.https://ru.hexlet.io/blog/posts/10-samyh-vliyatelnyh-mertvyh-yazykov-programmirovaniya#pl-i