Языки программирования — Lurkmore

Внимание!

Расположенная в этой статье информация принципиально никем не проверялась и, вероятнее всего, добавлена сюда исключительно для лулзов.

«Известны 10 преимуществ Паскаля перед Си:)

Я приведу только одно, но самое важное. На Си вы можете написать: for(;P("\n").R--;P("\ "))for(e=0x3DC;e--;P("_ "+(*u++/8)%2))P("| "+(*u/4)%2); На Паскале Вы НЕ МОЖЕТЕ такого написать.[1]

»

— Анонимус

Классификация языков программирования — быдлокодерский расовый холивар, поражающий одеревеневшую кору головы множества программистов. Не отрицая того, что концентрация быдлокодеров, работающих на отдельных языках, по объективным причинам выше, чем на других, анонимус спешит заметить, что участники настоящей специальной олимпиады вместо того, чтобы расширять свой кругозор и изучить что-то новенькое, тратят тысячи времени на попытки доказать, чей язык программирования лучше.

Быдлокодерские языки программирования

«Хороший язык программирования помогает программистам писать хорошие программы. Ни один из языков программирования не может запретить своим пользователям писать плохие программы »

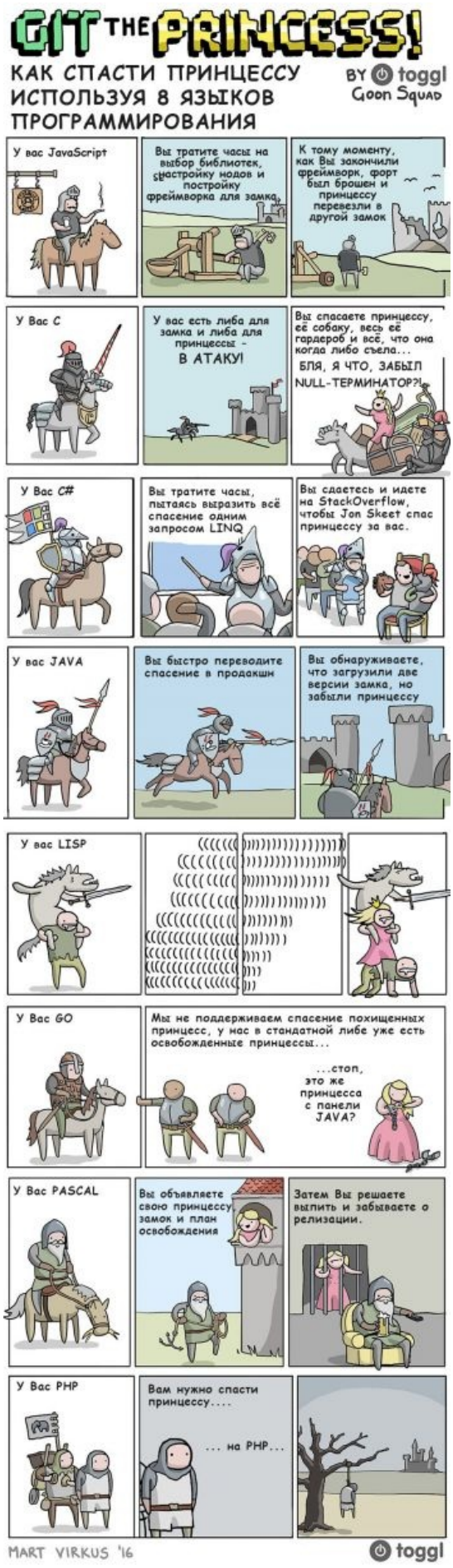
— Киз Костер

«Быдлокодерскими» злоехидные программисты называют любые языки программирования, на которых пишет чуть более, чем 3.5 анонимуса. Если, конечно, они не пишут на них сами.

- Visual Basic — зачастую — единственное что знают (и потому яростно любят) школьники. Удобен для тех, кто из всего «компьютерного» знает английский язык или где лежит словарь. Бывает «классический» и актуальный *VB.Net*. Овладение даже базовыми навыками программирования на *VBA* (*Visual Basic for Applications*, скриптовый язык для *MS Office*) даёт пользователю возможность выполнять в Ёксель-мокселе такие действия, которые раньше представлялись ему невозможными.
- PHP — почва для самореализации каждого начинающего веб-девелопера, благодаря чему этот язык здесь и находится. Обладает минимальным порогом входа (и выхода). На нем написана куча этих ваших CMS, включая популярные drupal, joomla, wordpress и нашу уютненькую mediawiki — которая, впрочем, будет скоро выпилена Новым Движком™ на расовом erlang. Язык и его интерпретатор разрабатываются группой веб-девелоперов в рамках проекта с открытым кодом.
- Python — идеальный язык для обучения программированию школоты, так как даже обезьяна осилит. Да что уж там обезьяна, даже 1С-ник или похапэшник не облажается. Тем не менее, на нём пишут и серьёзные вещи. Огромный плюс языка — синтаксис, всячески ограничивающий говнокодера в воплощении

своих влажных фантазий. В результате даже ОЧЕНЬ ПЛОХО написанный код таки поддаётся синтаксическому разбору среднестатистическим программистом. Ещё одно непревзойдённое достоинство языка и компиляторов для него — феерическая скорость работы программ.

- Ruby — тот же Python, только с принудительным ООП и парой-тройкой дополнительных свистоперделок. В своё время взлетел исключительно благодаря рельсам и, собственно, сейчас без них никому не упал. Породил целый пласт веб-разработчиков, мнящих себя (абсолютно безосновательно) илитой среди программистов. Имеет ужасную даже на фоне такого слоупока, как *Python*, производительность, в последнее время ситуация поменялась к лучшему, но на фоне ближайших конкурентов Ruby остается прожорливым тормозом.
- JavaScript — когда-то начинался как «язык сценариев HTML-страниц» (говоря проще, как хрень для сворачивания-разворачивания разделов при нажатии на кнопочки). С тех пор прошёл большой путь к славе, иногда недоброй. Незаслуженно считается простым, хотя *хорошо* писать с промисами и лямбдами умеет не так уж много людей. В 2021 году это язык, на котором пишется 9/10 ГУЯ (в 8 случаях из 10 GUI пишется для веба, ещё в одном случае — на основе *Chromium*, и только в последнем, десятом — что-нибудь на основе Qt/WPF/Delphi) и огромное количество серверных вещей — для примера гуглить *Node.JS* и *MongoDB*. Язык настолько популярен, что среди кодеров есть алкогольная игра: открой английский словарь, ткни пальцем в любое слово, прибавь к нему «.Js» и введи в поиск. Если найдётся такая библиотека — выпиваешь рюмку. Наклюкаться в хлам можно не быстро, а очень быстро.
- Pascal (и Delphi — IDE и диалект) — используется для обучения студентоты (учить Бейсику стыдно, чай не школьники, «будущие инженеры», а Си преподаватель зачастую сам не знает). Новых проектов на нём уже давно никто не начинает, но в прошлом на нём писали «ин ~~да~~ хаус» (то есть, свои внутренние продукты) многие большие конторы: и Apple, и Microsoft. В этой стране был популярен настолько, что программирующие иностранцы называли Россию 90-х «Pascal-land» и «Delphi-land».
- COBOL — мы говорим «Кобол», подразумеваем «w:мейнфреймы». Кощей Бессмертный в мире языков: помирает-помирает, да всё никак не помрёт (совершенно внезапно стал снова актуален в 2020 году в США, ибо оказалось, что написанный на нём софт пиндосской системы занятости не справляется с нагрузкой, выросшей из-за COVID-19). На Коболе написано огромное количество софта для всяких финансовых учреждений, возраст иных поделок приближается к 60 годам. До недавнего времени считался языком программирования, на котором было написано больше всего строк кода (не в последнюю очередь благодаря невероятно громоздкому синтаксису).
- 1С — Кобол, переведенный ПРОМТом.
- ABAP4 — внутренний язык SAPа. Как и 1С, является диалектом Cobol'а, причем местами даже более ущербным. При этом является самым высокооплачиваемым языком программирования. (Согласно городским легендам типичный САПёр представляет собой мужика в костюме, галстуке и недельной щетине, который спит с открытыми глазами с мешками под глазами на мешках с деньгами).
- Java — когда-то давно компания Sun так возненавидела Microsoft, что вместо зарабатывания денег (продажей серверов) решила последнюю изничтожить. Вспомнив, какую роль Basic сыграл в успехе мелкомягких (см. ниже), светло-фиолетовые решили осчастливить человечество новым языком, в надежде, что он вытеснит Васика и Микрософту поплохеет. Поплохело, однако, самим сан-техникам, в результате чего их купил Oracle, которому Жаба была нужна как собаке пятое колесо. Но не убивать же язык, который стал самым популярным в мире! (На пару с Си они делили первых два места во всех известных рейтингах).



Как спасти принцессу

В тот период, когда Жабу ещё не начала ебать Оракуловская гадюка, и язык кагбе считался дающимся всем и даром, и пусть никто не уйдёт обиженным, его попыталась прихватизировать корпорация добра, сделав основным языком разработки под Андроид. После оборакуливания она об этом очень пожалела, получив повестку в суд. Шаблон у энтерпрайзников, за долгие годы привыкших к халяве, затрещал, но выдержал.

- C# — ответочка Microsoft не заставила себя долго ждать. Так как Sun явно дала понять, что Embrace, extend and extinguish не пройдёт, пришлось Microsoft делать свою *Java*. С дотнетом и студией. Языком в MS занимался тот же чел, что в своё время привёл Delphi к успеху. Как и Жаба, является стандартом де-факто энтерпрайза. Язык (как и стандартная библиотека, да и платформа .Net в целом) всем хорош, кроме одного: анальной зависимости от Microsoft, которая проиграла битву за мобильную ОСь и тысячи других битв и сама не знает, чем будет заниматься послезавтра.
- Kotlin — посмотрев, как сильно популярности дотнета помогло отделение виртуальной машины от собственно языка программирования (писать под дотнет можно на любом языке, был бы компилятор — а они есть), чешско-русская контора JetBrains запилила другой язык (не Java!) для *Java Virtual Machine*. Поскольку Жаба, как язык, не настолько хороша, как о ней говорят (Oracle нехотя развивает её, но очень медленно), а энтерпрайзникам совместимости с JVM достаточно, чтобы не беспокоиться, язык откусил у Жабы приличную долю рынка.
- SQL — язык табличных СУБД (а это чуть менее, чем все СУБД вообще — остальные так и называются: «no-SQL» и пока остаются модной хипстотой). В своё время считалось, что этот язык обязан знать каждый компьютерно-грамотный бухгалтер. (Серьёзно, во времена терминалов полагали, что бухгалтер для составления своих отчётов должен писать SELECT'ы). Язык так и остался уделом программистов. Поскольку именно запросами на SQL управляются 95% баз данных, все программы, которые хранят данные в базах (в первую очередь — веб-сайты) так или иначе их содержат. В худшем случае запросы на SQL в виде текста вложены в говнокод на PHP, в чуть более хорошем случае — они генерируются при помощи колдунства под названием ORM: программист создаёт, меняет и уничтожает объекты, а под капотом библиотека шлёт SQL в СУБД. А ещё, поскольку считается неправильным тянуть по сети все данные на клиента, массово их менять, и заливать потом обратно, на этом языке пишутся w:хранимые процедуры (и их разновидность — «триггеры»), которые выполняются самой СУБД на сервере по запросу от клиента.
- ActionScript — язык Flash. На нем уже не пишутся, в частности, игры и приложения Вcuntакте, а также видеоплееры на YouTube и других видеохостингах.
- C++ — сюрпри-и-из! На нём пишет всякое быдло, Линус Торвальдс гарантирует это! Если серьёзно, очень популярен, многим известен, везде поддерживается.
- Go(вно)™ — ЯП от гугла для полных джунов в гугле. Имеет неплохую стандартную библиотеку. Применяется в сетевых сервисах и криптовалютах. Примечательно, что сделан язык Гуглом, но никто в Гугле не смог придумать уникального имени, которого легко было бы гуглить. Пришлось запиливать «golang» в качестве ключевого слова для поиска материалов по языку.
- Lua — скриптовый язык, напоминающий C. Используется во многих играх (WOW, например) и чуть реже, чем всегда, когда кому-то понадобился встроенный язык. Грядет на замену убогим шаблонам MediaWiki, в Википедиях уже доступен.
- Rust — в компании *Microsoft* однажды заметили, что 70% случаев surprise-sex'a происходит по вине программиста, что-то неправильно делающего с памятью. А в компании *Mozilla* написали язык, который, с одной стороны, обходится без автоматического управления памятью (что позволяет писать менее ресурсоёмкие программы), а с другой — вводит всякие правила (типа «один объект — один владелец»), которые сильно мешают потенциальным хакерам. Первоначально язык был личным проектом одного из сотрудников Мазилы, потом стал использоваться другими сотрудниками Мазилы, а в феврале 2021-ого (в связи с тем, что «денег нет, но вы держитесь») передан независимой организации. В которую входят: *AWS*, *Huawei*, *Google*, *Microsoft* и сама *Mozilla*. Что, как полагают, отныне обеспечит языку самое быдляцкое будущее.

Олдфаги не помнят, ньюфаги не знают...

Тот момент, когда ты уместил интерпретатор Бейсика в 4K и понял: сонибляди SOSNOOLEY!

...что и Бейсик, и Паскаль проектировались для написания исключительно учебных программ. Причём, Васик был создан как недо-Фортран, а Посраль — как недо-Алгол. (Предполагалось, что изучивший Васик на достигнутом бы не остановился, а освоившись с программированием перешёл бы с учебного языка на серьёзный Фортран. Ну, а изучивший Посраль, соответственно — мог бы перейти на Алгол).

Ирония в том, что на практике «учебные» языки отхватили (на довольно продолжительное время) изрядный кусок промышленного пирога.

Васик — язык, в 1964 году разработанный профессорами ~~Иннсмутского~~ Дартмутского колледжа Томасом Курцем и Джоном Кемени — дал путёвку в жизнь одной маленькой конторе (2 погроммиста-основателя + 1 подряженый временный кодер), делавшей такой крохотный софт, что его даже называли микро-софтом (sic! именно так, через дефис!). И софт этот риальне был настолько микро-, что даже эпик-! Первый коммерческий продукт Microsoft, Altair BASIC состоял из 2041 байта интерпретатора и 1307 байтов математической обвязки. Со всем говном («текстовыми ресурсами», если

так можно назвать ключевые слова) — **3348 байтов**! Щас так уже не пишут. Примечательно, что интерпретатор писал лично Гейтс (на пару с Полом Алленом, а некто Монти Давыдов написал математику), которого по незнанию школота считает не более, чем братцем-акулой капитализма. (Хотя он ещё и хакер от бога — кто сомневается, может сначала упихать свой Бейсик с командами и интерпретатором в 4 килобайта памяти, да ещё чтоб место для юзерских программ осталось).

~~Отдалённый потомок швейной машинки Зингера~~ Altair 8800. Первый массовый компьютер с отупляющим действием Бейсика.

Микрохвостовский Бейсик выстрелил с такой силой, что им немедленно начали комплектовать чуть менее, чем все бытовые компьютеры той эпохи (но не этот). Некто iBoз в своё время собирался запилить свой такой же, но так и не осилил плавающую точку (по поводу которой он кормил завтраками Джобса, а в итоге пришлось-таки лицензировать BASIC у MS). Памятуя о том, как простой народ любит эту поебень, Гейтс позже провернул трюк ещё пару раз, сначала запилив *GW-BASIC* под DOS (GW — инициалы какого-то эффективного менеджера, обосранного острым на язык Джоэлем), а потом и *Visual Basic*. Последний стал затычкой в каждой бочке: от платформы для создания программ под винды с нуля до скриптового языка для MS Office, и даже ВНЕЗАПНО несостоявшимся убийцей жабоскрипта в ишаке. Известная писькомерная даёт ему на весну 2021-ого аж 6%[2] неизвестно чего и целое 6-ое место.

В свою очередь, Pascal, «изобретённый» профессором Никлаусом Виртом путём усечения всего слишком сложного Алголу — не только ебал мозги студентам кибернетических специальностей, но и был, вы не поверите, официальной религией Apple. (Первые яблочные софты писали на жуткой смеси Поскакаля и ассемблера[нужен пруфлинк на репозитарий]). В 1983 году компилятор с приставкой *Turbo* написала одна конторка, подарив языку поистине ~~анти-~~народную славу. В этой стране не одно ванилько лило над ним слёзы в суровые 90-е и даже нулевые (в 80-х вода в бассейн была ещё слишком дорогая, плавать учили теоретически). В 1995-м та же контора разродилась *Delphi* — средой программирования для (тогда ещё) 16-битных виндов. Позже в «Delphi» переименовали и диалект Паскаля, который в этой среде использовался. По причинам неизвестного характера в постсовке Delphi нежно любили, можно сказать, считали винрарным[3]. Да и в других странах не отставали, программируя на нём много всякого ~~говна~~ софта, от *Nero Burning ROM* до *Skype*. В наши дни какое-то тёмное некромантское искусство вернуло его на 14-е место всё в том же индексе TIOBE.

Будьте осторожны, создавая что-то «учебное» для быдла: быдло не любит переучиваться, и ваш «мастер-класс для быдл» (*перумовск.*) может принять такой размах, что вам и не снилось.

Небыдлокодерские языки программирования

Небыдлокодерскими по определению являются ЯП с высоким порогом вхождения, а также полные комбинаторики, лямбда-исчисления и прочего матана. То есть, с одной стороны, это всякая функциональщина и эзотерика, с другой — приближенные к железу и системе вещи. По определению малопопулярны, а потому изучаются лишь задротами или большими энтузиастами, либо профессионалами для решения каких-то очень узкоспециальных (другими словами, никому не интересных) задач.

Академические языки

На них никто не программирует, но очень модно хвастать их знанием.

- M4 — специально спроектированный быдлокодероустойчивый язык, созданный с целью защиты от быдлокодеров в 1977 году. Долгое время препятствовал простому вхождению в разработку ПО, пока не появилась Delphi и другие революционные языки разработки в один (иногда бездумный) клик, на которых можно и толково зарабатывать деньги, а можно и быдлокодить.
- Objective-C — необыкновенно высокий порог вхождения которого обусловлен необходимостью быть геем для написания более-менее полезных программ. Всякие хелло-ворлды, на нём написанные, можно компилировать и под другие платформы, но большинству быдлокодеров это не нужно.
- Smalltalk — «мама» вышеуказанной НЁХ («папой» является ANSI C), разработанный яйцеголовыми бородачами из Xerox PARC. Первый (да в общем-то и единственный) Ъ-объектный язык в мире, отличается челябинской суровостью подхода (в нём ВСЁ объекты, даже цифра «3», которой можно передавать сообщения прямо в коде) и эпической слоупочностью. Вызывает у быдлокодеров короткое замыкание мозга, а у их компьютеров — работу кипятильником, особой популярности не снискал. Впрочем, есть мнение, что если бы не хреновая на момент появления производительность, вполне мог бы занять место жабы в предыдущем списке.
- Ада — боевой язык, разработанный американской военщиной в Пентагоне с целью установления военной гегемонии Соедёнённых Штатов в мире. Применялся и применяется в основном в ПО управления самолётами, подводными лодками, космическими ракетами и огромными боевыми человекоподобными роботами. Технически является жёстко расширенным и невъебически

усложнённым диалектом Паскаля. Вроде бы как должен был быть запредельно безопасным и удобным в разработке сложных систем, но сложилось как-то не очень. Название языка — традиционный источник шуток на тему сотонизма, Ада и Израиля. Например, известны случаи, когда старушки, увидев книгу с надписью «Язык Ада», либо орали на весь город, либо падали в обморок. А сам язык назван в честь самого первого в мире программера, который, кстати, был женщиной [1].
- Perl — старейший из ныне используемых скриптовых ЯП. 30 лет выносит быдлокодерам мозг своим синтаксисом, 10 лет из которых оналитики его закапывают. Объект лютого баттхёрта со стороны малолетних питоноводов, впадающих в когнитивный диссонанс при мысли о том, что не распиаренный гуглом язык может обладать такими же возможностями, как и их прелесть. Взрослые же питонщики существование перла игнорируют.
- Tcl — всем быдлокодерам вход строго воспрещен. Язык не отвечает достаточно большому количеству понятных для быдла стандартов. **Во-первых**, синтаксис — это ёбаный кошмар для всех поклонников мейнстримовых языков, такие конструкции, как *set set set* просто взрывают мозг неподготовленного хомячка. Далее — не нужно состоять ни в одной из популярных сект, в том числе не нужно быть борщехлебом и евангелистом GNU, язык не требует быть геем, не требует поклонение Дяди Билли как пророку мелкомягкой компании. **Во-вторых**, там метапрограммирование и требование к пониманию св. Матана. Да, анон, это один из немногих языков, в котором ты сможешь описать звук — стулом, а цвет — запахом. **В-третьих**, жёсткие требования к построению архитектуры приложений. Малоопытный программист превратит код на tcl в АдЪ и Израиль уже на уровне написания учебного калькулятора. **В-четвёртых**, очень большое количество синтаксического сахара, регулярок и списков, где любой Я-гей теряется, а сишарпист стреляется нахрен. Есть бесконечное число способов достичь результата и снести выстрелом в ногу нахуй свою голову и всех окружающим в радиусе пары сотен километров. **В-пятых**, язык придуман в том же ВУЗе, что и ЛСД и BSD.
- Make — язык написания программ для автоматизации процедур сборки других программ или иных хитрых целей, в сочетании с Autotools ~~выносит мозг~~ позволяет творить чудеса.
- APL — прославился тем, что требует специальной клавиатуры с математическими символами, а код выглядит как математическое выражение. Алсо, способен легко оперировать матрицами на уровне простейших операторов, не требуя каких-либо библиотек для этого.

Языки для функционального программирования

В той или иной степени зайчатки ФП поддерживаются во всех современных языках. Уж очень удобно писать что-то наподобие[4]:

```
var admins = users.where(x -> x.isAdmin);
```

Однако, есть языки, где функциональная парадигма возведена в культ. В теории считается, что правильно написанная программа с чистыми функциями и монадами не может содержать багов (в случае покрытия каждой функции тестами, естественно), поскольку в ней отсутствуют побочные эффекты. На практике же передозировка ФП приводит к написанию совершенно инопланетной логики, с которой никаких багов не надо. Например, если ты пишешь шутер и хочешь отнять здоровья у врага в результате попадания, надо вместо уменьшения значения переменной, хранящей его health points, сгенерировать целого нового врага, такого же, как старый, но с уменьшенным количеством HP, а старого развоплотить к хуям.

- ML — несколько функциональных языков разной степени кошерности. Характерны развитой системой вывода типов, благодаря чему языки статически и неявно типизированы одновременно (ИЧСХ, довести до ума эту систему не могут уже с OCamla, хотя альтернатив всё равно нет). Самый современный из адской семейки — F#, OCaml, скрещенный с типовой системой .net и зависимыми типами, тихий ужас.
- Lisp — позволяет достраивать синтаксис с помощью макросов и получать новые языки. Считается, что для этого достаточно трёх символов: двух скобок и пробела. На данный момент имеет как коммерческие реализации, так и бесплатные. Наиболее живым из последних можно назвать Racket и Clojure. Оба нелюбимы фонатами CommonLisp.
- Haskell — тот же матан, только шрифты заменили. И припаяли костыль в виде монад. По сути — лагерь штангистов разделился на две составляющие: 1. смотрят на пользователей монад как на гавно и продолжают дрочить матан и жать штангу, сохраняя волосы чистыми и шелковистыми и создавая чёрную дыру на мамкином диване при поглощении в космических масштабах такие вещи как мамкин борщ. 2. не знают мотана и удливляются, что их хацкель — функциональный язык. Всё решают через монады, не понимают ленивости и лямбд, скорее всего берут хацкель, чтобы рассказывать быдлокодирам из мира PHP, что научились забивать гвозди микроскопом (ну или чтобы в среде яблочников считали, что не гей).
- Erlang — единственный реально необходимый на продакшене функциональный язык, знающие который игнорируют существование других функциональных языков и на любой закидон касательно многочисленных костылей и недочетов в их любимце отвечают "Зато мне деньги платят". Алсо, на нем написано много вполне годных ынтэрпрайзных продуктов. Ситуация несколько изменилась после выхода Эликсира, коий суть есть Erlang с человеческим синтаксисом, созданный ради того, чтобы говнокодеры также сумели оценить скорость местной виртуальной машины.
- Prolog — некогда позиционировался, как серебряная пуля, которая убьёт все остальные языки, но выяснилось, что пуля на самом деле из говна, а почему блестит — неизвестно. Крайне упорот в своей сути, причем упоротость сия никакими практическими преимуществами не обосновывается. Взлетел,

но упал, и притом достаточно больно. Ныне почил в бездне отчаяния и безысходности. Goodnight, sweet prince.

- Scala — на сегодняшний день, единственный вменяемый функциональный язык на JVM, да и в мейнстриме вообще, имеющий внятные перспективы к дальнейшему развитию и коммерчески востребованный. Представляет собой адскую смесь из функциональщины а-ля OCaml и объектов Java-style с новомодными фишками вроде trait'ов. Несмотря на функциональную природу, довольно прост для изучения и довольно широко используется, имеется много библиотек на все случаи жизни, а кому не хватает - можно юзать джавовские без бубна. Стал популярен в этих ваших Европах на волне функционального хайпа и анти-Microsoft истерии в пику C#. Собственно, есть мнение, что Scala является этаким «ответом Чемберлену», поскольку C#-разработчики очень долго гнобили Java за отсутствие лямбд, extension-методов и прочих плюшек, из которых Scala состоит чуть более, чем наполовину.
- Wolfram - язык Стивена Вольфрама для символьных вычислений. На нём написана Wolfram Alpha. Имеет офигенную стандартную библиотеку и сайт с модулями, но лютая проприетарщина. Язык меметичен тем, что сам Вольфрам допрограммировался на нём до того, что узрел нашу Вселенную ~~пахнущей нефтью~~ в виде исполняемого кода, порождающего эти ваши энергии-гравитации-частицы. Кроме шуток, он уже заявил, что получил ОТО без космологического члена и квантовую механику. Нобелевский комитет судорожно подсчитывает, хватит ли у него медалей за все эти открытия.

Мёртворожденные языки

Как со всей очевидностью следует из названия, это языки, которые их хейтеры считают напрасно родившимися, напрасно коптящими небо и напрасно до сих пор не умершими. Разумеется, у их фанатов на этот счёт особое мнение, так что большая просьба не принимать этот раздел близко к серду.

Ты же не думал, что статья на Лурке обойдётся без расистских шуток?

- PL/1 — легендарный хтонический пиздец, которым пугали людей ещё во времена войн между Фортраном и Алголом. Мало кто осиливал язык в полном объёме, что ещё прокатывало когда писали код с нуля, но превращалось в полнейший пиздец, когда нужно было разбираться в чужом коде, написанном на незнакомой части языка. На масштабы пиздеца намекает тот факт, что с момента выхода первого стандарта не было реализовано ни одного компилятора, целиком его поддерживающего. Как ни странно, до сих пор иногда используется при программировании под мейнфреймы, ибо строгая типизация позволяет избавиться от многих проблем, характерных для программ на C.
- D — название намекает, что язык должен был вытеснить C и C++. К созданию приложил руку Андрей Александреску — весьма учОный кот по части плюсов, крестный папа библиотеки *boost*. Несмотря на название и на Александреску, так и не вытеснил. Переодически его выкапывают, насилуют в целях создания очередного «крутого продукта», задрачивают вусмерть и снова закапывают.
- Groovy — уродец на JVM, нечто среднее между «ужом» и «каменюками»[*ЩИТО?*]. Отсутствие вменяемых инструментов стало последнем гвоздём и язык передан в *Apache*, как и полагается: гомункулов — в кунсткамеру. На данный момент почти отовсюду вытеснен Скалой.
- Nemerle — попытка польской студентоты взять этот ёбаный C# и довести его до ума. Имеет нормальную функциональщину, метапрограммирование и макросы, при этом не скатился в ML с монадами, функторами и прочей хуитой. То есть удобен, мощен и относительно прост одновременно. Эталон промышленного языка. При этом оказался нахуй никому не нужен, sad but true.
- Dylan — попытка Корпорации Содомитов изобрести лишпек с синтаксисом без скобок. Провалилось, ибо геи не поняли, зачем нужен инструмент, где (в том числе) есть и вагина[*ЩИТО?*].

Наркоманские языки

Сознательно спроектированные с целью извлечения лулзов. Хотя, если задуматься…

- BrainFuck — ёбаная мозгодробительная херня, которая применяется биологами для моделирования ДНК и прочей псевдо-научной поебени. Больше непригоден ни для чего. Ну и для обучения курсу по написанию интерпретаторов. Создавался как самый минималистичный Тьюринг-полный язык программирования: имеет аж 9 операторов.
- Befunge — любопытный двумерный язык, в котором программа представляет поле, наполненное командами.

Для особых извращенцев есть 3-х мерные, 4-х мерные и прочие n-мерные версии.

- Whitespace — язык, состоящий из пробелов и табов целиком и полностью. Программа выглядит… Интересно.
- Malbolge — стоит особняком даже на фоне всей остальной наркомании. В языке 8 достаточно простых, но крайне упячных команд (например, «Сдвинуть регистр *d* на одну троичную цифру вправо и сохранить в регистры *d* и *a*»). Последовательность выполнения команд определяется остатком от деления на 96 суммы адреса текущей команды и её троичного значения. Вдобавок, после выполнения каждой команды она шифруется. Экспоненциальная мощь рекурсии в данном случае такова, что проще (и гораздо полезнее!) сгенерировать себе биткойн, чем листинг, который выводит самый банальный хелло-ворлд (кстати, вот он: (= *<`:9876Z4321UT.-Q+*)M'&%$H"!~}|Bzy?=| {z]KwZY44Eq0/{mlk**hKs_dG5[m_BA{?-Y;;Vb'rR5431M}/.zHGwEDCBA@98\6543W10/.R,+O<*). Поскольку гомо сапиенс на таком языке писать не может, генерировать примеры на Malbolge заставляют бессловесные компьютеры под управлением Lisp. За 14 лет был сгенерирован куайн. Страшно представить, что будет сгенерировано ещё через 14 лет…

Православный язык программирования. Мицгол одобряет.

- ДССП - советский тринарный язык, косплеивший Forth, для ЭВМ Сетунь работавшей на тринарной логике.
- Unlambda — минималистичный функциональный язык программирования. Название дословно означает «не-Лямбда» — иными словами, попытка сваять функциональный язык программирования без лямбда-оператора.
- Shakespeare — Ассемблер с кучей лишних слов, которые маскируют программу под пьесу Шекспира. Нет, правда.
- Iota Эталон минимализма. Тьюринг-полный функциональный язык, синтаксис которого состоит из одного единственного комбинатора.
- YOBAL — язык имиджборд. «ALLOW YOBA ETO TI? PSHH..PSHH… NE SLYSHNOW NE SLYSHNOW! YOBA YOBA VIZOVI STIRALLNOUY MASHINOW!! (VSE SOSNOOLEY)».
- YoptaScript — Русский ЯП с Русским синтаксисом похожий на JavaScript но сделаный для гопников и реальных пацанов!

```
гыы gop внатуре пиздишь, lt нах

куку йопта law()  жы
    вилкойвглаз(gop типа нечотко)  жы
        ксива.малява("Я и правда язык") нах
        gop сука чотко нах
    есть  иливжопураз  жы
        gop сука чотко нах
        потрещим(semki чоблясука трулио)  жы
            lt сука ксива.вычислитьЛоханоНомеру("list") нах
            ебало.шухер("Привет, йопта") нах
         есть
    есть
 есть
```

- Vala - для любителей GTK.

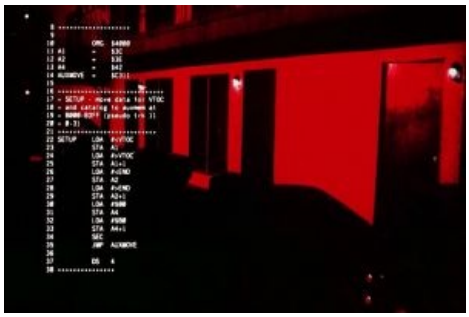
Какие вообще бывают?

«Все животные делятся на: а) принадлежащих Императору,
б) набальзамированных, в) прирученных, г) сосунков, д) сирен,
е) сказочных, ж) бродячих собак, з) включённых в эту классификацию,
и) бегающих как сумасшедшие, к) бесчисленных, л) нарисованных
тончайшей кистью из верблюжьей шерсти, м) и прочих, н) только что
разбивших кувшин, о) похожих издали на мух. »

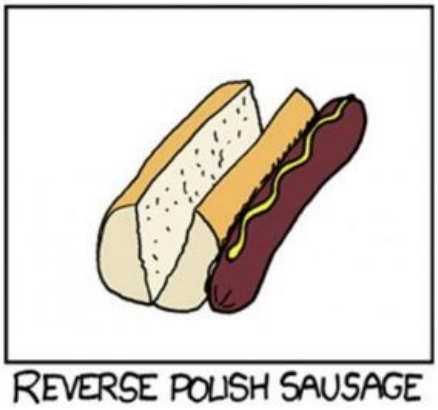
— *Борхес*

Нижеследующий текст ни в коем разе не следует воспринимать как сколько-нибудь полную или, упаси боже, научную классификацию.

Низко- и высокоуровневые языки



Нет, инструкции «УВЧ» там не найдёшь, как ни старайся.



Если бы создатели МК-61 делали хотдоги…

«Я весь последний год программировал на Си, но планирую опуститься ещё ниже! »

— Устами младенца…

«Выше к небу — ближе к богу »

— Почти АБС

Как легко догадаться, чем ближе язык к аппаратуре, на которой будет исполняться программа, тем он более низкоуровневый. Самый низкий уровень, ~~не~~доступный человеку, это программирование напрямую в машинных кодах. Никакого языка для этого не требуется, а требуется знать опкоды инструкций, поддерживаемых железом.

Немного выше находится ассемблер, он же «язык ассемблера» (*снобистск.*) — чуть более человеко- и терминаторо-читаемая запись этих же самых машинных кодов.

Основная трудность такого программирования для новичка — весьма непривычный подход ~~компьютеров~~ проектировщиков компьютеров ко многим привычным вещам, например, к вычислениям. Обычное сложение *x* + *y* превращается в помещение операндов (x и y) в регистры и вызов операции «целочисленное (sic!) сложение» (сам факт наличия отдельных операций исключительно для целых чисел может вынести неподготовленный мозг). Впрочем, советские калькуляторы были настолько суровы, что добровольно-принудительно обучали купивших их бедычей машинным кодам, поскольку были повёрнуты к лесу передом, а к пользователю — всеми этими кишками[5].

Основная трудность такого программирования для профессионала — ~~руки устают~~ приходится тратить время на разложение алгоритма буквально до атомов. Правда, человек может при этом пару атомов сэкономить, но это смотря какой человек. Отдельно НЕ доставляет тот факт, что компьютер в *класс*ификации не нуждается, и читать сплошную простыню такого кода так же просто, как статью, не разбитую на разделы, подразделы и абзацы.

В настоящее время код на асме видят, в основном, хакеры, крекеры, ~~спамы, куки~~, разработчики компиляторов и даже программисты на чём-то более высокоуровневом, когда дебаггер не может найти исходники. Раньше, однако, на нём педалились целые программные комплексы. Джоэл Спольски в подкасте с другим столь же старым пердуном вспоминал, как некая контора в конце 80-х вылетела с рынка, поскольку ВЕСЬ свой софт, в том числе текстовой редактор, писала исключительно на ассемблере[6].

Уровнем выше стоит (из самых известных) Си. В своё время его называли «самый высокоуровневый

ассемблер» и «универсальный ассемблер», уж больно легко для него было писать компиляторы подо все платформы. С другой стороны, многие называют его самым низкоуровневым из всех высокоуровневых языков: хоть в нём и появились имена для функций и переменных, такие милые особенности процессоров, как принципиально разная арифметика для целых и дробных чисел он сохранил во всей красе. За год до Си один Йода старый Forth язык написал, впрочем.

Ещё выше находятся языки, полные чуждых кремниевому процессору парадигм вроде ООП и ФП. Высокоуровневыми считаются *интерпретируемые* языки (хотя интерпретируемость, строго говоря, является свойством не языка, а сценария использования, и тот же BASIC может как интерпретироваться на лету, так и быть откомпилированым, многие языки специально создаются с прицелом на выполнение в интерпретаторе). По сходным причинам высокоуровневыми считаются языки, которые компилируются не в инструкции x86/ARM, а в инструкции виртуальной машины (Java и C#). Чем считать сами инструкции для виртуальной машины (джавовский байт-код и дотнетовский IL), высоко- или низкоуровневым языком — предмет спора не одного поколения учоных программистов.

Ручное управление памятью

Важным критерием низко/высокоуровневости языка является способ управления памятью: ручной или ~~автоматический~~ какой угодно другой, запрещающий грязными лапами лезть в память по произвольному адресу (не путать с наличием/отсутствием сборки мусора). Как известно, все баги можно поделить на два типа:

- Баги, связанные с порчей памяти и…
- …авотхуй, бля, нет больше никаких других типов багов, ибо на фоне первого типа все остальные — не баги, а добрые милые пушистые животные!

Об ошибках, связанных с несоответствием типов данных, программист узнаёт от компилятора. Об ошибках, связанных с нарушениям *лог*ики программист узнаёт из *лог*ов. Об ошибках, связанных с порчей памяти, программист узнаёт от разъярённых пользователей, у которых упала программа/сервер/хуй в самое неподходящее время, да ещё хорошо, если узнаёт вообще! Потому, что первым узнать о твоих переполняемых буферах может злобный хакер, а ты узнаешь уже о миллионах спизженных паролей.

Даже после того, как доказано существование ~~бога~~ бага, связанного с порчей памяти, найти и исправить его может оказаться труднее, чем пиздою улыбнуться. Секрет такой зловредности очень прост: программа может портить память в одном месте, а жопа вылезать в совершенно другом, да ещё и намного позже. Самый пиздец это memory corruption *высших степеней косвенности* — когда программа портит память в одном месте, затем при попытке с этой памятью работать — портит память где-нибудь ещё, ну ты понел. Фиксить такие баги экспоненциально труднее, правда, к счастью, и экспоненциально труднее написать такую длинную цепочку, чтобы она не порвалась. Но если ты счастливчик от природы — добро пожаловать в ад, приятель!

> О, канадский программист это особый тип. Он, ни на минуту не задумываясь, как рыцарь без страха и упрека, бросится фиксить самый свирепый баг китайского кода. Этот Баг живет там уже три года, и китайцы уже четырежды (каждый по разу) сообщали начальству, что он пофиксен. Но Баг каждый раз возвращался, как Бетмен в свой Готхем.
>
> *— Древний баян*

Истории известны случаи, когда программисты, которым было поручено исправление таких багов, просто увольнялись/были уволены. Известны и обратные примеры, когда программисты, сумевшие

Пара слов о виртуальных машинах

Выше написано, что C#, как и Java, компилируется «в инструкции виртуальной машины». Может ли за этим чисто техническим утверждением стоять драма? Ещё какая!

Начнём с того, что задолго до этого вашего жабовского байт-кода (ещё в 1977 году!) существовал p-code. Тогда же стали понятны плюсы и минусы компиляции высокоуровневого языка не напрямую в машинные инструкции, а в промежуточный универсальный код. Да, твои программы работают на любой платформе, для которой написан JIT, но работают они, сцуко, медленнее и не могут использовать фишки конкретного железа. Так что, ко времени Java-хайпа среди программистов уже вовсю ходила следующая шутка: «Говорить, что программы на Java лучше, потому, что работают на любой машине, это как говорить, что анальный секс лучше, потому, что им можно заниматься с человеком любого пола». Из чего следует, что в сознании типичного быдлопрограммиста JVM (*Java Virtual Machine*) поначалу находилась где-то на том же уровне, что и жопная гомоебля.

Когда Microsoft выпустил свой .Net, маркетологи мелкомягких почесали в затылке и то, что Sun назвала JVM, они окрестили CLR'ом (*Common Language Runtime*). Этот PR-ход был эквивалентен тотальной замене слова «ахтунг» на «личность небинарного гендера» — по сути то же самое, но вызывает куда меньшую социальную напряжённость. Самое смешное, что многие микрософтовские хомяки восприняли это на полном серьёзе и начали устраивать спецзабеги на программистских сайтах, доказывая джавистам, что у тех тормозное говно JVM, а у них самих — няшный CLRчик. На резонный вопрос: «А в чём разница?», ответы высасывались из всех возможных органов. Как бы то ни было, основная цель — заставить закоснелых сишников попробовать дотнет и убедиться, что не так уж он и плох — была достигнута.

Немного позже один из архитекторов Microsoft в каком-то интервью совершил знатный каминг-аут. На вопрос о

ВНЕЗАПНО пофиксить баг, к которому уже даже юзеры давно привыкли и перестали жаловаться, просаживали на радостях ползарплаты (с носа) на тусу с шампанским, блядями и кокаином.

На фоне вышеизложенного бывает ржачно почитать, как C++-ники катят баллоны на Javascript: у вас, мол, язык нетипо**безопасный**, а у нас несоответствие типов находит компилятор![7]

преимуществах CLR перед виртуальными машинами, он ответил нечто вроде: «Да бросьте, CLR и есть и всегда был виртуальной машиной». Разрыв шаблона у тех дотнетчиков, кто сам ещё не догадался, был сравним с таковым упоротого распиздяя, пришедшего в модный ночной клуб и под утро выяснившего, что он всю ночь отплясывал с чёрными властелинами.

Испортить память позволяют, в первую очередь C и C++, старый добрый Pascal и Rust (хоть он и содержит новые техники для затрудения этого прекрасного результата). VB, Javascript, Java и C# (без unsafe-элементов, добавленных для скорости) к порче памяти имунны.

Процедурные и объектные языки

Прямые потомки Алгола-58, откуда убрали оператор перехода (вернее, научились обходиться без него, а на уровне машинных команд прыжки вперёд через участки кода по-прежнему есть). Программы на них представляют собой последовательности действий, от «*x := 2*2*» до «*господьБог.создатьВселенную()*»:

- Алгол и Pascal
- C, C++, C#, Java и Oberon
- Ada, Simula и Modula
- Smalltalk и Objective-C
- ActionScript и Lua

— тысячи их! какой случайный язык, используемый (или использовавшийся) на практике, ни возьми — с высокой вероятностью он будет принадлежать к этой группе. Что и неудивительно: само название «Алгол» происходит от «Algorithmic Language», то есть язык для описания алгоритмов (но сам он, разделив судьбу птеродактилей, полетел в мир иной).

Язык C — особая мякотка. Несмотря на то, что он, строго говоря, является высокоуровневым языком, на фоне таких быдлоязычков, как Жаба или C# его называют низкоуровневым, так как уровень абстракций неприлично низок. Более того, именно на нем в 95% случаев пишут околосистемные вещи, вроде драйверов и ядер ОС, используя, впрочем, тонны ассемблерных вставок. Поэтому однозначно отнести его к какой-то категории, не вызвав бурления говн, невозможно. Иногда даже говорят, что C — «язык среднего уровня».

Прикладные языки

Это внезапно не Алгол! Языком прикладного программирования считали Фортран, на котором изначально предполагалось описывать вычислительные процессы. У него не было и до сих пор нет даже названия: «фортран» означает «транслятор формул», и так назывался не язык, а его компилятор. Изначально там не было даже правил видимости переменных, а все данные лежали в общей области данных: посчитал — и напечатал результат:

- Фортран всех версий - классика
- В нынешние времена благодаря Microsoft - Visual Basic (в том числе и VBA)
- Javascript (в том числе и node.js)
- Python, Perl, PHP, Ruby и Groovy

Иногда в прессе и на форумах поминают Фортран как что-то плохое. Возможно, так и было, но обижаться на то, что современные скриптовые языки происходят от него — глупость. Именно это направление развивалось больше всего (оно было прикладным) и именно на скриптах были обкатаны многие нововведения (такие, как словари), позже перенесённые в другие языки.

Лет 30-40 тому назад вопрос «Фортран или Алгол» был вполне насущным вопросом, и задавались им всерьёз. Даже был забытый ныне мем: «настоящие програмисты пишут на Фортране».

Языки для написания проводок

Они же «бизнес-ориентированные языки». Первоначально эту нишу занимал простецкий «Common Business-Oriented Language», который использовался в банках. Супермаркеты, ресторашки, кафэтэрии, склады, турагентства, таксопарки, железные дороги, госструктуры и т. п. — так до сих пор и используют его или один из его многочисленных диалектов:

- Кобол и ABAP
- PL/SQL
- Caché Script
- 1C

А теперь о грустном, да. Кобол был выблядком среди других языков, и проклятие тяготело над ним с самого начала: данные хранились вместе с кодом и внутри кода, в блоках «var» в заголовках модулей — и там неизбежно накапливался мусор. Позже этот приём перешёл в языки Алгол и Паскаль, а всё это ваше современное ООП — нежизнеспособный гибрид Алгола с Коболом. И какие молитвы ни возноси разным там компиляторам — не взлетит, и всё тут.

Привет из 50-х, коллеги!

Декларативные языки

Самым первым языком, пришедшим в эту нишу, доселе необитаемую, был Лисп, и всё, что здесь тусуется — прямые его потомки. По-видимому, условия здесь настолько не подходят для жизни, что не дохнут только немногие языки. Программы на них представляют собой не набор команд, а описание процесса: не *что* делать, а *как* делать. Эту группу надо разделить на функциональные и логические, а также смешанные:

Функциональные языки

Лисп когда-то был автокодом в скобочной записи, но этим дело не ограничилось:

- Scheme и Common Lisp
- Haskell
- РЕФАЛ
- ML с диалектами и расширениями(F#, Caml, OCaml, и т. д.)
- Erlang
- Agda
- Pure (term rewriting, кавай)

Логические языки

Изначально предназначались для создания экспертных систем aka искусственного интелекта. Но на практике, при попытке создать что-нибудь действительно полезное, выявились недостатки, которые привели к появлению гибридных функционально-логических языков.

- Prolog

Функционально-логические языки

- Mercury
- Curry

Глубокая идея функциональных языков на самом деле гораздо глубже, чем кажется на первый взгляд. Джон Маккарти, автор Лиспа, перевернул всё программирование 50-х и 60-х годов с головы на ноги: он поставил на первое место не данные, вокруг которых суетятся мелкие программки (как гномы, прыгающие вокруг Белоснежки в диснеевском мультике), а сами эти программки. Данные же он предлагал протаскивать через цепочки выполнения, а язык для описания этого дела был назван «List Processor», то есть Язык Обработки Списков.

А ещё, люди, которые пишут на Коболе, **не пишут** на этих языках. Так что сохраняй ледяное спокойствие, Анон, выгребая очередную порцию лживых обещаний с какого-нибудь X#$хантера.

Идеологии aka Парадигмы

> «Разделяй и властвуй (лат. divide et impera) »
>
> *— формула римского Сената*

Поскольку большая программа на олдовом процедурном языке по мере роста превращается в лютый пиздец размером с белого медведя при лисьей морде и лисьем же хвосте, возникли различные идеи по исправлению положения.

В разное время разным теоретикам приходили в голову разные умные мысли. Одни из этих идей вошли в арсенал процедурного программирования, другие остались чистой теорией, воплощённой в малоизвестных экзотических языках. В конце концов, практическое применение смогло найти ООП, оно же объектное программирование. Также особняком держится функциональное программирование.

Экскурс в историю

Многие элементы языков программирования, кажущиеся естественными, на самом деле были придуманы не сразу и не в готовом виде.

Процедурное программирование

Наиболее олдовая парадигма, в рамках которой и вывелся описанный выше песец, для борьбы с которым и придумали все прочие парадигмы.

Процедуры изобрёл ещё фон Нейман, до появления самих языков программирования, когда вычислительные устройства ползали и копошились где-то на уровне автомата Тьюринга. Идея заключалась в разделении больших программ на логически цельные подпрограммы (англ. *subroutine*), они же процедуры (англ. *procedure*), и во времена, когда компьютеры были большими, а программы маленькими, это изрядно помогало, но в конце концов разделения на процедуры оказалось недостаточно.

Структурное программирование

Придумано Дейкстрой, форсилось Виртом, придумка заключалась в дальнейшем разбиении кода подпрограмм на более мелкие фрагменты, а именно — на блоки *begin* и *end*, а также в отказе от оператора перехода GoTo. Но, кстати, вложенные *begin* и *end* появились в Алголе раньше, чем Дейкстра придумал свою парадигму, то есть Алгол появился раньше, чем идея о том, каким он «должен» быть *(спойлер:* это потому, что Дейкстра участвовал в разработке Алгола*)*. А ещё, в отличие от Алгола, в ранних реализациях Паскаля, который форсили как новейший структурный язык, дело с возможностью вложить в один *begin* и *end* сколько угодно других обстояло довольно туговато, так что он был шагом не вперёд, а назад…

Алсо, оператор GoTo стал на долгие годы предметом специальной олимпиады, в которой противники GoTo ссылались на Дейкстру, а сторонники — на Дональда Кнута. Сей именитый срач (со ссылкой на имена авторитетных теоретиков) постепенно затих с распространением ООП.

Модульное програмирование

Несмотря на все усилия, олдовые программы по-прежнему представляли собой длиннющий непрерывный листинг, печатаемый на длинной ленте. Идея модульности заключалась в том, чтобы этот длиннющий код разбить на более короткие и логически цельные == более лёгкие для чтения и восприятия куски.

Изначально для этого был запилен аж целый язык — Modula, представляющий собой модифицированный Pascal. Затем те же фишки добавили в обычный Pascal, а затем возможность разбивать программы на отдельные модули появилась и в других языках. Так что теперь практически все языки являются модульными.

Объектно-ориентированное програмирование

Появилось после того, как кто-то из хороших парней в стопицотый раз прошёлся по граблям с диким криком «почему йА». Идея заключалась в том, чтобы разделить код, разделить данные — а затем соединить соответствующие кусочки кода и данных. Но чтобы данные, ставшие ненужными — выбрасывались, в отличие от Кобола. Начальные подвижки к ООП появились уже в языке Simula, но первым объектно-ориентированным языком стал Smalltalk: собственно, его авторами и был введён сабжевый термин.

Однако кто-то что-то недопонял, поэтому Simula ВНЕЗАПНО обнаружила себя объектно-ориентированным языком, причем эталонным. Сначала ее недо-ООП слизал C++, а постепенно это веяние перешло и в другие языки. Самое смешное, что классическое ООП из Simula, используемое в Java, C#, C++ же и прочем мейнстриме, нихуя не канонично и имеет весьма мутное отношение к идеям, который вкладывались в ООП его создателями. Но всем как всегда.

ООП — не серебряная пуля и рядовому программистишке не даёт ничего. И при первой же возможности вырождается обратно в Кобол. *(спойлер:* Ведь проще всего, не думая, создать один-единственный класс на всю программу — и получается форменный Кобол с глобальными переменными*.)*

Все мейнстримовые языки сейчас — объектно-ориентированные. Так что применяется ООП почти везде, где надо и где не надо. Иногда — даже работает. Помимо классической C+±подобной реализации, характерные для C++ и потомков (Java, C#, PHP), есть более мягкая реализация без приватных полей (Python, Ruby), чуть другое ООР через prototype (характерен для JavaScript, но средний веб-проггер на него забивает) и более хитрый и новомодный «через подмешивание» (Scala, Rust, Go). Последний грозятся добавить в C++, чтобы олдфаги точно на стенку полезли.

Не могу молчать!

В своё время был эпичный срач, в котором олдфаги утверждали что любой алгоритм можно написать и без ООП, а потому оно не нужно. На что сторонники ООП утверждали, что ООП улучшает структуру программы, а «любой алгоритм» можно написать на любом тьюринг-полном языке, включая Brainfuck и Assembler. На это утверждение шёл ответ, что ООП не улучшает структуру, а превращает программу в mind-fuck с трудноуловимыми ошибками и утечками памяти. Что для нуба и в самом деле правда, и потому сторонники ООП это обвинение не только не отрицали, но и указывали на рукожопость и

скудоумие, особенно популярным троллингом было «старую собаку не научишь новым фокусам». Финальную точку в сраче ВНЕЗАПНО поставила Windows с ее объектно-ориентированным WinAPI (ЧСХ, со своей реализацией ООП на C). Это был EPIC WIN ООП.

Конкретно соснувшие хуйца олдфаги возненавидели лично Билла Гейтса (который в общем-то объективно говоря ничем не хуже Стива Джобса), и стали дружно фапать на тогда ещё полностью консольный UNIX с его чисто процедурным POSIX, вызывавший у них слезу умиления и ностальгию по тем временам, когда ещё не было ненавистных им юзеров, сидящих на винде. Ну а вчерашнее школоло, студенты-первокурсники, не понимая, где ООП нужен, а где нет, попыталось сесть на оба этих взаимоисключающих стула сразу: с одной стороны, проникли модным и крутым ООП, развивающимся под Windows же (sic!), а с другой — стали подражать олдфагам, ненавидящим Гейтса, без понимания причин этой ненависти. Вот так: весь этот ихний Масюкль — обезьянничание с ненавидимых и проклинаемых ими же технологий.

Продолжение банкета

Постепенно роль основных критиков ООП перешла к фанатам Лиспа. Забавно, что волну погнал Джон Бэкус, автор Фортрана: получая премию Тьюринга, он высказался в том смысле, прямо со сцены, что «все эти годы мы писали код неправильно». И вот недавно Micro$oft (уже без Била Гейтса) выпустила гибрид ежа с ужом — язык F#, сочетающий в себе «несовместимое»: ООП и функциональное программирование (на самом деле F# — это oCaml с блекджеком и LINQом под .net). Есть аналог и для JVM — Scala. И совсем уж левый Nemerle. Так что готовимся к продолжению банкета со вкусной едой.

Функциональное программирование

Появилось ещё до языков программирования, когда умный математик Алонзо Чёрч придумал лямбда-исчисление. Впервые было реализовано в Lisp (вторым по счёту ЯП после Фортрана). Заключается в низведении функций до уровня данных, а переменных до уровня констант. Идея здравая, поэтому настоящим программистам нравится, а заказчикам нет. А всё потому, что в функциональной парадигме нужно много думать, а потом написать десяток строчек, в то время как при императивном подходе можно написать 100 строчек вообще не думая — с тем же эффектом. Заказчик видит, что производительность (строчек в час) при использовании императивных языков выше, а значит, они лучше. Такие дела.

Функциональный код выглядит короче и наглядней во всех случаях, кроме тех, когда он выглядит длиннее и запутанней. А поддерживать и дебажить его — тот ещё ад, особенно если ошалевший небыдлокодер начинает необоснованно применять монады или функторы (а оно именно так и бывает в 95% случаев). Проблема в том, что большинство функциональщиков нихуя не владеют мат-базисом (λ-исчислением и теорией категорий), поэтому просто не понимают, зачем оно нужно (например, чтобы вычесть 1 из 1000000, требуется рекурсия глубиной как раз в 1000000), и используют по принципу «потому что могу» (порядка ради надо отметить, что императивщики дискретную математику вообще и теорию автоматов в частности знают не лучше, но для них это не критично).

Функциональное программирование окружает масса мифов. Перечитавшие Джоэла Спольски фонаты уверены, что любая программа на функциональном языке (даже если это домашка из первой главы SICP) автоматически масштабируется, летает и гарантирует, что его возьмут работать в Гугл. Конечно, функциональный код действительно неплохо масштабируются, но только если хорошенько попотеть. Даже общеизвестный LISP-овый факториал ни фига не пакетен, не масштабируем и тем более не поддерживает потоков. По сути, функциональная парадигма — это отличный инструмент для решения узкого круга задач, но уж никак не серебряная пуля, как рассказывают нам воннаби-небыдлокодеры. К примеру, попытка смоделировать систему с постоянно изменяющимся состоянием на каком-нибудь тру-функциональном Хаскеле превратится в монадный ад.

Поэтому функциональные языки всегда были в жопе, в серьёзном бизнесе не использовались и оставались уделом гиков, ибо деньги ими заработать нельзя. Так что функциональщикам только и остаётся, что после очередного высера на форуме: «вы все быдло с вашим ООП, смотрите как я получу бесконечную последовательность факториалов одной строчкой… как нахуй не надо?!», идти кодить на ненавистных Джавашарпах. Кушать-то хочется.

Событийно-ориентированное программирование

Парадигма, которая представляет программу как набор реакций на изменяющееся состояние. Ближайший аналог ИРЛ — условные рефлексы. Это позволяет здорово упростить представление: вместо одной обширной и необъятной блок-схемы, не умещающейся на листе ватмана формата A0, получается более удобный список небольших блок-схем, повешенных на события, что изрядно улучшает структуру обычной программы. Событийные языки поддерживают встроенный обработчик событий (к таковым относятся Delphi и C#), но при желании можно легко реализовать события на любом ООП языке с помощью паттерна en.w:Observer, чем и пользуются разработчики на C++ и Java. Раньше событийная парадигма была золотым стандартом фреймвёрков (её используют такие мастодонты, как Delphi, WinForms и ранние версии QT), но с популяризацией паттернов en.w:MVC и en.w:MVVM веяние сошло на нет.

Дело в том, что у событий есть одна маленькая проблема. Обработчики, висящие на событии, могут быть

объявлены где угодно и кроме как при отладке нигде не видны. Причем они могут изменить состояние вызвавшего их объекта (традиция передавать sender как первый параметр тянется уже добрых четверть века) или вызывать следующие обработчики (ведь события это ни что иное, как реинкарнация оператора goto на более высоком уровне). В итоге простейший код можно превратить в эталонную лапшу. Что быдло и делает.

Визуальное программирование

> «Вот она перед вами, коробка с карандашами
> В неё совершенно свободно помещается что угодно. »
>
> — *Владимир Приходько*

Бывает для взрослых и детей.

Для взрослых

Суть в том, что вместо написания кода рисуешь то, что нужно:

полностью визуальные

- ДРАКОН — просто рисуешь блок-схемы (требуется: космоплан «Буран»). Среди разработок примечательны: *алгоритм «Кровохарканье»* и *алгоритм «Острая диарея»*. В Википузии этот язык весьма активно проталкивает один из создателей, сверх-олдфаг В. Пароджанов (аж 1938 года рождения), который с помощью экс-админа Evacat даже выставлял статью на статус «хорошей», но не прокатило: обсуждающие накидали полную тележку замечаний. Да и в целом этот язык породил километровые срачи, которые Evacat одно время пытался разруливать, но потом плюнул и, видимо, свалил из проекта.
- LabVIEW — язык и среда разработки. На сегодняшний день является самым мощным и развитым инструментом визуального программирования. Основан на параллельно выполняющихся потоках данных. Позиционируется как инструмент для студентов, учёных и инженеров, не имеющих профильного образования в области программирования. Используется в основном для работы с железом и создания автоматизированных систем. Позволяет писать программы под Windows, macOS, Linux, операционные системы реального времени и FPGA (разумеется, только при наличии фирменного оборудования). Имеет бесчисленное количество библиотек, тулкитов и модулей на все случаи жизни — от обработки радиосигналов до разработки промышленных SCADA-систем. Язык имеет чрезвычайно низкий порог вхождения, однако содержит и продвинутые элементы: событийно-управляемое программирование, ООП, взаимодействие с .NET, Фреймворк акторов. Программирование в LabVIEW существенно отличается от того, к чему привыкли пользователи традиционных языков. Например, в нём считается дурным тоном использовать переменные, операторы if/else и последовательно выполняющийся код.

На деле является глючным проприетарным говном, стоящим тонны зелени, и крайне неудобном в работе: вместо кода программы разработчик думает о том, куда бы перетащить пиктограммы, чтобы не порвать провода и не наделать лишних связей, особенно доставляют неудобств элементы-рамки, о том, какого типа этот долбанный провод, и ожиданиепрокрутки окна при перетаскивании мышкой. Иногда поставляемые в комплекте модули зависают и глючат. Не говоря уже о несовместимости версий и вендор-локе. На основе LabView сделана среда программирования детских игрушек Lego Mindstorms, совершенно для этого непригодная даже для этого по тем же причинам, через несколько дней мытарств обычно ставится BricxCC, позволяющий писать на си-подобном языке без всяких «загрузок операционной системы». Задумка о визуализации исходника в виде графа вычислений неплохая, но только визуализации.

частично визуальные

- Delphi — рисуешь кнопочки и лепишь на них события: нажатие мышки, отжатие и так далее, «а всё остальное умный Дельфин сделает сам»;
- WinForms — то же, что и Delphi, только на C#.
- Qt — ещё один аналог, уже на C++. А вообще, тысячи их.
- SQL Manager Studio и аналоги — вместо написания SQL рисуешь структуру базы данных со связями *(спойлер:* работает через жопу, но тем не менее запросы быдлокодера, не знающего SQL, более-менее удовлетворяет — через жопу же! LOL~*)*

Для детей

А это учебные языки для маленьких детей, точнее, для родителей, желающих вырастить программера:

- Logo — самый знаменитый из таких языков, учит программировать алгоритмы, доставляет тем, что знаменитая «черепашка» для рисования на экране существует и в варианте программируемого робота => суть обучения сводится к весёлой игре «научи Черепашку рисовать на экране или на полу». Язык учит сразу функциональному программированию и основан на LISPe, а точнее, является

одной из немногих более-менее успешных попыток запилить в Лисп М-выражения, благодаря чему в отличие от чистого Лиспа гораздо более читабелен и понятен новичкам.
- Scratch — имеет занятное отображение алгоритмов: вместо традиционных блок-схем используется более красивое и наглядное отображение, показывающее, где у какой команды *begin*, а где *end*, вот так, что является нехилым плюсом для обучения и, без сомнения, WINом.
- Blockly — конструктор Лего от Google — косплеит Скретч.
- EToys также известный как Squeak — учит ООП и основан на Smalltalk. Любишь котегов? Так-то.
- Alice визуальный ООП язык
- КуМир и ПиктоМир — российские языки для детей, основанные на «Ершоле» (поскольку сочинены в основном ершовским учеником Кушниренко), наш ответ Scratch'у. Сначала осваивается ПиктоМир, а затем при помощи КуМира осуществляется переход от визуального программирования к абстрактному, подобно тому как Scratch является визуальной надстройкой над Смоллтоком. Кстати, Андрей Ершов и Джон МакКарти в жизни были большими друзьями, и последние версии ершола — это собственно такой лисп и есть. А Алан Кей, автор Смоллтока и Скрэтча, у МакКарти учился.

Какой язык учить?

Ибо ваистену,

> Какой язык учить?
>
> Хочешь программировать на выразительном и мощном языке: Python Нужно по-быстрому веб-сайт: PHP Желаешь в тусовку зовущих себя «рок-звездами» программирования: Ruby Реально нужно научиться программировать: C Ищешь просветления: Scheme Уйти в хандру: SQL Для офиса Microsoft: Microsoft Visual Basic Для получения постоянной, заурядной, но хорошо оплачиваемой работы по созданию финансовых приложений в офисной загородке под лампами дневного света: Java Тоже самое, но с аббревиатурами и списком сертификатов в своей подписи: C# Для получения волшебного, забытого в детстве, ощущения избавления от мании величия: Objective C

— С закрытой части быдлохабра

Но, однако,

> Какой язык учить?
>
> если реально нужно научиться программировать, но лень — си-шелл если вы недоучили C и хотите искупить вину страданием — шелл если у вас испорчено большинство кнопок клавиатуры — брейнфак если вы снисходительно относитесь к низшим по званию — ада если вам платят по строчкам кода — фортран если ваша пенсия по старости недостаточна — кобол если ваши программы записаны на магнитофонные кассеты — фокал если вы не доверяете компилятору перемножить два числа — язык ассемблера наоборот делать всегда если — форт если вы ностальгируете по модему без коррекции ошибок — перл если у вас чудовищная память, вмещающая две тысячи ключевых слов — PL/1 если вы хотите это, и то, и чтобы сразу — эрланг если процедуры вам удобно держать в массивах — алгол-68 если всё это для вас детский лепет — хаскелл

— Там же в мудрых комментариях

Алсо, немного занудства

> «вы все быдло и задроты. уважающий себя девелопер должен знать несколько языков и хотя бы иметь представления о многих технологиях. И в зависимости от задачи выбирать на чем писать. И не делать фетиш из языка. »
>
> *— Феерическая расстановка точек из смежного обсуждения*

Если вы вдруг нашли язык, с которым работаете, в разделах «быдлокодерские/небыдлокодерские языки», не спешите огорчаться или радоваться. Сами по себе вышеуказанные языки ничего не пишут и получающийся код является исключительно плодом вашего собственного моска и ручек. Другой вопрос, что некоторые языки проще для освоения и быстрого создания говна в промышленных масштабах, без значительных умственных и физических затрат, что делает их прекрасной средой для размножения быдлокодеров.

Для чего тогда сделано в этой статье вышеуказанное разделение? Just for lulz and holywars.

Также, закон Холивара применительно к языкам программирования звучит так:

> Ненависть к языку программирования обратно пропорциональна знанию этого языка, знанию и пониманию задач, для решения которых он был создан и умению применять его на практике.

Язык сам по себе ничего не решает, главное то, что находится в верхней голове, брат.

Галерея

Эпилептики, шизоиды, аутисты.

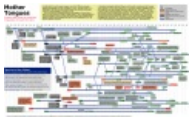
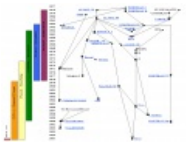


Схема языков.



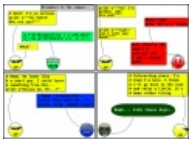
Развитие языков по времени.

Здесь и Фортран есть!

«Одна перфокарта — одно действие». Помнишь?

Как видят языки различные программисты.

Полиморфизм, инкапсуляция, наследование.



Python и все-все-все.

См. также

Копипаста:Программирование — дамп коллективного бессознательного /pr/, содержит провокационные суждения и радикальные мнения о разных языках и технологиях. Особенно интересен обзор языков от функциональщика.

Примечания

1. ↑ Неискажённый вариант примера:

   ```
   for (; P("\n"), R--; P("|"))
       for (e = C; e--; P("_" + (*u++ / 8) % 2))
           P("| " + (*u / 4) % 2);
   ```

   . Взят из шуточной истории языка Си. Пример в действии
2. ↑ Суммируйте Classic VB + VB
3. ↑ Пруфы тут, искать по слову «WinRAR».
4. ↑ Вместо того, чтобы описывать условие-предикат (позволяющее выделить из набора часть элементов) в виде отдельной функции, оно записано как *лямбда* — та же, в общем-то, функция, но анонимная и в виде выражения.
5. ↑ Как сложить «2 + 3» на калькуляторе w:Электроника МК-61? (На клавиатуре которого нет кнопки «=»!)
   - Ввести 2 в первый регистр.
   - Выполнить команду копирования из первого регистра во второй.
   - Ввести 3 (снова в первый регистр).
   - Нажать «+» и инициировать операцию сложения содержимого первого и второго регистров.
   - ?????
   - PROFIT!
6. ↑ Нет, они не были наркоманами. Идея состояла в том, что если программист осилил ассемблер, он хороший программист. Такая же кадровая политика, как ни забавно, используется по сей день, только вместо ассемблера теперь C++. Не потому, что нужен именно для этого проекта, а как фильтр профпригодности.
7. ↑ Такое несоответствие находится и чинится тривиальнейшим образом, в отличие от, а поверившего в подобные байки о безопасности C++ заказчика ждут увлекательные приключения вместе с программистами, которых он послушал.

Языки программирования

++i + ++i 1C AJAX BrainFuck C Sharp C++ Dummy mode Erlang Forth FUBAR God is real, unless explicitly declared as integer GOTO Haskell Ifconfig Java JavaScript LISP My other car Oracle Pascal Perl PHP Prolog Pure C Python RegExp Reverse Engineering Ruby SAP SICP Tcl TeX Xyzzy Анти-паттерн Ассемблер Быдлокодер Выстрелить себе в ногу Грязный хак Дискета ЕГГОГ Индусский код Инжалид дежице Капча КОИ-8 Костыль Лог Метод научного тыка Очередь Помолясь Проблема 2000 Программист Процент эс Рекурсия Свистелки и перделки Спортивное программирование СУБД Тестировщик Умение разбираться в чужом коде Фаза Луны Фортран Хакер

Языки программирования

Специальная олимпиада

AlexSword Avanturist Butthurt Check you DDoS Encyclopedia Dramatica/Атеист Fandom Grammar nazi IQ Livejournal.com Mac vs. PC S Special Olympics TeX X не умер Аборт Автосрачи Адекватная точка зрения Активная гражданская позиция Алкснис Аргументация в полемике Армата Арнольд Зукагой Артефакты Петербурга Атеизм Атеизм/Orthodox Edition Бесполезная наука Битва слона с китом Бодибилдинг Бокланопоцтит Бокс по переписке Ботинкометание Бульбосрач Бурление говн В/на Вайп Вандализм Ванкувер 2010 Леонид Василевский Вброс говна в вентилятор Веганы Великая Отечественная война Взлетит или не взлетит? Винилофилия ВиО Война правок Война пятницы тринадцатого Георгиевская ленточка Глобальное потепление ГМО Гоблин Говнарь Гогисрач Градус неадеквата Гражданская война в России Гринпис Демотивационный постер Детерминизм Диалог с собой Диванные войска Дружба между мужчиной и женщиной Дыхота Евромайдан Европейцы ли русские? Еда Жанрозадротство Женская логика Женя Духовникова Жестокость в компьютерных играх Иранский вопрос История древней Украины Как нам обустроить Россию Книга лучше Книга рекордов Гиннесса Комплексы Копирсрач Критерий Поппера Кровная месть Крокодил Кулинарный сноб Кургинян Курица или яйцо? Лавхейт Легалайз Ленд-лиз Лунный заговор Мавзолей Ленина Майдан Мицгол Моралфажество Моргенштерн Мужики vs бабы На самом деле Надмозг Наука vs религия Научный креационизм Национальная идея Не аниме Нот всего семь Обезьяна с гранатой