|  |
| --- |
| московский авиационный институт (национальный исследовательский университет)  Кафедра вычислительной математики и программирования  Дисциплина “логическое программирование” |
| Реферат |
| “Логические языки как первые языки для обучения программированию” |
| Группа: 8О-208Б  Студент: Захаров Игорь Сергеевич  Преподаватель: Сошников Дмитрий Валерьевич  Левинская Мария Александровна |

|  |
| --- |
| Москва 2018 |

Начнем с того, что мир программирования и в целом сфера информационных технологий сегодня очень стремительно развиваются. Эта сфера стала просто неотъемлемой частью нашей жизни. За последние 70 лет программирование изменило человечество. А ведь изначально компьютеры были большими и медленными, однако даже в таком виде они позволяли ускорить решение многих важных задач, преимущественно связанных с математикой, например, взлом шифров или расчёт артиллерийских таблиц. Самих языков программирования на то время не было. Теперь же, в наше время, мир языков программирования огромен, и продолжает расти. Так что давайте для начала заглянем в историю. Компьютеры и программирование активно развивались во время Второй Мировой войны, так как нужно было точно знать куда упадет бомба или расшифровывать карты, языки, да и финансовых средств на это выделялось гораздо больше во время войны, чем в мирное время. После войны компьютеры стали применяться в первую очередь именно для оборонных задач. Они создавались для решения серьезных математических проблем, и первыми программистами в большинстве своем были люди, которые составляли и выводили уравнения. Физики и математики тщательно разрабатывали алгоритмы. Они готовили подробную и точную документацию, анализировали решения своих коллег и искали математические доказательства. Однако по современным меркам, задачи, которые решали «Первопроходцы» были очень простыми.

Так и по сей день появляются новые и дополняются уже существующие языки. Каждый из них решает определенную задачу, так например php обычно используют для создания веб-страниц, javascript часто используют для создания прикладных приложений и скриптов в браузерах, язык Go, который чем-то похож на Си, часто используют для создания всяких бэкендов к приложениям. Из-за разнообразия языков, перед каждым человеком, который хочет стать программистом встает вопрос: С чего начать? Что для меня будет полезнее? Не пропадет ли у меня желание программировать, если я начну с этого языка? Что пригодится в будущем?

Выбрать первый язык для обучения, пожалуй, одна из самых сложных задач. Нужно учитывать не только в какой сфере ты хочешь работать, но и сложность синтаксиса, математического аппарата, структуры программ и многое другое. Давайте разберемся какие положительные и отрицательные стороны есть у логических языков, как первых языков программирования?

Логическое программирование принадлежит к декларативной парадигме программирования, которая основана на автоматическом доказательстве теорем. Элементы логического мышления создают основу, а логические утверждения служат в качестве компьютерной программы. Логическое программирование основано на теории и аппарате математической логики с использованием математических принципов резолюций. Языки логического программирования близки человеческому мышлению, в отличие от императивных языков, более ориентированных на компьютер. Самым известным, удобным и распространенным языком логического программирования является Пролог. Однако предшественником был язык Planner, который был создан Карлом Хьюитом в тысяча девятьсот шестьдесят девятом году. Этот язык предназначался для создания систем планирования роботов. Он мог выполнять поиск и анализ данных. В базе данных содержатся факты, которые можно создавать и изменять, требуется задавать логические отношения между заданными фактами, действия, которые разрешено производить, а также появился режим возврата, используемый при переборе вариантов во время решения задач. Режим возврата позволяет отбрасывать те варианты, которые не приводят к решению заданной цели.

Итак, познакомимся немного ближе с Прологом. Так в нём, как в наиболее популярном логическом языке программирования, описывается множество предикатов, задающих известные свойства изучаемой области, а решение выводится из этих свойств. При таком подходе внимание акцентируется на том, что представляет собой ответ, а не как его вычислить Логика Пролог-программы выражается в терминах отношений, представленных как правила (или факты). Факты - это не что иное, как заявления, они - истины нашей программы и имеют очень простой синтаксис. Они всегда начинаются с маленькой буквы и заканчиваются точкой. Факты могут также иметь аргументы в отношении формы (аргумент 1, аргумент 2 … аргумент N). Правила - это ключевое понятие в Прологе и позволяют делать выводы о нашем мире. Правило имеет вид: Заключение (arg1, arg2 ... argN): - отношение1, отношение2 ... отношение N. Вывод справедлив только в том случае, если все отношения между собой истинны. Запятые работают точно так же, как логические и. Так что это можно прочитать, поскольку вывод верен, если все, что происходит после: - также может быть доказано верно. Мы называем то, что происходит до: - головы, а затем - тела. Пролог имеет простой синтаксис, который легко понять, имея представление об описании логических формул. Этот язык имеет несколько важных качеств для начинающих студентов. Он не имеет привычных для программиста синтаксических конструкций из императивных языков, таких как циклы, ветвление. Решения на этом языке легко дополняются, то есть не нужно пытаться полностью решить задачу сразу, не нужно поддерживать большой шаблон, ведь всего несколько строк кода, обрабатывающих один единственный отдельный случай, являются рабочей программой. Если написать всего один случай одного из предикатов, можно сразу же протестировать его работу и добавить остальную часть кода позднее. Все это помогает студентам быстро осваивать новый для них язык программирования. Кроме того, предикаты в языке Пролог не имеют различия между входными параметрами и результатами. К примеру, один предикат пролога "append" может быть использован для объединения двух списков, а также чтобы делить список на две части, проверять, является ли элемент членом списка и для многого другого - каждое из этих применений требует отдельной программы в обычном языке программирования. Таким образом, Пролог приучает думать об обобщении и повторном использовании программ. Сами программы довольно компактны, они просты в написании и процессе отладки, так что скорость создания приложений на Прологе очень высока.

Важными понятиями, с которыми, как я считаю, стоит ознакомиться перед началом изучения Пролога, являются деревья, стек, рекурсия, backtracking. Так, для поиска ответа, Пролог строит дерево решений, и находит подходящий результат путем перебора. При неудаче или попытке поиска других решений осуществляется backtracking, или, другими словами, откат до места с которого возможно вхождение ответа. Знание рекурсии же необходимо, так как в Прологе нет циклов, и они реализуются с помощью рекурсии.

Логические языки программирования активно используются при создании систем искусственного интеллекта и экспертных систем, для обработки естественного языка и доказательств теорем, для разработки трансляторов, для работы с базами данных, так же эти языки используются в авиационной и медицинской отраслях.

Познакомившись немного ближе с Прологом можно сделать вывод, что языки логического программирования близки человеческому мышлению, в отличие от императивных языков, более ориентированных на компьютер.

Но что мы можем сказать про языки императивного типа? Императивная программа похожа на приказы, то есть представляют собой последовательность команд, которые должен выполнить компьютер. При императивном подходе к составлению кода широко используется присваивание. Наличие операторов присваивания увеличивает сложность модели вычислений и делает императивные программы подверженными специфическим ошибкам, не встречающимся при функциональном подходе.

Основные черты императивных языков: использование именованных переменных, использование оператора присваивания, использование составных выражений, использование подпрограмм и др.

Сравнивая языки императивного и декларативного типов можно отметить, что императивные языки вынуждают программиста следить за многими деталями, не связанными непосредственно с решаемой задачей, а это усложняет понимание, особенно у начинающих. Идея логического программирования и всех языков декларативного типа состоит в том, что программист не должен думать о внутреннем устройстве компьютера при написании программ. Это очень важное достоинство таких языков. В то же время, работая с логическими языками, нужно уметь определять ключевые свойства, которые определяют правильное решение задачи, нужно понимать, как выразить эти свойства логической формулой, которая может быть "выполнена" с помощью компьютера для получения ответа. Следовательно, начинающим изучать логический язык программирования следует знать основы математической логики и логики предикатов.

Несмотря на все преимущества логических языков, область их применения уже, чем у языков императивного типа, поэтому многим в дальнейшем будет необходимо выучить императивный язык программирования. Они не подходят для задач, связанных с обработкой большого количества вычислений и расчетов, для работы с мультимедийной информацией. При переходе к императивному типу у программиста могут возникнуть проблемы, нужно будет поменять стиль мышления и изучить много нового, чего нет в логическом программировании.

Пролог не был первым языком программирования с которым я познакомился. До него я уже писал и пишу программы на языках императивного типа, таких как C++, C. Освоить принципы написания и концепцию Пролога не стало большой проблемой для меня, поскольку я уже знаком с математической логикой, следовательно мне было несложно разобраться с правилами записи предикатов на Прологе. Мне нравилось, что Пролог довольно компактный язык, то, что занимало приличное количество строк на Си, на Прологе занимает всего пару строк.

Разобраться с написанием программы на языке Пролог не составило труда, однако, на мой взгляд, анализировать программу сложнее на Прологе, чем императивном языке.

Опыт логического программирования дает ученику не столько умение писать программы на данном языке, сколько показывает необходимость аналитического решения задачи перед написанием кода. Кроме того, этот подход учит разделять задачу на комбинацию более простых, то есть решать её сверху-вниз.

Для тех же, кто хочет продолжить изучать программирование, логический подход имеет большое преимущество. Им не нужно разбираться в языковых особенностях, чтобы описать задачу в компьютерном представлении. Данный подход учит логическому анализу задач, создавая таким образом бесценную основу для всей их будущей работы, связанной с программированием. Но, на мой взгляд, изучение логических языков полезно после освоения фундаментальных понятий программирования и информатики, после получения основных знаний и навыков работы с языками императивного типа и практики написания программ. Также мне кажется, что намного легче перестроиться от императивных языков к логическим, чем наоборот, потому что они близки человеческому мышлению, в то время как императивные языки имеют свою специфичную структуру, которую не так просто освоить. Таким образом, я считаю, что каждому нужно прикоснуться к парадигме логического программирования, но возможно не в начале своего обучения.

В этот сложный и объёмный мир IT нужен надёжный проводник. И Prolog –годится на эту роль отлично. Благодаря этому языку начинающий программист не только сможет быстро освоиться, но и научится думать в математическом смысле. Также на первое время использование Prolog позволит отказаться от проектирования вспомогательных алгоритмов, которые, казалось бы, никак не связаны с решением задачи, но в императивных языках без них обойтись невозможно.

Хотелось бы заметить, что мир информационных технологий, а именно программирование широко распространено не только в интересах, так называемого, старшего поколения, но и среди школьников. Я имею опыт работы со школьниками, и на сегодняшний день преподаю программирование детям разных возрастов. Немного уделю внимание этой теме, потому что я считаю стремление младшего поколения за компьютером не только играть, но и писать эти игры – очень похвальным. Поделюсь с вами своим опытом преподавания и с какими особенностями детской психологии я сталкивался. Вообще существует, как вы поняли, множество языков программирования и среди них есть часть классов, такие как логические и императивные по своему характеру и парадигме. Я бы выделил еще один тип, но его можно отнести к императивным типам, это визуальное программирование. К визуальным языкам программирования одним из самых популярных относят Scratch. Сам этот язык, как визуальная и событийно- ориентированная среда программирования для обучения детей младших классов получил очень немалую востребованность. Круг его использования популяризирован в робототехнике и создании первых для ребенка видео игр. В нём уже можно освоить пошаговость выполнения алгоритма, решающего изначально поставленную задачу, компьютером. Вообще говоря, в робототехнике для самых маленьких ребят есть множество платформ, программируемых в визуальных средах. Одна из таких платформ- тетра, это настоящий маленький компьютер, который подключается к вашему компьютеру и программируется в описанной выше среде Scratch. Ребенок получает электронную платформу, на которой самостоятельно устанавливает нужные ему датчики, диоды и так далее, всё это программирует с ориентировкой на выходы, называемые пинами, и уже понимает связь между программированием и электроникой. Для меня лично, робототехника еще в школьные годы тоже породила интерес к программированию. Самое интересное здесь то, что можно именно видеть прототип большого проекта, в котором работают все нужные функции, и да, они работают, код который ты пишешь и заливаешь в плату, можно сказать, отображает плоды твоих трудов. Это очень интересно- видеть, что код, который ты писал часами, это не только набор символов, а начинка, кровь, подпитка того, что ты хочешь разработать и показать людям и порадовать самого себя. Так вот, для ребят начальных классов языки визуального программирования, а в особенности, в применении в робототехнике, являются прекрасным началом правильного будущего. Они уже в раннем возрасте знакомятся с логикой, алгоритмами, решением задач и учатся думать последовательно, обращая внимание на все мелочи и исключительные случаи, которые могут встретиться в реализации. Всё это помогает по жизни, в повседневных ситуациях. Во время информационных технологий умение обращаться с компьютером уже просто жизненная необходимость. Если открыть доску вакансий, то уже практически невозможно найти работу без уверенного умения в пользовании ПК, всё потому, что интернет проник везде и никуда уходить не собирается, и новых технологий, более удобных, чем интернет, никем не предлагается. Вскоре умение программировать будет чем-то настолько же обыденным и привычным, что его начальные зачатки, вероятно, будут требовать практически на любой специальности, связанной с автоматизированным производством или офисными задачами, да практически любыми задачами. Ребенку, который еще с детства познал основы программирования, в дальнейшем будет гораздо легче обучаться, ведь новые знания в компьютерных технологиях будут ложиться на прочную базу, которую ему дадут преподаватели. Это как с математикой - не выучив числовой ряд, вы не сможете совершать примитивные операции сложения или вычитания, не говоря уже о более сложных вещах. А не имея совсем никакого абстрактного мышления, вы никогда не сможете понять ни одной формулы или закона математики. Что уж там говорить про математический анализ. Так и с программированием - без четкой структурной логики и ясного понимания принципа процессов ничего не выйдет.

Программирование развивает мышление и мозг в целом. И это не миф, а действительно правда. Еще в тысяча девятьсот восьмидесятом году профессор математики всемирно известного Массачусетского технологического университета Соединенных Штатов Америки, Сеймор Паперт, опубликовал книгу, в которой он утверждал, что дети, в детстве занимающиеся программированием, в будущем были более успешны в точных и логических науках. Позже его изыскания подтвердили серией исследований профессора Рой Реа и профессор Курланд. В 1984 они опубликовали статью, к которой на основании наблюдений подопечных сделали выводы, что обучение программированию в раннем возрасте с подробным объяснением всего механизма приводит к повышению когнитивных способностей детей, а также к тому, что им легче своих сверстников справляться с логическими и математическими задачами. Также польза программирования для повышения мозговой активности была неоднократно доказана учеными из MIT. В две тысячи тринадцатом году была создана образовательная платформа Code.org посвященная популяризации программирования. Среди спонсоров данной программы известные люди, получившие признание в индустрии информационных технологий, а это Марк Цукерберг- создатель популярной соц. сети Facebook, созданной изначально для студентов, Билл Гейтс- создатель огромной корпорации Microsoft, представляющей практически интересы той части людей, которая не предпочла яблочную технику, Джек Дорси- основатель одной из самых популярных соц. сети, являющейся на сегодняшний день площадкой и для общения и для рекламы, это Instagram. Данной программу столь признанные люди популяризируют для всех желающих освоить программирование.

Самое главное во всем этом — работать с хорошим преподавателем или курсом, который не будет вслепую учить ребенка нажимать клавиши, а объяснит, как работает каждая строчка кода, и что происходит в компьютере, когда мы запускаем написанную программу. Слепое обучение программированию не приводит ни к какому положительному результату, а, скорее, дает ложное представление о данном занятии и работе, и развивает логический аппарат в сторону замкнутости и закрытости к новым решениям. Недаром одна из самых больших проблем IT- индустрии- отсутствие новых, креативных идей и решений, даже не в части продукта, а в части реализации идеи продукта в реальности. Часто происходит так, что еще в детстве мозг ребенка ставят в ограничения, которые в будущем отражаются на всей его жизни- начиная от его выбора профессии и заканчивая его манерой говорить и решать сложные задачи. Такие ограничения крайне мешают успешному познанию сферы программирования и информационных технологий в целом. Конечно, здесь, как и везде, есть свой набор паттернов, но здесь эти паттерны должны обязательно быть кастомизированы под каждую задачу и не должны восприниматься как ограничения. Так же бесполезно обучение программированию без интереса обучаемого- если вы не заинтересуете ребенка новым видом деятельности, то он вряд ли станет им заниматься на постоянной основе, а именно так достигается тренировка логического мышления. Всегда помните, что программирование- это не только работа, но и эмоции, и если ребенку не симпатизирует данный вид активности, то и не надо его заставлять. И не нужно забывать хвалить его за успехи- программирование, все же, тоже челлендж — только перед самим собой. А иногда и перед начальством, и тогда хвалит уже начальство. Психология так же должна присутствовать здесь. В общем и целом, кодинг может быть тяжелым занятием, и не всем он подойдет — но он крайне хорош для развития логики и терпения. Не бойтесь программирования.

Ребята средних классов, примерно после шестого, способны писать уже код, понимать и запоминать синтаксис различных языков, иметь представление о предназначении языка. Я советую опять же, начинать именно с императивных языков, таких как Си, С++, python. Самым понятным для обучения из перечисленных для школьника будет python. Собственно говоря, связано это с тем, что многие функции, методы там уже реализованы, нет такой строгости написания как, например в Си. Хотя данная строгость именно и учит критически мыслить, относиться к коду с трепетом. Обучающийся должен постепенно осваивать алгоритмы. Одними из первых алгоритмов я советую работу с простыми числами, а именно разбор по цифрам, различные перестановки цифр. Подобные алгоритмы заставляют вспомнить немного математики, деление по модулям, домножение и пересчет чисел. Изучая синтаксис языка постепенно, эти алгоритмы возможно реализовать уже на втором или третьем занятии, когда ученик понимает что такое переменные и как работают некоторые виды циклов. Дальше можно работать с простейшими алгоритмами поисков максимумов и минимумов. Переходить к массивам данных и работать уже с более сложными и полезными алгоритмами. Со школьной скамьи некоторая часть школьников занимается предметными олимпиадами, так вот по информатике тоже недавно ввели и развивают данное направление. Задачи олимпиадного характера включают в себя очень серьезный подход к решению их, подразумевают знание учащимся алгоритмов дискретной математики и математического анализа, структур данных. В олимпиадном программировании как раз таки есть хитрость по использованию языка, на котором учащийся будет решать задачу, здесь есть ограничения по времени выполнения алгоритма, скорость выполнения как раз достигается и особенностью языка, самым популярным в таких задачах является с++ за счет своей скорости и гибкости, возможностью доступа к памяти. Но эти задачи требуют очень нестандартного подхода и мышления обучающегося. Обычно формулировки в олимпиадах имеют прикладной характер, и кажется, что задача действительно реальная, но и отрицать мы этого не можем, ведь практически любую жизненную ситуацию, требующую логического мышления, какого- то расчета или перебор возможных ситуаций можно запрограммировать. Например, теория графов, которую изучает такая наука, как Дискретная математика, поможет нам решить любую задачу связанную с выбором оптимального пути. Комбинаторика, которая также рассматривается в рамках данной науки, также поможет нам легко распределить между объектами процесса необходимые мощности, даст нам ответ на то, сколькими способами можно получить тот или иной исход и так далее.

Таким образом, в подведение итога, могу выделить одну главную мысль и пользу программирования. Заниматься данным видом деятельности и обучаться этому полезно с самых ранних возрастов. Существует множество видов программирования, которое даст вам пользу на любом уровне, и в любом возрасте. И не совсем значит, что, если взрослый человек не понимает, что происходит в обычном коде, то и ребенок не поймет. Хоть и дети бывают разных возрастов и уровней развития, в зависимости от которых и стоит подбирать соответствующий язык, среду и задачи. Но в любом случае, это пойдет только на пользу логическому мышлению. И не важно, императивный это язык или логический, просто у каждого своя специфика, требующая своего подхода. Однако мне кажется, что всё- таки лучше начинать с императивного, с него перестроиться под логику проще, нежели с логики исчисления предикатов на пошаговое выполнение кода компьютеров в стиле приказов.