ТУБ №12:
Поколения и уровни языков программирования

На сегодняшний день существует, по разным оценкам, от 2’500 до 10’000 языков программирования. Если при виде этих чисел Вам захотелось бросить всё и уйти в маляры, подождите! Вот, возьмите кисть и банку краски.

Конечно же, знание всех языков программирования не требуется для того, чтобы работать программистом. Да это и глупо было бы! Потому что как только Вы изучили бы все существующие языки программирования, обязательно нашёлся бы кто-то, кто решил бы создать ещё один, новый. Так можно всю жизнь изучением заниматься, а ведь кто-то должен на этих языках ещё и программы писать.

Есть и вторая крайность. Это когда начинающий программист посещает двухмесячные курсы по программированию на языке Compotik (подставьте любое название сами) и решает, что уж теперь-то он знает достаточно, чтобы до самой старости быть программистом №1 в мире по мнению Американской ассоциации стоматологов. Печальное зрелище.

На самом деле язык программирования — это не цель и не предмет гордости. Язык программирования — это просто инструмент для решения задачи. А задачи, как известно, бывают разные.

Возьмём, к примеру, молотки и им подобные инструменты. Согласитесь, глупо было бы, вешая картину, забивать гвоздь в стену кувалдой: как бы тут окна в квартиру к соседям непредвиденного не случилось. Или вот, к примеру, киянка — специальный молоток, который делают из дерева или резины. Надолго ли такого молотка хватит, если нужно сколотить скворечник или собачью будку, или целый деревянный забор? Зато если нужно аккуратно выправить металлическую пластину, ему нет равных.

Можно пойти и купить себе сразу комплект из всех возможных видов молотков, но возникает вопрос: означает ли это, что Вы сможете без специальной подготовки одинаково хорошо управиться и с киянкой, и с отбойным молотком?

Чтобы быть профессионалом, необходимо владеть разными инструментами, но брать при этом не количеством, а качеством — умением грамотно применять эти инструменты. Тогда можно и избу одним топором построить, если очень уж нужно будет. Но чтобы грамотно выбирать инструменты, нужно понимать, чем они отличаются, зачем создавались и какие вообще бывают задачи.

Поколения языков программирования

Со всем этим многообразием языков программирования универсальной общепринятой классификации для них не существует. Тем не менее, попытки хоть как-то сгруппировать похожие по каким-то признакам языки предпринимались всегда. Одна из таких попыток связана с выделением поколений языков программирования.

Давным-давно, когда компьютеры были большими, а программы маленькими, все вели разработку прямо в машинных кодах. Именно машинные коды и считаются языками первого поколения. Конечно, долго так продолжаться не могло, поэтому как только мощности стали позволять, все дружно переключились на так называемые языки ассемблера, которые являются языками второго поколения.

Чтобы осознать, насколько важным был этот шаг, вспомним о 8-битных игровых приставках Dendy, которые были особенно популярны у нас в 90-е годы XX века, да и сейчас пользуются спросом у любителей. Фрагмент программы для этой консоли, выполняющий копирование 256 байт из памяти игрового картриджа в так называемую видеопамять, а также настройку прокрутки фонового изображения, при записи в машинном коде выглядит примерно так (в двоичной системе счисления):

```
01001000
10101001 00000000
10001101 00000011 00100000
10101001 00000010
10001101 00010100 01000000
10101001 00000000
10001101 00000101 00100000
10001101 00000101 00100000
01101000
01000000
```

Этот же фрагмент программы при записи на языке ассемблера выглядит так:

```
pha

lda #$00
sta $2003
lda #$02
sta $4014

lda #$00
sta $2005
sta $2005

pla
rti
```

Это уже намного проще для понимания, особенно если знать, что LDA — это сокращение от «LoaD Accumulator», STA — «STore Accumulator», PHA — «PusH Accumulator», PLA — «PuLl Accumulator», а RTI — «ReTurn from Interrupt». Опытный программист к тому же знает, что означают записанные после названий команд числа. Ну и очевидно, в таком разнообразии символов намного сложнее запутаться, чем в нулях и единицах.

Конечно, развитие языков программирования на этом не остановилось и вскоре, примерно между 50-ми и 70-ми гг. XX века, начинается массовое использование языков третьего поколения. Именно эти языки получили наибольшее распространение и продолжают использоваться и создаваться и в наши дни. К ним относятся, например, широко известные Pascal, C, C++, Basic, C#, Java, Python, Ruby и т.п. Отличительной особенностью этих языков является возможность задавать алгоритм работы программы без привязки к особенностям конкретного устройства или класса устройств. Например, на языке ассемблера вычисление по формуле $y = \sin\left(x^2 + 2x + 3\right)$ может быть записано так (для сопроцессора x87):

```
fld     [x]
fld     st0
fmul    st0, st0
fxch    st1
fadd    st0, st0
faddp
fld1
fld1
fadd    st0, st0
faddp
faddp
fsin
fstp    [y]
```

Аналогичные действия на языке третьего поколения можно записать, например, так:

```
y = sin(x * x + 2 * x + 3);
```

а потом ещё и добавить ввод и вывод:

```
INPUT "Введите x: "; X
Y = SIN(X * X + 2 * X + 3)
PRINT "y = "; Y
```

или чуть более громоздко:

```
import math

x = int(input("Введите x: "))
y = math.sin(x * x + 2 * x + 3)
print("y = ", y)
```

или с полной самоотдачей:

```
import java.util.Scanner;

public class Main
{
    public static void main(String[] args) {
        System.out.print("x: ");
        Scanner in = new Scanner(System.in);
        double x = in.nextDouble();
        double y = Math.sin(x * x + 2 * x + 3);
          System.out.println("y = " + y);
    }
}
```

Совершенно очевидно, что такой способ записи намного более нагляден и разобраться в написанном полгода спустя намного проще, чем в случае языка ассемблера или машинного кода.

Тем не менее, развитие языков программирования на этом не остановилось. Решение многих задач всё ещё слишком громоздко выражается на языках третьего поколения, оставляя возможности для множества ошибок. Дальнейшее развитие ознаменовалось появлением языков четвёртого поколения (4GL).

Идея 4GL начинает развиваться в 70-х гг. XX века и основывается на стремлении при написании программ вместо вопроса «Какие действия нужно выполнить?» отвечать на вопрос «Какой результат должен быть получен?»

К началу XXI века акценты слегка изменяются и под 4GL начинают понимать среды программирования, которые позволяют программисту писать минимум кода (в идеальном случае — вообще не написать ни строчки кода) для достижения требуемых результатов. Такая среда программирования значительную часть кода генерирует самостоятельно, давая возможность разработчику оперировать более абстрактными понятиями, чем в языках третьего поколения. Основным применением языков четвёртого поколения становятся бизнес-задачи, а первыми кандидатами на автоматизацию оказываются работа с базами данных, генерация отчётов, разработка сложных пользовательских интерфейсов и т.п.

Значительная часть программы в языках четвёртого поколения разрабатывается с использованием визуального проектирования: разработчик работает не с текстом программы, а с графическим представлением отдельных её частей. Разумеется, универсального решения на все случаи жизни не существует, поэтому на сегодняшний день многие языки четвёртого поколения сохраняют возможности, больше характерные для предыдущих поколений: в тех случаях, когда требуется реализовать в программе специфическую для конкретной задачи логику, программист имеет возможность сформулировать решение «по старинке».

Ярким примером языка четвёртого поколения, получившего значительное распространение, является язык Delphi. Его отличительными особенностями по сравнению с языками третьего поколения, которые постепенно начинают поддерживать те или иные элементы 4GL, является наличие в самом языке конструкций, предназначенных для интеграции со средой визуального проектирования.

Иногда к языкам четвёртого поколения также относят так называемые функциональные языки программирования, однако их отличия от языков третьего поколения не настолько существенны, чтобы значительно упрощать разработку программ, как это происходило с предыдущими поколениями, а сами идеи, положенные в основу функциональных языков, успешно используются и при разработке с использованием других языков программирования.

В 80-е гг. XX века на волне возрастающей популярности языков четвёртого поколения началось также обсуждение языков пятого поколения. По задумке при работе с этими языками программист просто формулирует решаемую задачу и ограничения, которым должны соответствовать результаты её решения, а конкретные алгоритмы разрабатываются самим транслятором при обработке исходного кода программы. На практике это оказалось неосуществимым, поэтому, если не считать нескольких языков, показавших очень скромные результаты, языки пятого поколения по сути ещё не были созданы.

Языки высокого и низкого уровня

Одним из самым популярных способов сравнения двух языков является обсуждение уровня языка. Часто можно встретить понятия язык низкого уровня и язык высокого уровня, однако этим не ограничивается и нередко встречаются утверждения наподобие «Язык X более высокоуровневый, чем Y, но более низкоуровневый, чем Z».

Чёткого общепринятого определения того, что такое уровень языка, опять же не существует, однако неплохим приближением будет следующее:

> Уровень языка — среднее количество машинных команд, приходящихся на одну команду языка программирования.

Самые низкоуровневые языки программирования — машинный код и языки ассемблера. С машинным кодом всё понятно: одна команда в машинном кода — это и есть одна машинная команда. Что же касается языков ассемблера, то, как правило, в них тоже соответствие машинным командам однозначно, однако иногда трансляторы с этих языков позволяют использовать команды, которые не поддерживаются напрямую аппаратным обеспечением, однако могут быть представлены комбинацией машинных команд.

Условная граница между языками высокого и низкого уровня проходит там же, где и между языками второго и третьего поколений. Языки третьего и последующих поколений считаются высокоуровневыми и характеризуются тем, что не привязаны к особенностям аппаратного обеспечения, ввиду чего одной команде в таких языках могут соответствовать десятки и даже сотни машинных команд.

Как уже было отмечено ранее, различные языки программирования лучше адаптированы для решения различных задач. По этой причине субъективное ощущение уровня языка программирования может значительно отличаться у людей, решающих с его помощью разные задачи. Так что споры о том, какой язык высокоуровневый, а какой — не очень, похоже, не прекратятся никогда. И это замечательно, ведь именно в споре рождается истина.

[1] https://www.ibm.com/developerworks/ru/library/cl-3gl4glclouddev/index.html

[2] https://books.google.by/books?id=LUBv6K6Yrw4C&lpg=PP1&hl=ru&pg=PA14