Московский Авиационный Институт
(национальный исследовательский университет)
Факультет “Прикладная математика и информатика”
Кафедра 806

Реферат на тему «Классификация языков программирования»

Выполнил студент группы М8О-108Б-20
Попов Матвей Романович
Преподаватель: Трубченко Никита Михайлович

2020

Содержание

Введение

Несмотря на то, что не существует единой общепринятой таксономии языков программирования, есть огромное количество черт, позволяющих провести классификацию языков по некоторым параметрам.

Основные способы классификации:

- низкий/высокий уровень
- по способу реализации
- безопасный/небезопасный язык
- по парадигме
- по поколению

Уровень языка

Уровень языка (низкий или высокий) обычно определяется двумя показателями. Первым показателем является масштаб преобразований кода программы, необходимых для её исполнения (чем меньше масштаб, тем ниже уровень). Второй показатель — учитывает ли семантика (смысловое значение слов) языка мышление человека больше, чем «мышление» машины (чем язык «ближе к человеку», тем выше уровень данного языка).

Основными особенностями языков низкого уровня являются:

- реализация на аппаратном уровне
- невозможность одинаковой интерпретации одного и того же кода разными машинами
- объёмная запись простых конструкций (например, арифметических выражений)

К языкам программирования низкого уровня относят языки первого и второго поколений, то есть машинные языки, требовавшие реализации на аппаратном уровне. В течение роста сложности программ появилась потребность в языках программирования, которые бы легко модифицировались и переносились с одной машины на другую. Начали создаваться языки высокого уровня, первыми такими языками были Plankalkül и Fortran.

Достоинства высокоуровневых языков программирования в сравнении с низкоуровневыми:

- укороченный и более понятный на вид исходный код
- возможность одинаковой интерпретации исходного кода для разных машин

Недостатком высокоуровневых языков является увеличенное время работы программы, так как они являются лишь надстройкой над низкоуровневыми языками и должны быть обязательно переведены в машинный код.

Способ реализации

По способу реализации все языки программирования разделяются на три основные группы: компилируемые, интерпретируемые и встраиваемые.

Исходный код, написанный на компилируемом языке программирования, должен быть обязательно переведён в машинный код. Программа, переводящая исходный код в машинный, называется компилятором, а сам процесс — компилированием. К компилируемым языкам относят такие известные языки, как C, C++, Pascal, ALGOL, COBOL, Fortran, Haskell, Delphi, Rust, Swift.

Интерпретируемый язык программирования не требует перевода в машинный код, в нём операторы программы последовательно выполняются друг за другом с помощью некой программы-интерпретатора. Наиболее распространёнными интерпретируемыми языками являются Lisp, Python, JavaScript, Forth, PHP, Perl. Стоит также отметить, что для любого компилируемого языка можно написать интерпретатор.

И у того, и у другого способа есть свои достоинства и недостатки. Программы, написанные на компилируемых языках, работают гораздо быстрее, так как для выполнения машинного кода требуется гораздо меньше времени и вычислительной мощности, однако после каждого изменения исходного кода требуется перекомпиляция. В то же время в интерпретаторе возможно изменять код прямо во время исполнения, также на разработку и отладку таких программ уходит гораздо меньше времени.

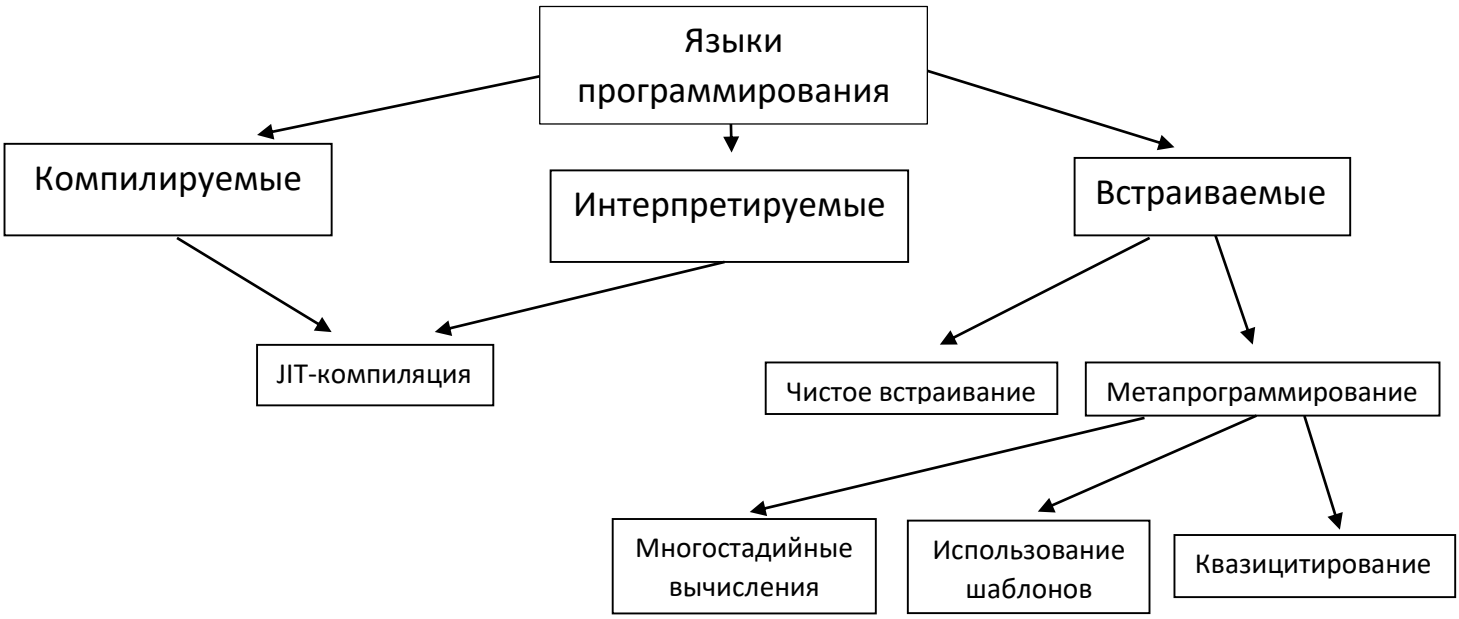
Существует также промежуточная стадия, когда исходный код компилируется в так называемый байт-код, а после этот код исполняется интерпретатором. Такой подход значительно ускоряет выполнение программы, однако не устраняет этот недостаток полностью. C#, Java, Ruby относятся к подобным языкам, также подобные интерпретаторы есть у языка Python.

Встраиваемые языки программирования — языки, реализующиеся внутри транслируемого языка, то есть языка, «переводящегося» на данный встраиваемый язык. Такие языки имеют ещё несколько реализаций:

- чистое встраивание
- метапрограммирование
    - многостадийные вычисления
    - использование шаблонов
    - квазицитирование

Метапрограммирование — написание программ, которые в ходе выполнения создают новые программы либо изменяют собственный код.

Схема классификации языков программирования по способу реализации



Безопасность языка

Чтобы уменьшить риск возникновения ошибок доступа к памяти, например, переполнение или утечка, была создана система типов — правила, ставящие в соответствие конструкциям языка (переменные, функции и т. д.) различные свойства. Система типов есть практически у каждого языка программирования, она сопровождает программы во время их выполнения.

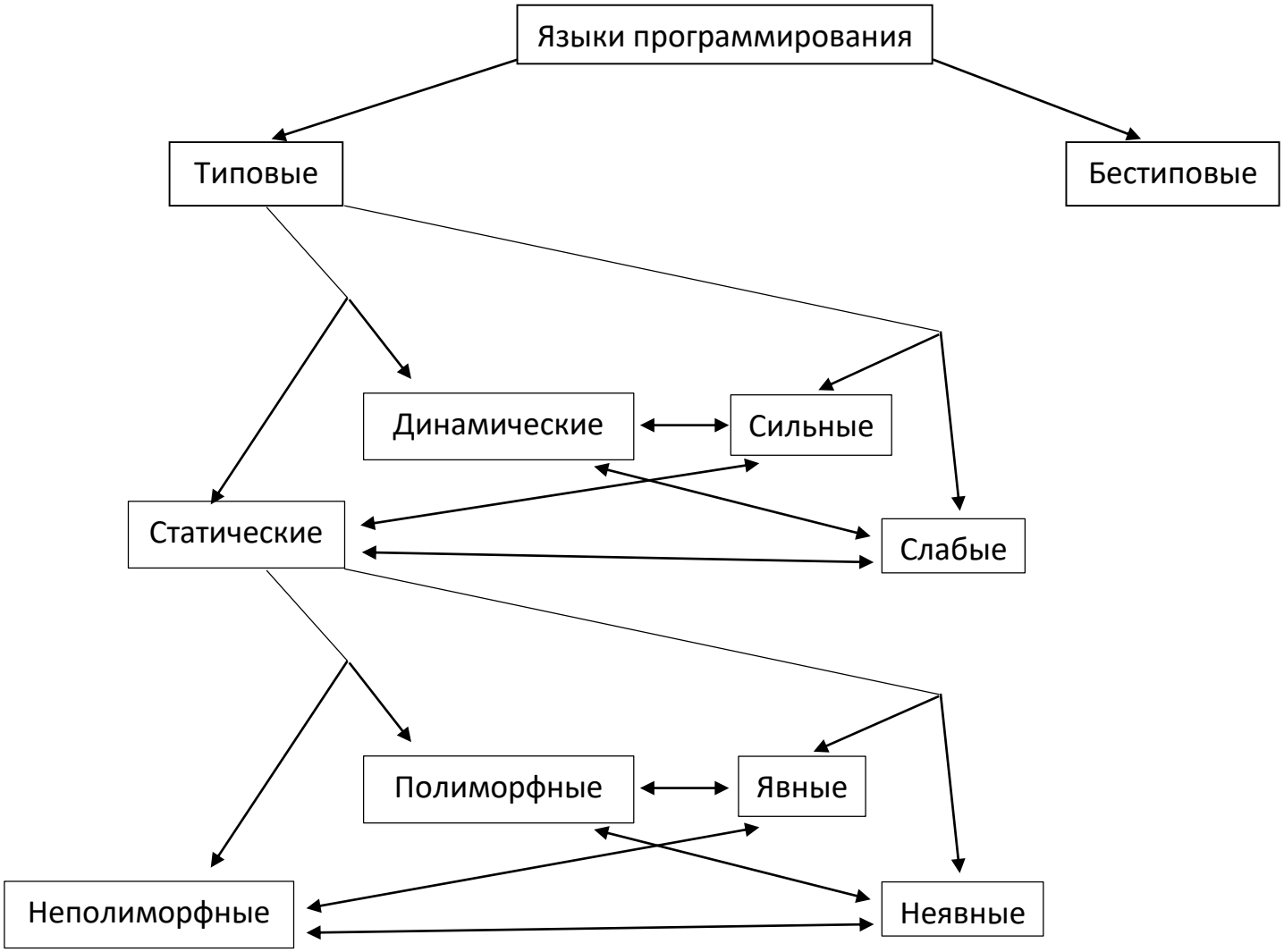
Системы типов бывают двух разновидностей: динамические и статические. Их основное отличие в том, что в динамической типизации переменная связывается со своим типом в момент присвоения ей какого-либо значения, а в статической типизации связывание переменной и её типа происходит одновременно с инициализацией переменной.

У языков программирования со статической типизацией существует дополнительный параметр для классификации — полиморфизм. Полиморфизм — это свойство, позволяющее исполнять один и тот же код для разных типов переменных и других данных. Помимо этого, статический тип подразделяется на явный (типы для объектов в программе требуют декларацию) и неявный (тип для объектов выводится самостоятельно).

Независимо от динамичности или статичности системы типов бывают сильные и слабые. Сильная система сохраняет тип для любого выражения постоянным, а слабая позволяет его изменять.

Таким образом, безопасный язык программирования — язык, программы на котором во время исполнения никогда не выходят за рамки исполнения, если они, разумеется, были приняты компилятором как правильно построенные. Программы, написанные на небезопасных языках, могут нарушить целостность данных и вызвать своё аварийное завершение или аварийное завершение операционной системы.

Схема классификации языков программирования по системе типов



Класс языка

Парадигма языка

Парадигма программирования — совокупность идей и понятий, позволяющих определить стиль написания программ для каждого языка. Также парадигма даёт понять, для каких целей лучше подходит тот или иной язык программирования. Парадигма языка определяет его класс. Один язык программирования может относиться к нескольким классам, равно как и один класс может содержать в себе несколько языков.

Императивное программирование

Основной парадигмой программирования является императивное программирование. Исходный код программы в императивном программировании представляет из себя список команд-инструкций, которые машина должна выполнить. В императивных языках базовыми действиями считаются использование именованных переменных, присваивания и подпрограмм. Первые высокоуровневые языки программирования были именно императивными, именно они являются наиболее популярными и по сей день. Императивное программирование дало старт множеству других парадигм.

Процедурное программирование

Процедурное программирование даёт возможность объединять несколько последовательно выполняемых операторов в подпрограммы. Другими словами, задачи в программе на процедурном языке разбиваются на множество шагов, которые последовательно выполняются машиной. Процедурными языками являются C, Fortran, Pascal. Также к процедурному программированию можно отнести машину Тьюринга.

Структурное программирование

В структурной парадигме программа представлена как иерархическая структура блоков (сгруппированных наборов идущих подряд команд). Существование такой парадигмы математически обосновывает знаменитая теорема Бойма-Якопини-Миллса, согласно которой любой алгоритм может быть приведён к структурированному виду.

Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) основывается на представлении программы как множества объектов. Определённые объекты

образуют класс, а классы объединяются в иерархию наследования. Фактически ООП позволяет улучшить метод структурного программирования благодаря введению объектов, которые могут реагировать на посылаемые им сообщения, используя свои данные. Разновидностями ООП являются агентно-ориентированное программирование (основное понятие — агент, обладающий поведением и взаимодействующий со средой), компонентно-ориентированное программирование (компонент — независимый модуль исходного кода, предназначенный для использования в других конструкциях), прототипное программирование (отсутствие классов, вместо них прототип — копируемый экземпляр объекта).

Аспектно-ориентированное программирование

В 2001 году для языка Java было разработано расширение, основная идея которого заключалась в разделении функциональности для улучшения разбиения программы на модули. Это расширение является одним из примеров аспектно-ориентированного программирования. Ранние парадигмы программирования были неспособны выделить некоторую функциональность в отдельную сущность. Аспектно-ориентированное программирование позволяет решить эту проблему благодаря введению новых понятий (аспект, совет, точка соединения).

Обобщённое программирование

Основной особенностью обобщённого программирования является возможность применения одного и того же алгоритма к разным типам данных. Обобщённое программирование в том или ином виде присутствует во многих языках программирования других парадигм.

Декларативное программирование

Данная парадигма программирования является полной противоположностью императивного программирования. Если в императивном программировании описывается алгоритм, то есть решение задачи, то в декларативном программировании описывается ожидаемый результат. В декларативных языках нет переменных, операторов присваивания, состояний. Основными подвидами декларативного программирования являются функциональное и логическое программирование.

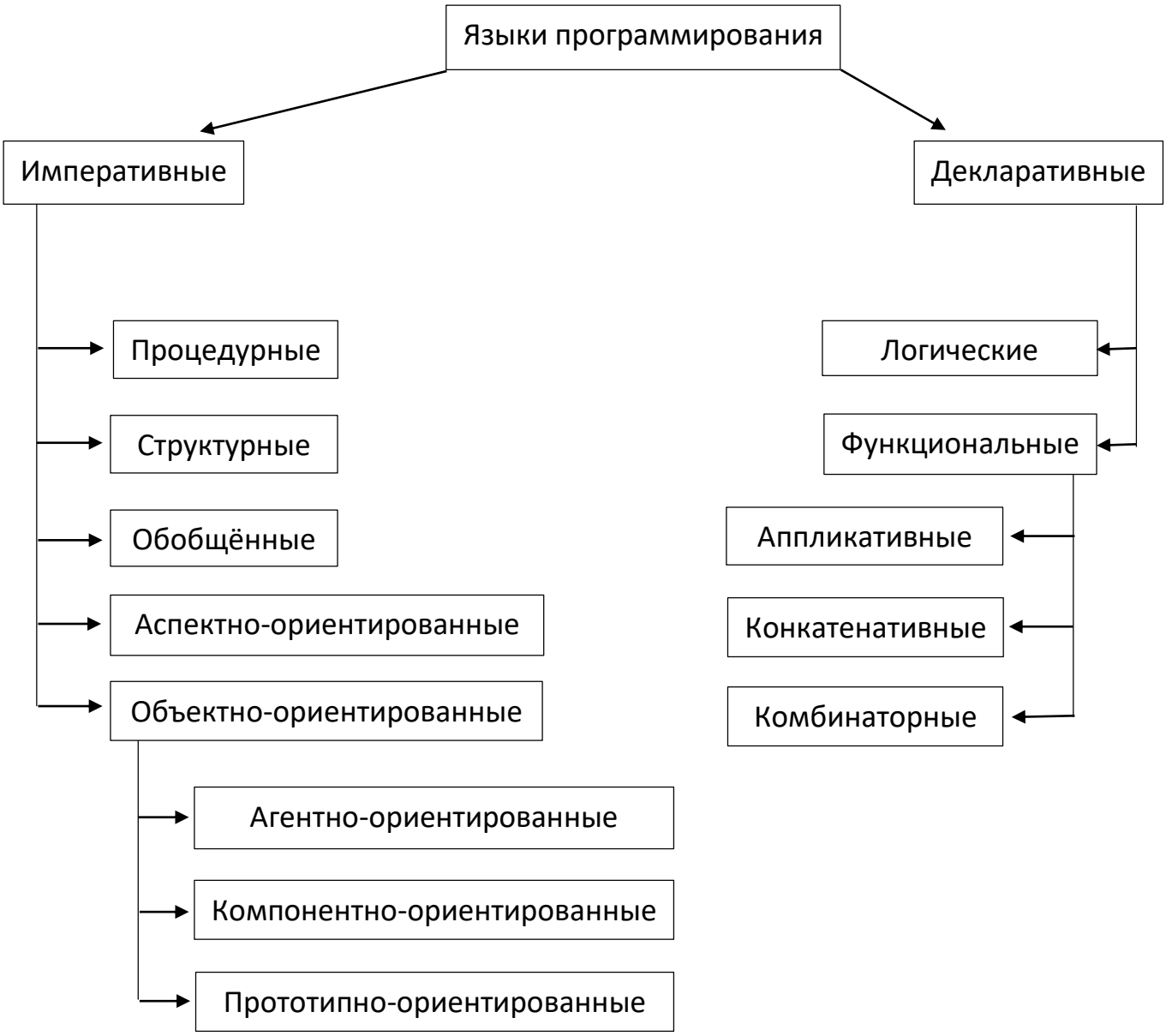
Функциональное программирование

В функциональном программировании процесс вычисления понимается как вычисление функций в их математическом понимании (правило, ставящее

в соответствие элементам одного множества элементы другого множества). Разновидностями функционального программирования являются аппликативное программирование (применение одного объекта к другому), конкатенативное программирование (склеивание фрагментов кода выражает их композицию), комбинаторное программирование (вместо переменных используются комбинаторы и композиции). Аппликативное программирование и конкатенативное программирование, так же, как и императивное и декларативное, являются противоположностями друг другу.

Таким образом, классификация языков программирования по классам является наиболее сложной, так как содержит в себе наибольшее число элементов и ветвлений.

Схема классификации языков программирования по классам



Чистота языка

Язык программирования называется чистым, если все функции этого языка являются чистыми. Функция называется чистой, если она является детерминированной и не обладает побочными эффектами. Детерминированность функции — невозможность возвращения разных значений при одинаковых входных аргументах. Побочными эффектами у функции могут быть чтение и модификация значений глобальных переменных, осуществление операций ввода-вывода, реакция на исключительные операции, вызов обработчиков. Чистые языки обладают рядом преимуществ в сравнении с нечистыми: их проще отлаживать, в них проще обнаруживать ошибки, их проще переписывать. Также чистые языки обладают параллелизмом — возможностью выполнения нескольких вычислений одновременно и взаимодействию их друг с другом.

Поколения языков программирования

Немаловажную роль в классификации языков программирования играет то, к какому поколению принадлежит данный язык.

Существует пять основных поколений:

- Первое поколение — все низкоуровневые машинные языки и языки ассемблера.
- Второе поколение — высокоуровневые языки программирования, возникновение компиляции.
- Третье поколение — независимость от аппаратного обеспечения, возникновение интерпретации.
- Четвёртое поколение — улучшение скорости и безопасности языков, ориентация на специальные области программирования.
- Пятое поколение — появление систем создания программ, позволяющих создавать программы с помощью визуального интерфейса.

Классификация некоторых языков программирования

C

- Высокоуровневый
- Компилируемый
- Статически слабый
- Императивный процедурный
- Третье поколение

C++

- Высокоуровневый
- Компилируемый
- Статически слабый
- Объектно-ориентированный, процедурный
- Третье поколение

C#

- Высокоуровневый
- Компилируемый в байт-код
- Статический
- Объектно-ориентированный, событийный, процедурный
- Четвёртое поколение

Python

- Высокоуровневый
- Интерпретируемый
- Динамический
- Объектно-ориентированный, структурный, обобщённый
- Четвёртое поколение

Java

- Высокоуровневый
- Компилируемый в байт-код
- Статический
- Объектно-ориентированный
- Четвёртое поколение

JavaScript

- Высокоуровневый
- Интерпретируемый
- Динамический
- Объектно-ориентированный, обобщённый, событийно-ориентированный
- Четвёртое поколение

PHP

- Высокоуровневый
- Интерпретируемый
- Динамический
- Объектно-ориентированный, обобщённый
- Четвёртое поколение

Ruby

- Высокоуровневый
- Интерпретируемый
- Динамический
- Объектно-ориентированный
- Четвёртое поколение

Lisp

- Высокоуровневый
- Компилируемый
- Динамический сильный
- Процедурный, объектно-ориентированный
- Второе поколение

Fortran

- Высокоуровневый
- Компилируемый
- Статический
- Процедурный
- Второе поколение

Pascal

- Высокоуровневый
- Компилируемый
- Статический сильный
- Императивный структурированный
- Третье поколение

Особые категории языков

Учебные языки

Потребность в языках программирования, предназначенных специально для обучения, возникла практически одновременно с появлением самих языков программирования. Основные требования к таким языкам — простота, ясность, понятность. Изначально такие языки, как BASIC и Pascal создавались специально для этой цели. Одним из самых популярных языков для обучения является Logo, который используется для обучения основам программирования детей.

Эзотерические языки

Обычно бесполезные на практике, эзотерические языки разрабатываются либо для исследования границ возможностей разработки языков программирования, либо, чаще всего, для развлечения.

Примеры эзотерических языков:

- Brainfuck — наиболее известный эзотерический язык, создавался с опорой на максимальный минимализм, имеет всего восемь команд, при этом обладает Тьюринг-полнотой.

  Программа «Hello World» на языке Brainfuck:

  ```
  ++++++++++++++++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++++.++++++++++++++++
  ++++++++++++.+++++++..+++.-------------------
  ----------------------------------------------
  ---------------.++++++++++++++++++++++++++++++
  ++++++++++++++++++++++++++.++++++++++++++++++
  ++++++.+++.------.--------.-------------------
  ----------------------------------------------
  ----.-----------------------.
  ```

- Whitespace — эзотерический язык программирования, в котором используются только пробел, табуляция и перенос строки.

  Программа «Hello World» на языке Whitespace:

- Malbolge — этот язык создавался как максимально сложный для написания программ.

  Программа «Hello World» на языке Malbolge:

  (=<`:9876Z4321UT.-
  Q+*)M'&%$H"!~}|Bzy?=|{z]KwZY44Eq0/{mlk**hKs_dG5[m_BA{?-
  Y;;Vb'rR5431M}/.zHGwEDCBA@98\6543W10/.R,+O<

Визуальные языки

В визуальных языках программист взаимодействует не с кодом, а с графическими объектами. Этот раздел программирования получил широкое развитие в пятом поколении языков программирования. Также существуют визуальные надстройки для уже существующих языков программирования. Одной из таких можно считать C++ Builder от Embarcadero Technologies для языка C++.

Заключение

Таким образом, в современном мире существует огромное множество языков программирования, применяющихся для самых разнообразных задач. Мы убедились, что существует множество признаков, по которым можно отличать и классифицировать языки программирования. По этим признакам можно также отслеживать эволюцию языков программирования, которая продолжается и по сей день. Вполне возможно, что через несколько лет появятся новые разделы программирования и новые задачи, для решения которых будут созданы более совершенные языки программирования, как, например, был создан язык PHP для разработки веб-страниц. Тем самым программирование как раздел информатики является одной из самых быстроразвивающихся наук, что делает профессию программиста полезной и востребованной.

Источники

- Гавриков М. М., Иванченко А. Н., Гринченков Д. В. Теоретические основы разработки и реализации языков программирования
- Криницкий Н. А., Миронов Г. А., Фролов Г. Д. Программирование
- Братчиков И. Л. Синтаксис языков программирования
- Лавров С. С. Основные понятия и конструкции языков программирования
- Х. Абельсон, Дж. Дж. Сассман, Дж. Сассман. Структура и интерпретация компьютерных программ
- Т. Пратт. Языки программирования: разработка и реализация