高级语言的各种分类方式及其对应的分类

1. 根据翻译方式分类：

计算机不能直接的理解高级语言，只能直接理解机器语言，所以必须要把高级语言翻译成机器语言，计算机才能执行高级语言的编写的程序。翻译的方式有两种，一个是编译，一个是解释。两种方式只是翻译的时间不同。

（1）编译型语言：

* 需通过编译器（compiler）将源代码编译成机器码，之后才能执行的语言。一般需经过编译（compile）、链接（linker）这两个步骤。编译是把源代码编译成机器码，链接是把各个模块的机器码和依赖库串连起来生成可执行文件。编译和执行是分开的，但是不能跨平台。
* 优点：编译器一般会有预编译的过程对代码进行优化。因为编译只做一次，运行时不需要编译，所以编译型语言的程序执行效率高。可以脱离语言环境独立运行。
* 缺点：编译之后如果需要修改就需要整个模块重新编译。编译的时候根据对应的运行环境生成机器码，不同的操作系统之间移植就会有问题，需要根据运行的操作系统环境编译不同的可执行文件。
* 代表语言：C、C++、Pascal、Object-C、swift等。

（2）解释型语言

* 解释性语言的程序不需要编译，相比编译型语言省了道工序，解释性语言在运行程序的时候才逐行翻译。每一个语句都是执行的时候才能翻译。这样解释性语言每执行一次要翻译一次，效率表较低。
* 优点：有良好的平台兼容性，在任何环境中都可以运行，前提是安装了解释器（虚拟机）。灵活，修改代码的时候直接修改就可以，可以快速部署，不用停机维护。
* 缺点：每次运行的时候都要解释一遍，性能上不如编译型语言。
* 代表语言：JavaScript、Python、Erlang、PHP、Perl、Ruby

（3）混合型语言

混合型语言也叫半编译型语言。比如java、C#，C#在编译的时候不是直接编译成机器码而是中间码，.NET平台提供了中间语言运行库运行中间码，中间语言运行库类似于Java虚拟机。.net在编译成IL代码后，保存在dll中，首次运行时由JIT在编译成机器码缓存在内存中，下次直接执行。

（java解释器采用生成与系统无关的字节代码指令技术。也就是说，在任何不同的操作系统上，只要正确安装了java运行系统，就有了编写调试java程序的平台，在分布式应用中，java的这个特点使同一个java程序能在不同的系统上运行，从而提高了软件生产效率。可移植性是跨平台特性的一个延伸，即具有了跨平台性，就保证了可移植性。java程序、java类库、java编译器、java系统都具有可移植性。）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 语言类型 | 描述 | 优点 | 缺点 | 例子 |
| 编译型语言 | 程序执行之前，已经编译好了，比如exe文件，下次执行不用再一次翻译。直接使用编译的结果即可 | 程序执行效率高 | 依赖编译器，跨平台性差 | C、C++、Pascal、Object Pascal、Delphi等 |
| 解释型语言 | 程序在执行程序的时候才翻译，边执行边翻译。 | 跨平台性好 | 执行一次要翻译一次，效率较低。 | Java、JavaScript、VBScript、Perl、Python、Ruby、MATLAB等 |
| 混合型语言 | 半编译，不是直接编译成机器码而是中间码，中间码不能执行。 | 具有可移植性，提高了软件生产效率 | 导致了复杂的环境。 | java、C#等 |

1. 根据运行是是否能够改变代码结构：

（1）动态结构语言

是一类在运行时可以改变其结构的语言：例如新的函数、对象、甚至代码可以被引进，已有的函数可以被删除或是其他结构上的变化。通俗点说就是在运行时代码可以根据某些条件改变自身结构。

主要动态语言：Object-C、C#、JavaScript、PHP、Python、Erlang。

（2）静态结构语言

与动态语言相对应的，运行时结构不可变的语言就是静态语言。如Java、C、C++。

|  |  |  |
| --- | --- | --- |
| 语言类型 | 描述 | 例子 |
| 动态结构语言 | 在运行时可以改变其结构的语言 | Object-C、C#、JavaScript、PHP、Python、Erlang |
| 静态结构语言 | 在运行时不可改变其结构的语言 | ava、C、C++等 |

三、根据数据类型检查的时间问题：

（1）动态类型语言

动态语言是在运行时确定数据类型的语言。变量使用之前不需要类型声明，通常变量的类型是被赋值的那个值的类型。

主要语言：Python、Ruby、Erlang、JavaScript、swift、PHP、SQL、Perl。

（2）静态类型语言

静态语言是在编译时变量的数据类型即可确定的语言，多数静态类型语言要求在使用变量之前必须声明数据类型。

主要语言：C、C++、C#、Java、Object-C。

【注意】

通常情况下解释型语言是动态类型语言，编译型语言是静态类型语言。但也不全是，swift是编译型语言但是它也是动态类型语言。C#和Java是解释型语言也是静态类型语言。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 语言类型 | 描述 | 优点 | 缺点 | 例子 |
| 动态类型语言 | 在执行期间检查数据的类型的语言。用这类语言编程，不会给变量指定类型，而是在附值时得到数据类型。 | 方便阅读 | 不方便调试，不利于理解 | Python、SQL、JavaScript、PHP、ASP、Ruby、Perl、ABAP、Unix Shell |
| 静态类型语言 | 在执行前编译时检查类型。在写代码时。没声明一个变量必须指定类型。 | IDE感知能力强，即编译器对代码检错能力强，便于调试 | 不方便阅读，代码冗长 | C++、Java、Delphi、C#等 |

四、数据类型能否被忽略：

（1）强类型语言：

一旦一个变量被指定了某个数据类型，如果不经过强制类型转换，那么它就永远是这个数据类型。你不能把一个整形变量当成一个字符串来处理。

主要语言：Java、C#、Python、Object-C、Ruby

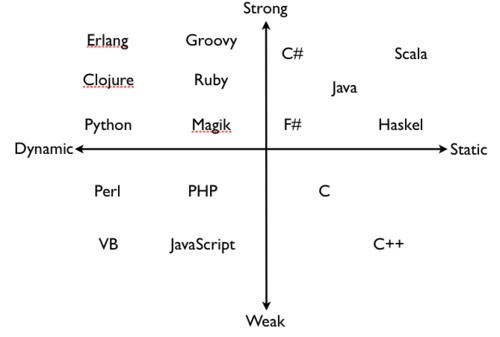
（2）弱类型语言：

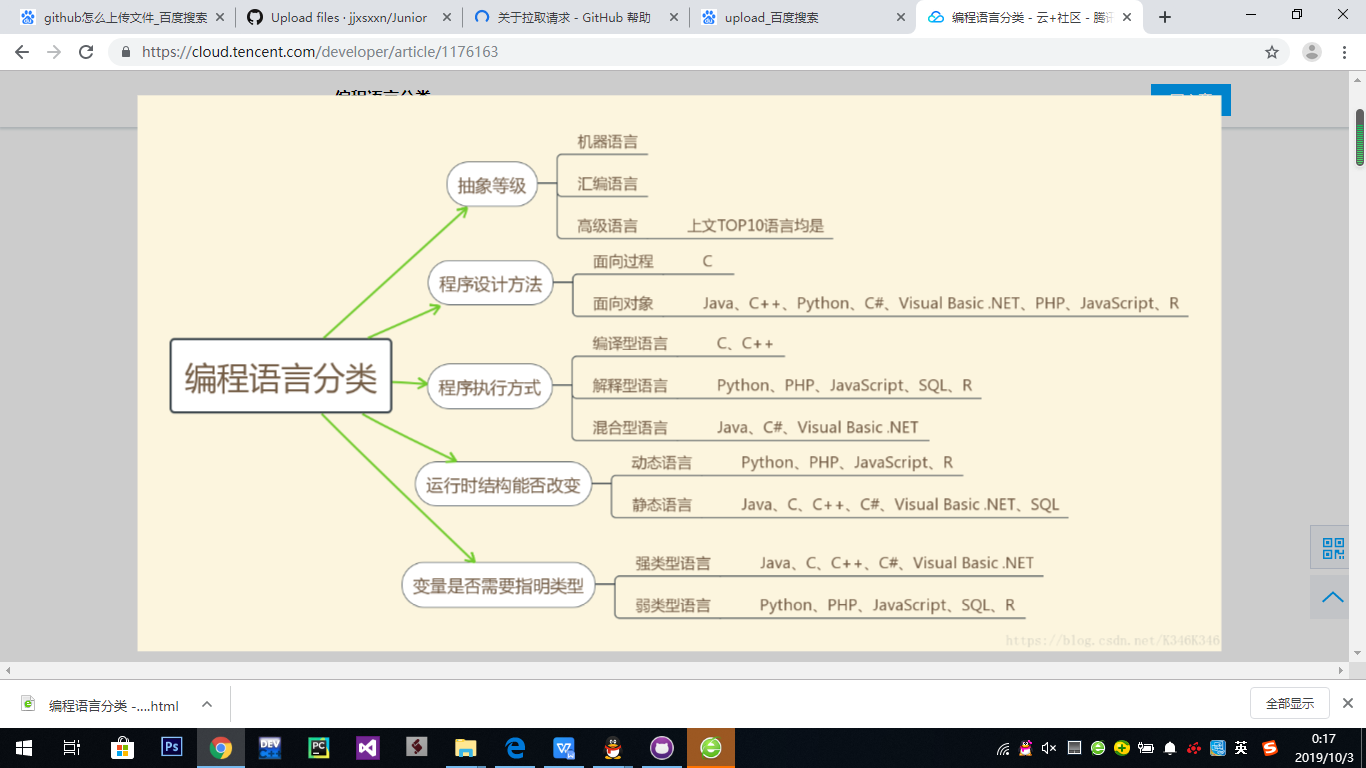
数据类型可以被忽略，一个变量可以赋不同数据类型的值。一旦给一个整型变量a赋一个字符串值，那么a就变成字符类型。

主要语言：JavaScript、PHP、C、C++（C和C++有争议，但是确实可以给一个字符变量赋整形值，可能初衷是强类型，形态上接近弱类型）

|  |  |  |
| --- | --- | --- |
| 语言类型 | 描述 | 例子 |
| 强类型语言 | 一旦一个变量被指定了某个数据类型，如果不经过强制类型转换，那么它就永远是这个数据类型。 | Java、C#、Python、Object-C、Ruby |
| 弱类型语言 | 数据类型可以被忽略，一个变量可以赋不同数据类型的值。 | JavaScript、PHP、C、C++等 |

语言类型分布图：





【可拓展：】

高级语言种类繁多， 还可以从应用特点和对客观系统的描述两个方面对其进一步分类。

|  |  |
| --- | --- |
| 应用角度 | 基础语言 |
| 结构化语言 |
| 专用语言 |
| 客观系统的描述 | 面向过程语言 |
| 面向对象语言 |

1、 从应用角度分类  
从应用角度来看，高级语言可以分为基础语言、[结构化语言](https://www.baidu.com/s?wd=%E7%BB%93%E6%9E%84%E5%8C%96%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)和专用语言。  
(1) 基础语言  
基础语言也称通用语言。它历史悠久，流传很广，有大量的已开发的软件库，拥有众多的用户， 为人们所熟悉和接受。 属于这类语言的有 FORTRAN、COBOL、BASIC、ALGOL等。

1. [结构化语言](https://www.baidu.com/s?wd=%E7%BB%93%E6%9E%84%E5%8C%96%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)  
   结构化语言直接支持结构化的控制结构，具有很强的过程结构和数据结构能力。PASCAL、C、Ada 语言就是它们的突出代表。  
   (3) 专用语言  
   是为某种特殊应用而专门设计的语言，通常具有特殊的语法形式。一般来说，这种语言的应用范围狭窄，移植性和可维护性不如结构化程序设计语言。应用比较广泛的有 APL 语言、Forth 语言、LISP 语言。  
   2、从客观系统的描述分类  
   从描述客观系统来看，程序设计语言可以分为面向过程语言和面向对象语言。  
   (1) 面向过程语言  
   以“数据结构+算法”程序设计范式构成的程序设计语言，称为面向过程语言。前面介绍的程序设计语言大多为面向过程语言。  
   (2) 面向对象语言  
   以“对象+消息”程序设计范式构成的程序设计语言，称为面向对象语言。目前比较流行的面向对象语言有 Delphi、Visual Basic、Java、C++等。

参考资料：

<https://zhidao.baidu.com/question/574568846.html>

<https://cloud.tencent.com/developer/article/1176163>

<https://blog.csdn.net/wordsin/article/details/80544602>

<https://blog.csdn.net/faterman/article/details/24733927>

<https://blog.csdn.net/YANG_Gang2017/article/details/72571635>

\*<https://blog.csdn.net/coding_dong/article/details/80920571>