张金源/76066001-162511 班

通过查资料，调研，列举 5 个目前常用的高级程序设计语言，简述其特点，并指出是编译型还是解释型，混合型，简述之

Java

Java 是一门面向对象编程语言，不仅吸收了 C++语言的各种优点，还摒弃了 C++里难以理解的多继承、指针等概念，因此 Java 语言具有功能强大和简单易用两个特征。Java 语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

Java 的特点：

1、**简单性，**Java 看起来设计得很像 C++，但是为了使语言小和容易熟悉，设计者们把 C++语言中许多可用的特征去掉了，这些特征是一般程序员很少使用的。

2、**面向对象，**Java 是一个面向对象的语言。对程序员来说，这意味着要注意应中的数据和操纵数据的方*（method）*，而不是严格地用过程来思考。在一个面向对象的系统中，类*（class）*是数据和操作数据的方法的集合。数据和方法一起描述对象*（object）*的状态和行。

3、**分布性，**Java 设计成支持在网络上应用，它是分布式语言。Java 既支持各种层次的网络连接，又以 Socket 类支持可靠的流*（stream）*网络连接，所以用户可以产生分布式的客户机和服务器。

4、**稳健性，**Java 原来是用作编写消费类家用电子产品软件的语言，所以它是被设计成写高可靠和稳健软件的。Java 消除了某些编程错误，使得用它写可靠软件相当容易。

5、**安全性** Java 的存储分配模型是它防御恶意代码的主要方法之一。Java 没有指针，所以程序员不能得到隐蔽起来的内幕和伪造指针去指向存储器。更重要的是，Java 编译程序不处理存储安排决策，所以程序员不能通过查看声明去猜测类的实际存储安排。

6、**可移植性，**Java 使得语言声明不依赖于实现的方面。Java 环境本身对新的硬件平台和操作系统是可移植的。Java 编译程序也用 Java 编写，而 Java 运行系统用 ANSIC 语言编写。

7、**高性能，**Java 是一种先编译后解释的语言，所以它不如全编译性语言快。但是有些情况下性能是很要紧的，为了支持这些情况，Java 设计者制作了“及时”编译程序，它能在运行时把 Java 字节码翻译成特定 CPU*（中央处理器）*的机器代码，也就是实现全编译了。

8、**多线索性，**Java 是多线索语言，它提供支持多线索的执行*（也称为轻便过程）*，能处理不同任务，使具有线索的程序设计很容易。

9、**动态性，**Java 语言设计成适应于变化的环境，它是一个动态的语言。例如，Java 中的类是根据需要载入的，甚至有些是通过网络获取的。

Java 语言是**解释性，**Java 编译程序生成字节码*（byte-code）*，Java 字节码提供对体系结构中性的目标文件格式，代码设计成可有效地传送程序到多个平台。Java 程序可以在任何实现了 Java 解释程序和运行系统*（run-time system）*的系统上运行。在一个解释性的环境中，程序开发的标准“链接”阶段大大消失了。

C 语言

C 语言是一门通用计算机编程语言，广泛应用于底层开发。C 语言的设计目标是提供一种能以简易的方式编译、处理低级存储器、产生少量的机器码以及不需要任何运行环境支持便能运行的编程语言。

C 语言特点：

1、**高级语言**：它是把高级语言的基本结构和语句与低级语言的实用性结合起来的工作单元。
2、**结构式语言**：结构式语言的显著特点是代码及数据的分隔化，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护以及调试。C 语言是以函数形式提供给用户的，这些函数可方便的调用，并具有多种循环、条件语句控制程序流向，从而使程序完全结构化。
3、**代码级别的跨平台**：由于标准的存在，使得几乎同样的 C 代码可用于多种操作系统，如 Windows、DOS、UNIX 等等；也适用于多种机型。C 语言对编写需要进行硬件操作的场合，优于其它高级语言。
4、**使用指针**：可以直接进行靠近硬件的操作，但是 C 的指针操作不做保护，也给它带来了很多不安全的因素。

C 语言是属于**编译性**，因为需要把源代码编译成机器语言的文件才可以执行，如 exe 格式的文件，以后要再运行时，直接使用编译结果即可，如直接运行 exe 文件。

C++

C++是 C 语言的继承，它既可以进行 C 语言的过程化程序设计，又可以进行以抽象数据类型为特点的基于对象的程序设计，还可以进行以继承和多态为特点的面向对象的程序设计。

C++的特点：

1、支持数据封装和数据隐藏在 C++中，类是支持数据封装的工具，对象则是数据封装的实现。C++通过建立用户定义类支持数据封装和数据隐藏。
2、支持继承和重用在 C++现有类的基础上可以声明新类型，这就是继承和重用的思想。通过继承和重用可以更有效地组织程序结构，明确类间关系，并且充分利用已有的类来完成更复杂、深入的开发。
3、支持多态性采用多态性为每个类指定表现行为。多态性形成由父类和它们的子类组成的一个树型结构。在这个树中的每个子类可以接收一个或多个具有相同名字的消息

C++是属于**编译性**，因为把源代码编译成机器语言的文件，如 exe 格式的文件，以后要再运行时，直接使用编译结果即可，如直接运行 exe 文件。

Python

Python 是一个有条理的和强大的面向对象的程序设计语言，它类似于 Perl，Ruby，Scheme，或 Java。

Python 的特点：

1、**简单**：Python 是一种代表简单主义思想的语言。阅读一个良好的 Python 程序就感觉像是在读英语一样。
2、**易学**：Python 极其容易上手，因为 Python 有极其简单的说明文档
3、**速度快：**Python 的底层是用 C 语言写的，很多标准库和第三方库也都是用 C 写的，运行速度非常快。
4、**可扩展性**：如果需要一段关键代码运行得更快或者希望某些算法不公开，可以部分程序用 C 或 C++编写，然后在 Python 程序中使用它们。
5、**可嵌入性**：可以把 Python 嵌入 C/C++程序，从而向程序用户提供脚本功能。
6、**丰富的库**：Python 标准库确实很庞大。它可以帮助处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-

RPC、HTML、WAV 文件、密码系统、GUI（图形用户界面）、Tk 和其他与系统有关的操作。

7、**可移植性**：由于它的开源本质，Python 已经被移植在许多平台上（经过改动使它能够工作在不同平台上）。

Python 是属于**解释性**，Python 语言写的程序不需要编译成二进制代码。可以直接从源代码运行 程序。在计算机内部，Python 解释器把源代码转换成称为字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行。

C#

C#是微软公司发布的一种面向对象的、运行于.NET Framework 之上的高级程序设计语言。并定于在微软职业开发者论坛(PDC)上登台亮相。C#是微软公司研究员 Anders Hejlsberg 的最新成果。C#看起来与 Java 有着惊人的相似。

C#的特点：

1、**简单**，C#中指针已经消失，不安全的操作比方说直接内存操作不被允许了，C#中"::"或"->"操作符是没用的.因为它是基于.NET 平台的,它继承了自动内存管理和垃圾回收的特点.

2、**面向对象，**C#支持数据封装,继承,多态和对象界面(即 java 中的 interface 关键字).

3、**类型安全** C#中我们不能进行不安全的类型转换象将 double 转换成 boolean，值类型(常量类型)被初始化为零值而引用类型(对象和类被编译器自动初始化为零值，数组类型下标从零开始而且进行越界检查，类型溢出将被检查.

C#是属于**解释性**，C#写的程序不需要编译成二进制代码。可以直接从源代码运行程序，二 C#运行于.NET Framework。

张金源/76066001-162511 班

介绍一个你熟悉的编译系统，描述其外部特征（功能、性能等）

GCC 6 的特性

GCC(GNU Compiler Collection) 是 GNU(GNU's Not Unix) 计划提供的编译器家族，它能够支持 C, C++, Objective-C, Fortran, Java 和 Ada 等等程序设计语言前端，同时能够运行在 x86, x86-64, IA-64, PowerPC, SPARC 和 Alpha 等等几乎目前所有的硬件平台上。每一个版本的 gcc 都会带来各种新的特性，这些新特性让使用 GCC 开发变得更加容易。在这我来介绍 GCC 6 的特征。

GCC 6 的外部特征：

1. OpenMP 4.5：OpenMP API 提供了一组简单、但高度灵活的接口来开发并行 C、C++和 Fortran 程序。2015 年 11 月发布了 OpenMP 4.5 规范，该规范在现有的 OpenMP4.0 基础上有了很多改进。GCC 6 将为 C 和 C++提供 OpenMP 4.5 的支持，对 Fortran 提供 OpenMP 4 支持以及对 offloading 一定程度的支持，尤其对于 Intel MIC 处理器。
2. 段寄存器支持： x86/x86_64 是一个分段的内存架构，但是 GCC 很大程度上忽略了 Intel 架构这方面的特性，而是依靠隐含的段寄存器。底层代码比如 linux 内核或者 glibc 通常必须意识到分段体系结构，传统上采用汇编语句进行显式的段寄存器操作。从 GCC 6 起，变量在声明时就可以指定到特殊的段，使用这些内存中变量时显式的段寄存器就会被操作。
3. 增强位置和其他诊断工作：GCC 6 的诊断机制引入了“增强位置”特性。这个特性允许 CC 跟踪一定范围的诊断信息，而不仅仅是单个点。举个例子，一个复杂表达式中的有疑问的子表达式可以被识别出来并向开发人员强调。诊断现在也包括“修复”性提示，建议开发者如何修改错误的代码。这个特性将会在 David Malcolm 的另一篇博客中详细介绍。
4. 目标克隆：GCC6 增加了一个新的“目标克隆”属性，用于指明特定函数需为不同的 ISA 变量进行多次编译，并创建程序分发器来检测这些正在使用中的 ISA 并调用适合的克隆。这使得对 ISA 的不同版本提供"特殊运行"程序支持更为容易。
5. 扩展存数规则：GCC6 新增一个新的 pragma“scalar_storage_order",用于指定在一个结构内字段的字节顺序。如果目标的字节顺序不同于正在被访问的对象的字节顺序，GCC6 会自动生成代码来按需 byte 交换这些对象。
6. 卸载/HSA:GCC5 最初已经支持通过 OpenMP 来卸载到 MIC，通过 OpenACC 卸载到 Nvidia GPUS. GCC6 改进了这些性能并新增了通过 HSA 加速器来卸载。
7. GCC 6 现在的默认值是 C++ 14. GCC 6 现在包括 C++ Concepts。GCC 6 也支持 C++17 的实验功能。
8. 改进诊断功能，包括改进位置、位置范围、标识符拼写错误改进建议、修复提示和新的警告。

9. 改进优化器。这影响了程序内优化、程序间优化、链接时间优化和各种目标后台。