编译原理 语义分析 语法分析 词法分析

关注者 87 被浏览 62,899

编译原理词法分析，语法分析，语义分析能检测出什么错误?

这三个分别能检测出代码的什么错误，请逐个说出来谢谢

关注问题 写回答 邀请回答 好问题 1 添加评论 分享 ···

查看全部 6 个回答

原点技术
公众号【原点技术】分享真正有用的东西!

+ 关注

8 人赞同了该回答 〉

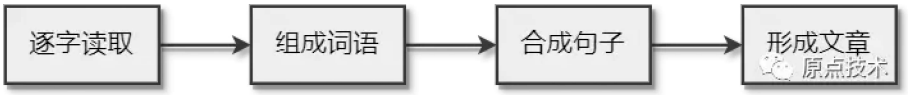
要回答这个问题，其实只要搞清楚词法、语法、语义的概念就行了!

先从我们的大脑是如何阅读一篇文章说起吧!

我们的大脑是怎么“编译”文章的?

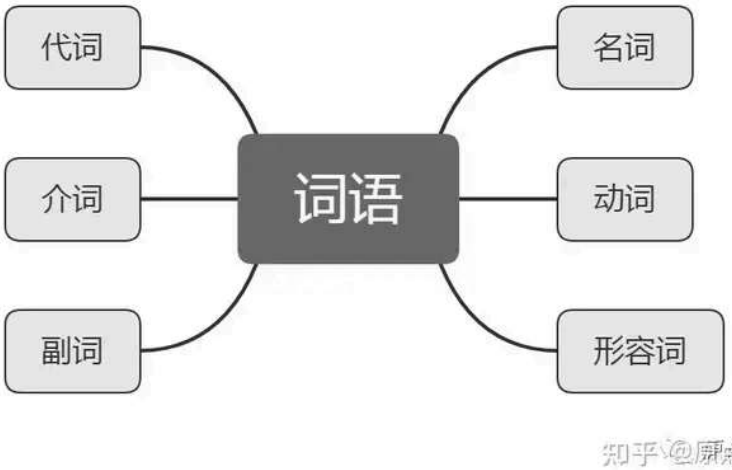
我们知道，中文的最小语言单位是字，字组成词，词组成句，句最终形成文。其实这也正是大脑对文章的一个“编译”过程。

大家不妨想一下，我们平时阅读一篇文章时，我们是怎么做的呢？是不是下面这样的:



逐字读取，组成词语

首先，逐个读取每个文字，在大脑中把这些单独的字组成一个个离散的词语。这个过程中，我们会在潜意识中给组成的词进行分类，比如动词、名词、形容词和副词等，并且会根据已有的知识，去判断这个词是不是在现实中存在并且有实际意义。



这个过程，就是**词法分析+**。

▲ 赞同 8 ▼ 添加评论 分享 收藏 喜欢 ···

关于作者

原点技术
公众号【原点技术】分享真...

回答 11 文章 14 关注者 744

关注他 发私信

被收藏 6 次

我的收藏 0 人关注
杜青 创建

计算机基础 0 人关注
李巨侠 创建

编译原理 0 人关注
囧冏 创建

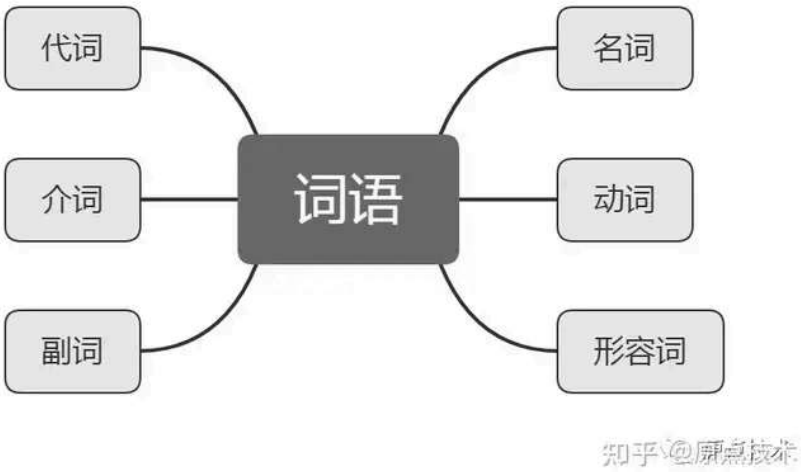
相关问题

编译原理语义分析阶段，现在已经做到了能生成语法树，基于语法树怎样实现语义分析和中间代码生成? 3 个回答

有什么“计算机语言的高级语义指导编译器写出低级优化很难优化好的地方”的例子? 2 个回答

根据句式，构造句子

然后，根据我们已经掌握的句式结构，把这些离散的词组合成一个个句子，如陈述句、疑问句、感叹句、祈使句、复合句等等。



这些句式都有固定的句型结构，如：

- 一个完整的句子一定要有主语、谓语、宾语。
- “虽然”后面一定跟着“但是”。
- “因为”后面一定跟着“所以”。

这个过程，就是**语法分析⁺**，而这些固定的句型结构，就是语法规则。

分析语句，理解含义

为了理解整篇文章所要表达的中心思想，我们需要结合上下文，理解句子所要表达的含义，并判断其是否符合常理。

比如，“人类是从石头里蹦出来的”这个陈述句，从语法上看没有任何问题，但是明显不符合常理（科学严谨地讲，猴哥不能算是真正的人类）。

这个过程，就是**语义分析⁺**。

是不是很简单？

为了加深对词法、语法、语义的理解，我们看一个最简单的例子。

从"山羊吃草"看词法、语法和语义

为简单起见，我们对一个合法的中文语句做以下规则约束：

- 一个完整的句子必须要有主语、谓语、宾语。
- 主语和宾语必须为名词，谓语必须是动词。
- 每个词语必须是现实存在且有意义的。
- 每个句子所表达的含义必须合乎常理。

现在，我们看下面四个句子：

- 山羊吃草。
- 山牛吃草。
- 山羊草。
- 草吃山羊。

思考一下，哪一句是合法的呢？不合法的句子分别有什么错误呢？

我们来分析一下。

山羊吃草

主语是“山羊”，谓语是“吃”，宾语是“草”。很显然，三条规则都满足，并且符合常识。因此，这句话是合法的。

山牛吃草

“山牛”这个词现实中不存在，是无意义的，因此不符合第三条规则。属于**词法错误**。

山羊草

缺少谓语，不能构成一个完整的句子，不符合第一条规则。属于**语法错误**。

草吃山羊

乍一看，这句话主、谓、宾都清晰明确，且都是现实存在并有实际意义的词语，所以，它是符合语法的。但这就意味着这句话是正确的吗？显然不是。现实世界中，草怎么可能吃山羊呢？这显然不符合常理，因此，这句话存在**语义错误**。

现在我们了解了词法、语法和语义的概念。那么，究竟什么是编译器*呢？

编译器是什么

人与人之间通过文字语言或者肢体语言进行交流，海豚之间通过发出不同频率的“脉冲声”进行交流，计算机之间则通过由0和1组成的电信号进行交流。

那么，人与计算机之间如何进行交流呢？人类无法直观地理解电信号所携带的信息，同样，计算机也无法理解人类使用的自然语言所表达的含义。

因此，要想让计算机能够“听懂”人话，帮人办事儿，就必须要能够把人类的语言，翻译成计算机能够识别的二进制机器指令。而这，就是编译器所要解决的最根本的问题。

简单来讲，**编译器就是把一种语言（通常是某种高度抽象的高级语言）转化为另一种语言（通常是某种低级语言）的计算机程序。**

下面以C语言为例，再次理解下词法、语法和语义分别对应哪些语言规则。

C语言词法、语法、语义举例

由于篇幅所限，不可能覆盖C语言所有的规则，这里仅举几个典型的例子，加深理解。

词法规则

- • 标识符由26个英文字母、数字和下划线组成，且不能以数字开头。
- • 单行注释必须以"//"标识，多行注释必须以"/*和*/"标识。
- • 字符串必须以双引号标识。

语法规则

- • 每个语句必须以分号结束。
- • 变量声明必须由类型开始，后面跟变量名。
- • "else"关键字前面必须有"if"关键字进行配对。
- • "case"语句必须和"switch"进行配对使用。

语义规则

- • 任何类型的变量都不能除以零。
- • 两个同类型的指针变量不能相加，但可以相减。
- • 函数调用时，实参个数必须和形参个数相等，且类型相互兼容。

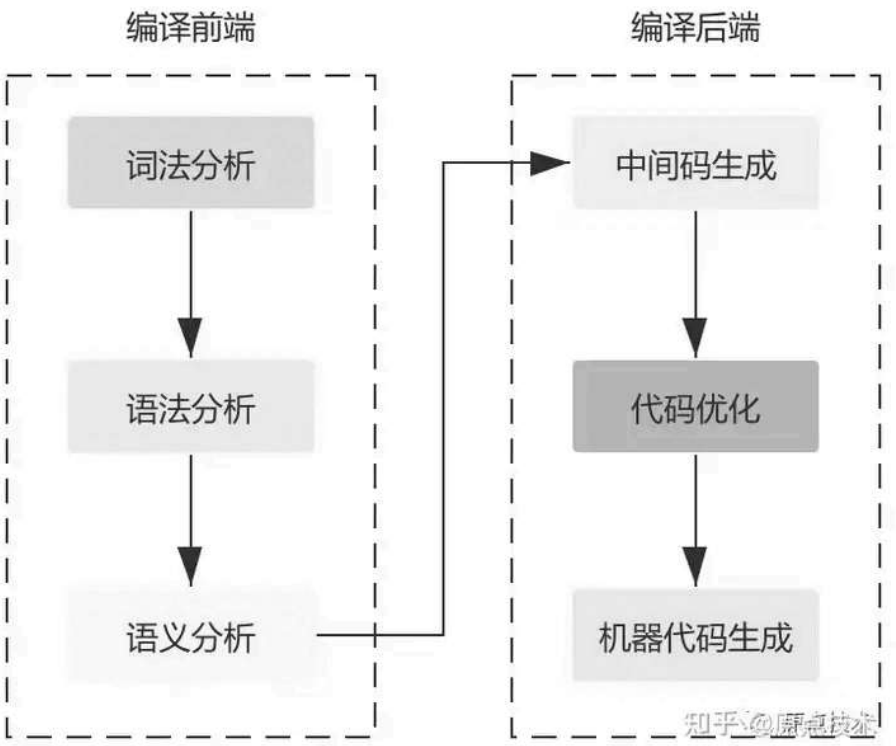
了解了词法、语法、语义的基本概念之后，我们来看看程序构建过程中，编译器是如何处理的。

编译器的实现原理

构建程序时，编译器对程序源代码的处理过程，和我们读书时大脑对文章的处理过程，是类似的。

典型的编译器都是分阶段实现的，每个阶段的输出都作为下一阶段的输入，把程序从一种表示形式转换成另外一种表示形式，最终转化成目标程序。

这几个阶段如下图所示：



词法分析 - Lexical Analysis

简单来说，词法分析阶的主要任务是：

- • 识别出源码文件中的词法单元，即token。
- • 对识别出来的词法单元进行分类，如关键字、常量、标识符、操作符等。
- • 根据词法规则，识别出非法的词法单元，并进行错误处理(一般是打印错误信息)。

词法分析阶段，对程序源码文件从左到右，逐字符进行扫描，根据词法规则，把这些字符组合成一系列单词序列，即"token"。

把字符组合成token之后，还要对其进行分类。以C语言为例，我们可以得到的token类型有关键字(如int, if, switch, for, while等)，操作符(如 +、-、*、/等)、标识符，字符串、常量等。

如下C语言代码片段：

```
int length = 2, width = 1;
```

经过词法分析后，我们可以得到如下token序列：

token	类型
int	关键字
length	标识符
=	赋值操作符
2	常量
,	逗号分隔符
width	标识符
=	赋值操作符
1	常量
;	分号分隔符

词法分析阶段还有一个很重要的工作，就是要根据词法规则，检测出非法的字符序列，并进行错误处理。

如C语言规范要求所有的变量名称必须以26个字母和'_'组成，且不能以数字开头。因此如下面代码片段：

```
int 4abc = 1;
```

在词法分析过程中，以数字开头的变量名"4abc"会被视为非法标识符，此时编译器会抛出词法错误的信息。

语法分析

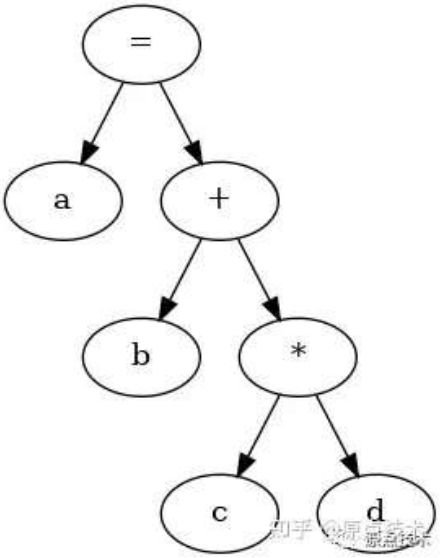
语法分析阶段的主要任务是根据语法规则，确定程序的语法结构，把词法分析阶段得到的token序列组合成表达式(expression)、语句(statement), 函数(function)等各种语法单元。

语法分析阶段的输出是一棵抽象语法树+(Abstract Syntax Tree)。

如下代码片段：

```
a = b + c * d;
```

经过语法分析后，会得到这样一棵抽象语法树：



语义分析

语义分析是在语法分析阶段构造出来的抽象语法树的基础上，对程序进行静态语义的分析，确保程序不仅符合语法规范，还应该能够正常执行。

对于静态类型的编程语言来说，语义分析阶段最重要的工作之一，就是类型检查。例如，传递给函数的实参类型是否和函数声明中的形参类型相互兼容，做减法运算的两个指针类型是否相同等。

而对于动态类型语言，如Python，则是把类型检查推迟到了运行时，这也是动态类型语言一般比静态类型语言运行效率低的原因之一。

中间代码+生成

经典古籍《离骚》，相信大家都印象深刻吧，通篇都是晦涩难懂的文言文，要想把它直接翻译成英文，是非常困难的一件事情。

但是，如果把晦涩难懂的文言文，先翻译成通俗易懂的现代白话文，然后再翻译成英文，这样操作起来，是不是难度就小的多了呢？

同样，高级语言一般都是高度抽象的，相比机器语言或目标语言来说，其语法结构和数据类型都更为复杂。因此，很难从高级语言直接生成目标语言，于是便有了中间代码，或者叫**中间表示(Intermediate Representation)**。

中间代码是一种在语法结构和数据类型更接近目标语言的表现形式。它处理起来要比源语言更加简单，便于进行逻辑优化，更加容易生成简洁、高效的目标代码。

不同的编译器，通常都有自己专用的中间代码形式，且一般互相之间并不兼容。如GCC，Clang/LLVM都有自己专有的中间代码。

三地址码+是一种比较简单也很常见的一种中间表示，一般格式如下：

```
x = y op z
```

之所以叫三地址码，是因为每条语句最多包含三个操作码，也就是：

结果变量 x 操作数 y和z 操作码 op

仍以下面的代码段为例：

```
a = b + c * d;
```

转换为三地址码为：

```
t1 = c * d;
a = b + t1;
```

其中，t1为引入的临时变量。

我们再看一个稍微复杂点的例子：

```
int test() {
    int i = 0, k = 3;

    for(i = 0; i <= 10; i ++) {
        k = k + 2 * i;
        if(k >= 20) {
            break;
        }
    }
    return k;
}
```

转化成三地址码：

```
test:
    i = 0;
    k = 3;
Label_1:
    if i > 10 goto Label_2;
    t1 = 2 * i;
    k = k + t1;
    if k >= 20 goto Label_2;
    i = i + 1;
    goto Label_1;
Label_2:
    return k;
```

代码优化✦

从抽象语法树到中间代码，往往是一个相对简单的操作，它更多考虑的是如何对语句进行等价转换，代码的效率不是主要考虑因素。这里的效率，包含时间效率和空间效率。

为了最终生成的程序能够更高效的执行和存储，现代编译器往往会花很大力气去对代码进行优化，而且会不止一次地对代码进行扫描和优化。

对于任何一个现代编译器项目，代码优化是整个项目中最关键也最复杂的核心功能。毫不夸张地说，一个编译器项目的成功与否，很大程度上取决于其代码优化模块。

常用代码优化方法有很多，如：

- • 死代码删除
- • 常量折叠
- • 循环展开
- • 跳转优化
- • 公共子表达式消除

需要强调一点，除了基于中间代码的优化之外，在生成目标代码时，往往也会进行优化。

目标代码生成

目标代码生成是编译的最后一个阶段（通常还会伴随着代码优化），它以优化后的中间代码作为输入，产生等价的目标代码作为输出。

对于如C语言这样的编译型语言，代码生成时，通常主要考虑的是：

- • 语义等价。这是最基本的要求，因为只有与原始代码语义上等价的目标代码，才能正确运行，得到期望的结果。
- • 高效。生成的代码要尽可能在时间和空间上保持高效，因此代码生成的过程中往往也伴随着代码优化，要充分考虑到对硬件资源的最优化使用，如寄存器、cache、流水线等。

如对上例中的test()的三地址码生成的汇编代码如下：

```
test:
    pushq %rbp
    movq  %rsp, %rbp
    movl  $0, -8(%rbp)   // i = 0;
    movl  $3, -4(%rbp)   // k = 3;

.Label_1:
    cmpl  $10, -8(%rbp)  // if i > 10 goto Label_2;
    jg .Label_2

    movl -8(%rbp), %eax  // t1 = 2 * i;
    addl  %eax, %eax
    addl  %eax, -4(%rbp)  // k = k + t1;
```

```
    cmpl  $20, -4(%rbp)   // if k >= 20 goto Label_2;
    jge .Label_2
    addl $1, -8(%rbp)    // i = i + 1;
    jmp .Label_1         // goto Label_1;

.Label_2:
    movl  -4(%rbp), %eax // return k;
    popq  %rbp
    ret
```

小结

作为程序员三大浪漫之一，编译原理人人向往，却又都敬而远之。我认为，很大一部分原因，是目前大部分介绍编译原理的书籍都聚焦于理论，缺乏实践，导致内容过于抽象，很容易让人萌生退意。

因此，本系列专题将重实践，轻理论。实践方面，从零开始，设计并开发一个简单的编程语言和编译器。理论方面，用最通俗易懂的语言，来介绍开发一个简单编译器所需的必要理论知识，力争让每个人都能看懂、掌握。

本文简要介绍编译器的一些重要概念和各个阶段完成的主要工作，让大家对编译器有一个整体的认识。后面文章会在实践过程中，对用到的理论知识做进一步详细的讲解。

别忘了点个赞！欢迎关注 @原点技术

发布于 2023-12-24 17:34 · IP 属地芬兰

更多回答

是莓莓呀

CMU / Facebook

了解它们具体做什么的，大概也就知道会报什么错了啊。

词法分析就是取出一个个词，然后给词归类、给个种别码什么的。所以遇到不认识的词或符号，一般就会报错。

语法分析就是根据语法规则识别出语法单位（赋值语句、条件语句之类），并检查语法单位在语法结构上的正确性。

语义分析是对语法单位进行静态的语义审查（动态的在运行时才可确定）。分析其含义，下一步就会用另一种接近目标语言或直接用目标语言去描述这个含义。此阶段要求语句的含义和使用规则正确。

比如嘛……我随便写个文法吧：

展开阅读全文 ⌄

匿名用户

这三个分别能检测出代码的什么错误，请逐个说出来谢谢

词法分析：

这 三个 分别 能 检测 出 代码 的 什么 错误，请 逐个 说 出来 谢谢

语法分析：

这 三个
分别
能
检测 出
代码 的
什么
错误
，
请

展开阅读全文 ⌄

▲ 赞同 15 ▼ 1 条评论 分享 收藏 喜欢 ···

查看全部 6 个回答