编程作业一

1 方案拟订项目概述

编程作业 I-IV 将指导您为 Cool 设计和构建编译器。每个作业将涵盖编译器的一个组件：词法分析、解析、语义分析和代码生成。每个分配最终将导致一个工作的编译器阶段，它可以与其他阶段接口。您将在 C++ 执行项目。

对于这项任务，您将使用名为 Flex 的词汇分析器生成器编写词汇分析器，也称为扫描器。您将以适当的输入格式描述 Cool 的令牌集，分析器生成器将生成用于识别 Cool 程序中令牌的实际代码。

项目所需所有工具的在线文件将在课程网站上提供。 这包括 FLEX 手册（在本作业中使用）、野牛文档以及 SPIM 模拟器手册。

你可以单独工作，也可以成对工作。

2 Flex 简介

Flex 允许您通过编写与用户定义的正则表达式匹配的规则并为每个匹配的模式执行指定的操作来实现词法分析器。 Flex 将您的规则文件(例如，“lexer.flex”)编译为 C 源代码，实现了识别规则文件中指定的正则表达式的有限自动机。 幸运的是，没有必要理解甚至查看实现规则的自动生成（而且通常非常混乱）文件。

flex 中的规则文件结构如下：

```
%{
声明
%}
定义
%%
规则
%%
用户子程序
```

声明和用户子程序部分是可选的，允许您在 C 中编写声明和帮助函数。定义部分也是可选的，但通常非常有用，因为定义允许您给正则表达式命名。 例如，定义

数字 [0-9]

允许您定义一个数字。 这里，DIGIT 是给正则表达式的名称，它匹配 0 到 9 之间的任何单个字符。 下表概述了 Flex 中可以指定的常见正则表达式：

字符“x”	
“x”	一个“x”，即使 x 是一个运算符。
x	一个“x”，即使 x 是一个运算符。
[xy]	字符 x 或 y.

[x-z] 字符 x，y 或 z.
["x] 任何角色，但 x.
. 任何角色，但 换行。
"在一行的开头。
当 Flex 处于启动条件 y 时，<y>xanx。
x$ 直线末尾的 x。
x? 一个可选的 x。
x* 0,1,2，...x 的实例。
x+ 1,2,3，...x 的实例。
x|y 一个 x 或 y。
(x) a x.
x/y ax，但只有在后面跟着 y。
{xx}从定义部分翻译 xx。
x{m，n}m 通过 x 的 n 次出现

词汇分析器最重要的部分是规则部分。 Flex 中的规则指定了一个操作，如果输入与规则开头的正则表达式或定义相匹配。 要执行的操作是通过编写规则的 C 源代码来指定的。 例如，假设一个数字在我们的语言中表示一个令牌(请注意，在 Cool 中并非如此)，规则：

{数字}{
cool_yylval.符号=inttable.add_string(Yytext)；
返回 DIGIT_TOKEN；}

记录全局变量中数字的值 cool_yylval 并返回适当的令牌代码。 (有关全局变量的更详细讨论，请参见第 5 节 cool_yylval 关于上述代码片段中使用的 inttable 的讨论请参见第 4.2 节。)

要记住的一个重要点是，如果当前输入(即函数调用 yylex（)的结果）匹配多个规则，Flex 选择匹配最多字符的规则。 例如，如果定义以下两个规则

[0-9]+{/行动 1}
[0-9a-z]+{/行动 2}

如果字符序列 2a 出现在被扫描的文件中，则将执行操作 2，因为第二个规则匹配的字符比第一个规则多。 如果多个规则匹配相同数量的字符，则选择文件中首先出现的规则。

在 Flex 中编写规则时，可能需要根据以前遇到的令牌执行不同的操作。 例如，在处理关闭注释令牌时，您可能感兴趣的是知道以前是否遇到了打开注释。 跟踪状态的一个明显方法是在声明部分声明全局变量，当遇到某些感兴趣的令牌时，这些变量被设置为 true。 Flex 还通过使用状态声明（如：）为实现类似的功能提供语法糖

开始交流

可以通过编写 BEGIN(COMMENT)将其设置为 true。 要执行一个操作，只有在前面遇到一个开头注释时，您才能使用语法：在 COMMENT 上断言您的规则

<评论>{
//你剩下的规则...

还有一个特殊的默认状态，称为初始状态，它是活动的，除非您显式地指示新状态的开始。 您可能会发现此语法对于此赋值的各个方面都很有用，例如错误报告。

3 文件和目录

要启动，请登录其中一个水稻机并运行以下命令

/AFS/ir.stanford.edu/class/cs143/bin/pa_fetch PA1<project_directory>

您指定的项目目录将在必要时创建，并将包含一些供您编辑的文件和一堆您不应该编辑的东西的符号链接。（事实上，如果您制作和修改这些文件的私有副本，您可能会发现无法完成作业。）请参阅 README 文件中的说明。 您需要修改的文件是：

- 很酷
  此文件包含用于 Cool 的词法描述的骨架。有注释表明您需要在哪里填写代码，但这不一定是一个完整的指南。 作业的一部分是让您确保您有一个正确和工作的 lexer。 除了所示的部分，欢迎您对我们的骨架进行修改。 实际上，您可以使用骨架描述构建扫描仪，但它没有多大作用。 您应该阅读 Flex 手册，以了解骨架描述的功能。 您希望编写的任何辅助例程都应该直接添加到适当的部分中的此文件中（参见文件中的注释）。
- test.cl
  此文件包含一些要扫描的示例输入。它不能锻炼所有的词汇规范，但它仍然是一个有趣的测试。它不是一个好的测试开始，也没有提供足够的测试您的扫描仪。 您的部分任务是提出良好的测试输入和测试策略。(不要掉以轻心——良好的测试输入是很难创建的，而忘记测试是评分过程中失分的最可能原因。)
  您应该使用您认为充分锻炼扫描仪的测试来修改此文件。我们的 test.cl 类似于真正的 Cool 程序，但您的测试不需要。 你可以尽可能少地保留我们的测试。 我们将对您的测试文件的覆盖范围进行评分-也就是说，您的文件在多大程度上练习了尽可能多的 lexer 部分。
- 自述文件
  此文件包含分配的详细说明以及一些有用的提示。您应该编辑此文件以包含您的 SUNETID（详见第 7 节）。

虽然这些文件不完整，但 lexer 确实编译并运行。 要构建 lexer，必须键入 make lexer。

4 扫描结果

在此赋值中，您将编写与定义 Cool 中有效令牌的适当正则表达式匹配的 Flex 规则，如 Cool 手册第 10 节和图 1 所述，并执行适当的操作，例如返回正确类型的令牌、在适当的情况下记录 lexeme 的值遇到错误时报告错误。 在开始此作业之前，请确保阅读 Cool 手册第 10 节和图 1；然后研究 Cool-parse.h 中定义的不同令牌。您的实现需要定义与定义 cool-parse.h 中定义的每个令牌的正则表达式匹配的 Flex 规则，并为每个匹配的令牌执行适当的操作。 例如，如果您在令牌 BOOLCONST 上匹配，则 Lexer 必须记录其值是真还是假；类似地，如果您在 TYPEID 令牌上匹配，则需要记录类型的名称。 请注意，并非每个令牌都需要存储额外的信息；例如，只返回令牌类型就足以满足一些令牌，如关键字。

您的扫描仪应该是健壮的-它应该适用于任何可以想象的输入。 例如，您必须处理错误，例如发生在

字符串或注释中间的 EOF，以及太长的字符串常量。这些只是可能发生的一些错误；其余的请参阅手册。

如果发生致命错误，您必须为优雅的终止做一些规定。 核心转储或未记录的异常是不可接受的。

4.1 错误处理

所有错误都应该传递给解析器。 你的 lexer 不应该打印任何东西。 错误通过返回一个名为 ERROR 的特殊错误令牌传递给解析器。 (注意，您应该忽略这个赋值的名为 error（小写）的标记；它在 PA2 中被解析器使用。) 报告和从词汇错误中恢复有几个要求：

- 当遇到一个无效字符（不能开始任何令牌）时，应该返回一个只包含该字符的字符串作为错误字符串。恢复以下字符。
- 在为多行字符串报告行号时，使用最后一行。 一般来说，行号应该是令牌结束的地方。 这使实现保持简单——否则您将不得不跟踪其他状态。
- 当字符串太长时，在错误令牌中的错误字符串中报告错误为“字符串常量太长”。 如果字符串包含无效字符（即空字符），则将其报告为“String 包含空字符”。 如果字符串包含一个未分隔的换行符，则报告为“未终止字符串常量”。 只报告要发生的第一个字符串常量错误，然后在字符串结束后恢复 lexing。字符串的末尾定义为任意一个
  1. 下一行的开头，如果未转义的换行发生在字符串中的任何点；或
  2. 结束后“否则。
- 当遇到 EOF 时，如果注释仍然打开，请使用“注释中的 EOF”消息报告此错误。不要仅仅因为终止符丢失而标记注释的内容。 同样，对于字符串，如果在闭引号之前遇到 EOF，则将此错误报告为“字符串常量中的 EOF”。
- 如果您在注释之外看到“*)”，则将此错误报告为“无与伦比的*)”，而不是将其标记为*和)。

从讲座中回想一下，编译器的这一阶段只捕获非常有限的一类错误。**不要检查本作业中没有错误的错误。**例如，在使用前不应检查变量是否已声明。 确保您完全理解编译器的 lexing 阶段所做的错误，并且在开始之前不检查。

4.2 字符串表

程序往往有许多相同的词汇出现。 例如，标识符通常在程序中引用不止一次（否则它不是很有用！）。为了节省空间和时间，常见的编译器实践是将词汇存储在字符串表中。我们提供了一个字符串表实现。详情见以下各节。

在决定如何处理基本类(Object、Int、Bool、String)、SELF_TYPE 和 Self 的特殊标识符时存在问题。然而，这个问题直到编译器的后期阶段才会出现——扫描器应该像对待其他标识符一样对待特殊标识符。

不要测试整数文字是否适合在 Cool 手册中指定的表示中-简单地创建一个符号，以整个文字的文本作为其内容，而不管它的长度。

4.3 串

扫描器应该将字符串常量中的转义字符转换为正确的值。 例如，如果程序员键入这八个字符：

0h00hh00

您的扫描仪将返回令牌 STRCONST，其语义值为这 5 个字符：

回回屈日回

其中|\n|表示换行符的文字 ASCII 字符。

按照 Cool 手册第 15 页的规范，必须返回包含文字空字符的字符串的错误。 然两个字符的序列.

00

允许，但应转换为一个字符(数字零，ASCII 值 48)

回.

4.4 其他说明

您的扫描仪应该维护变量 curr_lineno，该变量指示当前正在扫描源文本中的哪一行。 此功能将帮助解析器打印有用的错误消息。

你应该忽略令牌 LETSTMT。 它仅由解析器(PA2)使用。 最后，请注意，如果词汇规范不完整（有些输入没有匹配的正则表达式），那么 flex 生成的扫描器就会做不受欢迎的事情。 *确保您的规范是完整的。*

5 执行说明

- 扫描器上的每个调用都从输入返回下一个令牌和词汇。 函数 cooLyylex 返回的值是一个整数代码，表示句法类别(例如整数文字、分号、if 关键字等)。 所有令牌的代码都在 cool-parse.h 文件中定义。 第二个组件，语义值或词汇，被放置在全局联合 cooLyylval 中，这是 YYSTYPE 类型的。 类型 YYSTYPE 也在 cool-parse.h 中定义.. 单个字符符号（例如，和“，”）的标记仅由字符本身的整数(ASCII)值表示。 所有的单个字符标记都在 Cool 手册中的 Cool 语法中列出。
- 对于类标识符、对象标识符、整数和字符串，语义值应该是存储在字段 cooLyylval.symbol 中的符号。对于布尔常数，语义值存储在 cooLyylval.bolean 字段中。 除了错误（见下文），其他令牌的词汇不携带任何有趣的信息。
- 我们为您提供了一个字符串表实现，这将在 Cool 支持代码的巡演和代码中的文档中详细讨论。 目前，您只需要知道字符串表条目的类型是符号。
- 当遇到词汇错误时，例程 cooLyylex 应该返回令牌错误。 语义值是表示错误消息的字符串，存储在字段 cool_yylval 中.error_msg（注意此字段是普通字符串，不是符号）.. 有关输入错误消息的信息，请参见上一节。

6 测试扫描仪

至少有两种方法可以测试你的扫描仪。第一种方法是生成示例输入并使用 lexer 运行它们，它打印出扫描器识别的每个令牌的行号和 lexeme。您可以将这里的输出与在/afs/ir.stanford.edu/class/cs143/bin/lexer 上找到的示例 lexer 的输出进行比较。 另一种方法是，当您认为您的扫描仪正在工作时，尝试运行 my coolc，

将您的 lexer 与所有其他编译器阶段（我们提供的）一起调用。 这将是一个完整的 Cool 编译器，您可以尝试任何测试程序。

7 什么是自首

在提交之前，请确保您已经做了以下工作：

- 您有责任确保您提交的最终版本没有任何调试打印语句，并且您的词汇规范是完整的（每个可能的输入都有一些与之匹配的正则表达式）
- 请删除 README 中的说明部分。 本节是从开头开始，并运行到下面一行（包括这一行）

  ---8<---------- 8<---------8<---------8<---- 切在这里-----8<---------8<---------8<---------8<----

  README 的这一部分只是为您提供关于需要编辑哪些文件的额外指导，您必须在实际提交作业之前删除它

- 确保您已经编辑了 README，以便第一行以以下格式包含您的用户名：

  用户：<your_sunet_id>

  如果您在两组中工作，则应该有两个单独的用户 linesthe 提交脚本将失败，如果两个 SUNETID 都包含在同一行中。 特别是，如果两个具有 SUNETID 的学生 abcdef 和 bcdefg 一起工作，他们的 README 应该以以下方式提到 SUNETID：

  用户：ABCDEF
  用户：bcdefg

  重要的是，以下内容将不起作用：

  用户：abcdef，bcdefg

  您的 SUNETID 是您用来登录水稻机的名称(不是您的 8 位学生 ID 号)，可以使用 whoami 命令查询。

- 一旦 README 就绪，从项目目录中运行以下内容：

  提交

  这将打包项目目录中的大部分文件(它将忽略核心转储和 Emacs 备份文件等内容)，并将它们复制到提交文件夹中，同时附上时间戳。

  您提交的作业的最后一个版本将是分级的。 每次提交都覆盖上一次。 然而，我们保留保留旧的提交书的权利，以供在发生任何争端时参考。 请记住，您有三天的延迟时间用于整个季度；详情请参阅课程网站上的延迟政策页面。

说服我们你理解材料的负担在你身上。 省略代码会对你的成绩产生负面影响，所以花点时间让你的代码可读性。

- 重要事项：我们将只接受通过提交脚本提交的作业。 请在截止日期前至少 24 小时测试-提交您的作业，以确保脚本为您工作，即使您没有完成。 提前检查问题是你的责任。