编译原理实验1报告

姓名：朱宸慷 学号：2021110908

1. 程序实现的功能
2. 所使用的数据结构

本次实验在Flex和Bison所提供的功能的基础上，自行实现了语法树SyntaxTree.c，提供Node结构体用于保存分析过程中产生的数据。该结构体采用左孩子右兄弟模式，将多叉树保存为二叉树，方便在输出语法树时进行前序遍历。

基于实验中对于打印节点的要求，构造共六种枚举类型。其中前五种用于识别Token的类型，最后一种用于标识非终结符

文本

描述已自动生成

提供了基于该语法树的构造函数。针对词法分析，主要收集Token的字面量，而在语法分析中，主要是结合产生式，通过提供可变参数完成语法树的构建。



1. 词法分析

在词法分析中，笔者实现了以下功能：识别单行注释和多行注释；识别十进制、八进制、十六进制的整型数；识别小数和带科学计数法的浮点数；识别标识符；识别类型以及控制流关键字；识别其他的符号；对错误的三种类型整型数、错误的浮点数、错误的标识符的识别。

笔者基本上是基于正则表达式来实现以上功能，以下依次介绍部分有特色的内容。

识别行注释：匹配以“//”开头的内容，当识别到该标识时，重复调用input()函数直到输入了一个换行符，此时就在词法分析器中跳过了该行所有内容。

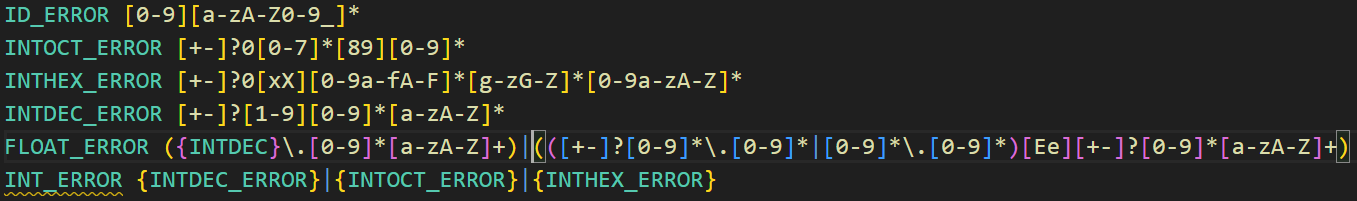
识别块注释：首先匹配以/\*开头的内容，之后匹配\*之前的所有非\*字符，该部分是注释的正文，其次匹配一个或多个\*，要求不以/结尾，最后匹配到一个\*/，表示块注释结束。

文本

描述已自动生成

识别拼写错误的词法单元：笔者通过构造错误的正则表达式实现了对于三种整型、浮点型以及标识符的识别。在整型数中，通过给出不符合进制要求的数字来识别错误，例如八进制中出现8和9，十进制中出现字母、十六进制出现g到z之间的字母等。在浮点数中，通过识别除了e之外其他字母的出现来判断是否错误使用了科学计数法。在标识符中，将数字开头的标识符视为错误标识符。

笔者此处给出的正则表达式不太严谨，比如错误浮点数其实没有排除字母e在外，但是由于Flex识别是按序执行的，而错误处理在代码文件最后，因此不会干扰正常的Token识别。



完成词法定义后，笔者在对应的分析器行为中为五类Token构建了对应的词法节点，并且在发生词法错误后通过全局变量LexError向主函数报告错误。

1. 语法分析

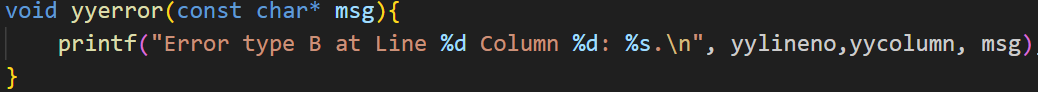
笔者在语法分析中主要实现了以下功能：语法分析树的构建和打印，报错信息的扩展，错误处理与恢复和部分移进-归约冲突的解决。以下依次介绍。

语法分析树的构建：每当Bison完成一个产生式的归约时，就将执行一个指定的动作。笔者在这里为产生式左部符号新建一个非终结符节点，并且将其右部的符号都作为该节点的子代传入构造函数中。除了该节点的基本信息外，该函数还将接受一个指定孩子数量的参数和多个Node类型的可变参数。在函数的内部实现中，首先为左侧节点申请足够的内存并复制基本信息，之后调用可变参数库，根据指定的数量接受传入的节点，将第一个节点指定为孩子后，其余的节点作为该孩子节点的兄弟节点依次连接。



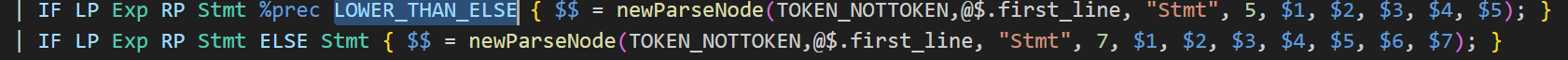
语法分析树的打印：笔者采用递归打印的方式。在调用打印函数时传入语法树根节点以及初始深度0，打印函数首先判断结束条件，如果当前传入指针为空，则停止打印。之后根据指导书要求，打印对应的缩进和节点的名称。此处由于函数携带参数深度，因此直接打印深度\*2个空格即可完成缩进，而名称字段已经保存在结构体中，对应输出即可。其次根据该节点的不同类型，打印不同的内容，例如非终结符打印行号，终结符打印词素或者面值等。完成以上打印工作后，递归调用该函数，先打印孩子节点，再打印自己的兄弟节点，即可完成NLR的前序打印。

报错信息的扩展：通过在.l文件中定义YY\_USER\_ACTION来扩展词法分析器的行为，当产生报错的时候可以通过词法分析器得到该词法单元所在的位置。在.y文件中首先使用%locations选项来通知语法分析器需要定位错误，再通过%define parse.error verbose选项来使得语法分析器在报错的时候输出额外的报错信息，例如缺失某个符号时，语法分析器将提示该符号缺失。



错误处理与恢复：利用Bison的特性，为语法分析器提供恐慌模式的处理策略。在产生式中添加error项并在其后紧跟着一个终结符，这样的产生式会让语法分析器在检查到错误进入恐慌模式时持续向后扫描直到发现该终结符，之后完成该产生式的归约并继续检查之后的Token。

部分移进-归约冲突的解决：当存在形如if if else这种代码的时候，我们要求else与第二个if匹配，但是由于产生式在此处仍然需要递归展开Statement，语法分析器将在解析else时产生移入-归约冲突。所以我们有必要让语法分析器在如上情形中决定先移入else，从而保证语义的正确性。因此，我们可以定义一个名为LOWER\_THAN\_ELSE的终结符，并在Bison中声明该符号的优先级小于ELSE，当语法分析器分析到该语句时，就会自动选择优先级更高的ELSE，从而执行移入动作。



1. 程序的编译方法
2. 处理.y文件

在code文件夹下，首先使用Bison编译Parse.y文件。在执行命令时添加-d参数，可以同时生成Parse.tab.h和Parse.tab.c，其中.c文件是最终可执行文件链接的一部分，而.h文件需要在词法分析中使用，所以此处先编译.y文件，防止后续文件产生依赖问题。

1. 处理.l文件

在Parse.tab.h文件生成完毕后，使用Flex处理Lexer.l文件，生成lex.yy.c，该文件依赖Parse.tab.h文件中定义的token类型

1. 编译可执行文件

完成前置文件Parse.tab.h，Parse.tab.c，lex.yy.c三个文件的生成后，就可以开始编译整个语法分析器的可执行文件。此时需要收集目录下所有.h和.c文件，添加-lfl和-ly参数，即执行gcc -Wall -g -o parser SyntaxTree.c main.c Parse.tab.c -lfl -ly 命令编译生成可执行文件。

1. 自动化执行makefile

以上步骤可以通过使用makefile自动执行。在定义相关文件名变量之后，可以根据上述的依赖关系使用makefile来生成并编译文件。除此之外，笔者还搭配了一个bash脚本用于自动化执行makefile以及输入测试用例进行测试。

1. 结果展示
2. 错误的声明语句，包括非法符号、错误ID、错误整型和浮点型，缺少分号

图片包含 图形用户界面

描述已自动生成

文本

描述已自动生成

该错误样例最后包含了一个没有分号的声明语句，于是语法分析器检查到了“}”而没有等待到“;”从而产生语法报错，同时标识此处期待一个分号。

1. 其他的语法错误，包括嵌套注释、未闭合注释、错误的数组访问

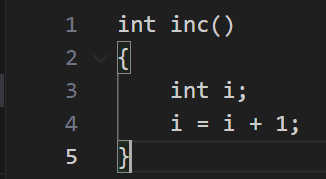
图形用户界面, 文本

描述已自动生成

文本

描述已自动生成

1. 正常运行时，程序将打印生成树，篇幅所限部分生成树内容被折叠

图片包含 图形用户界面

描述已自动生成