Lab01-Scanner and Parser

> 实验名称：NJU-编译原理(2022秋)-Lab01
> 实验人员：201220096 钟亚晨
> 完成日期：2022/10/05
> 更多实现细节可见个人博客文章：(pawx2's Blog)【密码：`chcp65001`】

程序实现了哪些功能?

- 实现了`Appendix_A.pdf`中要求的所有词法、文法规定下的词法分析器与语法分析器，以及：
  - 单行注释、多行注释；
  - 整数`INT`的十六进制、八进制表示；
  - 浮点数`FLOAT`的科学计数法表示；

实现构成

> `lexical.l`和`syntax.y`中使用同一套token类型集合。

lexical.l：

- 为所有的token类型定义对应的正则表达式用于匹配规则，并且返回相应的token类型供语法解析；
  - INT的十六进制、八进制以及FLOAT的科学计数法可直接用正则表达式基本操作实现；
  - 单行注释、多行注释通过input函数以及简单自动机实现（单行注释读到\n进入终止态，多行注释读到连续读到*/进入终止态）；
  - 提供空格与换行的匹配规则，用于略过空白符以及增加相关语义操作；
  - 非法符号使用`.`进行匹配，其意义在于：如果我们定义的正则匹配规则都未匹配到，那么说明其不在C--文法规定范围内，则为非法符号；
- 设定token的默认类型为`Node *`类型，即自定义的树节点指针类型，并且对每个匹配规则中的token调用相关函数进行初始化；
  - 使用了`YYSTYPE`宏定义，统一规定token类型，相比于`%union`在调用时节省了很多代码，且适用于当下类型统一的场景；（该部分在`syntax.y`中也有进行）
  - 将`Node`类型对象的初始化过程封装进了相关函数，并且在`lexical.l`中提供宏定义，其意义在于简化函数调用代码，减省统一、重复代码；
- 利用内置变量`yylineno`在词法分析报错时指明错误位置；

syntax.y：

- 提供所有的token类型定义，此处与`lexical.l`中相统一；
- 使用`%nonassoc`、`%left`、`%right`相关关键字以及声明顺序规定了操作符的结合性、优先级，以解决移入-规约冲突；
- 定义了所有产生式，并且在适当地方增添`error`来使用Bison提供的错误恢复；
  - 该部分也是实现过程中遇到的一个问题，后通过阅读手册、Bison官方文档以及对产生式进行推导尝试，完成了理解
  - 首先在所有产生式体可能出错的位置使用`error`进行替换，而后将形成的带`error`的产生式体增加到对应产生式中；
  - 而后通过分析顶层产生式和底层产生式，可以发现很多顶层产生式的`error`永远不可达（在其包含的底层产生式中就已经被恢复了），因而删除这些冗余的顶层产生式中的相关产生式体；

    - 再进行编译运行，可以发现一些移入-规约冲突，此时分析底层产生式中相关的`error`产生式体，对于造成冲突的产生式体进行删除；
- 依照实验手册在`lexical.l`中定义`YY_USER_ACTION`宏及相关操作，重写报错函数，提供了语法分析中每个token的位置，同时该位置可用于发生语法错误时指明错误位置；

utils、sat_gen.c、general.h：

- general.h 提供了必要的结构体类型定义，包括`YYLTYPE`、`TreeNode`(名为Node)，以及建立语法分析树和初始化节点所需的相关函数声明；
    - 树节点类型定义：`(符号类型[终结符/非终结符]，token类型名，字符串类型存储的值，子节点链表)`；
    - 即多叉树使用的是链表-链表的数据结构；
- sat_gen.c 提供了初始化节点以及建立、打印语法分析树的相关函数；
- utils 提供了将特定类型，如八进制INT、十六进制INT、科学计数法FLOAT转化为C语言基本类型的相关函数；

实现亮点

- 使用了`YYSTYPE`宏统一规定token默认类型，相较于`%union`和`token<type_name>`减省了大量不必要代码，且易于后续代码维护和更改；
- 建树过程中使用到的`add_children`函数使用了带有不定参数的C函数，结合`$$, $1, ... $n`相关符号，减省了不必要代码，易于后续代码维护和更改；

实现过程中遇到的相关问题

> Q01 : 依照实验手册上直接使用宏定义`YYSTYPE`来指定默认token类型无法通过链接，导致报错；

在`lexical.l`当中：`#define YYSTYPE`要放置在`#include syntax.tab.h`之前。（这一点通过查看`syntax.tab.h`的源代码显而易见）

> Q02 : 依照实验手册上直接使用宏定义`YY_USER_ACTION`时，频繁出现`yylloc not defined`错误，导致编译不通过；

其解决办法在于：

- 创建一个`.h`头文件，在其中包含`YYLTYPE`类型的定义，并且该头文件需要被`lexical.l`和`syntax.y`所引用，而后才可在`lexical.l`中使用`YY_USER_ACTION`的宏定义
- 解决方案来自于`StackOverFlow`的两个问答：
  Solution01
  Solution02

> Q03 : 实验手册尚未指明科学计数法表示下的FLOAT类型TOKEN格式，因而导致较早提交OJ时WA；

通过自行摸索和反复尝试，大致对FLOAT科学计数法的格式约定有一个认知：

- 底数部分：
  1.小数点必须有；
  2.小数点前有数字，而小数点后没有；
  3.小数点前无数字，而小数点后有；
  4.小数点前后都有数字；
- `e/E`：必须有，有且仅有一个；
- 指数部分:
  1.可以有一个正/负号，或者没有正/负号；
  2.指数格式同INT，为整数；

> Q04 : 最初定义节点类型后，指定token的默认类型就为Node，从而导致生成树为一个奇怪的东西；

通过在`syntax.y`中添加相关`printf`操作打印地址，得知同一个token对象出现在不同产生式中但内存地址不同，立刻幡然醒悟，意识到自己忽略了局部变量的问题，因此更改为指针类型，对于字符指针的相关操作全部添加了`malloc/free`操作。

> Q05 : 最初在词法规定上纠结于Negative操作和Minus操作分别属于不同优先级和结合性，但是却使用同一个token。

反复阅读和尝试理解后，意识到Negative操作实际上为Minus操作去除左边部分，此时由于Negative没有了左边部分，即使其为左结合性，效果也同右结合性一致，另外后续`syntax.y`中的`Exp`产生式也对这两者的使用进行了规定，从而不存在冲突。

未来展望

当前token节点中对值的存储仍旧使用字符串存储，在需要打印时才使用相关工具函数进行转换，该类型定义形式目前尚不知是否有利于未来相关功能的实现，不过由于当前使用到大量的宏、不定参数的函数以及函数封装，未来即便要变更节点定义也仅需修改相关函数，该部分的灵活性还是很高的。

程序应当如何进行编译?

提供了`makefile`文件，具体编译规则见文件内容，进入到`./Code/`文件夹下，使用命令`make parser`即可完成编译，获得可执行文件`parser`。

而后可通过指令`./parser <target_cmm_file_path>`来对相应的`.cmm`C--源程序进行解析，如果出错则打印错误信息，否则打印出其语法分析树。