Small编译器 开发日志

# 2013/09/17 启动

项目正式启动。决定采用C#语言、VS2012作为核心开发工具。暂不考虑开发GUI，以控制台命令行方式运行。

工作流程：分析并修改语法结构🡪词法分析程序设计🡪语法分析程序设计🡪语义分析程序设计🡪解释器设计。

# 2013/09/20 词法分析程序完成

完成整个词法分析过程设计。采用逐字符分析后拼接成单词，并将单词加入单词流的方法。

# 2013/09/28 语法分析程序框架完成

在单词流基础上进行语法分析。由于已证明small语言为LL（1）语法，故采用自顶向下分析方法。已将所有已定义的语法要素全部编写完毕。

# 2013/10/04 语法分析程序扩充

主要扩充点：符号表的设计。

# 2013/10/05 解释器初步设计完成

参考书本的中间语言设计解释器。但之后发现语法结构有区别，故先将其搁置。容日后研究。

初步想法是可能需要重新设计中间语言。但现在当务之急是搞清语法结构和符号表，以及错误处理。

# 2013/10/08 重构

之前代码由于修改过多已经出现混乱。故重构代码，将词法分析，语法及语义分析相分离，采用项目间引用的方式以避免相互影响。同时定义错误信息表。

# 2013/10/9 尝试Lex和Yacc

从实践上来看，用Lex和Yacc要比手写方便得多。于是开始学习这两个工具的使用。

以资料中的计算器为蓝本，准备在此基础上扩充。

gcc编译提示缺少头文件解决方法：在PATH中加入GCC所在目录，重启即可。

# 2013/10/10 使用LEX和Yacc开工

奋战一天，程序基本完成，但是出现了规约顺序不正确的BUG，经查是语法结构出了问题。明天继续修改、研究。

# 2013/10/14 修正if语句归约问题，完成不带扩展的smallcc

由于之前的if语句总是出现后续语句先归约而判断条件后归约的问题，因此修改了语法结构：

在if-then，if-then-else 和 repeat-until 三种语句中，若执行语句多于一个，必须使用begin-end 词对包含其中。其中end 后的分号可以不加。

至此不带扩展的smallcc完成。后续工作为测试外加扩展：

计划扩展如下：

1. 增加while-do 语句
2. 增加自增自减运算符“++”“--”
3. 增加ODD判断奇偶
4. 求余运算%
5. 布尔运算 and or not xor
6. 位运算
7. 跳过行注释
8. 数组

# 2013/10/15 解释器BUG修复。扩展：跳过/\*\*/和//注释，增加求余运算，增加ODD运算

新发现的BUG：若if-then-else语句中的执行代码段没有加begin-end，则else不会被解析。原因和解决方案待查。

原因：由于本来的语句定义后没有分号，所以导致在遇到单语句的时候先归约。

解决方案：更改语言定义，每个语句后都必须有分号。

# 2013/10/16 更新：增加了数组的定义与使用，增加while-do语句，增加布尔运算

生成的中间代码的指令增加：

stm,将数存储入数组名所指向的地址加上偏移量的位置

ldm，将数组名所指向的地址加上偏移量的位置的值取出

new，说明该标识符是一个数组的首地址

alloc，必须紧跟在new之后，为数组开辟空间

and，or，not，xor，不多说了

修复bug：jz指令应为false跳转，为此修正了repeat-until语句和if-then-else语句的生成代码。

# 2013/10/17 更新：增加exit语句和位运算

exit语句用于结束程序，翻译成halt指令，当程序执行到halt指令时程序结束，不再执行后续语句。

位运算仅支持位与‘&’，位或‘|’，位反‘~’，将异或运算归类为位运算。

# 2013/10/21 更新：增加“++”“--”运算

可以支持单个变量的自增++，自减--运算。有如下四种情况：

1. 前置运算符，即++a,--a
2. 后置运算符，即a++,a--
3. 前置赋值，即b = ++a,b=--a;
4. 后置赋值，即b = a++, b= a--;

同时在ex.c文件中加入一个变量wait，即是否需要等待语句分析结束才处理自增自减运算符

|  |  |
| --- | --- |
| Wait的值 | 说明 |
| 0 | 没有自增（自减）操作，不做处理 |
| 1 | 后置自增操作，若有赋值语句，则语句等价为b=a;a=a+1; |
| 2 | 后置自减操作，和上一条类似 |
| 3 | 前置自增（自减）操作，直接输出inc或dec的操作符 |

另外在语句处理末尾也加入判断，避免了因没有赋值号导致的漏操作。

增加两个中间代码操作符：inc和dec，为栈顶元素增加1或减少1

BUG报告：解释器对堆栈的管理尚不完善，会发生栈溢出的现象，有待进一步调试修正

BUG修正：某些指令没有及时将栈顶的数据弹出导致栈越堆越高，加上弹出栈的操作后BUG基本被消除。

至此想做的扩展基本都完成了。接下来以测试和BUG修正为主，不再增加新的功能。

# 2013/10/22 BUG修正：in指令由于未接受合法字符串的问题

修正了在in指令后如果没有输入合法字符串（即空串）导致的异常问题。改为如不输入则一直提示输入

修正了in语句之后堆栈未弹出问题

# 2013/10/22 更新：增加了详细的错误处理

之前一直忽略了错误处理，调试时发现如果没有错误处理，则编译程序会在错误处停下来，这明显不符合要求。查找资料后发现需要在yacc中用到error这个关键字，经反复试验后能够实现分析出所有的错误并且能跑完整个程序，不会因错误而停止编译。

目前可能的错误如下：

1. 语句后缺少分号
2. If语句中缺少then
3. While语句中缺少do
4. 数组的定义用圆括号
5. 其他错误，统一用“未定义错误”表示

# 2013/10/29 BUG修复：内存泄漏问题（by Jiashuo Li）

Li同学报告了一个问题：词法分析阶段分析出标识符以后，由于使用了malloc()函数分配内存，但是没有地方释放，因此存在可能的内存泄漏问题。

解决方法：在freeNode()函数里，加入判断节点类型若是typeId则释放这个节点所指向标识符字符串的指针。

# 2013/11/03 BUG修复：++，--运算符问题

面对赋值语句和直接调用时++，--运算符的归约顺序出了问题，在赋值语句分析的时候加上一个判断变量，当它为1时为赋值语句，先归约，否则先归约++，--运算符。