C0编译程序的实现

1. C0编译程序的总体设计
2. C0编译程序词法分析的设计与实现
3. C0编译程序语法分析的设计与实现
4. C0编译程序语义分析的设计与实现
5. C0编译程序语法错误处理的实现
6. C0编译程序代码生成的实现
7. 基于C++代码解释器的设计与实现

分工：

A：词法分析getch(),getsym()

B：表格管理、出错处理、部分语法函数编写condition(),loop()

C：语法分析，语义分析block(),statement()

1.1 C0编译程序的总体设计

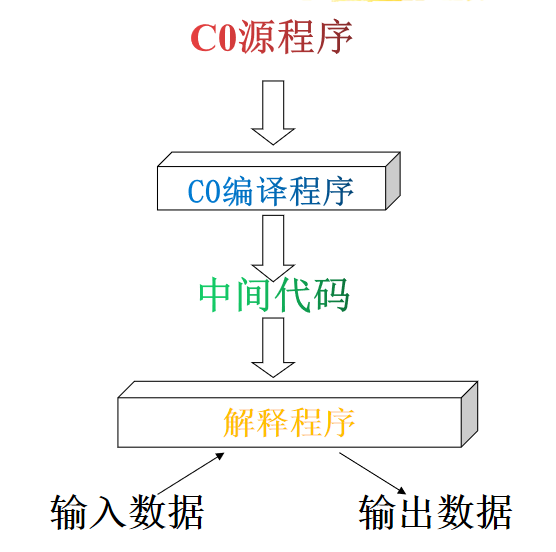
（1）单趟方式

（2）以语法、语义分析程序为核心，词法分析程序和代码生成程序都作为一个过程，当语法分析需要读单词时就调用词法分析程序，而当语法、语义分析正确，需要生成相应的目标代码时，则调用代码生成程序。

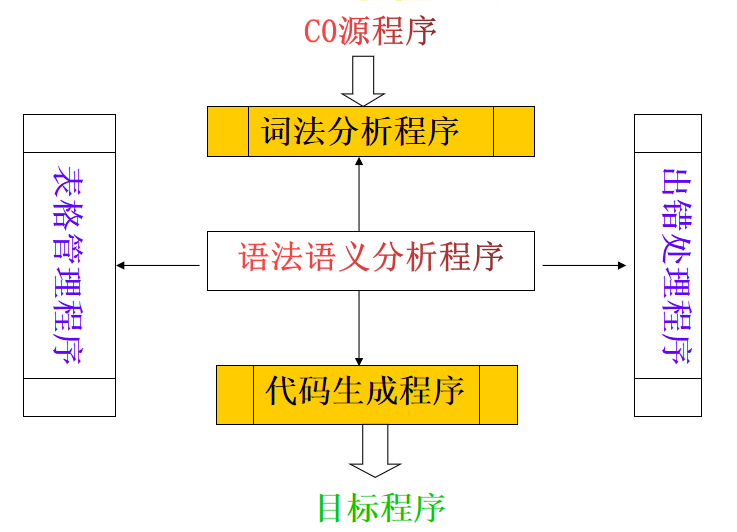
（3）表格管理程序实现变量、常量和过程标识符的信息的登录和查找。

（4）出错处理程序，对词法和语法、语义分析遇到的错误给出源程序中出错的位置和与错误性质有关的编号，并进行错误恢复。

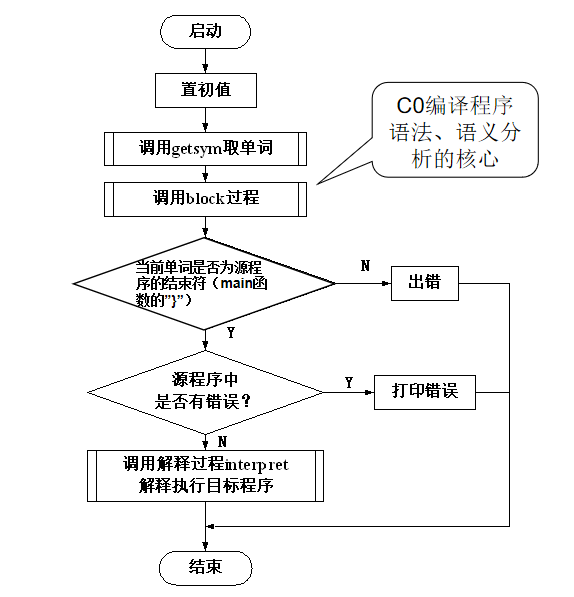
C0编译程序：



C0编译程序的结构：



C0编译系统的总体流程图：



2.1 C0编译程序词法分析的实现

词法分析函数getsym()所识别的单词：

1. 保留字或关键字：int,void,main,if,else,while,return,scanf,printf
2. 运算符：+,-,\*,/,=
3. 标识符：用户定义的变量名，常数名，过程名
4. 常数：如：1,20,180等整数
5. 界符：’,’，’;’，’(‘，’)’，’{‘，’}’

在编译程序中，单词的表示形式 ：(sym,id/num)

1. C0编译程序语法分析的设计与实现

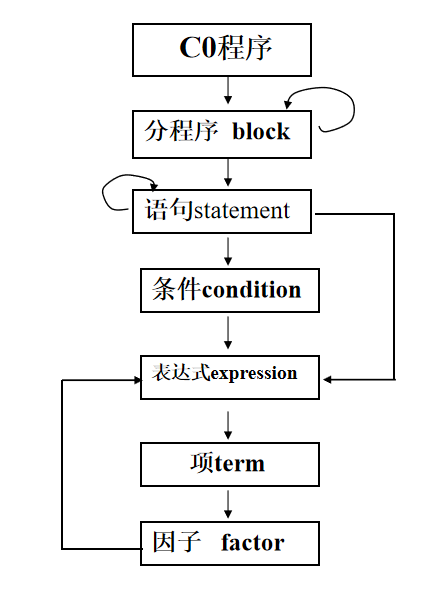
1.自顶向下的语法分析

2.递归子程序法（递归下降分析器recursive-descent parser）：

3.对应每个非终结符（语法成分），编一个独立的处理子程序。

4.由<程序>开始，按规则右部进行分析，遇到非终结符，则调用相应的处理过程，遇到终结符，则判断当前读入的单词是否与该终结符相匹配，若匹配，再读取下一个单词继续分析。

语法调用关系图：



1. C0编译程序语义分析的设计与实现
2. 说明部分的分析与处理

（1）对每个过程（含主程序）说明的对象（变量，常量和过程）造符号表

（2）登录标识符的属性。（包括种类，所在层次,值和分配的相对位置）

2.表格管理

3.语句的分析与处理

1. C0编译程序语法错误处理的实现

错误处理的原则：尽可能准确指出出错位置，错误性质，尽可能进行校正。

六、C0编译程序代码生成的实现

七、解释器的设计与实现

目标代码存放在数组CODE中（程序地址寄存器p）

解释程序定义一个一维整型数组S作为运行栈

栈顶寄存器（指针）t，基址寄存器（指针）b，指令寄存器i（当前正在解释的目标指令）