



**编译原理实验**

学 院 计算机学院

专 业 计算机科学与技术

年级班别 17级6班

序 号 52

学 号 3117004652

学生姓名 郭家豪

指导教师 李小妹

成 绩

2020 年 06 月

**报告：**

**内容：** □详细　 □完整　 □不完整

**设计方案：** □非常合理　 □合理　 □较差

**实现：** □全部实现　 □部分实现　 □未实现

**文档格式：** □规范　 □基本规范　 □不规范

**答辩：**

□理解题目透彻，问题回答流利

□理解题目较透彻，回答问题基本正确

□部分理解题目，部分问题回答正确

□未能完全理解题目，答辩情况较差

**总评成绩：**

□优　　　□良　　　□中　　　□及格　　　□不及格

**目 录**

[1. 任务一 1](#_Toc42788621)

[1.1. 内容 1](#_Toc42788622)

[1.2. 要求 1](#_Toc42788623)

[1.3. 实验过程 2](#_Toc42788624)

[1.3.1. 代码修改 2](#_Toc42788625)

[1.3.2. 测试代码 4](#_Toc42788626)

[1.3.3. 实验结果 5](#_Toc42788627)

[2. 任务二 6](#_Toc42788628)

[2.1. 内容 6](#_Toc42788629)

[2.2. 要求 6](#_Toc42788630)

[2.3. 实验过程 6](#_Toc42788631)

[2.3.1. 代码修改 6](#_Toc42788632)

[2.3.2. 测试代码 11](#_Toc42788633)

[2.3.3. 实验结果 11](#_Toc42788634)

[3. 任务三 12](#_Toc42788635)

[3.1. 内容 12](#_Toc42788636)

[3.2. 要求 12](#_Toc42788637)

[3.3. 语法分析 12](#_Toc42788638)

[3.4. 实验过程 13](#_Toc42788639)

[3.4.1. 代码修改 13](#_Toc42788640)

[3.4.2. 代码分析 14](#_Toc42788641)

[3.4.3. 测试代码 15](#_Toc42788642)

[3.4.4. 实验结果 16](#_Toc42788643)

[4. 总结 17](#_Toc42788644)

1. 任务一
   1. 内容

读懂程序GetSym()：

1. 识别保留字
2. 识别标识符
3. 拼数
4. 拼双字符单词
5. 识别单字符单词
   1. 要求

设计测试方式(将语句必须写在getsym()合适的地方)，测试单词是否能被识别（只完成了词法分析部分，代码运行出错不用理会）

1. **Form1->printfs (“关键字”);**
2. **Form1->printfs ("标识符");**
3. **Form1->printfs("数字");**
4. **Form1->printfs(“双符号");**
5. **Form1->printfs("单字符");**
   1. 实验过程
      1. 代码修改

在GetSym()合适的地方找到相应单词识别的位置，给其添加输出语句。

void GetSym() {

int i,J,K; ALFA A;

while (CH<=' ') GetCh();

if (CH>='A' && CH<='Z') { /\*ID OR RESERVED WORD\*/

K=0;

do {

if (K<AL) A[K++]=CH;

GetCh();

}while((CH>='A' && CH<='Z')||(CH>='0' && CH<='9'));

A[K]='\0';

strcpy(ID,A); i=1; J=NORW;

do {

K=(i+J) / 2;

if (strcmp(ID,KWORD[K])<=0) J=K-1;

if (strcmp(ID,KWORD[K])>=0) i=K+1;

}while(i<=J);

if (i-1 > J) {

SYM=WSYM[K];

Form1->printfs("关键字");

}

else {

SYM=IDENT;

Form1->printfs("标识符");

}

}

else

if (CH>='0' && CH<='9') { /\*NUMBER\*/

K=0; NUM=0; SYM=NUMBER;

do {

NUM=10\*NUM+(CH-'0');

K++; GetCh();

}while(CH>='0' && CH<='9');

Form1->printfs("数字");

if (K>NMAX) Error(30);

}

else

if (CH==':') {

GetCh();

if (CH=='=') { SYM=BECOMES; GetCh();/\*代码改动Form1->printfs("关键字:=");\*/ }

else SYM=NUL;

}

else /\* THE FOLLOWING TWO CHECK WERE ADDED

BECAUSE ASCII DOES NOT HAVE A SINGLE CHARACTER FOR <= OR >= \*/

if (CH=='<') {

GetCh();

if (CH=='=') { SYM=LEQ; GetCh();Form1->printfs("双字符>=");}

else {

SYM=LSS;

Form1->printfs("单字符<");

}

}

else

if (CH=='>') {

GetCh();

if (CH=='=') { SYM=GEQ; GetCh();Form1->printfs("双字符<=");}

else {

SYM=GTR;

Form1->printfs("单字符>");

}

}

else { SYM=SSYM[CH]; GetCh();Form1->printfs("单字符"); }

} /\*GetSym()\*/

* + 1. 测试代码

PROGRAM EX01;

VAR A,B,C;

BEGIN

A:=88;

READ(B);

C:=A+B\*(3+B);

WRITE(C)

END.

* + 1. 实验结果





1. 任务二
   1. 内容
2. 增加保留字：ELSE，FOR，TO，DOWNTO，RETURN
3. 增加双字符单词：\*=，/=，++，--，&，||，！
4. 修改单词：不等号# 改为 <>
   1. 要求

设计测试不等号的NEQL.PL0源程序，并运行

* 1. 实验过程
     1. 代码修改

1. 实验中关键字会增加ELSE，FOR，TO，DOWNTO，RETURN，所以NORW要在原来基础上加上5：

const NORW = 19; /\* # OF RESERVED WORDS \*/

要增加关键字，首先要扩展SYMBOL和SYMOUT的内容

typedef enum { NUL, IDENT, NUMBER, PLUS, MINUS, TIMES,

SLASH, ODDSYM, EQL, NEQ, LSS, LEQ, GTR, GEQ,

LPAREN, RPAREN, COMMA, SEMICOLON, PERIOD,

BECOMES, BEGINSYM, ENDSYM, IFSYM, THENSYM,

WHILESYM, WRITESYM, READSYM, DOSYM, CALLSYM,

CONSTSYM, VARSYM, PROCSYM, PROGSYM, ELSESYM,

FORSYM, TOSYM, DOWNTOSYM, RETURNSYM, TIMESBECOMES,

SLASHBECOMES, PLUSONE, MINUSONE, ANDSYM, ORSYM, NOTSYM

} SYMBOL;

char \*SYMOUT[] = {"NUL", "IDENT", "NUMBER", "PLUS", "MINUS", "TIMES",

"SLASH", "ODDSYM", "EQL", "NEQ", "LSS", "LEQ", "GTR", "GEQ",

"LPAREN", "RPAREN", "COMMA", "SEMICOLON", "PERIOD",

"BECOMES", "BEGINSYM", "ENDSYM", "IFSYM", "THENSYM",

"WHILESYM", "WRITESYM", "READSYM", "DOSYM", "CALLSYM",

"CONSTSYM", "VARSYM", "PROCSYM", "PROGSYM", "ELSESYM", "FORSYM",

"TOSYM", "DOWNTOSYM", "RETURNSYM", "TIMESBECOMES", "SLASHBECOMES",

"PLUSONE", "MINUSONE", "ANDSYM", "ORSYM", "NOTSYM" };

1. 这里以SYMSET SymSetUnion(SYMSET S1，SYMSET S2)为例子，与其后面SymSetAdd、SymSetNew……等等出现33的地方都修改为45，因为新加入的运算符和关键字共12个，以免栈溢出。

SYMSET SymSetUnion(SYMSET S1, SYMSET S2) {

SYMSET S=(SYMSET)malloc(sizeof(int)\*45);

for (int i=0; i<45; i++)

if (S1[i] || S2[i]) S[i]=1;

else S[i]=0;

return S;

}

1. 在GetSym()中添加\*=，/=，++，--，||五个双字符的识别

else if(CH=='+'){

GetCh();

if(CH=='+'){

SYM=PLUSONE; GetCh();

}

else SYM=PLUS;

}

else if (CH=='-'){

GetCh();

if (CH=='-'){

SYM=PLUSONE; GetCh();

}

else SYM=MINUS;

}

else if(CH=='\*'){

GetCh();

if(CH=='='){

SYM=TIMESBECOMES; GetCh();

}

else SYM=TIMES;

}

else if(CH=='/'){

GetCh();

if(CH=='='){

SYM=SLASHBECOMES; GetCh();

}

else SYM=SLASH;

}

else if(CH=='|'){

GetCh();

if(CH=='|'){

SYM=ORSYM; GetCh();

}

}

1. 在GetSym()相应位置，修改“#”为“<>”

else /\* THE FOLLOWING TWO CHECK WERE ADDED

BECAUSE ASCII DOES NOT HAVE A SINGLE CHARACTER FOR <= OR >= \*/

if (CH=='<') {

GetCh();

if (CH=='=') { SYM=LEQ; GetCh(); }

else if (CH=='>'){

SYM=NEQ; GetCh();

}

else {

SYM=LSS;

}

}

1. 按字母表顺序插入，在单字符中注释掉“#”号添加“！”和“&”，并在最后修改DECLBESYS、STATBEGSYS、FACBEGSYS的大小为45

void \_\_fastcall TForm1::ButtonRunClick(TObject \*Sender) {

for (CH=' '; CH<='^'; CH++) SSYM[CH]=NUL;

strcpy(KWORD[ 1],"BEGIN"); strcpy(KWORD[ 2],"CALL");

strcpy(KWORD[ 3],"CONST"); strcpy(KWORD[ 4],"DO");

strcpy(KWORD[ 5],"DOWNTO"); strcpy(KWORD[ 6],"ELSE");

strcpy(KWORD[ 7],"END"); strcpy(KWORD[ 8],"FOR");

strcpy(KWORD[ 9],"IF"); strcpy(KWORD[10],"ODD");

strcpy(KWORD[11],"PROCEDURE");strcpy(KWORD[12],"PROGRAM");

strcpy(KWORD[13],"READ"); strcpy(KWORD[14],"RETURN");

strcpy(KWORD[15],"THEN"); strcpy(KWORD[16],"TO");

strcpy(KWORD[17],"VAR"); strcpy(KWORD[18],"WHILE");

strcpy(KWORD[19],"WRITE");

WSYM[ 1]=BEGINSYM; WSYM[ 2]=CALLSYM;

WSYM[ 3]=CONSTSYM; WSYM[ 4]=DOSYM;

WSYM[ 5]=DOWNTOSYM; WSYM[ 6]=ELSESYM;

WSYM[ 7]=ENDSYM; WSYM[ 8]=FORSYM;

WSYM[ 9]=IFSYM; WSYM[10]=ODDSYM;

WSYM[11]=PROCSYM; WSYM[12]=PROGSYM;

WSYM[13]=READSYM; WSYM[14]=RETURNSYM;

WSYM[15]=THENSYM; WSYM[16]=TOSYM;

WSYM[17]=VARSYM; WSYM[18]=WHILESYM;

WSYM[19]=WRITESYM;

SSYM['+']=PLUS; SSYM['-']=MINUS;

SSYM['\*']=TIMES; SSYM['/']=SLASH;

SSYM['(']=LPAREN; SSYM[')']=RPAREN;

SSYM['=']=EQL; SSYM[',']=COMMA;

SSYM['.']=PERIOD; /\*SSYM['#']=NEQ;\*/

SSYM[';']=SEMICOLON; SSYM['&']=ANDSYM;

SSYM['!']=NOTSYM;

strcpy(MNEMONIC[LIT],"LIT"); strcpy(MNEMONIC[OPR],"OPR");

strcpy(MNEMONIC[LOD],"LOD"); strcpy(MNEMONIC[STO],"STO");

strcpy(MNEMONIC[CAL],"CAL"); strcpy(MNEMONIC[INI],"INI");

strcpy(MNEMONIC[JMP],"JMP"); strcpy(MNEMONIC[JPC],"JPC");

DECLBEGSYS=(int\*)malloc(sizeof(int)\*45);

STATBEGSYS=(int\*)malloc(sizeof(int)\*45);

FACBEGSYS =(int\*)malloc(sizeof(int)\*45);

for(int j=0; j<45; j++) {

DECLBEGSYS[j]=0; STATBEGSYS[j]=0; FACBEGSYS[j] =0;

}

* + 1. 测试代码

PROGRAM NEQL;

VAR A,B;

BEGIN

READ(A);

B:=0;

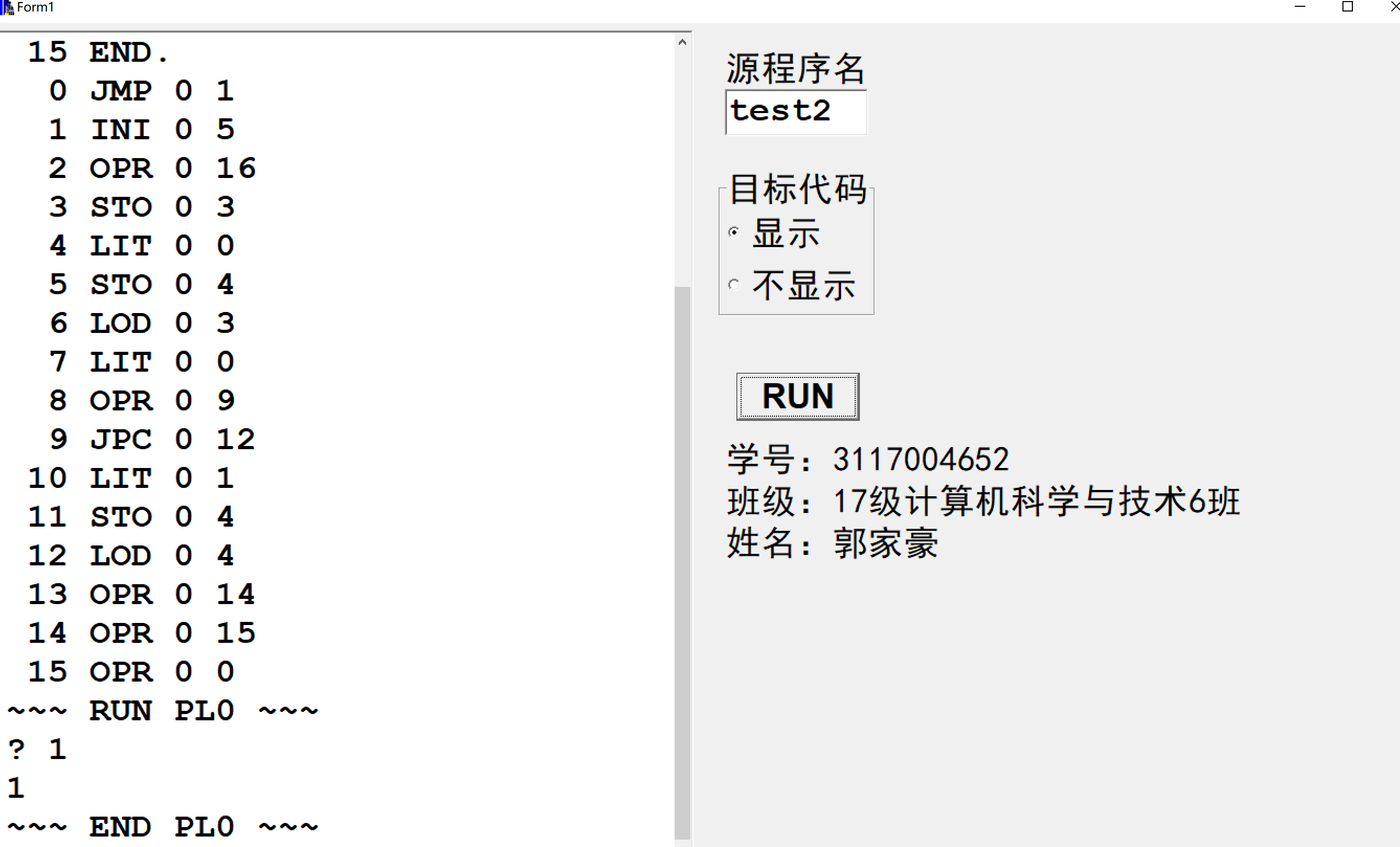
IF A<>0 THEN B:=1;

WRITE(B);

END.

* + 1. 实验结果

输入值1：



1. 任务三
   1. 内容

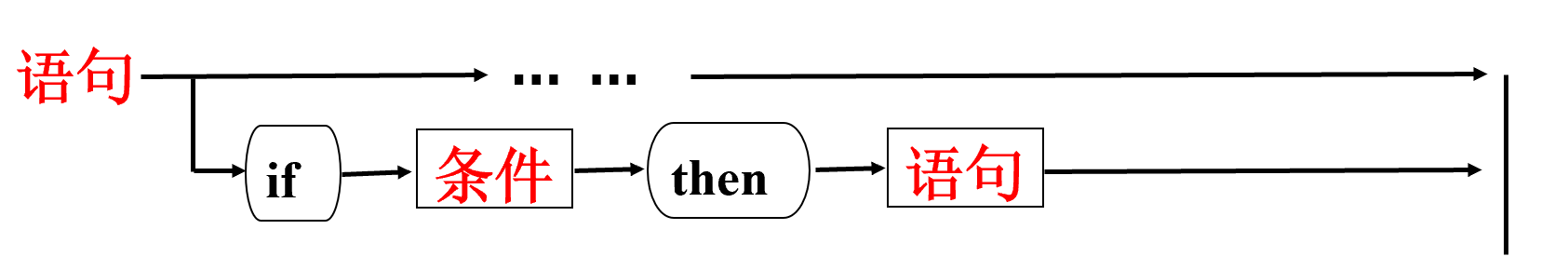
增加条件语句的ELSE子句，改写编译程序，设计测试源码ELSE.PL0，并能正确运行。

* 1. 要求

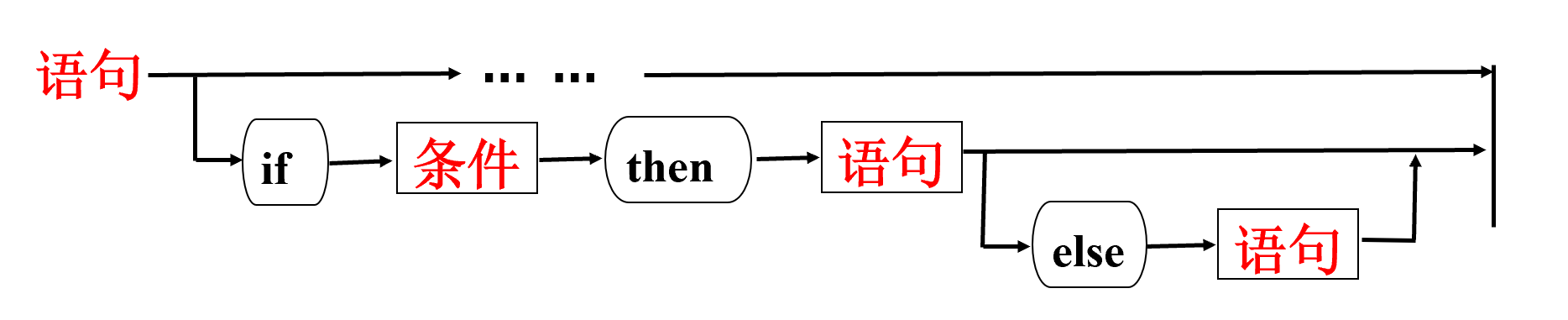
写出相关文法，语法图，语义规则。

* 1. 语法分析

在未修改的程序中，IF语句的语法图应该是这样的：



而现在添加ELSE语句的功能，而ELSE语句的内容会出现在then紧接着的语句后面，所以在这个位置需要给予程序判断，后续是否有标识符ELSE，如果有，产生ELSE功能的语法，如果没有，就只是原来的IF THEN的语法。那么IF语句的语法图将修改为这个样子：



* 1. 实验过程
     1. 代码修改

需要修改语义分析过程Statement函数对应的IF语句的功能，如下

case IFSYM:

GetSym();

CONDITION(SymSetUnion(SymSetNew(THENSYM,DOSYM),FSYS),LEV,TX);

if (SYM==THENSYM) GetSym();

else Error(16);

CX1=CX; GEN(JPC,0,0);

/\*STATEMENT(FSYS,LEV,TX); CODE[CX1].A=CX;\*/

/\*以下为ELSE功能扩展代码添加\*/

STATEMENT(SymSetUnion(SymSetNew(ELSESYM),FSYS),LEV,TX);

if(SYM!=ELSESYM) CODE[CX1].A=CX;

else{

GetSym();

CX2=CX;

GEN(JMP,0,0);

CODE[CX1].A=CX;

STATEMENT(FSYS,LEV,TX);

CODE[CX2].A=CX;

}

break;

* + 1. 代码分析

按照语法图的分析，在原来IF语句的语法识别中，会生成一条目标代码JPC，条件判断跳转。它会根据IF条件内容目标代码的逻辑真假跳转到相应的目标代码地址。目标代码的形式为：指令名称（FCT F，int L， int A），其中A这个位置就是地址。而JPC这一条目标代码会在成功识别THEN标识符后通过GEN()函数生成。但此时的的目标代码为JPC（0，0，0），还没有明确跳转的地址。由于CONDITION生成IF判断条件目标代码后，目标代码的现地址会指向下一条准备生成的地址，所以在成功识别THEN标识符后和通过GEN()函数生成JPC才做代码前，用CX1记录下这个地址，那么这个地址就是这个JPC目标代码的地址。在等待STATEMENT生成THEN后面语句的目标代码后，此时的地址CX就位于这些目标代码后面。通过CODE[CX1].A=CX;把CX赋值给JPC的A位置，其中CODE[CX1]指的是第CX1条目标代码，也就是之前记录下的JPC的位置。那么当IF逻辑真，JPC就不会跳转按顺序执行，如果IF为逻辑假，那么JPC就会跳转到这个CX位置，从而实现跳过THEN后面的目标代码。根据这个道理修改代码，在生成THEN后面的目标代码添加一个识别ELSE标识符的操作。注意的是在IF语义代码开头部分：

CONDITION(SymSetUnion(SymSetNew(THENSYM,DOSYM),FSYS),LEV,TX);这一条语句中出现的

SymSetUnion(SymSetNew(THENSYM,DOSYM),FSYS),LEV,TX，

目的是在FSYS字符集的基础上添加THEN和DO这两个标识符，形成一个新的字符集，用来识别THEN和DO两个字符，也就是说如果没有这条语句，IF后面出现THEN和DO两个标识符将不会被识别出来。同样道理，如果想在THEN后面继续识别ELSE，最起码的就是给字符集添加ELSE字符的识别。所以在扩展代码后面出现了一句

STATEMENT(SymSetUnion(SymSetNew(ELSESYM),FSYS),LEV,TX);

语句中也用到了(SymSetUnion(SymSetNew(ELSESYM),FSYS),LEV,TX)

在添加完识别ELSE标识符的操作后进行判断，如果语句后面不是接着ELSE，那么就可以按IF THEN的语句形式进行。而如果出现了ELSE，就要实现逻辑假的操作。可以知道在生成THEN后面的目标代码，此时的地址位置我们设为CX2，这里我们添加一条JMP无条件跳转语句，此时地址CX为CX2+1，也就是ELSE部分那一段目标代码的首地址，此时才再通过CODE[CX1].A=CX;，把现在的地址CX赋值给JPC。这样做的原因是当IF逻辑真的时候，会按照目标代码的顺序执行，如果执行完THEN后面的目标代码没有JMP这条目标代码，就会顺序执行ELSE部分的目标代码，这样的程序是错误的，所以JMP的作用就是逻辑真跳出ELSE部分的目标代码。同理，逻辑假的时候，就会通过JPC跳转到CX2+1的位置，执行ELSE部分的目标代码。

以上就是这次代码改动的分析。

* + 1. 测试代码

PROGRAM NEQL;

VAR A,B,C,D;

BEGIN

READ(A);

B:=0;

IF A<>0 THEN B:=1 ELSE B:=2;

WRITE(B);

END.

* + 1. 实验结果

输入值2：



输入值0：



1. 总结

这次实验有三个任务，难度都依次递增，任务一让我知道C++Builder的基本操作和正确运行程序后的模样，任务二让我了解如何手动添加单词的识别和对部分字符识别的修改。任务三是难度最大但也是三个任务中收获最多的任务。老师在发布实验任务后，我一直不知道从代码的哪个地方开始动手，直到老师对代码的讲解才了解到PLO语法分析的知识。相对于前两个任务而言，第三个任务不只是单纯的修改代码，还要了解这样修改的原因，修改的内容虽然不多，但是关联到的内容真的不少。其中目标代码地址跳转的操作我也在课后思考了好久才逐渐明白。有了这次的实验收获，我相信会对我课程设计有很大的帮助。