12/04/2021

A7: GENERATION OF INTERMEDIATE CODE USING LEX AND YACC

Vanathi G
185001188
CSE C

CODE :

a7_v5.y:

```
%{
   #include <stdio.h>
   #include <stdlib.h>
   #include <math.h>
   void yyerror();

   struct info
   {
          char var[10];
          char code[100];
          char true[10];
          char false[10];
          char out[10];

   };

   void newTemp(int count, char *var)
   {
          char str_count[3];
          char varname[] = "t";
          sprintf(str_count, "%d", count);
          strcat(varname, str_count);
          strcpy(var, varname);
   }

   void newLabel(int count, char *label)
   {
          char str_count[3];
          char labelname[] = "L";
          sprintf(str_count, "%d", count);
          strcat(labelname, str_count);
```

```
        strcpy(label, labelname);
    }

    struct info* makeNode(int count, char type)
    {
        struct info *temp;
        temp = malloc(sizeof(struct info));

        if(type == 't')
        {
            newTemp(count, temp->var);
        }
        else
        {
            newLabel(count, temp->true);
            if(type == 'i')
                newLabel(count+1, temp->out);
            else {
                newLabel(count+1, temp->false);
                newLabel(count+2, temp->out);
            }

        }

        strcpy(temp->code, "");

        return temp;
    }

    int err_flag = 0;
    int tempvar_count=0;
    int label_count=0;

%}

%union {
    struct info *node;
    char name[50];
    char keyword;
}

%token <keyword> IF THEN ELSE ENDIF BEG END TYPE
%token <name> ID RELOP CONST
%type <node> AS I T F E
```

```
%%
START : DECL PROGRAM
        | DECL
        | PROGRAM

DECL : DECL D
        | D

D : ID':'TYPE';'
  | ID':'TYPE'='CONST';'

PROGRAM : BEG B END

B : B S
  | S

S : AS {printf("%s", $1->code);}
  | I {printf("%s", $1->code);}

I : IF'('ID RELOP ID')' THEN AS ELSE AS ENDIF{
                $$ = makeNode(label_count, 'e');
                label_count += 3;

                char if_code[30];
                sprintf(if_code, "\tif %s %s %s goto %s\n\tgoto %s\n", $3, $4, $5, $$->true,
$$->false);

                sprintf($$->code, "%s%s:%s\tgoto %s\n%s:%s%s:", if_code, $$->true,
$8->code, $$->out, $$->false, $10->code, $$->out);
        }
  | IF'('ID RELOP ID')' THEN AS ENDIF{
                $$ = makeNode(label_count, 'i');
                label_count += 2;

                char if_code[30];
                sprintf(if_code, "\tif %s %s %s goto %s\n\tgoto %s\n", $3, $4, $5, $$->true,
$$->out);

                sprintf($$->code, "%s%s:%s%s:", if_code, $$->true, $8->code, $$->out);
        }

AS : ID'='E';'{
                $$ = makeNode(0, 't');
```

```
                sprintf($$->code, "%s\t%s = %s\n", $3->code, $1, $3->var);
        }

E : T'*'E{
                $$ = makeNode(tempvar_count, 't');
                tempvar_count++;
                sprintf($$->code, "%s%s\t%s = %s * %s\n", $1->code, $3->code, $$->var,
$1->var, $3->var);
        }

  | T'/'E{
                $$ = makeNode(tempvar_count, 't');
                tempvar_count++;
                sprintf($$->code, "%s%s\t%s = %s / %s\n", $1->code, $3->code, $$->var,
$1->var, $3->var);
        }

  | T {$$ = $1;}

T : T'+'F{
                $$ = makeNode(tempvar_count, 't');
                tempvar_count++;
                sprintf($$->code, "%s%s\t%s = %s + %s\n", $3->code, $1->code, $$->var,
$1->var, $3->var);
        }

  | T'-'F{
                $$ = makeNode(tempvar_count, 't');
                tempvar_count++;
                sprintf($$->code, "%s%s\t%s = %s - %s\n", $3->code, $1->code, $$->var,
$1->var, $3->var);
                if(strlen($3->code) > 0)
                        strcat($$->code, $3->code);
        }

  | F {$$ = $1;}

F : ID {$$ = makeNode(0, 't'); strcpy($$->var, $1);}
  ;

%%

void yyerror()
{
```

```
    return;
}

int main()
{
    printf("-------------------------\nINTERMEDIATE CODE
GENERATION\n-------------------------\n");

    FILE *fp = fopen("input.txt", "r");
    char c = fgetc(fp);
        while (c != EOF)
        {
        printf ("%c", c);
        c = fgetc(fp);
        }
        fclose(fp);

    printf("\n-------------------------\nGENERATED CODE\n-------------------------\n");

    yyparse();
    printf("\n");
    return 0;
}
```

a7_v5.l:

```
%{
    #include <stdio.h>
    #include "y.tab.c"
%}

letter [a-zA-Z]
digit [0-9]
relop "<"|"<="|"=="|"!="|">"|">="
type "integer"|"real"|"char"

digits {digit}+
optFrac \.{digits}
optExp E("+"|"-")?{digits}
numberconst {digits}({optFrac})?({optExp})?

charconst \'{letter}\'
```

```
constant {numberconst}|{charconst}

%%

"if" {return IF;}
"then" {return THEN;}
"else" {return ELSE;}
"endif" {return ENDIF;}
"begin" {return BEG;}
"end" {return END;}
{type} {yylval.keyword = yytext[0]; return TYPE;}

{constant} {strcpy(yylval.name, yytext); return CONST;}
{relop} {strcpy(yylval.name, yytext); return RELOP;}
{letter}({letter}|{digit})* {strcpy(yylval.name, yytext); return ID;}
[' '] { };
['\t'] { };
['\n'] { };
. return yytext[0];

%%

int yywrap(){
   return 1;
}
```

OUTPUT :

```
vanathi@vanathi-HP-Pavilion-x360:~/Desktop/Semester 6/Compiler Design/Lab/A7
 vanathi@vanathi-HP-Pavilion-x360  ~/Desktop/Semester 6/Compiler Design/Lab/A7  yacc a7_v5.y
 vanathi@vanathi-HP-Pavilion-x360  ~/Desktop/Semester 6/Compiler Design/Lab/A7  lex a7_v5.l
 vanathi@vanathi-HP-Pavilion-x360  ~/Desktop/Semester 6/Compiler Design/Lab/A7  gcc lex.yy.c -lm -w
 vanathi@vanathi-HP-Pavilion-x360  ~/Desktop/Semester 6/Compiler Design/Lab/A7  ./a.out < input.txt
---------------------------
INTERMEDIATE CODE GENERATION
---------------------------
a:integer;
b:real=5.253;
c:real=15;
ch:char='y';
op:char='5';
p:integer=10;
q:integer=11;

begin
if(p < q) then
        a = p * b + c;
else
        a = q / b - c;
endif
if(ch == op) then
        p = a * b * c;
endif
q = a + b + c;
end
```

```
----------------------------
GENERATED CODE
----------------------------
        if p < q goto L0
        goto L1
L0:     t0 = b + c
        t1 = p * t0
        a = t1
        goto L2
L1:     t2 = b - c
        t3 = q / t2
        a = t3
L2:     if ch == op goto L3
        goto L4
L3:     t4 = b * c
        t5 = a * t4
        p = t5
L4:     t6 = a + b
        t7 = t6 + c
        q = t7

 vanathi@vanathi-HP-Pavilion-x360  ~/Desktop/Semester 6/Compiler Design/Lab/A7
```