

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Date : 22/03/2022 | | | | | | | |
|  | CSPC62 : COMPILER DESIGN  **LAB-5** | | | | | |  |
|  |  | | | | |  | |
|  | | |  |  | | | |
|  | | | Roll no. : 106119100Name : Rajneesh PandeySection : CSE-B |  | | | |
|  | |  | | |  | | |

Construct Syntax tree and generate intermediate code for assignment statement and expressions in C.

Code:

lexer.l

%{

#include"parser.tab.h"

%}

%%

[\t ] ;

[0-9]+ {yylval.symbol = (char)(yytext[0]);return NUMBER;}

[a-z] {yylval.symbol = (char)(yytext[0]);return LETTER;}

">=" {return GE;}

"<=" {return LE;}

"==" {return EQ;}

"!=" {return NE;}

"&&" {return AND;}

"||" {return OR;}

. {return yytext[0];}

\n {return 0;}

%%

int yywrap(){

    return 1;

}

parser.y

%{

    #include<string.h>

    #include<stdio.h>

    int yylex(void);

    int yyerror(const char \*s);

    char addtotable(char,char,char[]);

    int index1=0;

    char temp = 'A'-1;

    struct expr{

    char operand1;

    char operand2;

    char operator[2];

    char result;

};

%}

%union{

    char symbol;

}

    %left OR

    %left AND

    %left GE LE NE EQ

    %left '<' '>'

    %left '+' '-'

    %left '/' '\*'

    %token GE NE LE EQ AND OR

    %token <symbol> LETTER NUMBER

    %type <symbol> exp

    %start statement

%%

statement: LETTER '=' exp ';' {addtotable((char)$1,(char)$3, "=");};

exp: exp '+' exp {$$ = addtotable((char)$1,(char)$3,"+");}

|exp '-' exp {$$ = addtotable((char)$1,(char)$3,"-");}

|exp '/' exp {$$ = addtotable((char)$1,(char)$3,"/");}

|exp '\*' exp {$$ = addtotable((char)$1,(char)$3,"\*");}

|exp '<' exp {$$ = addtotable((char)$1,(char)$3,"<");}

|exp '>' exp {$$ = addtotable((char)$1,(char)$3,">");}

|exp AND exp {$$ = addtotable((char)$1, (char)$3, "&&");}

|exp OR exp {$$ = addtotable((char)$1, (char)$3, "||");}

|exp GE exp {$$ = addtotable((char)$1, (char)$3, ">=");}

|exp LE exp {$$ = addtotable((char)$1, (char)$3, "<=");}

|exp NE exp {$$ = addtotable((char)$1, (char)$3, "!=");}

|exp EQ exp {$$ = addtotable((char)$1, (char)$3, "==");}

|'(' exp ')' {$$ = (char)$2;}

|NUMBER {$$ = (char)$1;}

|LETTER {$$ = (char)$1;}

;

%%

struct expr arr[20];

int yyerror(const char \*s){

    printf("%s",s);

}

char addtotable(char a, char b, char o[]){

    temp++;

    arr[index1].operand1 = a;

    arr[index1].operand2 = b;

    strcpy(arr[index1].operator, o);

    arr[index1].result=temp;

    index1++;

    return temp;

}

void threeAdd(){

    int i=0;

    while(i<index1){

        printf("%c := ",arr[i].result);

        printf("%c ",arr[i].operand1);

        printf("%c%c ",arr[i].operator[0], arr[i].operator[1]);

        printf("%c",arr[i].operand2);

        i++;

        printf("\n");

    }

}

int main(){

    printf("Enter the expression: ");

    yyparse();

    threeAdd();

    printf("\n");

    return 0;

}

Input:





Run :

Text

Description automatically generated

Output :

**Text

Description automatically generated**

**Text

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated with low confidence**

**Text

Description automatically generated**