|  |  |
| --- | --- |
| #include<stdio.h> |  |
|  | #include<ctype.h> |
|  | #include<string.h> |
|  | char keyword[24][30]={"int","while","break","for","do","if","float","char","switch","double","short","long", |
|  | "unsigned","sizeof","else","register","extern","static","auto","case","break","volatile","enum","typedef"}; |
|  |  |
|  | int check\_keyword(char s[]) //linear search |
|  | { |
|  | for(int i=0;i<24;++i) |
|  | if(strcmp(s,keyword[i])==0) |
|  | return 1; |
|  | return 0; |
|  | } |
|  | /\*write a function to store identifier in symbol table |
|  | Check whether the id is already available in the symbol table, if available, |
|  | ignore. otherwise add it.\*/ |
|  | void store\_symb\_tab(char id[], char symb\_tab[][30]) |
|  | { |
|  | int i; |
|  | for(i=0; strcmp(symb\_tab[i],"")&&i<20;++i) |
|  | if(!strcmp(symb\_tab[i],id)) |
|  | return; |
|  | if(i==20) |
|  | { printf("\nOverflow!"); return;}// create linked list to avoid this |
|  | strcpy(symb\_tab[i],id); //adds id to symb\_tab |
|  | } |
|  |  |
|  | int main() |
|  | { |
|  | FILE \*fp1,\*fp2; |
|  | char c,id[30], num[10]; |
|  | int state=0,i=0,j=0; |
|  | fp1=fopen("x.txt","r");//input file containing src prog |
|  | fp2=fopen("y.txt","w");//output file name |
|  | //declare symbol table as a doubly dimensional array of characters. |
|  | char symb\_tab[20][30]; |
|  | for(int i=0;i<20;++i) |
|  | strcpy(symb\_tab[i],""); //initialise symb\_table with empty strings to clear garbage |
|  |  |
|  | while((c=fgetc(fp1))!=EOF) |
|  | { |
|  | switch(state) |
|  | { |
|  | case 0: |
|  | if(isalpha(c)||c=='\_') |
|  | { state=1; id[i++]=c; } |
|  | else if(isdigit(c)) |
|  | { state=3; num[j++]=c; } |
|  | else if(c=='<' || c=='>') |
|  | state=5; |
|  | else if(c=='=' || c=='!') |
|  | state=8; |
|  | else if(c=='/') |
|  | state=10; |
|  | else if(c==' ' || c=='\t' || c=='\n' || c=='\r'); //state remains 0; |
|  | else |
|  | fprintf(fp2,"\n%c",c); |
|  | break; |
|  | case 1: |
|  | if(isalnum(c)||c=='\_') |
|  | id[i++]=c; //state remains 1 |
|  | else{ |
|  | id[i]='\0'; |
|  | if(check\_keyword(id)) |
|  | fprintf(fp2," \n%s : keyword ",id); |
|  | else{ |
|  | fprintf(fp2,"\n%s : identifier",id); |
|  | // call a function which stores id in symbol table |
|  | store\_symb\_tab(id,symb\_tab); |
|  | } |
|  | state=0; |
|  | i=0; |
|  | ungetc(c,fp1); |
|  | } |
|  | break; |
|  | case 3: |
|  | if(isdigit(c)) |
|  | num[j++]=c; //state remains 3 |
|  | else{ |
|  | num[j]='\0'; |
|  | fprintf(fp2," \n%s: number",num); |
|  | state=0; |
|  | j=0; |
|  | ungetc(c,fp1); |
|  | } |
|  | break; |
|  | case 5: |
|  | if(c=='='){ |
|  | //write code to print specific operator like <= or >= |
|  | fseek(fp1,-2,SEEK\_CUR); //go back 2 chars |
|  | c=fgetc(fp1); // read '<' or '>' again |
|  | if(c=='<') |
|  | fprintf(fp2,"\n<=: relational operator LE"); |
|  | else |
|  | fprintf(fp2,"\n<=: relational operator GE"); |
|  | c=fgetc(fp1); // read '=' again |
|  | state=0; |
|  | } |
|  | else{ |
|  | //write code to print specific operator like <, >, <= or >= |
|  | fseek(fp1,-2,SEEK\_CUR); //go back 2 chars |
|  | c=fgetc(fp1); // read '<' or '>' again |
|  | if(c=='<') |
|  | fprintf(fp2,"\n<: relational operator LT"); |
|  | else |
|  | fprintf(fp2,"\n>: relational operator GT"); |
|  | //c=fgetc(fp1) // read '=' again |
|  | state=0; |
|  | //ungetc(c,fp1); |
|  | } |
|  | break; |
|  | case 8: |
|  | if(c=='='){ |
|  | //write code to print specific operator like == or != |
|  | fseek(fp1,-2,SEEK\_CUR); //go back 2 chars |
|  | c=fgetc(fp1); // read '!' or '=' again |
|  | if(c=='=') |
|  | fprintf(fp2,"\n==: relational operator EQ"); |
|  | else |
|  | fprintf(fp2,"\n!=: relational operator NE"); |
|  | c=fgetc(fp1); // read '=' again |
|  | state=0; |
|  | } |
|  | else{ |
|  | fprintf(fp2,"\n!"); |
|  | ungetc(c,fp1); |
|  | state=0; |
|  | } |
|  | break; |
|  | case 10: |
|  | if(c=='\*') |
|  | state=11; |
|  | else{ |
|  | fprintf(fp2,"\n/%c: invalid lexeme",c); |
|  | state=0; |
|  | } |
|  | break; |
|  | case 11: |
|  | if(c=='\*') |
|  | state=12; |
|  | //else |
|  | // state=11; |
|  | break; |
|  | case 12: |
|  | if(c=='\*'); //state remains 12 |
|  | else if(c=='/') |
|  | state=0; |
|  | else |
|  | state=11; |
|  | break; |
|  |  |
|  | }//End of switch |
|  | }//end of while |
|  | if(state==11) |
|  | fprintf(fp2,"\ncomment did not close"); |
|  | fclose(fp1); |
|  | fclose(fp2); |
|  | // To print symbol table remove these comments |
|  | //for(int i=0; strcmp(symb\_tab[i],"")&&i<20;++i) |
|  | // printf("\n%s",symb\_tab[i]); |
|  | return 0; |
|  | } |