```c
/*
Compiler Design
Assignment 3
Himansu Rathi,R Om prakash
*/

#include<stdio.h>
#include<ctype.h>
#include<limits.h>
#include<string.h>
#define MAX_TOKENS 70

/*different accepted tokens */
char *keywords[]= {"auto","extern","sizeof","break","float","static","case",
                   "for","struct","char","goto","switch","const","if","typedef",
                   "continue","int","union","default","long","unsigned","do",
                   "register","void","return","double","volatile","else","short",
                   "while","enum","signed"};
char *relational_op[]= {"<",">","<=",">=","==","!="};
char *arithmetic_op[]= {"+","-","*","/","%"};
char *conditional_op[]= {"&&","||","!"};
char *assignment_op[]= {"="};
char *incdec_op[]= {"++","--"};
char *delimiter[]= {"{","}","(",")","[","]"};

/*structure for indivisual token */
typedef struct token
{
    char name[30];
    int code;
    unsigned int ptr[100];
    int occur;
} token;
token st[MAX_TOKENS];

/*Symbol Table valid insertion*/
void symbol_table()
{
    int k,j,len;
    k=0;
    /*valid keyword*/
    len=sizeof(keywords)/sizeof(*keywords);
    while(k<len)
    {
        strcpy(st[k].name,keywords[k]);
        st[k].code=(k+1);
        st[k].ptr[0]=-1;
        st[k].occur=0;
        k++;
    }

    j=k;
    k=0;
    /*valid relational operator*/
    len=sizeof(relational_op)/sizeof(*relational_op);
    while(k<len)
    {
```

```
58          strcpy(st[j].name,relational_op[k]);
59          st[j].code=j;
60          st[j].ptr[0]=(k+1);
61          st[j].occur=0;
62          k++;
63          j++;
64      }
65      k=0;
66      /*valid arithmetic operator*/
67      len=sizeof(arithmetic_op)/sizeof(*arithmetic_op);
68      while(k<len)
69      {
70          strcpy(st[j].name,arithmetic_op[k]);
71          st[j].code=j;
72          st[j].ptr[0]=(k+1);
73          st[j].occur=0;
74          k++;
75          j++;
76      }
77      k=0;
78      /*valid conditional operator*/
79      len=sizeof(conditional_op)/sizeof(*conditional_op);
80      while(k<len)
81      {
82          strcpy(st[j].name,conditional_op[k]);
83          st[j].code=j;
84          st[j].ptr[0]=(k+1);
85          st[j].occur=0;
86          k++;
87          j++;
88      }
89      k=0;
90      /*valid assignment operator*/
91      len=sizeof(assignment_op)/sizeof(*assignment_op);
92      while(k<len)
93      {
94          strcpy(st[j].name,assignment_op[k]);
95          st[j].code=j;
96          st[j].ptr[0]=(k+1);
97          st[j].occur=0;
98          k++;
99          j++;
100     }
101     k=0;
102     /*valid increment-decrement operator*/
103     len=sizeof(incdec_op)/sizeof(*incdec_op);
104     while(k<len)
105     {
106         strcpy(st[j].name,incdec_op[k]);
107         st[j].code=j;
108         st[j].ptr[0]=(k+1);
109         st[j].occur=0;
110         k++;
111         j++;
112     }
113     k=0;
114     /*valid Delimiter*/
```

```
115       len=sizeof(delimiter)/sizeof(*delimiter);
116       while(k<len)
117       {
118           strcpy(st[j].name,delimiter[k]);
119           st[j].code=j;
120           st[j].ptr[0]=(k+1);
121           st[j].occur=0;
122           k++;
123           j++;
124       }
125   }
126
127   void print_stm()
128   {
129       int i=0,j;
130       printf("\n-----------------------------------------------\nSYMBOL TABLE\n");
131       printf("\n%10s \t%s \t%s\n","NAME","CODE","POINTER");
132       for(i=0; i<MAX_TOKENS; i++)
133       {
134           if(st[i].code!=0 && st[i].occur>=1)
135               {
136                 printf("%10s \t%d \t%d",st[i].name,st[i].code,st[i].ptr[0]);
137                 for(j=1;j<st[i].occur;j++)
138                     printf(",%d",st[i].ptr[j]);
139                 printf("\n");
140               }
141       }
142   }
143   /* Read from File*/
144   void fileread(FILE *f, char *input)
145   {
146       char c;
147       int j=0;
148       do
149       {
150           c=fgetc(f);
151           input[j++]=c;
152       }
153       while(c!=EOF);
154       input[j]='\0';
155   }
156   /* print input file*/
157   void input_print(char *in)
158   {
159       printf("Input file:\n");
160       printf("\n%s",in);
161   }
162   /* For other symbols as delimiter*/
163   void other(char *input)
164   {
165   int i,j,k=0,count=0;
166   char temp[100];
167     for(i=0;i<strlen(input);i++)
168       {
169          /* Non Alpha Numeric Characters*/
170         if((!isalnum(input[i])) || (count==1))
171          {
```

```
172         if(input[i]=='+' || input[i]=='-' || input[i]=='>' || input[i]=='<' || input [i]=='=' || input[i]=='!'|| input[i]=='&' ||input[i]=='|' )
173         {
174             //for +,-,<,>
175             count++;
176             // for <=,>=,==,&&,!=,++
177             temp[k++]=input[i];
178             if(count<2)
179              continue;
180         }
181         else if(count!=1)
182         {
183             temp[k++]=input[i];
184             temp[k]='\0';
185         }
186             for(j=0;j<56;j++)
187             {
188                 /*compare with existing structure*/
189                 if(strcmp(st[j].name,temp)==0)
190                 {
191                   st[j].occur=1;
192                   break;
193                 }
194             }
195            if(j==56 && (!(strcmp(temp," ")==0 || strcmp(temp,"\n")==0 || strcmp(temp ,"\t")==0)))
196            {
197              /* invalid tokens*/
198              strcpy(st[58].name,"INVALID");
199              st[58].ptr[st[58].occur]=&st[58].ptr[st[58].occur];
200              st[58].occur++;
201            }
202            temp[0]='\0' ;
203            k=0;
204            count=0;
205          }
206      }
207
208  }
209  /*Parsing Lexical Analyzer*/
210  void lexical_analyzer(char* input)
211  {
212      int j,k,flag=0,val=56;
213      char *temp;
214      char delim[]=": =+-*/%^{}!()&|<>,.\"\'\n\t\r";
215
216      for(temp=strtok(input,delim); temp!=NULL; temp=strtok(NULL,delim))
217      {
218          flag=0;
219          /* Checking which keywords are appearing*/
220          for(j=0; j<sizeof(keywords)/sizeof(*keywords); j++)
221              if(strcmp(temp,keywords[j])==0)
222                  {
223                      st[j].occur=1;
224                      flag=1;
225                      break;
226                  }
```

```
227             if(flag==0)
228             {
229                 //then temp is an identifier and so add to st
230                 if(isdigit(temp[0]))
231                     {
232                       for(k=1;k<strlen(temp);k++)
233                       {
234                          if(isdigit(temp[k]))
235                               st[56].code=56;
236                          else
237                               {
238                                   /*For Invalid characters*/
239                                 st[58].code=58;
240                                 flag=1;
241                                 val=58;
242                                 break;
243                               }
244                        }
245                        /*For Constants*/
246                        if(flag==0)
247                            {
248                              strcpy(st[56].name,"CONSTANT");
249                              val=56;
250                            }
251                     }
252                 else
253                 {
254                   for(k=0;k<strlen(temp);k++)
255                   {
256                      /*Alpha numeric characters*/
257                      if(isalpha(temp[k]))
258                           {
259                              st[57].code=57;
260                              val=57;
261                           }
262                      else
263                           {
264                             st[58].code=58;
265                             val=58;
266                             break;
267                           }
268                    }
269                 if(flag==0)
270                 {
271                   /* add Identifier*/
272                    strcpy(st[57].name,"IDENTIFIER");
273                    val=57;
274                 }
275                 }
276                 if(flag)
277                   {
278                     /*add invalid charcter*/
279                     strcpy(st[58].name,"INVALID");
280                     val=58;
281                   }
282                   if(!(strcmp(temp," ")==0 || strcmp(temp,"\n")==0 || strcmp(temp,"\t"
                    )==0))
```

```
283                 {
284                   /* both for identifiers and Invalid words*/
285                   st[val].ptr[st[val].occur]=&st[val].ptr[st[val].occur];
286                   st[val].occur++;
287                 }
288             }
289           }
290       }
291
292   int main()
293   {
294       char input[MAX_INPUT],filename[50];
295       /*Input file name asked from user*/
296       printf("Enter the file name with.txt extension\n");
297       scanf("%s", filename);
298       FILE *fp=fopen(filename, "r");
299       fileread(fp,input);
300       input_print(input);
301       /*For Parsing of code to get keywords ,operators and identifiers*/
302       symbol_table();
303       other(input);
304       lexical_analyzer(input);
305       /* Printing output symbol table*/
306       print_stm();
307       return 0;
308   }
309
```