Program for implementing Shift Reduce Parsing using C

**ALGORITHM:**

1. Get the input expression and store it in the input buffer.
2. Read the data from the input buffer one at the time.
3. Using stack and push & pop operation shift and reduce symbols with respect to production rules available.
4. Continue the process till symbol shift and production rule reduce reaches the start symbol.
5. Display the Stack Implementation table with corresponding Stack actions with input symbols.

**PROGRAM:**

#include

#include

#include

#include

char ip\_sym[15],stack[15];

int ip\_ptr=0,st\_ptr=0,len,i;

char temp[2],temp2[2];

char act[15];

void check();

void main()

{

clrscr();

printf("\n\t\t SHIFT REDUCE PARSER\n");

printf("\n GRAMMER\n");

printf("\n E->E+E\n E->E/E");

printf("\n E->E\*E\n E->a/b");

printf("\n enter the input symbol:\t");

gets(ip\_sym);

printf("\n\t stack implementation table");

printf("\n stack \t\t input symbol\t\t action");

printf("\n\_\_\_\_\_\_\_\_\t\t\_\_\_\_\_\_\_\_\_\_\_\_\t\t\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("\n $\t\t%s$\t\t\t--",ip\_sym);

strcpy(act,"shift");

temp[0]=ip\_sym[ip\_ptr];

temp[1]='\0';

strcat(act,temp);

len=strlen(ip\_sym);

for(i=0;i<=len-1;i++)

{

stack[st\_ptr]=ip\_sym[ip\_ptr];

stack[st\_ptr+1]='\0';

ip\_sym[ip\_ptr]=' ';

ip\_ptr++;

printf("\n $%s\t\t%s$\t\t\t%s",stack,ip\_sym,act);

strcpy(act,"shift");

temp[0]=ip\_sym[ip\_ptr];

temp[1]='\0';

strcat(act,temp);

check();

st\_ptr++;

}

st\_ptr++;

check();

}

void check()

{

int flag=0;

temp2[0]=stack[st\_ptr];

temp2[1]='\0';

if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))

{

stack[st\_ptr]='E';

if(!strcmpi(temp2,"a"))

printf("\n $%s\t\t%s$\t\t\tE->a",stack,ip\_sym);

else

printf("\n $%s\t\t%s$\t\t\tE->b",stack,ip\_sym);

flag=1;

}

if((!strcmpi(temp2,"+"))||(strcmpi(temp2,"\*"))||(!strcmpi(temp2,"/")))

{

flag=1;

}

if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E\E"))||(!strcmpi(stack,"E\*E")))

{

strcpy(stack,"E");

st\_ptr=0;

if(!strcmpi(stack,"E+E"))

printf("\n $%s\t\t%s$\t\t\tE->E+E",stack,ip\_sym);

else

if(!strcmpi(stack,"E\E"))

printf("\n $%s\t\t%s$\t\t\tE->E\E",stack,ip\_sym);

else

if(!strcmpi(stack,"E\*E"))

printf("\n $%s\t\t%s$\t\t\tE->E\*E",stack,ip\_sym);

else

printf("\n $%s\t\t%s$\t\t\tE->E+E",stack,ip\_sym);

flag=1;

}

if(!strcmpi(stack,"E")&&ip\_ptr==len)

{

printf("\n $%s\t\t%s$\t\t\tACCEPT",stack,ip\_sym);

getch();

exit(0);

}

if(flag==0)

{

printf("\n%s\t\t\t%s\t\t reject",stack,ip\_sym);

exit(0);

}

return;

}

**OUTPUT:**

                             SHIFT REDUCE PARSER

          GRAMMER

          E->E+E

          E->E/E

          E->E\*E

          E->a/b

          Enter the input symbol: a+b

Stack Implementation Table

         Stack              Input Symbol                  Action

         -------           -----------------               ---------

         $                 a+b$                             --

          $a                  +b$                              shift a

          $E                  +b$                              E->a

         $E+                  b$                              shift +

          $E+b                  $                              shift b

          $E+E                  $                              E->b

         $E                   $                              E->E+E

         $E                    $                              ACCEPT