**Compiler Design**

**Shift Reduce Parsing**

**Sai Ram Pendyala**

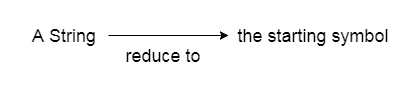
**RA1911003010696**

**Aim:**

To implement shift reduce parser using any programming language.

**Process:**

* Shift reduce parsing is a process of reducing a string to the start symbol of a grammar.
* Shift reduce parsing uses a stack to hold the grammar and an input tape to hold the string.



* Sift reduce parsing performs the two actions: shift and reduce. That's why it is known as shift reduces parsing.
* At the shift action, the current symbol in the input string is pushed to a stack.
* At each reduction, the symbols will be replaced by the non-terminals. The symbol is the right side of the production and non-terminal is the left side of the production.

**Program:**

#include<stdio.h>

#include<string.h>

*int* k=0,z=0,i=0,j=0,c=0;

*char* a[16],ac[20],stk[15],act[10];

*void* check();

*int* main()

{

    puts("GRAMMAR is E->E+E \n E->E\*E \n E->(E) \n E->id");

    puts("enter input string ");

    gets(a);

    c=strlen(a);

    strcpy(act,"SHIFT->");

    puts("stack \t input \t action");

    for(k=0,i=0; j<c; k++,i++,j++)

    {

        if(a[j]=='i' && a[j+1]=='d')

        {

            stk[i]=a[j];

            stk[i+1]=a[j+1];

            stk[i+2]='\0';

            a[j]=' ';

            a[j+1]=' ';

            printf("\n$%s\t%s$\t%sid",stk,a,act);

            check();

        }

        else

        {

            stk[i]=a[j];

            stk[i+1]='\0';

            a[j]=' ';

            printf("\n$%s\t%s$\t%ssymbols",stk,a,act);

            check();

        }

    }

}

*void* check()

{

    strcpy(ac,"REDUCE TO E");

    for(z=0; z<c; z++)

        if(stk[z]=='i' && stk[z+1]=='d')

        {

            stk[z]='E';

            stk[z+1]='\0';

            printf("\n$%s\t%s$\t%s",stk,a,ac);

            j++;

        }

    for(z=0; z<c; z++)

        if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')

        {

            stk[z]='E';

            stk[z+1]='\0';

            stk[z+2]='\0';

            printf("\n$%s\t%s$\t%s",stk,a,ac);

            i=i-2;

        }

    for(z=0; z<c; z++)

        if(stk[z]=='E' && stk[z+1]=='\*' && stk[z+2]=='E')

        {

            stk[z]='E';

            stk[z+1]='\0';

            stk[z+1]='\0';

            printf("\n$%s\t%s$\t%s",stk,a,ac);

            i=i-2;

        }

    for(z=0; z<c; z++)

        if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')

        {

            stk[z]='E';

            stk[z+1]='\0';

            stk[z+1]='\0';

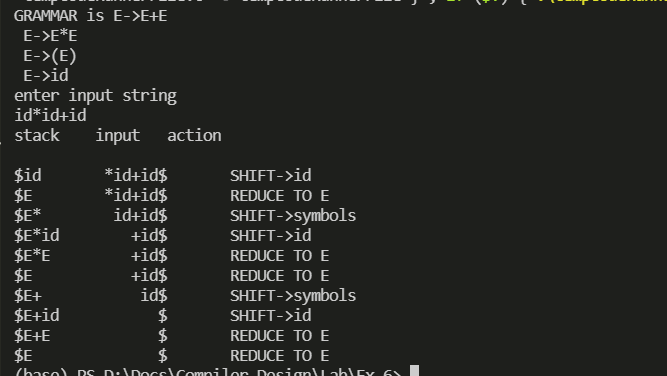
            printf("\n$%s\t%s$\t%s",stk,a,ac);

            i=i-2;

        }

}

**Output:**



**Result:**

Hence, the shift reduce parser is implemented and tested successfully.