SHIFT REDUCING PARSER

EX. NO. 7

Parth Langalia

Date:6/3/23

AIM: To write a program to implement the shift-reducing parsing algorithm.

ALGORITHM:

Shift Reduce Parser requires two Data Structures

- Input Buffer
- Stack

There are the various steps of Shift Reduce Parsing which are as follows −

There are the various steps of Shift Reduce Parsing which are as follows −

- It uses a stack and an input buffer.
- Insert $ at the bottom of the stack and the right end of the input string in Input Buffer.

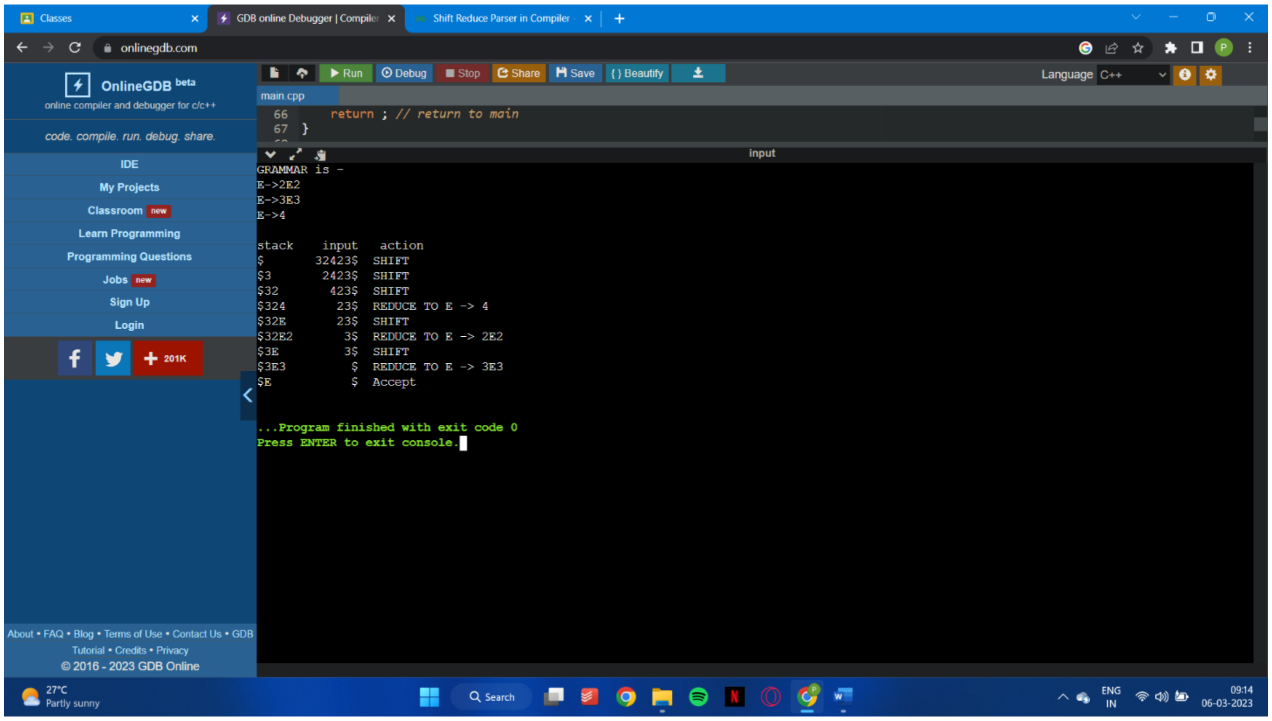
- **Shift −** Parser shifts zero or more input symbols onto the stack until the handle is on top of the stack.
- **Reduce −** Parser reduce or replace the handle on top of the stack to the left side of production, i.e., R.H.S. of production is popped, and L.H.S is pushed.
- **Accept −** Step 3 and Step 4 will be repeated until it has detected an error or until the stack includes start symbol (S) and input Buffer is empty, i.e., it contains $.

PROGRAM:

```
// Including Libraries
#include <bits/stdc++.h>
using namespace std;

// Global Variables
int z = 0, i = 0, j = 0, c = 0;

// Modify array size to increase
// length of string to be parsed
char a[16], ac[20], stk[15], act[10];

// This Function will check whether
// the stack contain a production rule
// which is to be Reduce.
// Rules can be E->2E2 , E->3E3 , E->4
void check()
{
// Copying string to be printed as action
strcpy(ac,"REDUCE TO E -> ");

// c=length of input string
for(z = 0; z < c; z++)
{
// checking for producing rule E->4
if(stk[z] == '4')
{
        printf("%s4", ac);
        stk[z] = 'E';
        stk[z + 1] = '\0';

        //printing action
        printf("\n$%s\t%s$\t", stk, a);
    }
    }

    for(z = 0; z < c - 2; z++)
    {
    // checking for another production
    if(stk[z] == '2' && stk[z + 1] == 'E' &&
                                        stk[z + 2] == '2')
    {
        printf("%s2E2", ac);
        stk[z] = 'E';
        stk[z + 1] = '\0';
        stk[z + 2] = '\0';
        printf("\n$%s\t%s$\t", stk, a);
        i = i - 2;
    }

    }

    for(z = 0; z < c - 2; z++)
    {
    //checking for E->3E3
    if(stk[z] == '3' && stk[z + 1] == 'E' &&
                                        stk[z + 2] == '3')
    {
        printf("%s3E3", ac);
        stk[z]='E';
        stk[z + 1]='\0';
        stk[z + 2]='\0';
        printf("\n$%s\t%s$\t", stk, a);
        i = i - 2;
}
}
return ; // return to main
}

// Driver Function
int main()
{
printf("GRAMMAR is -\nE->2E2 \nE->3E3 \nE->4\n");

// a is input string
strcpy(a,"32423");

// strlen(a) will return the length of a to c
c=strlen(a);

// "SHIFT" is copied to act to be printed
strcpy(act,"SHIFT");

// This will print Labels (column name)
printf("\nstack \t input \t action");

// This will print the initial
// values of stack and input
```

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

// This will Run upto length of input string
for(i = 0; j < c; i++, j++)
{
// Printing action
printf("%s", act);

// Pushing into stack
stk[i] = a[j];
stk[i + 1] = '\0';

// Moving the pointer
a[j]=' ';

// Printing action
printf("\n$%s\t%s$\t", stk, a);

// Call check function ..which will
// check the stack whether its contain
// any production or not
check();
}

// Rechecking last time if contain
// any valid production then it will
// replace otherwise invalid
check();

// if top of the stack is E(starting symbol)
// then it will accept the input
if(stk[0] == 'E' && stk[1] == '\0')
printf("Accept\n");
else //else reject
printf("Reject\n");
}
// This code is contributed by Shubhamsingh10
```

INPUT/OUTPUT :



```
    return ; // return to main
}
```

```
GRAMMAR is -
E->2E2
E->3E3
E->4

stack    input   action
$       32423$  SHIFT
$3       2423$  SHIFT
$32       423$  SHIFT
$324       23$  REDUCE TO E -> 4
$32E       23$  SHIFT
$32E2       3$  REDUCE TO E -> 2E2
$3E         3$  SHIFT
$3E3         $  REDUCE TO E -> 3E3
$E           $  Accept


...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT :

The shift reducing parsing algorithm has been implemented successfully.