RA1911026010033

T Manikanta

# Compiler Design

## Week - 7

Topic :

Shift Reducing Parsing

AIM: To implement Shift Reduce Parser in C.

ALGORITHM:

1. Start the program.
2. Initialize the required variables.
3. Enter the input symbol.
4. Perform the following:
    - for top-of-stack symbol, s, and next input symbol, a Shift x: (x is a STATE number)
    - Push a, then x on the top of the stack
    - Advance ip to point to the next input symbol.
    - Reduce y: (y is a PRODUCTION number)
    - Assume that the production is of the form A→ß
    - Pop 2 * |ß| symbols of the stack.
    - At this point the top of the stack should be a state number, say s'.
    - Push A, then goto of T[s',A] (a state number) on the top of the stack.
    - Output the production A→ß.
5. Print if string is accepted or not.
6. Stop the program.

Code :

```c
#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();
        }
    }
    printf("\n");
}
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';
```

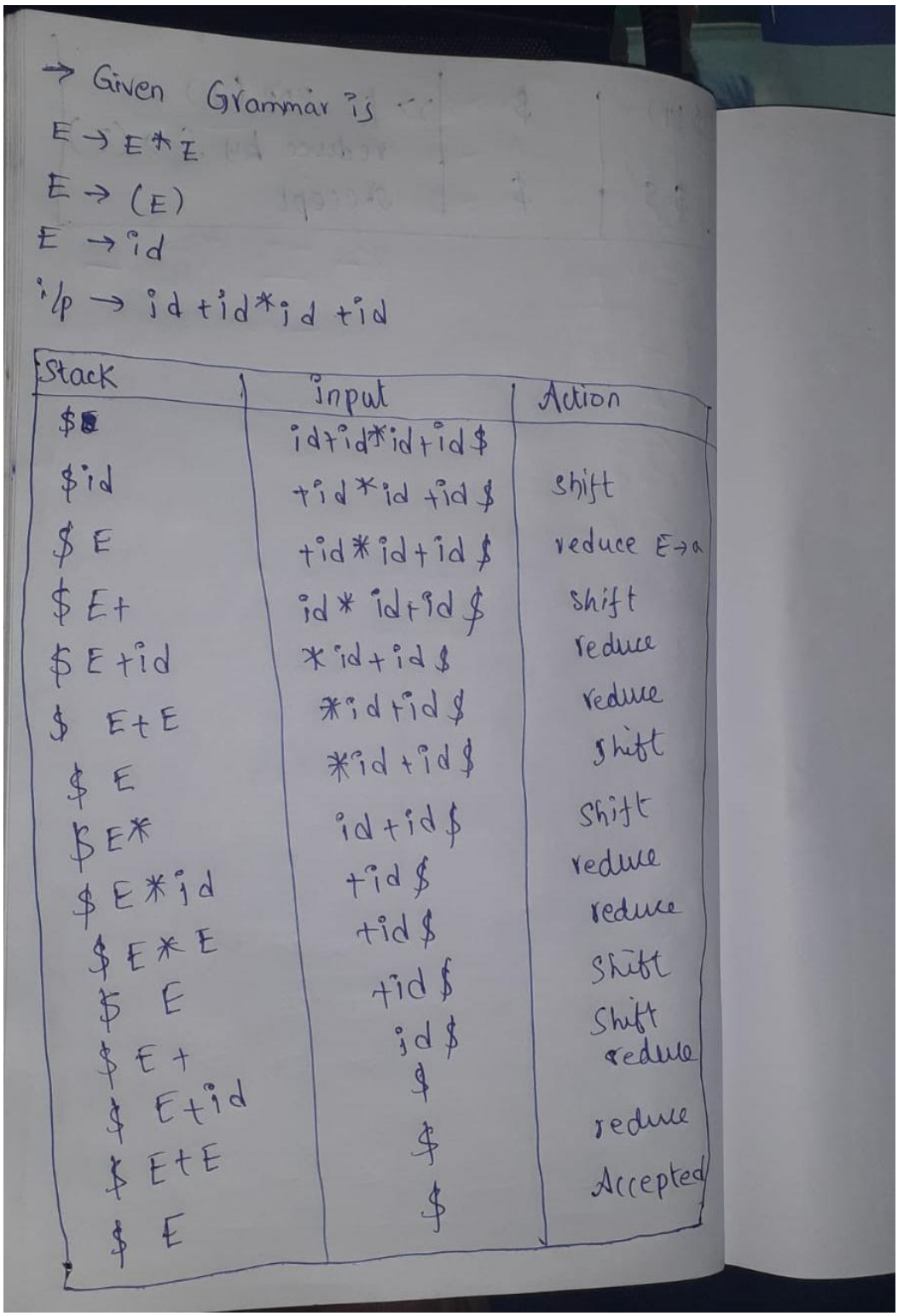
```
            stk[z + 1] = '\0';
            stk[z + 2] = '\0';
            printf("\n$%s\t%s$\t%s", stk, a, ac);
            i = i - 2;
        }
    for (z = 0; z < c; z++)
        if (stk[z] == 'E' && stk[z + 1] == '*' && stk[z + 2] == 'E')
        {
            stk[z] = 'E';
            stk[z +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

```
GRAMMAR is E->E+E
 E->E*E
 E->(E)
 E->id
enter input string
id+id*id+id
stack    input   action

$id       +id*id+id$    SHIFT->id
$E        +id*id+id$    REDUCE TO E
$E+        id*id+id$    SHIFT->symbols
$E+id        *id+id$    SHIFT->id
$E+E         *id+id$    REDUCE TO E
$E           *id+id$    REDUCE TO E
$E*           id+id$    SHIFT->symbols
$E*id           +id$    SHIFT->id
$E*E            +id$    REDUCE TO E
$E              +id$    REDUCE TO E
$E+              id$    SHIFT->symbols
$E+id              $    SHIFT->id
$E+E               $    REDUCE TO E
$E                 $    REDUCE TO E
```

Manual Calculation:



→ Given Grammar is

E → E*E

E → (E)

E → id

i/p → id+id*id+id

| Stack | input | Action |
|---|---|---|
| $ | id+id*id+id$ | |
| $id | +id*id+id$ | shift |
| $ E | +id*id+id$ | reduce E→a |
| $ E+ | id*id+id$ | shift |
| $ E+id | *id+id$ | reduce |
| $ E+E | *id+id$ | reduce |
| $ E | *id+id$ | shift |
| $E* | id+id$ | shift |
| $E*id | +id$ | reduce |
| $E*E | +id$ | reduce |
| $ E | +id$ | shift |
| $ E+ | id$ | shift |
| $ E+id | $ | reduce |
| $ E+E | $ | reduce |
| $ E | $ | Accepted |

Result:

The C Implementation of Shift Reduce Parser was compiled, executed and verified successfully.