RA2011031010046

EXPERIMENT : 6

PREDICTIVE PARSING TABLE

DATE : 22/02/23

AIM : To write a program in C/C++ to implement a predictive parsing table.

PROCEDURE:

1. Start the program.
2. 2. Initialize the required variables.
3. 3. Get the number of coordinates and productions from the user.
4. 4. Perform the following
5. for (each production A → α in G) {
6. for (each terminal a in FIRST(α))
7. add A → α to M[A, a];
8. if (ε is in FIRST(α))
9. for (each symbol b in FOLLOW(A))
10. add A → α to M[A, b];
11. 5. Print the resulting stack.
12. 6. Print if the grammar is accepted or not.
13. 7. Exit the program.

PROGRAM:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main()
{
   char fin[10][20], st[10][20], ft[20][20], fol[20][20];
   int a = 0, e, i, t, b, c, n, k, l = 0, j, s, m, p;
   printf("enter the no. of productions\n");
   scanf("%d", &n);
   printf("enter the productions in a grammar\n");
   for (i = 0; i < n; i++)
      scanf("%s", st[i]);
   for (i = 0; i < n; i++)
      fol[i][0] = '\0';
   for (s = 0; s < n; s++)
   {
      for (i = 0; i < n; i++)
      {
         j = 3;
         l = 0;
         a = 0;
      l1:
```

```
    if (!((st[i][j] > 64) && (st[i][j] < 91)))
    {
        for (m = 0; m < l; m++)
        {
            if (ft[i][m] == st[i][j])
                goto s1;
        }
        ft[i][l] = st[i][j];
        l = l + 1;
    s1:
        j = j + 1;
    }
    else
    {
        if (s > 0)
        {
            while (st[i][j] != st[a][0])
            {
                a++;
            }
            b = 0;
            while (ft[a][b] != '\0')
            {
                for (m = 0; m < l; m++)
                {
                    if (ft[i][m] == ft[a][b])
                        goto s2;
                }
                ft[i][l] = ft[a][b];
                l = l + 1;
            s2:
                b = b + 1;
            }
        }
    }
    while (st[i][j] != '\0')
    {
        if (st[i][j] == '|')
        {
            j = j + 1;
            goto l1;
        }
        j = j + 1;
    }
    ft[i][l] = '\0';
}
```

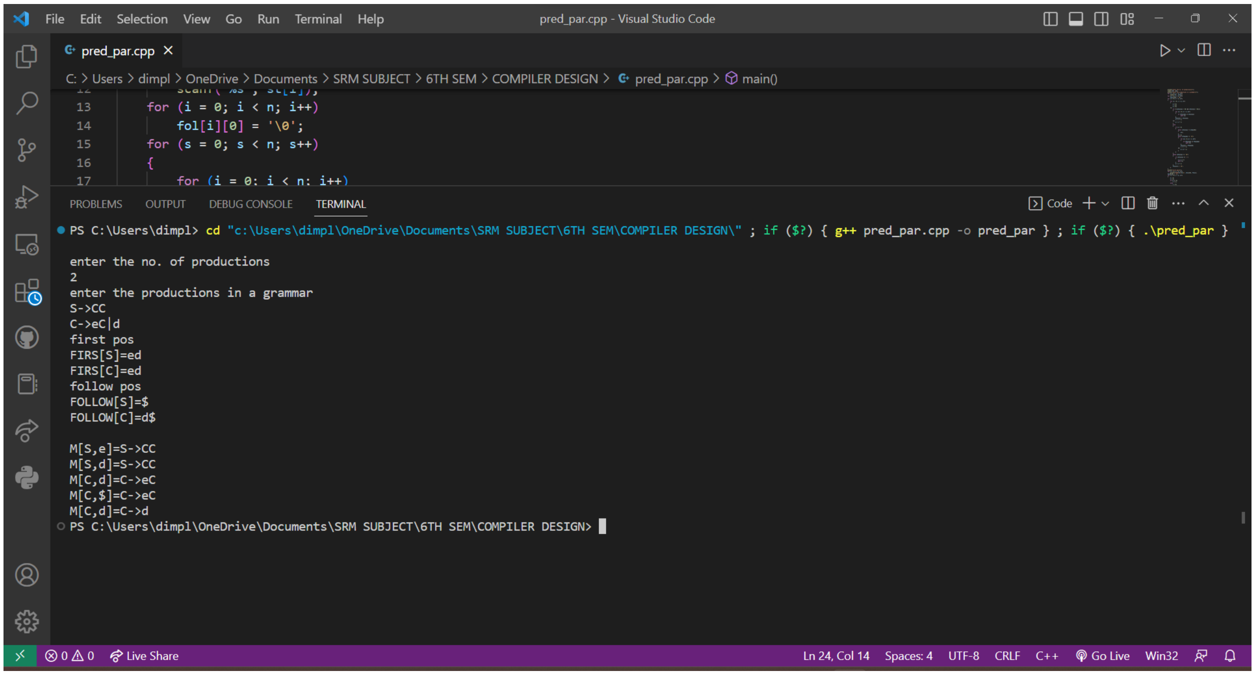
```
}
printf("first pos\n");
for (i = 0; i < n; i++)
    printf("FIRS[%c]=%s\n", st[i][0], ft[i]);
fol[0][0] = '$';
for (i = 0; i < n; i++)
{
    k = 0;
    j = 3;
    if (i == 0)
        l = 1;
    else
        l = 0;
k1:
    while ((st[i][0] != st[k][j]) && (k < n))
    {
        if (st[k][j] == '\0')
        {
            k++;
            j = 2;
        }
        j++;
    }
    j = j + 1;
    if (st[i][0] == st[k][j - 1])
    {
        if ((st[k][j] != '|') && (st[k][j] != '\0'))
        {
            a = 0;
            if (!((st[k][j] > 64) && (st[k][j] < 91)))
            {
                for (m = 0; m < l; m++)
                {
                    if (fol[i][m] == st[k][j])
                        goto q3;
                }
                fol[i][l] = st[k][j];
                l++;
            q3:
                p++;
            }
            else
            {
                while (st[k][j] != st[a][0])
                {
                    a++;
        }
        p = 0;
        while (ft[a][p] != '\0')
        {
            if (ft[a][p] != 'e')
            {
                for (m = 0; m < l; m++)
                {
                    if (fol[i][m] == ft[a][p])
                        goto q2;
                }
                fol[i][l] = ft[a][p];
                l = l + 1;
            }
            else
                e = 1;
        q2:
            p++;
        }
        if (e == 1)
        {
            e = 0;
            goto a1;
        }
    }
}
else
{
a1:
    c = 0;
    a = 0;
    while (st[k][0] != st[a][0])
    {
        a++;
    }
    while ((fol[a][c] != '\0') && (st[a][0] != st[i][0]))
    {
        for (m = 0; m < l; m++)
        {
            if (fol[i][m] == fol[a][c])
                goto q1;
        }
        fol[i][l] = fol[a][c];
        l++;
    q1:
        c++;
```

```
                }
            }
            goto k1;
        }
        fol[i][l] = '\0';
    }
    printf("follow pos\n");
    for (i = 0; i < n; i++)
        printf("FOLLOW[%c]=%s\n", st[i][0], fol[i]);
    printf("\n");
    s = 0;
    for (i = 0; i < n; i++)
    {
        j = 3;
        while (st[i][j] != '\0')
        {
            if ((st[i][j - 1] == '|') || (j == 3))
            {
                for (p = 0; p <= 2; p++)
                {
                    fin[s][p] = st[i][p];
                }
                t = j;
                for (p = 3; ((st[i][j] != '|') && (st[i][j] != '\0')); p++)
                {
                    fin[s][p] = st[i][j];
                    j++;
                }
                fin[s][p] = '\0';
                if (st[i][t] == 'e')
                {
                    b = 0;
                    a = 0;
                    while (st[a][0] != st[i][0])
                    {
                        a++;
                    }
                    while (fol[a][b] != '\0')
                    {
                        printf("M[%c,%c]=%s\n", st[i][0], fol[a][b], fin[s]);
                        b++;
                    }
                }
                else if (!((st[i][t] > 64) && (st[i][t] < 91)))
                    printf("M[%c,%c]=%s\n", st[i][0], st[i][t], fin[s]);
                else
                {
                    b = 0;
                    a = 0;
                    while (st[a][0] != st[i][3])
                    {
                        a++;
                    }
                    while (ft[a][b] != '\0')
                    {
                        printf("M[%c,%c]=%s\n", st[i][0], ft[a][b], fin[s]);
                        b++;
                    }
                }
                s++;
            }
            if (st[i][j] == '|')
                j++;
        }
    }
    return 0;
}
```

INPUT:

```
Enter the no. of nonterminals
2
Enter the productions in a grammar
S->CC
C->eC | d
```

OUTPUT:



```
        scanf("%s", st[i]);
    for (i = 0; i < n; i++)
        fol[i][0] = '\0';
    for (s = 0; s < n; s++)
    {
        for (i = 0; i < n; i++)
```

```
PS C:\Users\dimpl> cd "c:\Users\dimpl\OneDrive\Documents\SRM SUBJECT\6TH SEM\COMPILER DESIGN\" ; if ($?) { g++ pred_par.cpp -o pred_par } ; if ($?) { .\pred_par }

enter the no. of productions
2
enter the productions in a grammar
S->CC
C->eC|d
first pos
FIRS[S]=ed
FIRS[C]=ed
follow pos
FOLLOW[S]=$
FOLLOW[C]=d$

M[S,e]=S->CC
M[S,d]=S->CC
M[C,d]=C->eC
M[C,$]=C->eC
M[C,d]=C->d
PS C:\Users\dimpl\OneDrive\Documents\SRM SUBJECT\6TH SEM\COMPILER DESIGN>
```

RESULT:

Thus, we have successfully implemented the predictive parsing table.