College Of Engineering Trivandrum

Compiler Design Lab

CS431

Submitted By:
Neethu S
S6 CSE Roll No:42

TVE18CS043

Department of Computer Science

January 4, 2021

Contents

Cycle 2 Experiment 1

1 Operator Precedence Parser

1.1 Aim

Develop an operator precedence parser for a given language.

1.2 Theory

Operator Precedence Parser constructed for operator precedence grammar. Operator precedence grammar is a grammar that doesn't contain epsilon productions and does not contain two adjacent non-terminals on R.H.S. of any production. Operator precedence grammar is provided with precedence rules. Operator Precedence grammar could be either ambiguous or unambiguous.

1.3 Algorithm

```
1.Start
2.Read a string from a user.
3.Add $ at both the ends.
4.Scan the input string from left until '>' is encountered.
5.Scan towards left over all equal precedence until first leftmost '<'
    is encountered.
6.Everything between '<' and '>' is handled.
7.While reaching '$' , parsing is successful and stops.
8.Display the operator precedence table
9.Stop
```

1.4 Code

```
#include<stdio.h>

void main() {
  char stack[20], ip[20], opt[10][10][1], ter[10];
  int i, j, k, n, top = 0, col, row;

  for (i = 0; i < 10; i++) {
    stack[i] = NULL;
    ip[i] = NULL;
    for (j = 0; j < 10; j++) {
      opt[i][j][1] = NULL;
    }
  }
  printf("Enter the no.of terminals :\n");
  scanf("%d", & n);
  printf("\nEnter the terminals :\n");
  scanf("%s", & ter);
```

```
printf("\nEnter the table values :\n");
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++) {
    printf("Enter the value for %c %c:", ter[i], ter[j]);
    scanf("%s", opt[i][j]);
  }
}
printf("\n**** OPERATOR PRECEDENCE TABLE ****\n");
for (i = 0; i < n; i++) {
  printf("\t%c", ter[i]);
}
printf("\n");
for (i = 0; i < n; i++) {
  printf("\n%c", ter[i]);
  for (j = 0; j < n; j++) {
    printf("\t%c", opt[i][j][0]);
  }
}
stack[top] = '$';
printf("\nEnter the input string:");
scanf("%s", ip);
i = 0;
printf("\nSTACK\t\t\tINPUT STRING\t\t\tACTION\n");
printf("\n%s\t\t\t%s\t\t\t", stack, ip);
while (i <= strlen(ip)) {
  for (k = 0; k < n; k++) {
    if (stack[top] == ter[k])
      col = k;
    if (ip[i] == ter[k])
      row = k;
  }
  if ((stack[top] == '$') && (ip[i] == '$')) {
    printf("String is accepted\n");
    break;
  } else if ((opt[col][row][0] == '<') || (opt[col][row][0] == '=')) {
    stack[++top] = opt[col][row][0];
    stack[++top] = ip[i];
    printf("Shift %c", ip[i]);
    i++;
  } else {
    if (opt[col][row][0] == '>') {
      while (stack[top] != '<') {
        --top;
      }
      top = top - 1;
      printf("Reduce");
    } else {
      printf("\nString is not accepted");
      break;
    }
  }
  printf("\n");
```

```
    for (k = 0; k <= top; k++) {
      printf("%c", stack[k]);
    }
    printf("\t\t\t");
    for (k = i; k < strlen(ip); k++) {
      printf("%c", ip[k]);
    }
    printf("\t\t\t");
  }
}
```

1.5 Output

```
s1842@administrator-rusa:~/s7$ ./a.out
Enter the no.of terminals :
4

Enter the terminals :
+*i$

Enter the table values :
Enter the value for + +:>
Enter the value for + *:<
Enter the value for + i:<
Enter the value for + $:>
Enter the value for * +:>
Enter the value for * *:>
Enter the value for * i:<
Enter the value for * $:>
Enter the value for i +:>
Enter the value for i *:>
Enter the value for i i:=
Enter the value for i $:>
Enter the value for $ +:<
Enter the value for $ *:<
Enter the value for $ i:<
Enter the value for $ $:A

**** OPERATOR PRECEDENCE TABLE ****
        +       *       i       $

+       >       <       <       >
*       >       >       <       >
i       >       >       =       >
$       <       <       <       A
Enter the input string:i+i*i$

STACK                   INPUT STRING                    ACTION

$                       i+i*i$                  Shift i
$<i                     +i*i$                   Reduce
$                       +i*i$                   Shift +
$<+                     i*i$                    Shift i
$<+<i                   *i$                     Reduce
$<+                     *i$                     Shift *
$<+<*                   i$                      Shift i
$<+<*<i                 $                       Reduce
$<+<*                   $                       Reduce
$<+                     $                       Reduce
$                       $                       String is accepted
s1842@administrator-rusa:~/s7$
```

1.6 Result

Implemented the program for the Operator Precedence Parser using C language in Ubuntu 20.04 with kernel and the above outputs were obtained.