19CSE401 Compiler

Design Lab

Anindita Das Badhan

CH.EN.U4CSE22180

CSE-B, 4th year

Lab Exercises - 03

Aim: To implement LL(1) parsing using C program.

Code:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char s[20], stack[20];

// Parsing table for predictive parsing (non-terminal x terminal)
char *m[5][6] = {
    /* i     +      *      (      )      $  */
    {"tb",  "",     "",    "tb",  "",     ""},   // e
    {"",    "+tb",  "",    "",    "n",    "n"},  // b
    {"fc",  "",     "",    "fc",  "",     ""},   // t
    {"",    "n",    "*fc", "",    "n",    "n"},  // c
    {"i",    "",    "",    "(e)", "",     ""}    // f
};

int size[5][6] = {
    {2, 0, 0, 2, 0, 0},  // e
    {0, 3, 0, 0, 1, 1},  // b
    {2, 0, 0, 2, 0, 0},  // t
    {0, 1, 3, 0, 1, 1},  // c
    {1, 0, 0, 3, 0, 0}   // f
};

int main()
{
    int i, j, k;
    int str1, str2;
    int n;

    printf("\nEnter the input string: ");
    scanf("%s", s);

    strcat(s, "$");
    n = strlen(s);

    stack[0] = '$';
    stack[1] = 'e';
    i = 1; // top of stack index
    j = 0; // input pointer index

    printf("\nStack\tInput\n");
    printf("______\n\n");

    // Continue until BOTH stack top and input symbol are '$'
```

```
// Continue until BOTH stack top and input symbol are '$'
while (!(stack[i] == '$' && s[j] == '$')) {
    if (stack[i] == s[j]) {
        // Match terminal
        i--;
        j++;
    } else {
        // Get row for non-terminal on top of stack
        switch (stack[i]) {
            case 'e': str1 = 0; break;
            case 'b': str1 = 1; break;
            case 't': str1 = 2; break;
            case 'c': str1 = 3; break;
            case 'f': str1 = 4; break;
            default:
                printf("\nERROR: Invalid non-terminal %c\n", stack[i]);
                exit(0);
        }

        // Get column for current input symbol
        switch (s[j]) {
            case 'i': str2 = 0; break;
            case '+': str2 = 1; break;
            case '*': str2 = 2; break;
            case '(': str2 = 3; break;
            case ')': str2 = 4; break;
            case '$': str2 = 5; break;
            default:
                printf("\nERROR: Invalid input symbol %c\n", s[j]);
                exit(0);
        }

        if (m[str1][str2][0] == '\0') {
            printf("\nERROR: No rule for [%c][%c]\n", stack[i], s[j]);
            exit(0);
        } else if (m[str1][str2][0] == 'n') {
            // 'n' means epsilon production (pop)
            i--;
        } else if (m[str1][str2][0] == 'i') {
            // 'i' means push 'i' on stack
            stack[i] = 'i';
        } else {
            // Push RHS of production in reverse order
            for (k = size[str1][str2] - 1; k >= 0; k--) {
```

Output:

```
ubuntu:~$ gcc third.c
ubuntu:~$ ./a.out
```

```
Enter the input string: i*i+i

Stack   Input
______

$bt     i*i+i$
$bcf    i*i+i$
$bci    i*i+i$
$bc     *i+i$
$bcf*   *i+i$
$bcf    i+i$
$bci    i+i$
$bc     +i$
$b      +i$
$bt+    +i$
$bt     i$
$bcf    i$
$bci    i$
$bc     $
$b      $
$       $

SUCCESS
ubuntu:~$
```

Results:

The program to implement left factoring and left recursion has been successfully executed.