College of Engineering Trivandrum

Compiler Design Lab

Abhishek Manoharan
S7 CSE Roll No:2

TVE17CS002

Department of Computer Science

October 15, 2020

Exp 11

1 Shift Reduce Parser

1.1 Aim

Construct a Shift Reduce Parser for a given language.

1.2 Theory

Shift Reduce parser

Shift Reduce parser attempts for the construction of parse in a similar manner as done in bottom up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). A more general form of shift reduce parser is LR parser.
This parser requires some data structures i.e.

- A input buffer for storing the input string.
- A stack for storing and accessing the production rules.

Basic Operations –

1. **Shift**: This involves moving of symbols from input buffer onto the stack.
2. **Reduce:** If the handle appears on top of the stack then, its reduction by using appropriate production rule is done i.e. RHS of production rule is popped out of stack and LHS of production rule is pushed onto the stack.
3. **Accept:** If only start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accept action is obtained, it is means successful parsing is done.
4. **Error:** This is the situation in which the parser can neither perform shift action nor reduce action and not even accept action.

1.3 Algorithm

Algorithm 1: Algorithm for Shift Reduce parser

```
loop forever :
    for top - of - stack symbol , s , and next input symbol , a case
    action of T [s , a ]
        shift x : ( x is a STATE number )
            push a , then x on the top of the stack and
            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 == > beta
            pop 2 * | beta | 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 == > beta .
        accept :
            return --- a successful parse .
        default :
            error --- the input string is not in the language .
```

1.4 Code

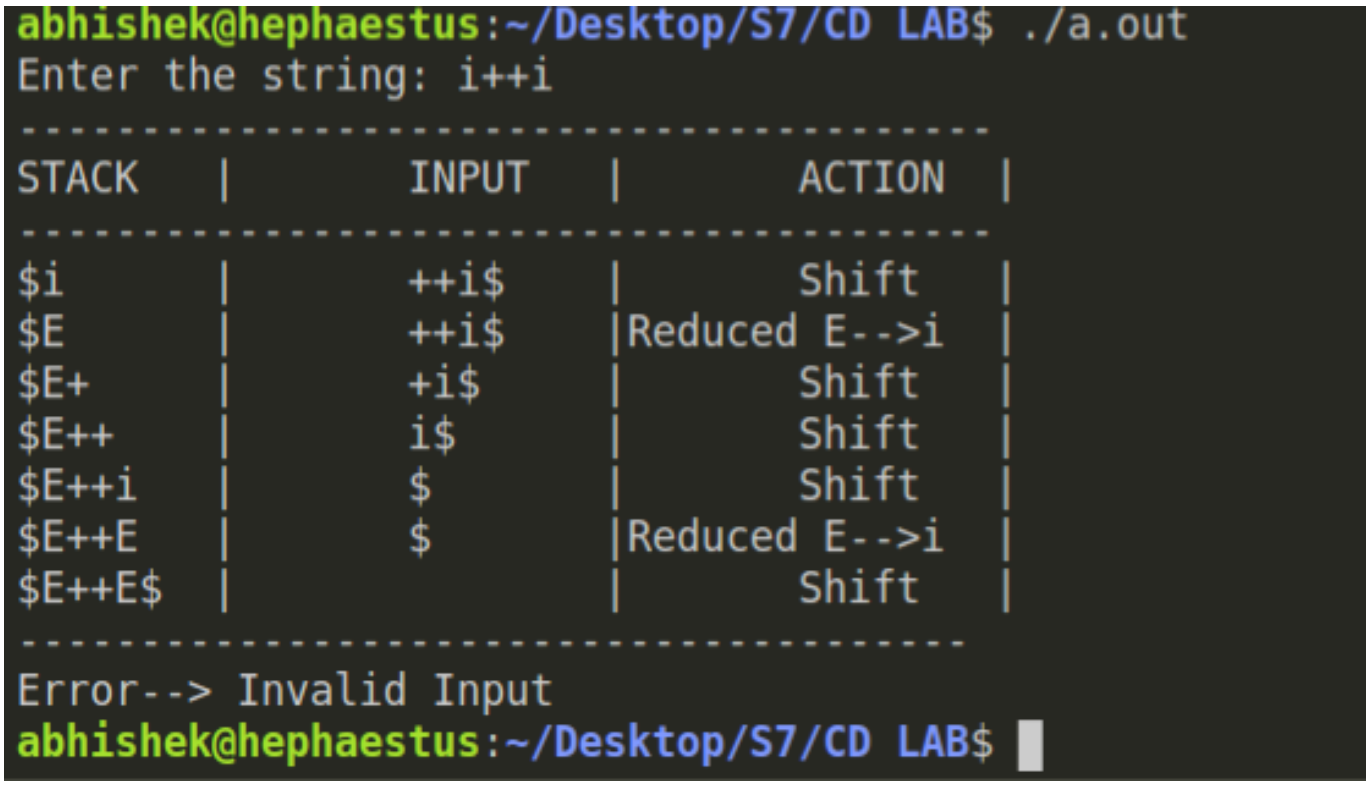
```cpp
#include <bits/stdc++.h>
using namespace std;
void print_stack(stack<char> check)
{
    string s = "";
    while (!check.empty())
    {
        s = check.top() + s;
        check.pop();
    }
    cout << s;
}
string last_3(stack<char> check)
{
    string res = "";
    for (int i = 0; i < 3; ++i)
    {
        res = check.top() + res;
        check.pop();
    }
    return res;
}

int main()
{
    vector<char> lhs = {'E'};
    unordered_set<string> rhs = {"E+E", "(E)", "i", "E*E"};
    cout << "Enter the string: ";
    string s;
    cin >> s;
    s += "$";
    int n = s.size(), count = 1, i = 0;
    stack<char> SR;
    char a, b, c;
    SR.push('$');
    cout << "----------------------------------------" << endl;
    cout << "STACK\t|\tINPUT\t|\tACTION\t|" << endl;
    cout << "----------------------------------------" << endl;
    while (true)
    {
        if (count >= 3)
        {
            string over = last_3(SR);
            //cout << "string found is " << over << endl;
            if (over == "$E$")
            {
                cout << "-------------------------------------" << endl;
                cout << "Parsing successfully finished, valid input" << endl;
                break;
            }
            if (rhs.find(over) != rhs.end())
            {
                SR.pop();
                SR.pop();
                SR.pop();
                SR.push('E');
                print_stack(SR);
                cout << "\t|\t";
                cout << s.substr(i, n - i) << "\t|Reduced E-->" << over << "|" << endl;
                // cout << "-------------------------------------" << endl;
                count -= 2;
                continue;
            }
        }
        if (SR.top() == 'i')
        {
            SR.pop();
            SR.push('E');
            print_stack(SR);
            cout << "\t|\t";
            cout << s.substr(i, n - i) << "\t|Reduced E-->i\t|" << endl;
            //cout << "-------------------------------------" << endl;
            continue;
        }
```

```
        if (i >= n)
        {
            cout << "---------------------------------------" << endl;
            cout << "Error--> Invalid Input" << endl;
            break;
        }
        SR.push(s[i]);
        print_stack(SR);
        cout << "\t|\t";
        cout << s.substr(i + 1, n - i) << "\t|\tShift\t|" << endl;
        //cout << "---------------------------------------" << endl;
        count++;
        i++;
    }
    return 0;
}
```

1.5 Output

```
abhishek@hephaestus:~/Desktop/S7/CD LAB$ ./a.out
Enter the string: i+i
----------------------------------------
STACK   |       INPUT   |       ACTION  |
----------------------------------------
$i      |       +i$     |       Shift   |
$E      |       +i$     |Reduced E-->i  |
$E+     |       i$      |       Shift   |
$E+i    |       $       |       Shift   |
$E+E    |       $       |Reduced E-->i  |
$E      |       $       |Reduced E-->E+E|
$E$     |               |       Shift   |
----------------------------------------
Parsing successfully finished, valid input
abhishek@hephaestus:~/Desktop/S7/CD LAB$ g++ shift-reduce.cpp
abhishek@hephaestus:~/Desktop/S7/CD LAB$ ./a.out
Enter the string: i*(i+i)
----------------------------------------
STACK   |       INPUT   |       ACTION  |
----------------------------------------
$i      |       *(i+i)$ |       Shift   |
$E      |       *(i+i)$ |Reduced E-->i  |
$E*     |       (i+i)$  |       Shift   |
$E*(    |       i+i)$   |       Shift   |
$E*(i   |       +i)$    |       Shift   |
$E*(E   |       +i)$    |Reduced E-->i  |
$E*(E+  |       i)$     |       Shift   |
$E*(E+i |       )$      |       Shift   |
$E*(E+E |       )$      |Reduced E-->i  |
$E*(E   |       )$      |Reduced E-->E+E|
$E*(E)  |       $       |       Shift   |
$E*E    |       $       |Reduced E-->(E)|
$E      |       $       |Reduced E-->E*E|
$E$     |               |       Shift   |
----------------------------------------
Parsing successfully finished, valid input
abhishek@hephaestus:~/Desktop/S7/CD LAB$
```




```
abhishek@hephaestus:~/Desktop/S7/CD LAB$ g++ shift-reduce.cpp
abhishek@hephaestus:~/Desktop/S7/CD LAB$ ./a.out
Enter the string: i+i
---------------------------------------
STACK   |       INPUT   |       ACTION  |
---------------------------------------
$i      |       +i$     |       Shift   |
$E      |       +i$     |Reduced E-->i  |
$E+     |       i$      |       Shift   |
$E+i    |       $       |       Shift   |
$E+E    |       $       |Reduced E-->i  |
$E      |       $       |Reduced E-->E+E|
$E$     |               |       Shift   |
---------------------------------------
Parsing successfully finished, valid input
abhishek@hephaestus:~/Desktop/S7/CD LAB$ g++ shift-reduce.cpp
abhishek@hephaestus:~/Desktop/S7/CD LAB$ ./a.out
Enter the string: i*(i+i)
---------------------------------------
STACK   |       INPUT   |       ACTION  |
---------------------------------------
$i      |       *(i+i)$ |       Shift   |
$E      |       *(i+i)$ |Reduced E-->i  |
$E*     |       (i+i)$  |       Shift   |
$E*(    |       i+i)$   |       Shift   |
$E*(i   |       +i)$    |       Shift   |
$E*(E   |       +i)$    |Reduced E-->i  |
$E*(E+  |       i)$     |       Shift   |
$E*(E+i |       )$      |       Shift   |
$E*(E+E |       )$      |Reduced E-->i  |
$E*(E   |       )$      |Reduced E-->E+E|
$E*(E)  |       $       |       Shift   |
$E*E    |       $       |Reduced E-->(E)|
$E      |       $       |Reduced E-->E*E|
$E$     |               |       Shift   |
---------------------------------------
```

```
Parsing successfully finished, valid input
abhishek@hephaestus:~/Desktop/S7/CD LAB$ ./a.out
Enter the string: i++i
---------------------------------------
STACK   |       INPUT   |       ACTION  |
---------------------------------------
$i      |       ++i$    |       Shift   |
$E      |       ++i$    |Reduced E-->i  |
$E+     |       +i$     |       Shift   |
$E++    |       i$      |       Shift   |
$E++i   |       $       |       Shift   |
$E++E   |       $       |Reduced E-->i  |
$E++E$  |               |       Shift   |
---------------------------------------
Error--> Invalid Input
abhishek@hephaestus:~/Desktop/S7/CD LAB$
```

1.6 Result

Implemented the program to construct a Shift Reduce parser. It was compiled using g++ version 9.3.0, and executed in Ubuntu 20.04 and the above output was obtained.