Practical - 7

# Aim:

Implement a program for Recursive Descent Parser for the given grammar. E →T + E | T

T → F \* T | F F→ id

# Code

# #include <iostream>

# #include <string>

# using namespace std;

# string input;

# size\_t idx = 0;

# char currentChar;

# void parseE();

# void parseT();

# void parseF();

# void match(char expectedChar);

# int main() {

# cout << "Enter the input string: ";

# cin >> input;

# 

# idx = 0;

# currentChar = input[idx];

# 

# try {

# cout << "Parsing steps:" << endl;

# parseE();

# if (idx == input.length()) {

# cout << "\nParsing successful." << endl;

# } else {

# cout << "\nError: Unexpected input after parsing." << endl;

# }

# } catch (const exception &e) {

# cout << e.what() << endl;

# }

# 

# return 0;

# }

# // E → T + E | T

# void parseE() {

# parseT();

# if (currentChar == '+') {

# cout << "E -> T + E" << endl;

# match('+');

# parseE();

# }

# }

# // T → F \* T | F

# void parseT() {

# parseF();

# if (currentChar == '\*') {

# cout << "T -> F \* T" << endl;

# match('\*');

# parseT();

# }

# }

# // F → id

# void parseF() {

# if (currentChar == 'i') {

# cout << "F -> id" << endl;

# match('i');

# match('d');

# } else {

# throw runtime\_error("Error: Expected id but found '" + string(1, currentChar) + "'");

# }

# }

# void match(char expectedChar) {

# if (currentChar == expectedChar) {

# idx++;

# if (idx < input.length()) {

# currentChar = input[idx];

# }

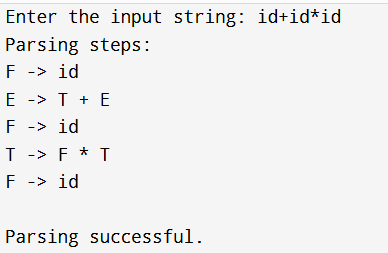
# } else {

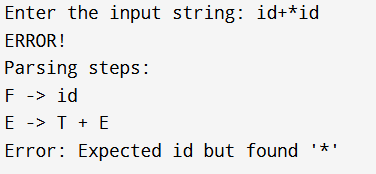
# throw runtime\_error("Error: Expected '" + string(1, expectedChar) + "' but found '" + string(1, currentChar) + "'");

# }

# }

# Output:



****

# Conclusion:

In this practical I implemented a program for Recursive Descent Parser for the given grammar. This implementation handles basic arithmetic expressions involving identifiers, addition, and multiplication. I check different string which is parse or not from this parser.