# UIT1601 - Compiler Design

Implementation of a Parser

Name: M Badri Narayanan

**Reg No:** 185002018

Semester: VI

**Date:** April 3, 2021

## Recursive Descent Parser

#### Grammar

- $E \rightarrow TE'$
- $E^{'} \rightarrow +TE^{'} | \epsilon$
- ullet T o FT'
- $\bullet \ T^{'} \to *FT^{'}|\ \epsilon$
- $id \rightarrow FT'$

### **Program**

## Header File

```
typedef char string[30];
typedef enum
    FAIL,
    SUCCESS
} Result;
void indent(const int level)
    for (int i = 0; i < level; i++)</pre>
        printf(" ");
Result T(string, int *, int);
Result E(string, int *, int);
Result Eprime(string, int *, int);
Result Tprime(string, int *, int);
Result F(string, int *, int);
Result recursiveParser(string input)
    int index = 0;
    return E(input, &index, 0);
```

```
Result E(string input, int *idx, int depth)
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("E()\n");
    Result result = T(input, idx, depth + 1);
    if (result == FAIL)
        return FAIL;
    result = Eprime(input, idx, depth + 1);
    return result;
}
Result Eprime(string input, int *idx, int depth)
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("E'()\n");
    Result result;
    int current_idx = *idx;
    if (input[*idx] == '+')
    {
        indent(depth);
        printf("Non terminal \'+\' found!\n");
        (*idx) = (*idx) + 1;
        result = T(input, idx, depth + 1);
        if (result == FAIL)
            return FAIL;
        result = Eprime(input, idx, depth + 1);
        if (result == SUCCESS)
            return SUCCESS;
    }
    *idx = current_idx;
    indent(depth);
    printf("Non terminal \'+\' found!\n");
    return SUCCESS;
Result T(string input, int *idx, int depth)
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("T()\n");
    Result result = F(input, idx, depth + 1);
    if (result == FAIL)
        return FAIL;
    result = Tprime(input, idx, depth + 1);
    return result;
}
Result Tprime(string input, int *idx, int depth)
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("T'()\n");
```

```
Result result;
    int current_idx = *idx;
    if (input[*idx] == '*')
    {
        indent(depth);
        printf("Non terminal \'*\' found!\n");
        (*idx) = (*idx) + 1;
        result = F(input, idx, depth + 1);
        if (result == FAIL)
            return FAIL;
        result = Tprime(input, idx, depth + 1);
        if (result == SUCCESS)
            return SUCCESS;
    *idx = current_idx;
    indent(depth);
    printf("Non terminal \'*\' not found!\n");
    return SUCCESS;
}
Result F(string input, int *idx, int depth)
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("F()\n");
    int current_idx = *idx;
    if (input[*idx] == 'i' && input[*idx + 1] == 'd')
        (*idx) += 2;
        indent(depth);
        printf("Non terminal \'id\' found!\n");
        return SUCCESS;
    else if (input[*idx] == '(')
        indent(depth);
        printf("Non terminal \'(\' found!\n");
        Result result = E(input, idx, depth + 1);
        if (result == FAIL)
            return FAIL;
        if (input[*idx] == ')')
            return SUCCESS;
    return FAIL;
}
```

### Main Program

```
#include <stdio.h>
#include <string.h>
#include "Procedure.h"
void printResult(Result);
int main(void)
   printf("\n\nRecursive Descent Parser \n\n");
    string input;
    int opt = -1;
    while (opt != 0)
       printf("Enter the input string: ");
       scanf("%s", input);
       printResult(recursiveParser(input));
       printf("----\n\n");
       printf("Do you want to continue 1/0: ");
       scanf("%d", &opt);
   printf("\n\n End \n\n");
}
void printResult(Result result)
    if (result == SUCCESS)
       printf("Given string is accepted!\n");
    else
       printf("Given string is not accepted!\n");
}
```

## Output

Figure 1: Recursive Descent Parser Output

```
badri@DESKTOP-IV11987: /mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/Parser
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/Parser$ gcc Parser.c -o Parser
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/Parser$ ./Parser
Recursive Descent Parser
Enter the input string: id+id*id
E()
T()
           F()
Non terminal 'id' found!
           T'()
Non terminal '*' not found!
     E'()
Non terminal '+' found!
          tei.
T()
F()
                Non terminal 'id' found!
T'()
Non terminal '*' found!
                F()
Non terminal 'id' found!
Non terminal '*' not found!
           E'()
Non terminal '+' found!
Given string is accepted!
Do you want to continue 1/0: 1
Enter the input string: id+*id
Enc.
E()
T()
           F()
Non terminal 'id' found!
T'()
           Non terminal '*' not found!
     E'()
Non terminal '+' found!
T()
F()
F()
not accepted
Given string is not accepted!
Do you want to continue 1/0: 0
The End
badri@DESKTOP-IV11987:/mnt/c/Users/badri/Desktop/SemVI/Assignments/CompilerDesign/Parser$ 💂
```