**Aim**

To develop a **C program** to compute the **FIRST( )** set of a given **context-free grammar (CFG)**, which is useful for **predictive parsing**.

**Program**

c

CopyEdit

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX 10

char productions[MAX][MAX]; // Stores grammar productions

char first[MAX][MAX]; // Stores FIRST sets

int numProductions; // Number of productions

// Function to check if a character is a terminal

int isTerminal(char c) {

return !(c >= 'A' && c <= 'Z');

}

// Function to find FIRST of a given non-terminal

void findFirst(char symbol, int index) {

// If it is a terminal, FIRST(symbol) = {symbol}

if (isTerminal(symbol)) {

first[index][0] = symbol;

first[index][1] = '\0';

return;

}

// Scan all productions

for (int i = 0; i < numProductions; i++) {

if (productions[i][0] == symbol) { // If production starts with the symbol

int j = 2; // Start from RHS of production (A -> α)

while (productions[i][j] != '\0') {

char nextSymbol = productions[i][j];

if (isTerminal(nextSymbol)) {

// Add terminal to FIRST set

strncat(first[index], &nextSymbol, 1);

break;

} else {

// Recursively find FIRST of non-terminal

for (int k = 0; k < numProductions; k++) {

if (productions[k][0] == nextSymbol) {

findFirst(nextSymbol, k);

strcat(first[index], first[k]); // Add FIRST(Y) to FIRST(X)

}

}

}

// If FIRST(nextSymbol) contains ε, check next symbol

if (strchr(first[index], 'ε') == NULL) {

break;

}

j++;

}

}

}

}

// Function to print FIRST sets

void printFirst() {

printf("\nFIRST Sets:\n");

for (int i = 0; i < numProductions; i++) {

printf("FIRST(%c) = { ", productions[i][0]);

for (int j = 0; j < strlen(first[i]); j++) {

printf("%c ", first[i][j]);

}

printf("}\n");

}

}

int main() {

printf("Enter number of productions: ");

scanf("%d", &numProductions);

printf("Enter the productions (e.g., A->BC):\n");

for (int i = 0; i < numProductions; i++) {

scanf("%s", productions[i]);

}

// Compute FIRST sets

for (int i = 0; i < numProductions; i++) {

findFirst(productions[i][0], i);

}

// Print FIRST sets

printFirst();

return 0;

}

**Example Input and Output**

**Input**

rust

CopyEdit

Enter number of productions: 3

Enter the productions (e.g., A->BC):

E->TR

R->+TR|ε

T->F

**Output**

r

CopyEdit

FIRST Sets:

FIRST(E) = { T }

FIRST(R) = { + ε }

FIRST(T) = { F }