**Aim**

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

**C Program to Compute FOLLOW( )**

CopyEdit

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX 10

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

char follow[MAX][MAX]; // Stores FOLLOW sets

int numProductions; // Number of productions

char startSymbol; // Start symbol of grammar

// Function to check if a character is a terminal

int isTerminal(char c) {

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

}

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

void findFollow(char symbol, int index) {

// If it is the start symbol, FOLLOW(start) includes '$'

if (symbol == startSymbol) {

strcat(follow[index], "$");

}

// Scan all productions

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

int j = 2; // Skip the "X->" part

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

if (productions[i][j] == symbol) {

int k = j + 1;

// Case: A → αBβ (add FIRST(β) to FOLLOW(B))

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

if (isTerminal(productions[i][k])) {

strncat(follow[index], &productions[i][k], 1);

break;

} else {

// If non-terminal, check its FIRST

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

if (productions[m][0] == productions[i][k]) {

strcat(follow[index], follow[m]);

}

}

}

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

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

break;

}

k++;

}

// Case: A → αB (add FOLLOW(A) to FOLLOW(B))

if (productions[i][k] == '\0' && productions[i][0] != symbol) {

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

if (productions[m][0] == productions[i][0]) {

strcat(follow[index], follow[m]);

}

}

}

}

j++;

}

}

}

// Function to print FOLLOW sets

void printFollow() {

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

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

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

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

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

}

printf("}\n");

}

}

int main() {

printf("Enter the number of productions: ");

scanf("%d", &numProductions);

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

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

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

}

// Set start symbol (first production's LHS)

startSymbol = productions[0][0];

// Compute FOLLOW sets

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

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

}

// Print FOLLOW sets

printFollow();

return 0;

}

**Example Input and Output**

**Input**

rust

CopyEdit

Enter the number of productions: 3

Enter the productions (e.g., S->AaAb):

S->AaAb

A->ε

B->ε

**Output**

ruby

CopyEdit