# Program 16

Write a C Program to Generate the Three address code representation for the given input statement.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX 100

int tempVarCount = 1; // Temporary variable counter

// Function to generate TAC for an expression

void generateTAC(char expression[MAX]) {

   char tokens[MAX][MAX]; // Token storage

   int tokenCount = 0;

   char \*token = strtok(expression, " "); // Tokenize input

   // Tokenizing the input expression

   while (token != NULL) {

       strcpy(tokens[tokenCount++], token);

       token = strtok(NULL, " ");

   }

   printf("\nGenerated Three-Address Code:\n");

   char tempVars[MAX][MAX]; // Store temporary variables

   int tempIndex = 0;

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

       if (strcmp(tokens[i], "\*") == 0 || strcmp(tokens[i], "/") == 0) {

           // Multiplication and division have higher precedence

           printf("t%d = %s %s %s\n", tempVarCount, tokens[i - 1], tokens[i], tokens[i + 1]);

           sprintf(tempVars[tempIndex], "t%d", tempVarCount++);

           strcpy(tokens[i - 1], tempVars[tempIndex]); // Replace left operand with temp var

           for (int j = i; j < tokenCount - 2; j++) {

               strcpy(tokens[j], tokens[j + 2]); // Shift remaining tokens left

           }

           tokenCount -= 2;

           i--; // Re-evaluate at same position

       }

   }

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

       if (strcmp(tokens[i], "+") == 0 || strcmp(tokens[i], "-") == 0) {

           // Addition and subtraction

           printf("t%d = %s %s %s\n", tempVarCount, tokens[i - 1], tokens[i], tokens[i + 1]);

           sprintf(tempVars[tempIndex], "t%d", tempVarCount++);

           strcpy(tokens[i - 1], tempVars[tempIndex]); // Replace left operand with temp var

           for (int j = i; j < tokenCount - 2; j++) {

               strcpy(tokens[j], tokens[j + 2]); // Shift remaining tokens left

           }

           tokenCount -= 2;

           i--; // Re-evaluate at same position

       }

   }

   // Final assignment

   printf("%s = t%d\n", tokens[0], tempVarCount - 1);

}

int main() {

   char expression[MAX];

   printf("Enter an arithmetic expression (use spaces between operators & operands):\n");

   fgets(expression, MAX, stdin);

   expression[strcspn(expression, "\n")] = 0; // Remove trailing newline

   generateTAC(expression);

   return 0;

}

