CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX_EXPR_SIZE 100
typedef struct {
    char data[MAX_EXPR_SIZE];
    int top;
} Stack;
void initStack(Stack *stack) {
    stack->top = -1;
}
void push(Stack *stack, char item) {
    stack->data[++stack->top] = item;
}
char pop(Stack *stack) {
    return stack->data[stack->top--];
}
int isOperator(char c) {
    return (c == '+' || c == '-' || c ==
     '*' || c == '/');
}
int getPrecedence(char c) {
    switch (c) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        default:
            return 0;
    }
}
void infixToPostfix(char infix[], cha
r postfix[]) {
    Stack stack;
    initStack(&stack);

    int i = 0, j = 0;

    while (infix[i] != '\0') {
        if (isalnum(infix[i])) {
            postfix[j++] = infix[i++];
        } else if (isOperator(infix[i])) {
            while (!isOperator(stack.data[stack.top]) &&
                   getPrecedence(stack.data
                   [stack.top]) >= getPrecedence(infix[i])) {
                postfix[j++] = pop(&stack);
            }
            push(&stack, infix[i++]);
        } else if (infix[i] == '(') {
            push(&stack, infix[i++]);
        } else if (infix[i] == ')') {
            while (stack.data[stack.top] != '(') {
                postfix[j++] = pop(&stack);
            }
            pop(&stack); // Pop '('
            i++;
        } else {
            // Invalid character in the infix
            expression
            printf("Invalid character in the infix
            expression: %c\n", infix[i]);
            exit(EXIT_FAILURE);
        }
    }
    while (stack.top != -1) {
        postfix[j++] = pop(&stack);
    }
    postfix[j] = '\0';
}
void generateThreeAddressCode(char postfix[]) {
    Stack stack;
    initStack(&stack);
    int tempCount = 1;
    for (int i = 0; postfix[i] != '\0'; i++) {
        if (isalnum(postfix[i])) {
            push(&stack, postfix[i]);
        } else if (isOperator(postfix[i])) {
            char op = postfix[i];
            char tempVar[3];
            sprintf(tempVar, "t%d", tempCount++);
            printf("%c = %c %c %c\n", tempVar[0],
             pop(&stack), op, pop(&stack));
            push(&stack, tempVar[0]);
        }
    }
}
int main() {
    char infix[MAX_EXPR_SIZE];
    char postfix[MAX_EXPR_SIZE];
    printf("Enter the infix expression: ");
    scanf("%s", infix);
    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);
    printf("Three-address code:\n");
    generateThreeAddressCode(postfix);
    return 0;
}
```

OUTPUT

```
Enter the infix expression: a+b*c
Postfix expression: abc*+
Three-address code:
t = b * c
t = a + t
```

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
    char icode[10][30], str[20], opr[10];
    int i = 0;
    printf("\nEnter the set of intermediate
     code (terminated by exit):\n");
    // Input 3-address code
    do {
        scanf("%s", icode[i]);
    } while (strcmp(icode[i++], "exit") != 0);
    printf("\nTarget code generation");
    printf("\n**********************");
    i = 0;
    // Generate target code
    do {
        strcpy(str, icode[i]);

        switch (str[3]) {
            case '+':
                strcpy(opr, "ADD");
                break;
            case '-':
                strcpy(opr, "SUB");
                break;
            case '*':
                strcpy(opr, "MUL");
                break;
            case '/':
                strcpy(opr, "DIV");
                break;
            default:
                printf("Invalid operator in
                 intermediate code\n");
                return 1;  // Indicate an error
        }
        printf("\n\tMov R%d,%c", i, str[2]);
        printf("\n\t%s R%d,%c", opr, i, str[4]);
        printf("\n\tMov %c,R%d", str[0], i);
        i++;
    } while (strcmp(icode[i], "exit") != 0);
    printf("\n");
    return 0;
}
```

OUTPUT

```
Enter the set of intermediate code (terminated by exit):
t=b*c
t=a+t
exit

Target code generation
**********************
        Mov R0,b
        MUL R0,c
        Mov t,R0
        Mov R1,a
        ADD R1,t
        Mov t,R1
```