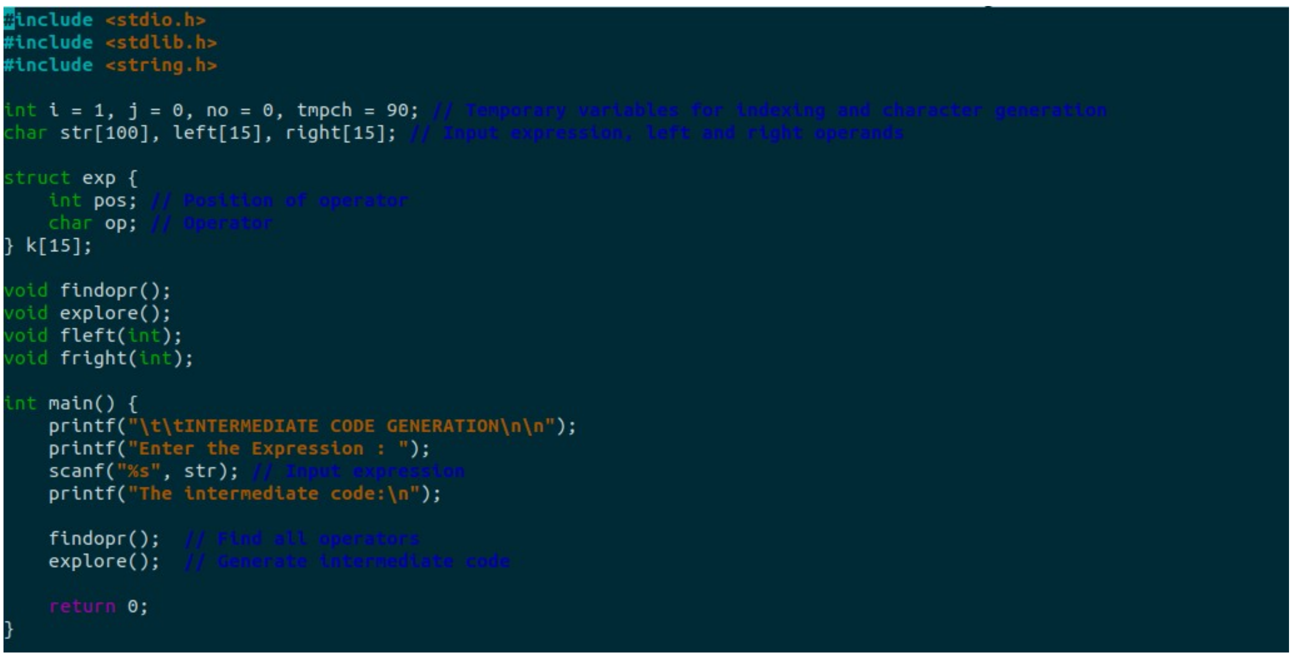
19CSE401 Compiler Design Lab

Anindita Das Badhan
CH.EN.U4CSE22180
4th Year CSE-B

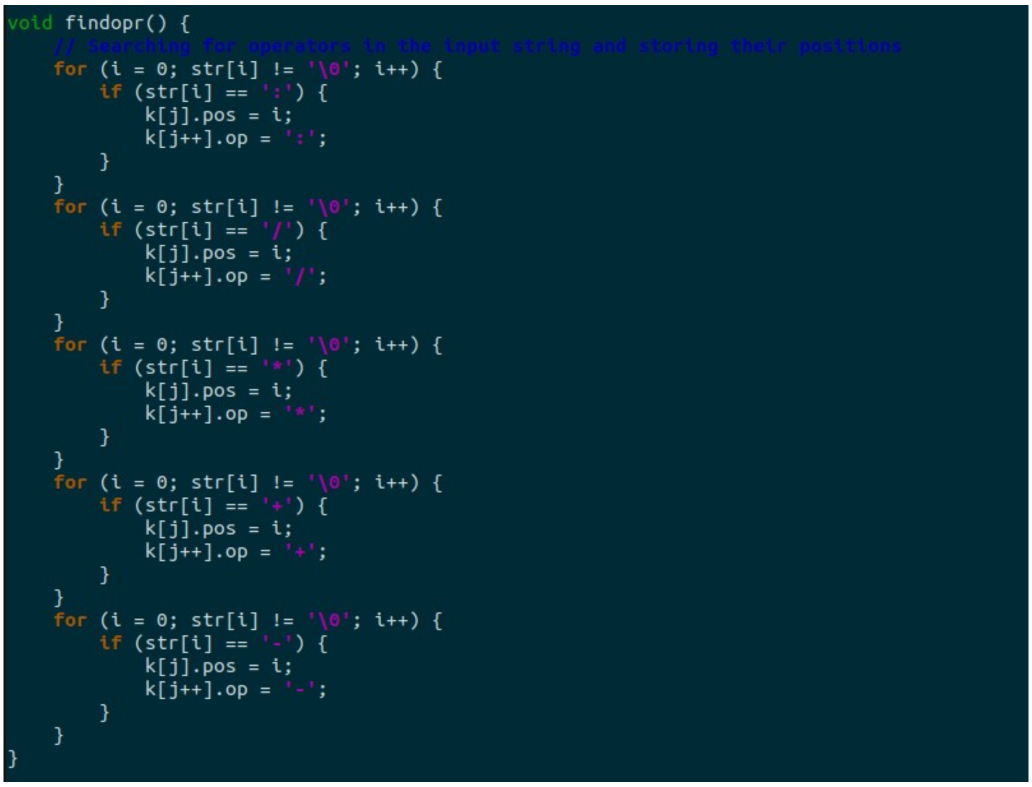
<u>Lab Exercise- 06</u>

Aim: To Implement Intermidiate Code generation

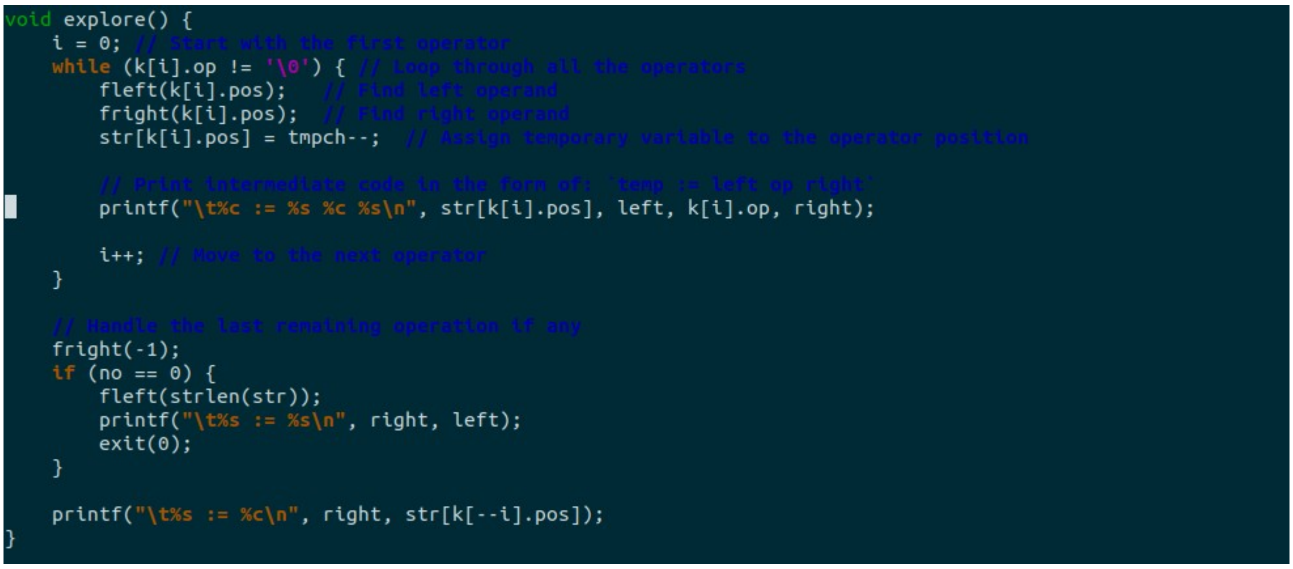
Code:



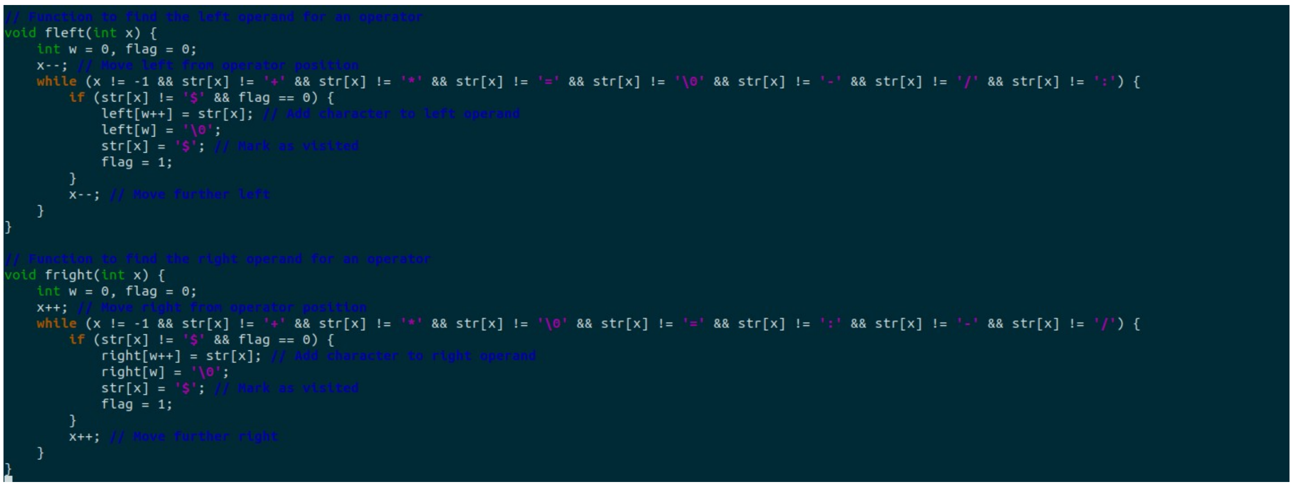
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int i = 1, j = 0, no = 0, tmpch = 90; // Temporary variables for indexing and character generation
char str[100], left[15], right[15]; // Input expression, left and right operands

struct exp {
    int pos; // Position of operator
    char op; // Operator
} k[15];

void findopr();
void explore();
void fleft(int);
void fright(int);

int main() {
    printf("\t\tINTERMEDIATE CODE GENERATION\n\n");
    printf("Enter the Expression : ");
    scanf("%s", str); // Input expression
    printf("The intermediate code:\n");

    findopr();  // Find all operators
    explore();  // Generate intermediate code

    return 0;
}
```



```c
void findopr() {
    // Searching for operators in the input string and storing their positions
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == ':') {
            k[j].pos = i;
            k[j++].op = ':';
        }
    }
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == '/') {
            k[j].pos = i;
            k[j++].op = '/';
        }
    }
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == '*') {
            k[j].pos = i;
            k[j++].op = '*';
        }
    }
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == '+') {
            k[j].pos = i;
            k[j++].op = '+';
        }
    }
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == '-') {
            k[j].pos = i;
            k[j++].op = '-';
        }
    }
}
```

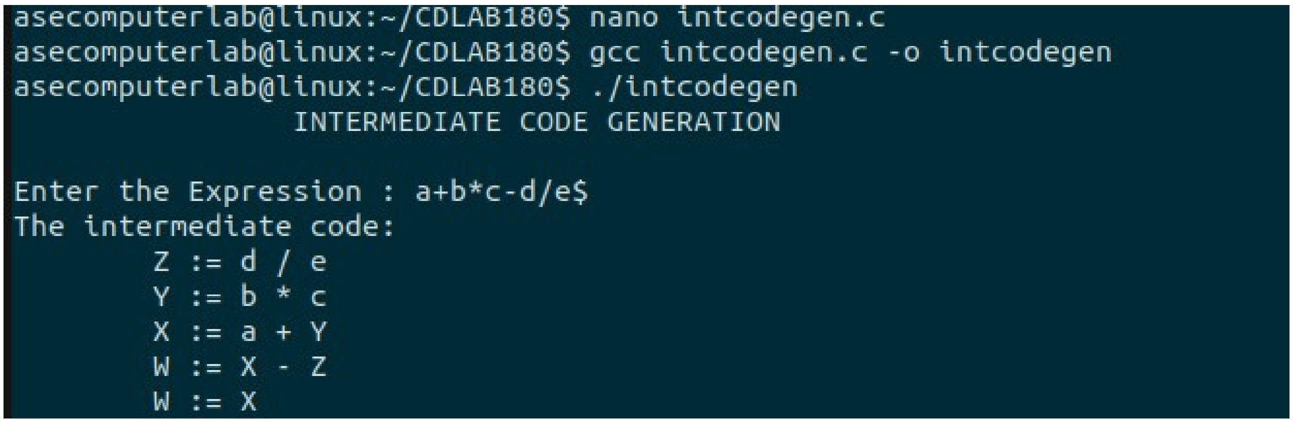


```c
void explore() {
    i = 0; // Start with the first operator
    while (k[i].op != '\0') { // Loop through all the operators
        fleft(k[i].pos);   // Find left operand
        fright(k[i].pos);  // Find right operand
        str[k[i].pos] = tmpch--;  // Assign temporary variable to the operator position

        // Print intermediate code in the form of: `temp := left op right`
        printf("\t%c := %s %c %s\n", str[k[i].pos], left, k[i].op, right);

        i++; // Move to the next operator
    }

    // Handle the last remaining operation if any
    fright(-1);
    if (no == 0) {
        fleft(strlen(str));
        printf("\t%s := %s\n", right, left);
        exit(0);
    }

    printf("\t%s := %c\n", right, str[k[--i].pos]);
}
```



```c
// Function to find the left operand for an operator
void fleft(int x) {
    int w = 0, flag = 0;
    x--; // Move left from operator position
    while (x != -1 && str[x] != '+' && str[x] != '*' && str[x] != '=' && str[x] != '\0' && str[x] != '-' && str[x] != '/' && str[x] != ':') {
        if (str[x] != '$' && flag == 0) {
            left[w++] = str[x]; // Add character to left operand
            left[w] = '\0';
            str[x] = '$'; // Mark as visited
            flag = 1;
        }
        x--; // Move further left
    }
}

// Function to find the right operand for an operator
void fright(int x) {
    int w = 0, flag = 0;
    x++; // Move right from operator position
    while (x != -1 && str[x] != '+' && str[x] != '*' && str[x] != '\0' && str[x] != '=' && str[x] != ':' && str[x] != '-' && str[x] != '/') {
        if (str[x] != '$' && flag == 0) {
            right[w++] = str[x]; // Add character to right operand
            right[w] = '\0';
            str[x] = '$'; // Mark as visited
            flag = 1;
        }
        x++; // Move further right
    }
}
```

Output:



```
asecomputerlab@linux:~/CDLAB180$ nano intcodegen.c
asecomputerlab@linux:~/CDLAB180$ gcc intcodegen.c -o intcodegen
asecomputerlab@linux:~/CDLAB180$ ./intcodegen
                INTERMEDIATE CODE GENERATION

Enter the Expression : a+b*c-d/e$
The intermediate code:
        Z := d / e
        Y := b * c
        X := a + Y
        W := X - Z
        W := X
```

Result: Thus, the program to implement intermediate code generation has been executed successfully