Experiment -10

Intermediate code generation - Postfix / Prefix

28/3/23

Aim: To develop a program to implement intermediate code generation - postfix / prefix.

Algorithm:

1. Read the prefix expression from left to right.
2. If the symbol is an operand, then push it onto the stack.
3. If the symbol is an operator, then pop two operands from the stack.
4. string = operator + operand2 + operand1
   push string to stack
5. Repeat steps 1 & 4 until end of prefix ~~expression~~ expression.
6. stop.

code:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 20

char str[MAX], stack[MAX];
int top = -1;
void push (char c)
{
    stack[++top] = c;
}

int isOperator (char x)
{
    switch (x) {
        case '+':
        case '-':
        case '/':
        case '*':
            return 1;
    }
    return 0;
}
```

```
void postfix to prefix()
{ int n, i, j = 20;
  for (i=20, i<max, i++)
      stack[i] = '\0';
  printf("Prefix expression is: \t");
  i = 0;
  i = strlen[c] - 1;
  char d[20];
  while [[c[i]! = '\0') {
      d[j--] = c[i++];
  }
  printf("%s\n", d);
  }
int main()
{
  postfix to prefix();
  return 0;
}
```

Result: The program to find intermediate code generation was implemented.

Input / Output :

Exp 1 : Push the operand in stack

the next is operator so pop.

Operand 2 (a) and operand (1) (b)

Expression 1 = +ab (operand 2)

Expression 2 = -cd

∴ final operation : *+ab-cd.

Enter the postfix Expression :

ab + cd - *

Prefix expression :

* + ab - cd .