|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No. 03**

|  |  |
| --- | --- |
| Semester | S.E. Semester III – Computer Engineering |
| Subject | Data Structures Lab (CSL301) |
| Subject Professor In-charge | Prof. Swapnil S. Sonawane |
| Assisting Teachers | Prof. Swapnil S. Sonawane |

|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**Title:**

Evaluate Postfix Expression using Stack ADT.

**Objective:**

Students will be able to choose appropriate data structure and apply it to solve problems in various domains.

**Explanation:**

A Stack is used to convert and evaluate the following 3 types of notations:

1. Infix Notation: In this operator is used in between operands.

Example: A+B

1. Prefix Notation: In this, operator is used before operands.

Example: +AB

1. Postfix Notation: In this, operator is used after operands

Example: AB+

Infix expressions are readable and solvable by humans. We can easily distinguish the order of operators, and also can use the parenthesis to solve that part first during solving mathematical expressions. The computer cannot differentiate the operators and parenthesis easily, that’s why postfix conversion is needed.

Following is algorithm for evaluation postfix expressions.

1) Create a stack to store operands (or values).

2) Scan the given expression and do the following for every scanned element.

…..a) If the element is a number, push it into the stack

…..b) If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack

3) When the expression is ended, the number in the stack is the final answer

**Program Code:**

#include<stdio.h>

#include<string.h>

#define N 15

typedef struct stack

{

int a[N];

int top;

}stack;

void push(stack \*s,int x)

{

s->top++;

s->a[s->top]=x;

}

int pop(stack \*s)

{

int x;

x=s->a[s->top];

s->top--;

return x;

}

int isoperand(char x)

{

if(x>='0'&&x<='9')

return 1;

else

return 0;

}

int evaluate(char postfix[])

{

stack s;

s.top=-1;

int op1,op2,x,v,i;

for(i=0;i<strlen(postfix);i++)

{

x=postfix[i];

if(isoperand(x))

{

push(&s,(int)x-'0');

}

else//jab x operator rahega

{

op1=pop(&s);

op2=pop(&s);

if(x=='+')

{

v=op2+op1;

}

else if(x=='-')

{

v=op2-op1;

}

else if(x=='\*')

{

v=op2\*op1;

}

else

{

v=op2/op1;

}

push(&s,v);

}

}

return pop(&s);

}

int main()

{

char postfix[10];

int z;

printf("Enter postfix expression=");

gets(postfix);

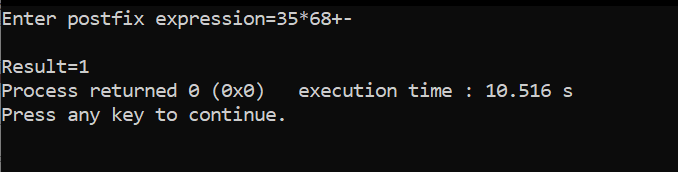
z=evaluate(postfix);

printf("\nResult=%d",z);

return 0;

}

**Output:**

****

**Conclusion:**

Through this experiment, we learned how to evaluate a postfix expression and wrote a program to do the same.