28/12/23

Week - 2

Lab Program - 2

#

Write a Program to Convert a given Valid Parenthesized infix arithmetic Expression to Postfix Expression. The Expression Consists of Single Character Operands & binary Operators + (Plus), - (minus), * (multiply), / (divide) and ^ (POWER).

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

#define MAX 100

char stack[MAX];
char infix[MAX];
char postfix[MAX];
int top = -1;

void push(char);
char pop();
int isEmpty();
void inToPost();
void Print();
int Precendence(char)

int main()
```

```
{
Print f ("enter infix Expression :");
Gets (infix);
inTo Post ();
Print ();
return 0;
}

Void inTo Post ()
{
int i, J = 0;
Char Symbol, next;
for (i=0; i< Strlen (infix); i++)
{
   Symbol = infix[i];
   Switch (Symbol);
   break;
Case ')':
   While ((next = POP()) != '(')
      Post fix [j++] = next;
      break;
Case '+':
Case '-':
Case '*':
Case '/':
While (!is Empty () && Precendence (Stack[top])
   >= Precendence (symbol))
```

```
        Postfix [j++] = Pop();
        Push (Symbol);
        break;
    default :
        Postfix [j++] = Symbol;
    }
}
while (Precendence (Char Symbol))
{
    Switch (Symbol)
    {
    Case '^' :
        return 3;
    Case '/' :
    Case '*' :
        return 2;
    Case '+' :
    Case '-' :
        return 1;
    default :
        return 0;
    }
}
```

```
void Print()
{
    int i = 0;
    Printf("The Equivalent Postfix Expression
            is :");
    while (Postfix[i])
    {
        Printf("%c", Postfix[i++]);
    }
    Printf("\n");
}
void Push(char c)
{
    if (top == MAX-1)
    {
        Printf(" Stack Overflow ");
        return;
    }
    top++;
    Stack[top] = c;
}
char Pop()
{
    char c;
    if (top == -1)
```

```
{
    Printf ("Stack Underflow");
    exit (1);
}
c = stack [top];
top = top - 1;
return c;
}
int isEmpty ()
{
    if (top == -1)
        return 1;
    else
        return 0;
}
```

NLP 18/1/24

Output:

```
Enter infix Expression : a*b+c*d-e
```

The equivalent Postfix Expression is

a b * c d * + e -

Lab Program - 3

28/12/23

> Write a Program to Convert a given valid Parenthesized infix arithmetic Expression to Postfix Expression. The Expression consists of Single character Operands & binary operators and power

Write a Program to demonstrate Postfix evaluation:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_SIZE 100
int stack [MAX_SIZE];
int top = -1;
void Push (int element)
{
  if (top >= MAX_SIZE - 1)
  {
    Printf ("Stack Overflow");
      return ;
  }
  else {
    Stack [++top] = Element;
  }
}
int POP () {
  if (top < 0) {
    Printf ("Stack Underflow \n");
    return -1;
  }
  else {
    return
Stack [top--];
```

18/1/24

```
}
int Evaluate Postfix (char * exp) {
    int i, result;
    int len = Strlen (exp);
    for (i=0 ; i<len ; i++)
{
    if
    (isdigit (exp [i])) {
        Push (exp [i] - '0');
    }
      Else {
    int ~~Operator~~ Operand2 = POP ();
    int Operand 1 = POP ();
    Switch (exp[i])
    {
        Case '+';
Push (Operand 1 + Operand 2);
    break;
        Case '-':
    Push (Operand 1 - Operand 2);
    break;
Case '*';
Push (Operand 1 * Operand 2);
        break;
    Case '/';
```

```
        Push (operand1 / Operand2);
        break;
    }
  }
}
result = POP();
return result;
}
int main() {
char exp [Max_Size];
Printf ("Enter the Postfix Expression :");
Scanf ("%s", exp);
int res = EvaluatePostfix (exp);
Printf ("Result = %d \n", res);
return 0;
}
```

Output:

```
Enter the Postfix Expression:
23* 5+
Result : 11
```

Week-3

Program - 3

> Write a Program to demonstrate Postfix Evaluation:

> WAP to Simulate working of a Queue of Integers using an array.

```
#include <stdio.h>
#define MAX 50

void insert();
void delete();
void display();
int Queue_array[MAX];
int rear = -1;
int front = -1;
main()
{
   int choice;
   while(1)
   {
    Printf("insert Element to Queue\n");
    Printf("Delete Element from Queue\n");
   Printf("Display all Elements of Queue\n");
  Printf("Quit\n");
  Printf("Enter your choice");
  Scanf("%d", &choice);
  Switch (choice)
{
  Case 1:
  insert();
  break;
```

```
        case 2:
        delete();
        break;
        case 3:
        display();
        break;
        case 4:
        exit();
        default:
        Printf("Wrong choice \n");
    }
}

void insert()
{
    int add_item;
    if (rear == MAX-1)
    Printf ("Queue Overflow \n");
    else
    {
    if (front == -1)
    front = 0;
    Printf ("insert the element in Queue:");
    Scanf ("%d", &add_item);
    rear = rear + 1;
    Queue_array [rear] = add_item;
    }
}
```

NLP 18/11/2024

```
void delete()
{
 if (front == -1 || front > rear)
 {
  Printf ("Queue Underflow \n");
  return;
 }
else
{
 Printf ("Element deleted from
  Queue is : %d \n", Queue_array[front]);
   front = front + 1;
 }
}

void display()
{
  int i;
  if (front == -1)
   Printf ("Queue is Empty \n");
 else
 {
  Printf ("Queue is : \n");
  for (i = front; i <= rear ; i++)
   Printf ("%d", Queue_array[i]);
  Printf ("\n");
 }
}
```

output

1. Insert Element to Queue
2. Delete Element from Queue
3. display all Elements of Queue
4. Quit

Enter your choice : 1

Insert the Element in Queue : 42

Enter your choice : 3

Queue is 42

Enter your choice : 2

Element deleted from Queue is 42

Enter your choice 4

Quit

```
include <stdio.h>

define MAX 3

oid insert();
oid delete();
oid display();
nt queue_array[MAX];
nt rear = - 1;
nt front = - 1;
ain()

    int choice;
    while (1)
    {
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(1);
            default:
            printf("Wrong choice \n");
        }
    }


oid insert()

    int add_item;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
```

```
}

oid insert()

   int add_item;
   if (rear == MAX - 1)
   printf("Queue Overflow \n");
   else
   {
       if (front == - 1)

       front = 0;
       printf("Inset the element in queue : ");
       scanf("%d", &add_item);
       rear = rear + 1;
       queue_array[rear] = add_item;
   }


oid delete()

   if (front == - 1 || front > rear)
   {
       printf("Queue Underflow \n");
       return ;
   }
   else
   {
       printf("Element deleted from queue is : %d\n", queue_array[front]);
       front = front + 1;
   }

oid display()

   int i;
   if (front == - 1)
       printf("Queue is empty \n");
   else
   {
       printf("Queue is : \n");
       for (i = front; i <= rear; i++)
           printf("%d ", queue_array[i]);
       printf("\n");
   }
```

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 10
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 10
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 20
Wrong choice
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Queue Underflow
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :

1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4

Process returned 1 (0x1)   execution time : 17.302 s
Press any key to continue.
```

```c
#include<ctype.h>

char stack[100];
int top = -1;

void push(char x) {
    stack[++top] = x;
}

char pop() {
    if (top == -1) {
        return -1;
    } else {
        return stack[top--];
    }
}

int priority(char x) {
    if (x == '(')
        return 0;
    if (x == '+' || x == '-')
        return 1;
    if (x == '*' || x == '/')
        return 2;
    return -1;
}

int main() {
    char exp[100];
    char *e, x;
    printf("Enter the expression: ");
    scanf("%s", exp);
    e = exp;

    while (*e != '\0') {
        if (isalnum(*e)) {
            printf("%c", *e);
        } else if (*e == '(') {
            push(*e);
        } else if (*e == ')') {
            while ((x = pop()) != '(') {
                printf("%c", x);
            }
        } else {

            while (top != -1 && priority(stack[top]) >= priority(*e)) {
```

```
}

int priority(char x) {
    if (x == '(')
        return 0;
    if (x == '+' || x == '-')
        return 1;
    if (x == '*' || x == '/')
        return 2;
    return -1;
}

int main() {
    char exp[100];
    char *e, x;
    printf("Enter the expression: ");
    scanf("%s", exp);
    e = exp;

    while (*e != '\0') {
        if (isalnum(*e)) {
            printf("%c", *e);
        } else if (*e == '(') {
            push(*e);
        } else if (*e == ')') {
            while ((x = pop()) != '(') {
                printf("%c", x);
            }
        } else {

            while (top != -1 && priority(stack[top]) >= priority(*e)) {
                printf("%c", pop());
            }
            push(*e);
        }
        e++;
    }


    while (top != -1) {
        printf("%c", pop());
    }

    return 0;
}
```

```
nter the expression: a*b+c*d-e
ab*cd*+e-
Process returned 0 (0x0)   execution time : 10.191 s
Press any key to continue.
```

```c
int stack[3];
int top = -1;

void push(int x)
{
    stack[++top] = x;
}

int pop()
{
    return stack[top--];
}

int main()
{
    char exp[20];
    char *e;
    int n1,n2,n3,num;
    printf("Enter the expression :: ");
    scanf("%s",exp);
    e = exp;
    while(*e != '\0')
    {
        if(isdigit(*e))
        {
            num = *e - 48;
            push(num);
        }
        else
        {
            n1 = pop();
            n2 = pop();
            switch(*e)
            {
            case '+':
            {
                n3 = n1 + n2;
                break;
            }
            case '-':
            {
                n3 = n2 - n1;
                break;
            }
            case '*':
            {
                n3 = n1 * n2;
```

```
int n1,n2,n3,num;
printf("Enter the expression :: ");
scanf("%s",exp);
e = exp;
while(*e != '\0')
{
    if(isdigit(*e))
    {
        num = *e - 48;
        push(num);
    }
    else
    {
        n1 = pop();
        n2 = pop();
        switch(*e)
        {
        case '+':
        {
            n3 = n1 + n2;
            break;
        }
        case '-':
        {
            n3 = n2 - n1;
            break;
        }
        case '*':
        {
            n3 = n1 * n2;
            break;
        }
        case '/':
        {
            n3 = n2 / n1;
            break;
        }
        }
        push(n3);
    }
    e++;
}
printf("\nThe result of expression %s = %d\n\n",exp,pop());
return 0;
```

```
nter the expression :: 23*5+

he result of expression 23*5+ = 11


rocess returned 0 (0x0)   execution time : 8.538 s
ress any key to continue.
```