Data Structures and its Applications

Dinesh Singh
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Evaluation of Postfix expression and Parenthesis matching

Dinesh Singh
Department of Computer Science & Engineering

Evaluation of Postfix Expression - Algorithm

- Each operator in a postfix string refers to the previous two operands.
- Each time an operand is read , it is pushed on to the stack
- When an operator is reached, its operands will be the top two elements on to the stack.
- The two elements are popped out, the indicated operation is performed on them and result is pushed on the stack so that it will be available for use as an operand of the next operator.

Evaluation of Postfix Expression - Algorithm

```
opndstk is the empty stack
while(not end of the input) // scan the input string
{
 symb=next input character;
 if (symb is an operand)
   push(opndstk,symb)
else
   {
    opnd2=pop(opndstk);
    opnd1=pop(opndstk);
    value = result of applying symb to opnd1 and opn2;
   push(opndstk, value);
  }
 return(pop(opndstk));
}
```

Evaluation of Postfix Expression – Trace of the Algorithm

Infix : 3 + 5 * 4

Postfix expression : 3 5 4 * +

Symb	Opnd1	Opnd2	Value	opndstk
3	-			3
5				3, 5
4				3 , 5 , 4
*	5	4	20	3 , 20
+	3	20	60	60

Evaluation of Postfix Expression - implementation

```
int postfix_eval(char* postfix)
 {
  int i,top,r;
  int s[10];//stack
  top=-1;
  i=0;

  while(postfix[i]!='\0')
  {
    char ch=postfix[i];
    if(isoper(ch))
    {
      int op1=pop(s,&top);
      int op2=pop(s,&top);
```

Evaluation of Postfix Expression - implementation

```
switch(ch)
      {
        case '+':r=op1+op2;
                 push(s,&top,r);
                 break;
        case '-':r=op2-op1;
                  push(s,&top,r);
                  break;
         case '*':r=op1*op2;
                 push(s,&top,r);
                  break;
        case '/':r=op2/op1;
                  push(s,&top,r);
                 break;
       }//end switch
    }//end if
```

Evaluation of Postfix Expression - implementation

```
else
      push(s,&top,ch-'0');//convert charcter to
integer and push
     i++;
    }  //end while
  return(pop(s,&top));
 }
```

Parenthesis Matching – overview of the Algorithm

Examples

1. (( )) : Valid Expression
2. ((( )) : Invalid Expression (Extra opening parenthesis)
3. (( ))) : Invalid Expression (Extra closing parenthesis)
4. ((}) : Invalid Expression (Parenthesis mismatch)
5. (( )] : Invalid Expression (Parenthesis mismatch)

Parenthesis Matching – overview of the Algorithm

1.Read the input symbol from the input expression
2. If the input symbol is one of the open parenthesis (‘(‘ , ‘ { ‘ or ‘ [‘), it is pushed on to the stack
3. If the input symbol is of closing parenthesis, stack is popped and the popped parenthesis is compared with the input symbol, if there is a mismatch in the type of the parenthesis, return 0 (Mismatch of parenthesis)
4. If there is a match in the parenthesis , the next input symbol is read.
5. If during this process, if the stack becomes empty, return 0 (Extra closing parenthesis)
6. If at the end of the expression, if the stack is not empty, return 0 (Extra opening parenthesis)
7. If at the end of the input expression, if the stack is empty, return 1. (Parenthesis are matching)

Data Structures and its Applications

Parenthesis Matching - Implementation

```
int match(char *expr)
 {
    int i,top;
    char s[10],ch,x;//stack
    i=0;
     top=-1;

    while(expr[i]!='\0')
    {
      ch=expr[i];
      switch(ch)
      {
          case '(':
          case '{':
          case '[':push(s,&top,ch);
                   break;
```

Parenthesis Matching

```
case ')':if(!isempty(top))
                {
                  x=pop(s,&top);
                  if(x=='(')
                    break;
                  else
                    return 0;//mismatch of parenthesis
                }
                else
                  return 0;//extra closing parenthesis
```

Parenthesis Matching

```
case '}':if(!isempty(top))
         {
          x=pop(s,&top);
          if(x=='{')
            break;
          else
            return 0;//mismatch of parenthesis
         }
         else
          return 0;//extra closing parenthesis
```

Data Structures and its Applications

Parenthesis Matching

```
case ']':if(!isempty(top))
                {
                 x=pop(s,&top);
                 if(x=='[')
                   break;
                 else
                   return 0;//mismatch of parenthesis
                }
                else
                 return 0;//extra closing parenthesis

        }//end switch
        i++;
    }//end while
   if(isempty(top))
      return 1;
    return 0;//extra opening parenthesis
  }
```

THANK YOU

Dinesh Singh
Department of Computer Science & Engineering

dineshs@pes.edu

+91 8088654402