```
int avaliar (char *e)
{
   char symbol;
   int i=0;
   PILHA_ENC pilha_operandos;
   cria_pilha (&pilha_operandos);
   while (symbol = e[i++])
      if (eh_operando(symbol))
         push (&pilha_operandos, symbol-'0');
      else
      {
         int op2=top_pop(&pilha_operandos),
         op1=top_pop(&pilha_operandos);
         push (&pilha_operandos, aplicar (op1, symbol, op2));
      }
   return (top_pop(&pilha_operandos));
}
```

```
int eh_operando(char op)
{
   return (op != '+' && op != '-' && op != '*' && op != '/' && op
    != '^');
}
int aplicar (int operando1, char operador, int operando2)
{
   switch (operador)
   {
      case '+': return (operando1 + operando2);
      case '-': return (operando1 - operando2);
      case '*': return (operando1 * operando2);
      case '/': return (operando1 / operando2);
      case '^': return ((int)pow(operando1, operando2));
   }
}
```

```
main()
{
   char expr [MAXCOLS]; /*# define MAXCOLS  ?*/
   int position = 0;
   while ((expr[position++] = getchar ()) != '\n');
   expr[--position]='\0';
   printf ("%s%s","a expressão posfixada
   original eh ",expr);
   printf (" = %d\n", avaliar(expr));
}
```

Notações: in, pré e posfixada

Já construímos um algoritmo para avaliar expressões posfixadas, agora, vamos tratar o problema de converter uma expressão infixada em uma posfixada.

Inicialmente, trataremos apenas a questão da precedência, deixando a questão da utilização de parênteses para uma análise posterior.

Defina uma função simples, cujo objetivo é verificar a precedência entre operadores. **int prcd (char op1, char op2) ;**

UNIVASF

```
int prcd (char op1, char op2)
{
   if ((op1=='+' || op1=='-') && (op2=='+' || op2=='-'))
      return (1);
   if ((op1=='*' || op1=='/') && (op2=='+' ||
   op2=='-' || op2=='*' || op2=='/'))
      return (1);
   if (op1=='^' && (op2=='^' || op2=='+' ||
   op2=='-' || op2=='*' || op2=='/'))
      return (1);
   return (0);
}
```

Notações: in, pré e posfixada

Agora, implemente uma função que converte uma string infixada sem parênteses em uma string posfixada.

```
void converter_s_p(char *o, char *d);
void converter_s_p(char *o, char *d)
{
   char symbol;
   int i1=0, i2=0;
   PILHA_ENC opstk;
   cria_pilha (&opstk);
```


UNIVASF

```
while (symbol=o[i1++])
   if (eh_operando(symbol))
      d[i2++]=symbol;
   else
   {
      while (!eh_vazia(opstk) && prcd
(top(opstk),symbol))
         d[i2++]=top_pop(&opstk);
      push(&opstk, symbol);
   }
while (!eh_vazia(opstk))
   d[i2++]=top_pop(&opstk);
d[i2]='\0';
}
```

Notações: in, pré e posfixada

Que alterações seriam necessárias para que o algoritmo anterior seja capaz de converter uma string infixada com parênteses em uma string posfixada?

São necessárias apenas pequenas alterações.

Quando um parêntese de abertura for lido, ele deverá ser introduzido na pilha, ou seja, (prcd(op,'(') é FALSE para todo símbolo de op diferente de um parêntese direito).

Além disso, também vamos definir prcd('(', op) como FALSE para todo símbolo de op exceto ')'.

UNIVASF

Notações: in, pré e posfixada

Quando um parêntese de fechamento for lido, todos os operadores até o primeiro parêntese de abertura deverão ser retirados da pilha e inseridos na string posfixada.

Isso pode ser feito definindo-se prcd(op, ')') como TRUE para todos os operadores op diferentes de um parêntese esquerdo.

Quando esses operadores forem removidos da pilha e o parêntese de abertura for descoberto, uma ação especial deve ser tomada: o parêntese de abertura deve ser removido da pilha e descartado, juntamente com o parêntese de fechamento, em vez de ser colocado na string posfixada ou na pilha.

Definiremos então prcd ('(', ')') como FALSE.

UNIVASF

Notações: in, pré e posfixada

Isto garantirá que, o parêntese de abertura não seja inserido na string posfixada.

Entretanto, como o parêntese de fechamento não deve ser introduzido na pilha, faremos a seguinte alteração:

```
if (symbol==‘)’)
    pop(&opstk);
else
    push(&opstk, symbol);
```

Com estas pequenas alterações podemos fazer uma função para converter uma string infixada qualquer em uma string posfixada.

Notações: in, pré e posfixada - Exercício

Com base no que vimos, construa um programa, na linguagem C, que leia da entrada padrão uma string representando uma expressão infixada, com presença de parênteses, a converta em posfixada e a avalie retornando na saída padrão o resultado da avaliação.