Avaliação de Expressões

- Como efetuar o cálculo de uma expressão em um computador?

  Exemplo: A / (B * (C – D) + E)

  → Regras usuais da matemática.
  → Os parênteses alteram a ordem das expressões:
  O que dificulta a avaliação das expressões:

  1º - a existência de prioridades diferentes.
  Exemplo: A+B*C

  2º - a existência de parênteses
  Exemplo: A*(B+C)

Jan Lukasienwicz conseguiu criar uma representação para expressões onde não existem prioridades e nem a necessidade de uso dos parênteses.

Notação Polonesa

Para somar dois números normalmente escrevemos A+B, entretanto existem três formas:

- Infixa: (A+B)
- Prefixa: (+AB) → Notação Polonesa
- Posfixa: (AB+) → Notação Polonesa Reversa

Na NPR:

→ Cada operador aparece imediatamente após os valores que ele deve operar.
→ A ordem que o operador aparece é a mesma que ele deve ser efetuado.
→ Não há necessidade do uso de parênteses.

Exemplos:

Notação Infixa (Normal)	Notação Posfixa
A+B*C	ABC*+
A*(B+C)	ABC+*
(A+B)/(C-D)	AB+CD-/
(A+B)/(C-D)*E	AB+CD-/E*

Conversão de Expressões Infixa em Posfixa:

- Parentetizar completamente a expressão (definir a ordem de avaliação);
- Varrer a expressão da esquerda para a direita e, para cada símbolo:
    a. Se for parêntese de abertura, ignorar;
    b. Se for operando, copiá-lo para a expressão posfixa (saída);
    c. Se for operador, fazê-lo aguardar;
    d. Se for parêntese de fechamento, copiar o último operador acessado.

Conversão da Expressão A+B*C-D:

1º Etapa: Parentetização

a) A + (B * C) – D → efetua multiplicação
b) (A + (B * C)) – D → efetua adição
c) ((A + (B * C)) – D) → efetua subtração

2º Etapa: Varredura

Símbolo	Adição	Pilha	Saída
(	ignora	P:[]	
(	ignora	P:[]	
A	copia	P:[]	A
+	aguarda	P:[+]	A
(	ignora	P:[+]	A
B	copia	P:[+]	A B
*	aguarda	P:[*,+]	A B
C	copia	P:[*,+]	A B C
)	remover	P:[+]	A B C *
)	remove	P:[]	A B C * +
–	aguarda	P:[–]	A B C * +
D	copia	P:[–]	A B C * + D
)	remove	P:[]	A B C * + D –

Exercícios

A)
1. A – B * (D – C)

Etapa 1:
A – (B * (D – C))
(A – (B* (D – C)))

Etapa 2:
ABDC – * –

2. A * B + C – (D / E + F)

Etapa 1:
((A * B) + (C – ((D / E) + F)))

Etapa 2:
A B * C D E F + / – +

```
function Npr(E:String):String;
var      P:Pilha;
         S:String;
         i:Integer;

begin

  Init(P);
  S:='';

  for i:=1 to length(E) do
    case E[i] of
         'A'..'Z': S :=S+E[i];       {copia}
         '+','-',
         '*','/' : Push(P,E[i]);     {aguarda}
         ')'     : S := S + Pop(P);  {remove}
    end;

  Npr := S;

end;
```

Algoritmo de Conversão Infixa para Posfixa:

Fundamentos para criação de algoritmos

- A ordem dos operandos não é alterada quando a expressão é convertida.
- Os operadores devem mudar de ordem.
- O que determina a posição do operador na posfixa é a prioridade que ele possui na infixa (os de maior procedência aparecem primeiro).
- Quando um operador é encontrado no processo de varredura, ele de aguardar até que apareça outro operador com prioridade menor ou igual a dele, somente então ele poderá ser copiado na saída.
- Quando um parêntese de abertura é encontrado no processo de varredura, ele deve ser imediatamente empilhado. Uma vez empilhado, sua ordem de prioridade deve ser a menor possível para não influenciar o resultado.

Ordem de Prioridades na Atualização da Expressão:

Operador	Prioridade
(	1
+ –	2
* /	3

```
function Prio(S:char) : integer;
begin
  case S of
     '(': Prio := 1;
     '+',
     '-': Prio := 2;
     '*',
     '/': Prio := 3;
  end;
end;
```

O Processo de conversão infixa para posfixa pode ser efetuado da seguinte forma:

- Inicie com uma pilha vazia;
- Realize uma varredura na expressão infixa, copiando todos os identificadores encontrados diretamente para a expressão de saída.
  a) Ao encontrar um operador:
     1. Enquanto a pilha não estiver vazia e houver no seu topo um operador com prioridade maior ou igual ao encontrado, desempilhe o operador e copie-o na saída;
     2. Empilhe o operador encontrado;
  b) Ao encontrar um parêntese de abertura, empilhe-o;
  c) Ao encontrar um parêntese de fechamento, remova um símbolo da pilha e copie-o na saída, até que seja desempilhado o parêntese de abertura correspondente.
- Ao final da varredura, esvazie a pilha, movendo os símbolos desempilhados para a saída.

Veja o funcionamento da versão final do algoritmo na conversão da expressão A*(B+C)/D:

Símbolo	Ação	Pilha	Saída
A	Copia para a saída	P:[]	A
*	Pilha vazia, empilha	P:[*]	A
(	Sempre deve ser empilhado	P:[(,*]	A
B	Copia para a saída	P:[(,*]	A B
+	Prioridade maior, empilha	P:[+,(,*]	A B
C	Copia para a saída	P:[+,(,*]	A B C
)	Desempilha até achar '('	P:[*]	A B C +
/	Prioridade igual, desempilha	P:[/]	A B C +*
D	Copia para a saída	P:[/]	A B C +* D
	Final, esvazia pilha	P:[]	A B C * + D /

```
function Posfixa(E:string) : string;
var P : Pilha;
    S : string;
    i : integer;
    x : char;
begin
  Init(P);
  S:='';

  for I:=1 to length(E) do
     case E[i] of
        'A'..'Z': S := S + E[I];
        '+'..'-',
        '*'..'/': begin
                    while not IsEmpty(P) and
                       (Prio(Top(P))>=Prio(E[i]))
                    do S := S + Pop(P);
                    Push(P, E[i]);
                  end;

        '('     : Push(P, E[i]);
        ')'     : begin
                    while Top(P)<>'(' do S := S + Pop(P);
                       x:= Pop(P);
                  end;
     end;
   while not IsEmpty(P) do S := S + Pop(P);
       Posfixa := S;
end;
```

Avaliando uma Expressão Posfixa:

Percorrendo uma expressão posfixa da esquerda para a direita, ao encontrarmos um operador, sabemos que ele deve operar as duas últimas variáveis encontradas, ou seja, as duas últimas serão as primeiras a serem processadas.

Seja a expressão:

AB+CD-/E*

Supondo A=7, B=3, C=6, D=4 e E=9

Podemos proceder da seguinte maneira:

- Iniciamos com uma pilha vazia;
- Varremos a expressão e, para cada elemento encontrado, fazemos:
  a) Se for operando, então empilhar seu valor;
  b) Se for operador, então desempilhar os dois últimos valores, efetuar a operação com eles e empilhar de volta o resultado obtido;
- No final do processo, o resultado da avaliação estará no topo da pilha.

Analise o funcionamento do algoritmo de conversão no exemplo a seguir:

Expressão	Elemento	Ação	Pilha
AB+CD-/E*			P:[ ]
B+CD-/E*	A	Empilha valor de A	P:[7]
+CD-/E*	B	Empilha valor de B	P:[3,7]
CD-/E*	+	Desempilha y=3	P:[7]
		Desempilha x=7	P:[ ]
		Empilha x+y	P:[10]
D-/E*	C	Empilha valor de C	P:[6,10]
-/E*	D	Empilha valor de D	P:[4,6,10]
/E*	-	Desempilha y=4	P:[6,10]
		Desempilha x=6	P:[10]
		Empilha x-y	P:[2,10]
E*	/	Desempilha y=2	P:[10]
		Desempilha x=10	P:[ ]
		Empilha x/y	P:[5]
*	E	Empilha valor de E	P:[9,5]
	*	Desempilha y=9	P:[5]
		Desempilha x=5	P:[ ]
		Empilha x*y	P:[45]

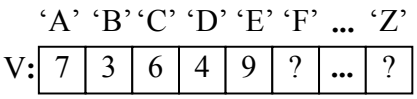
Implementação da NPR:

Em Pascal é possível implementar um vetor cujo índice é um caracter. Exemplo:

```
var
V = Array['A'..'Z'] of Real;
```

Dessa forma, os elementos do vetor V poderão ser acessados conforme sugere a figura:

Representação interna das variáveis



Usando um vetor desse tipo é possível criar uma rotina para receber os valores de uma expressão, conforme a seguir:

```
procedure Atribui(var V:Valores);
var   N:char;
begin
   Writeln('Digite . para finalizar...');

   repeat
     write('Nome: ');
     readln(N);
     if N in ['A'..'Z'] then
       begin
         write('Valor: '); 'A'
         readln(V[N]);
       end;
   until N = '.';
end;
```