

Выражение в инфиксной форме:

Выражение в постфиксной форме:

$$(a + b * (a + c)) / (a - d)$$

$$a b a c + * + a d - /$$

Вычисление значения выражения в постфиксной форме с помощью стека

<i>a</i>	<i>b</i>	<i>a</i>	<i>c</i>	$\langle a + c \rangle$	$\langle b * \langle a + c \rangle \rangle$	$\langle a + b * \langle a + c \rangle \rangle$
	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>a</i>	
		<i>a</i>	<i>b</i>	<i>a</i>		
			<i>a</i>			

<i>a</i>	<i>d</i>	$\langle a - d \rangle$
$\langle a + b * \langle a + c \rangle \rangle$	<i>a</i>	$\langle a + b * \langle a + c \rangle \rangle$
	$\langle a + b * \langle a + c \rangle \rangle$	

$$\langle a + b * \langle a + c \rangle \rangle / \langle a - d \rangle$$

Приоритет входного символа операции

'(' ICP = 0

'*', '/' ICP = 1

'+', '-' ICP = 2

')' ICP = 3

Внутрисктековый приоритет символа операции

'*', '/' ISP = 1

'+', '-' ISP = 2

'(', '#' ISP = 3

Стек	E[i]	ICP		ISP	Выражение
#					"
(#	(0	<	3	"
(#	<i>a</i>				' <i>a</i> '
+ (#	+	2	<	3	' <i>a</i> '
+ (#	<i>b</i>				' <i>a b</i> '
* + (#	*	1	<	2	' <i>a b</i> '
(* + (#	(0		1	' <i>a b</i> '
(* + (#	<i>a</i>				' <i>a b a</i> '
+ (* + (#	+	2	<	3	' <i>a b a</i> '
+ (* + (#	<i>c</i>				' <i>a b a c</i> '
(* + (#)	3	>	2	' <i>a b a c +</i> '
* + (#)	3	=	3	' <i>a b a c +</i> '
+ (#)	3	>	1	' <i>a b a c + *</i> '

Стек	E[i]	ICP		ISP	Выражение
(#)	3	>	2	'a b a c + * +'
#)	3	=	3	'a b a c + * +'
/ #	/	1	<	3	'a b a c + * +'
(/ #	(0	<	1	'a b a c + * +'
(/ #	a				'a b a c + * + a'
- (/ #	-	2	<	3	'a b a c + * + a'
- (/ #	b				'a b a c + * + a d'
(/ #)	3	>	2	'a b a c + * + a d -'
/ #)	3	=	3	'a b a c + * + a d - /'
#					

```
const
  N = 100;
type
  TStackArr = array [ 1..N ] of variant;
  TVarRec = record
    V:boolean;
    Num:integer;
  end;
  TVarArr = array [ 'a' .. 'z' ] of TVarRec;
  TStack = class
    TopS : integer;
    S : TStackArr;
    procedure Init;
    procedure Push( E : variant);
    function Pop : variant;
    function Empty : boolean;
    function Full : boolean;
  end;
```

```
TCalcExpr = class (TStack)
  PostExpr : string;
  Vars : TVarArr;
  function ICP (C : char) : byte;
  function ISP ( C: char) : byte;
  procedure InfixToPostfix( _Expr : string);
  function InpData(var _Data : TStringGrid) : string;
  function Calc( _Data : TStringGrid) : string;
  function ShowPost : string;
end;
```

```
function TStack.Pop : variant;  
var  
  tmp : variant;  
begin  
  tmp := S[TopS];  
  TopS := TopS-1;  
  Pop := tmp;  
end;
```

```
procedure TStack.Push ( E : variant);  
begin  
  TopS := TopS + 1;  
  S[TopS] := E;  
end;
```

```
function TCalcExpr.ICP( C : char) : byte;
begin
  case C of
    '(' : ICP := 0;
    '*', '/' : ICP := 1;
    '+', '-' : ICP := 2;
    ')' : ICP := 3;
  end;
end;
```

```
function TCalcExpr.ISP(C : char) : byte;
begin
  case C of
    '*', '/' : ISP := 1;
    '+', '-' : ISP := 2;
    '(', '#' : ISP := 3;
  end;
end;
```



```
procedure TCalcExpr.InfixToPostfix(_Expr : string);
var
  tmp :string;
  c : char;
  i : integer;
begin
  Init;
  Push( '#' );
  PostExpr := "";
  for c:='a' to 'z' do
    Vars[c].V:=false;
```

```

for i := 1 to length(_Expr) do
  if _Expr[ i ] in [ 'a' .. 'z' ] then
    begin
      PostExpr := PostExpr + _Expr[ i ];
      Vars[_Expr[ i ]].V := true;
    end
  else
    begin
      tmp := Pop;
      while ISP(tmp[ 1 ]) < ICP(_Expr[ i ]) do
        begin
          if tmp <> '(' then
            PostExpr := PostExpr + tmp;
            Tmp := Pop;
          end;
          if tmp[1] <> '(' then
            Push(tmp);
          if _Expr[ i ] <> ')' then
            Push(_Expr[ i ]);
          end;
        end;
      end;
    end;
  end;
end;

```

Выражение

$(a+b)*c/(a-c)$

=

-17,50

Переменная	Значение
a	3
b	4
c	5

ab+cas-/*

Ввести значения!

CV

C

```
function TCalcExpr.Calc(_Data : TStringGrid): string;
var
  Op1 , Op2 , Res: single;
  i : integer;
begin
  for i :=1 to length(PostExpr) do
    begin
      if PostExpr[ i ] in [ 'a' .. 'z' ] then
        Push(_Data.Cells[ 1, Vars[PostExpr[ i ]].Num])
      else
        begin
          Op2 := Pop;
          Op1 := Pop;
          case PostExpr[i] of
            '+' : Push(Op1 + Op2);
            '-' : Push(Op1 - Op2);
            '*' : Push(Op1 * Op2);
            '/' : Push(Op1 / Op2);
          end;
        end;
      end;
    end;
  Res:=Pop;
  Calc:=FloatToStrF(Res,ffFixed,8,2);
end;
```