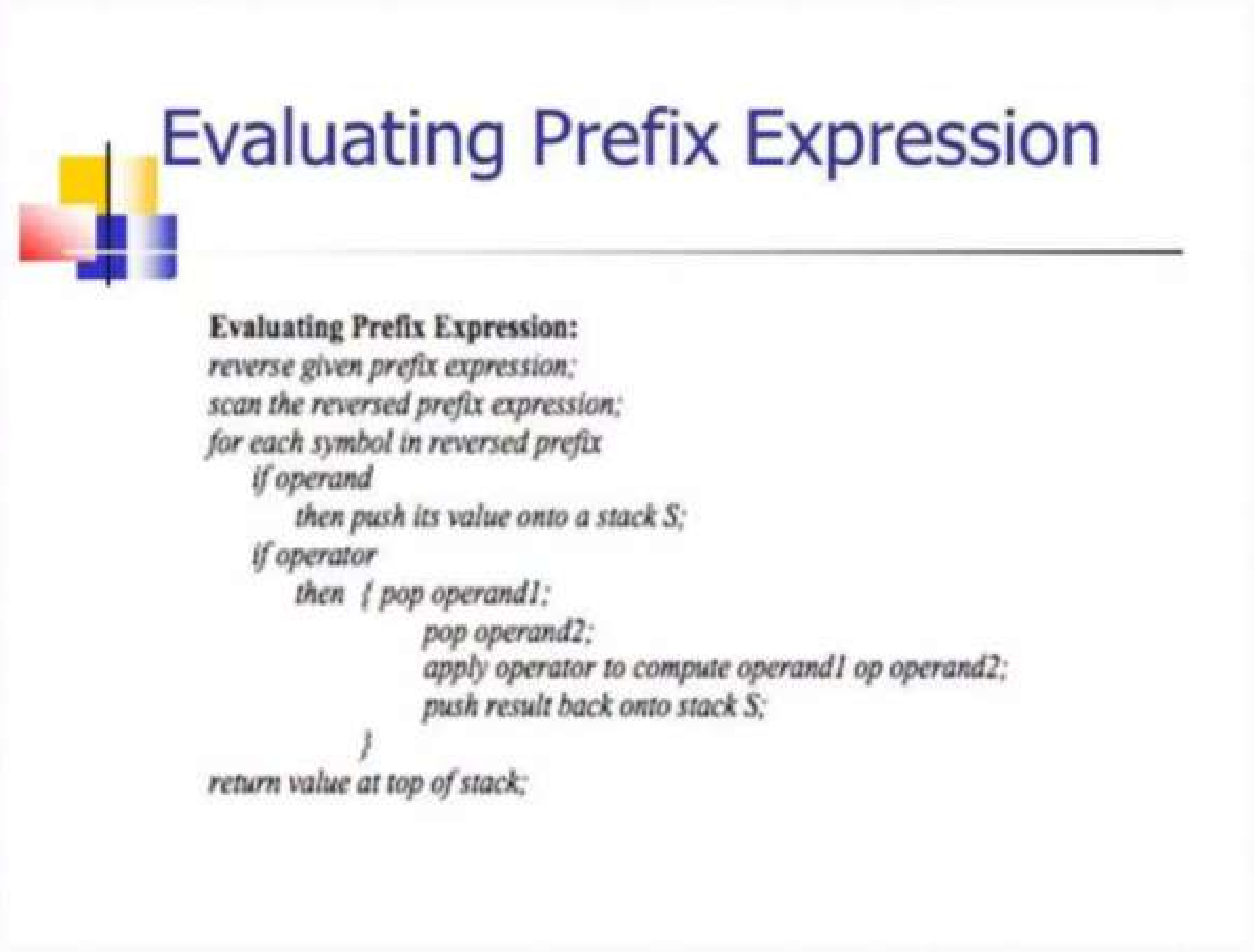
EVALUATE_PREFIX (STRING)

- Step 1: Put a pointer P at the end of the end.
- Step 2: If character at P is an operand push it to Stack
- Step 3: If the character at P is an operator pop two elements from the Stack. Operate on these elements according to the operator, and push the result back to the Stack.
- Step 4: Decrement P by 1 and go to Step 2 as long as there are characters left to be scanned in the expression.
- Step 5: The Result is stored at the top of the Stack, return it,
- Step 6: End

EXPRESSION

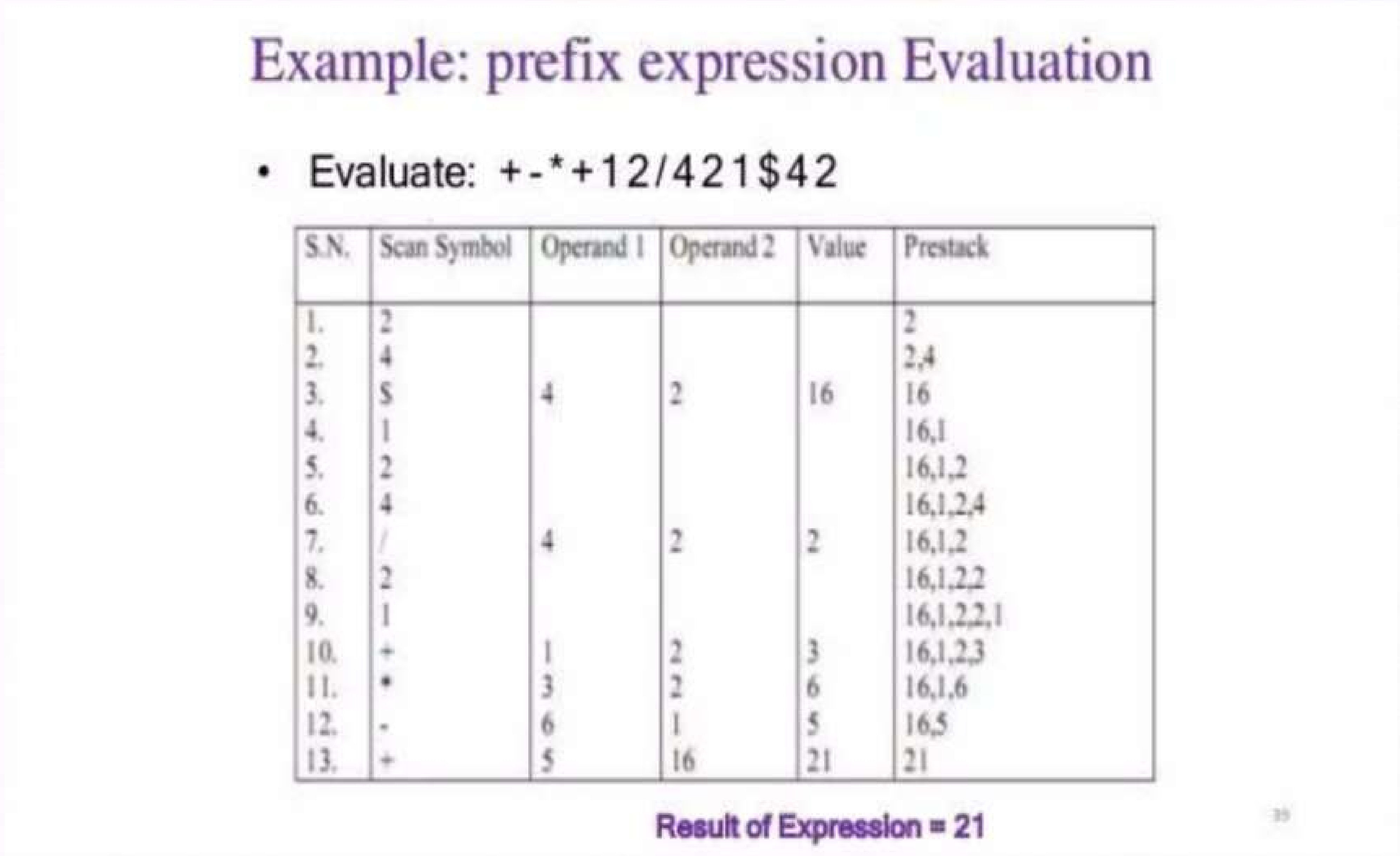


Evaluating Prefix Expression

Evaluating Prefix Expression:

```
reverse given prefix expression;
scan the reversed prefix expression;
for each symbol in reversed prefix
    if operand
        then push its value onto a stack S;
    if operator
        then { pop operand1;
                    pop operand2;
                    apply operator to compute operand1 op operand2;
                    push result back onto stack S;
             }
return value at top of stack;
```

EXAMPLE



Example: prefix expression Evaluation

- Evaluate: +-*+12/421$42

S.N.	Scan Symbol	Operand 1	Operand 2	Value	Prestack
1.	2				2
2.	4				2,4
3.	$	4	2	16	16
4.	1				16,1
5.	2				16,1,2
6.	4				16,1,2,4
7.	/	4	2	2	16,1,2
8.	2				16,1,2,2
9.	1				16,1,2,2,1
10.	+	1	2	3	16,1,2,3
11.	*	3	2	6	16,1,6
12.	-	6	1	5	16,5
13.	+	5	16	21	21

Result of Expression = 21

Evaluation of Prefix expressions

- An expression: 2*3 +5*4-9

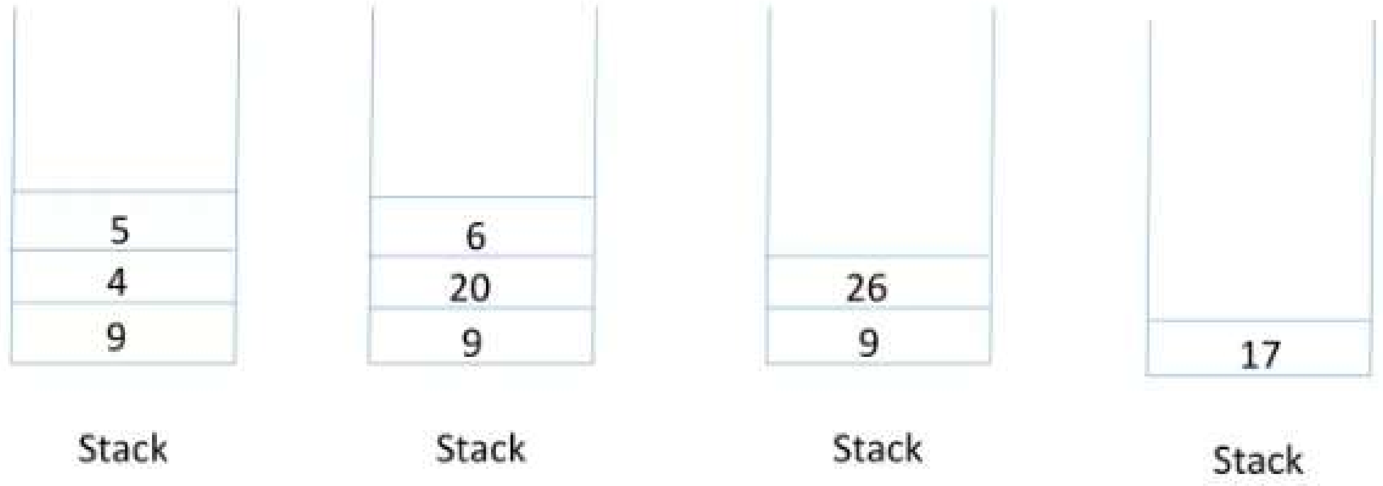
- Can be written as:
  {(2*3)+(5*4)}-9
  {(*2 3)+(*5 4)}-9
  {+(*2 3) (*5 4)}-9
  -{+(*2 3) (*5 4)}9

- We can get Rid of Paranthesis
  -+*2 3 *5 4 9

Evaluation of Prefix expressions

- We have to scan it from right

  -+*2 3 *5 4 9


5
4
9
Stack
6
20
9
Stack
26
9
Stack
17
Stack