

стек

---

Примеры  
использования  
стека

# Абстрактный тип данных Стек

---

- *Стеком* называется последовательность элементов одного и того же типа, к которой можно добавлять новые элементы и удалять элементы последовательности.

Причем как добавление элементов, так и удаление элементов производится с одного и того же конца последовательности, называемого *вершиной стека*.

# Операции со стеком

---

- ***CreateStack()*** - создает пустой стек
- ***DeleteStack ()*** – уничтожает стек
- ***IsEmpty()*** – функция определения пустоты стека ли стек
- ***Push(NewElement)*** – добавляет новый элемент *NewElement* в стек
- ***Pop()*** – удаляет верхний элемент из стека
- ***Peek()*** – возвращает значение верхнего элемента (вершины стека) без его удаления

# Алгебраические выражения

---

- *Инфиксная запись выражений:*  
каждый бинарный оператор помещается между своими операндами
- *Префиксная запись выражений (Prefix):*  
каждый бинарный оператор помещается перед своими операндами  
(Польская запись)
- *Постфиксная запись выражений (Postfix):*  
каждый бинарный оператор помещается после своих операндов  
(Обратная Польская запись)

# Алгебраические выражения

<i>Инфиксная форма</i>	<i>Префиксная форма</i>	<i>Постфиксная форма</i>
$A+B$	$+AB$	$AB+$
$A+B*C$	$+A*BC$	$ABC*+$
$(A+B)*C$	$*+ABC$	$AB+C*$

# Преобразование инфиксной формы в Prefix и Postfix

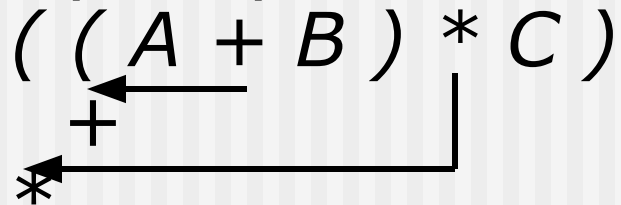
---

- Допустим, инфиксное выражение полностью заключено в скобки
- Преобразование в префиксную форму: каждый оператор перемещается на позицию соответствующей открывающейся скобки (перед операцией)
- Преобразование в постфиксную форму: каждый оператор перемещается на позицию соответствующей закрывающейся скобки (после операцией)
- Скобки убираются

# Примеры

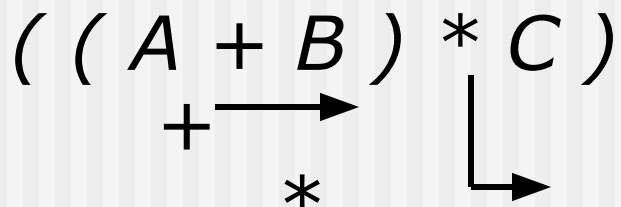
---

- Преобразование в префиксную форму:

$$((A + B) * C)$$


$$*+ABC$$

- Преобразование в постфиксную форму:

$$((A + B) * C)$$


$$AB+C*$$

# Преимущества префиксной и постфиксной форм записи

---

- Не нужны приоритеты операций, правила ассоциативности, скобки
- Алгоритмы распознавания выражений и вычисления более просты



# Вычисление постфиксных выражений

---

- Допустим необходимо вычислить выражение:  
 $2*(3+4)$
- Его постфиксная запись:  
 $234+*$
- Порядок выполнения операций:
  - Помещаем в стек значения всех операндов, пока не встретим знак операции
  - Выталкиваем из стека 2 операнда
  - Производим с ними соответствующую операцию
  - Результат помещаем в стек
  - Повторяем операции до тех пор, пока не кончится строка символов

# Пример:

<i>Символ</i>	<i>Действия</i>	<i>Состояние стека</i>
2	Push(2)	2
3	Push(3)	2 3
4	Push(4)	2 3 4
+	Op2=Peek() Op1=Peek() Result=Op1+Op2	2 3 2
	Push(Result)	2 7
*	Op2=Peek() Op1=Peek() Result=Op1*Op2	2 -
	Push(Result)	14

# Псевдокод алгоритма

---

Предположения:

- Строка содержит синтаксически правильное выражение
- Унарные операции и операции возведения в степень не используются
- Операнды задаются строчными буквами

# Псевдокод алгоритма

---

```
For (каждый символ ch в строке)
{ if (ch является операндом)
  // помещаем ch в стек
  Push(ch);
else
  { Oprign=ch;
    Op2=Pop(); //извлекаем значение из вершины
    Op1=Pop(); //извлекаем значение из вершины
    // выполняем соответствующую операцию
    Result=Op1 Opsign Op2;
    // помещаем результат в стек
    Push(Result);
  }; // конец оператора if
} // конец оператора For
```

# Преобразование инфиксных выражение в постфиксные

---

Будем использовать:

- Стек для хранения операций и скобок
- Строку PostfixExp для формирования постфиксного выражения

# Преобразование инфиксных выражение в постфиксные

Алгоритм:

- Если встретился операнд – помещаем его в строку
- Если встретилась '(' – помещаем в стек
- Если встретился оператор:
  - Если стек пуст – помещаем оператор в стек
  - Если стек не пуст – операторы более высокого приоритета выталкиваются и помещаются в строку, пока не встретится '(' или оператор более низкого приоритета
- Если встретился символ ')', то все элементы выталкиваются из стека и помещаются в строку, пока не встретится соответствующая '('
- Достигнув конца строки, все элементы стека добавляются в строку

# Пример: $A-(B+C*D)/F$

Символ	Стек	Строка
A		A
-	-	
(	-(	A
B	-(	AB
+	-(+	AB
C	-(+	ABC
*	-(+*	ABC
D	-(+*	ABCD
)	-(+ -( -	ABCD* ABCD*+ ABCD*+
/	-/ -	ABCD*+ ABCD*+F
F	-/ -	ABCD*+F
		ABCD*+F/

# Пример: $A+B*(C/V+Z*(A+D))$

Символ	Стек	Строка	Символ	Стек	Строка
<b>A</b>		<b>A</b>	*	<b>+(+*</b>	<b>ABCB/Z</b>
<b>+</b>	<b>+</b>	<b>A</b>	<b>(</b>	<b>+(+*(</b>	<b>ABCB/Z</b>
<b>B</b>	<b>+</b>	<b>AB</b>	<b>A</b>	<b>+(+*(</b>	<b>ABCB/ZA</b>
<b>*</b>	<b>+(+*</b>	<b>AB</b>	<b>+</b>	<b>+(+*(+*</b>	<b>ABCB/ZA</b>
<b>(</b>	<b>+(+*(</b>	<b>AB</b>	<b>D</b>	<b>+(+*(+*</b>	<b>ABCB/ZAD</b>
<b>C</b>	<b>+(+*(</b>	<b>ABC</b>	<b>)</b>	<b>+(+*</b>	<b>ABCB/ZAD+</b>
<b>/</b>	<b>+(+*(/</b>	<b>ABC</b>	<b>)</b>		<b>ABCB/ZAD+*+*+</b>
<b>B</b>	<b>+(+*(/</b>	<b>ABCB</b>			
<b>+</b>	<b>+(+*(+*</b>	<b>ABCB/</b>			
<b>Z</b>	<b>+(+*(+*</b>	<b>ABCB/Z</b>			



Пример:  $A+B*(C/B+Z*(A+D))$

---

$A+B*(C/B+Z*(A+D))$

The diagram illustrates the order of operations for the expression  $A+B*(C/B+Z*(A+D))$ . It shows the sequence of operations from left to right, following the standard order of operations (PEMDAS):

- 1.  $A$  (indicated by an arrow pointing to the first 'A')
- 2.  $B$  (indicated by an arrow pointing to 'B')
- 3.  $(A+D)$  (indicated by an arrow pointing to the innermost parentheses)
- 4.  $Z$  (indicated by an arrow pointing to 'Z')
- 5.  $(C/B)$  (indicated by an arrow pointing to the division operation)
- 6.  $+$  (indicated by an arrow pointing to the addition operation between  $(C/B)$  and  $Z*(A+D)$ )
- 7.  $*$  (indicated by an arrow pointing to the multiplication operation between  $B$  and the sum)

Результат:  $ABC B/ZAD+*+*+$

# Псевдокод алгоритма:

---

```
For (ch)
{ Swith (ch)
  { case operand:
    PostfixExp= PostfixExp+ch;
    break;
  case '(' :
    Push(ch);
    break;
  case ')'':
    While( '(')
    { PostfixExp= PostfixExp + Peek();
      Pop();
    };
  }
```

# Псевдокод алгоритма:

---

```
case operator:
    While (!IsEmpty() и значение вершины != '('
           и Приоритет ch не превосходит
приоритета вершины) { PostfixExp=
PostfixExp+Peek();
    Pop();
    } // конец While
    Push(ch);
break;
} // конец Switch;
} // конец For
While(! IsEmpty() )
    { PostfixExp= PostfixExp + Peek();
      Pop();
    }; // конец While
```