

Стек & Опашка

SHUNTING YARD АЛГОРИТЪМ И ИЗЧИСЛЯВАНЕ НА ИЗРАЗИ

Александар Айтov

ФМИ СДП 2025

Roadmap

Въведение

Мотивация и учебни
цели

Shunting Yard Algorithm

История и правила
на алгоритъма

Преговор

Структурите стек и
опашка

Имплементация

C++ класове и начин
на работа

Въведение



— Какво ще научим днес?

- 1 Структурите от данни stack и queue и техните приложения
- 2 Целта и методологията на Shunting Yard алгоритъма
- 3 Парсирайте и изчислявайте математически изрази

Защо е важно?

Как компютърът разбира: $2 + 3 * 4 = ?$

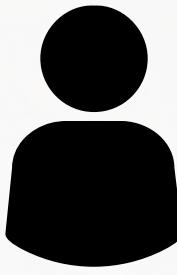


$$(2 + 3) * 4 = 20$$



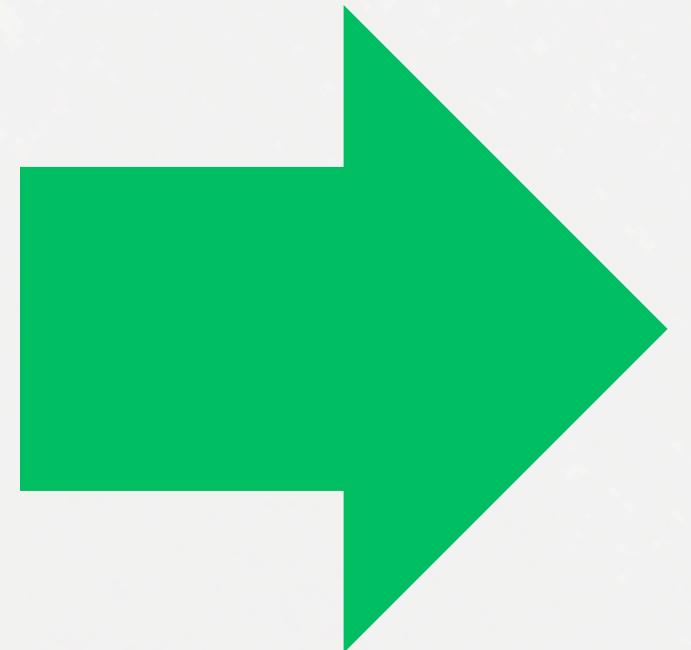
$$2 + (3 * 4) = 14$$

Инфиксна срещу постфиксна нотация



Инфиксна нотация
(подходяща за хора)

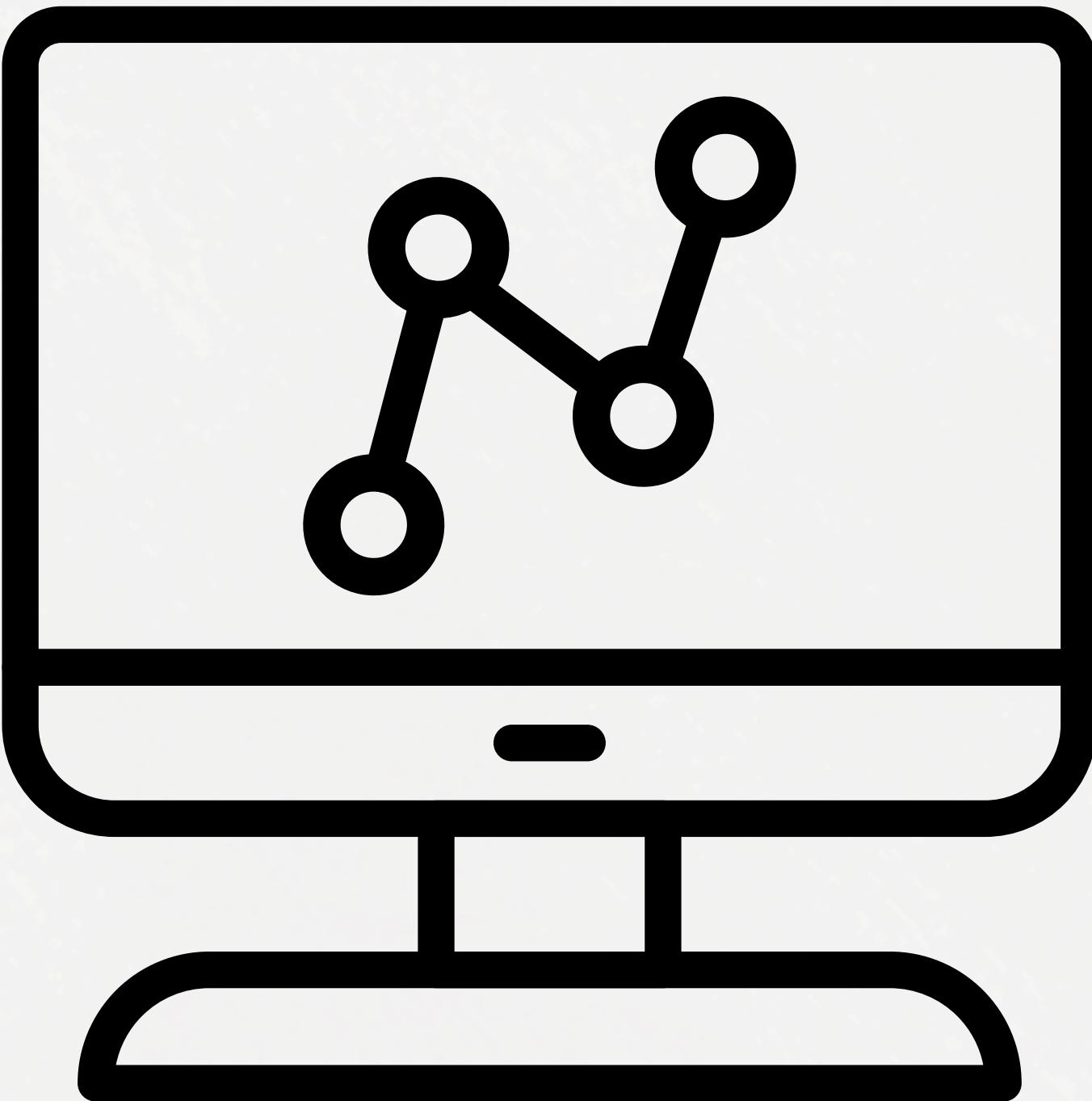
$3 + 4$



Постфиксна нотация
(подходяща за компютри)

$3 \ 4 \ +$

Преговор



Стек (Stack)

Last In, First Out (LIFO)

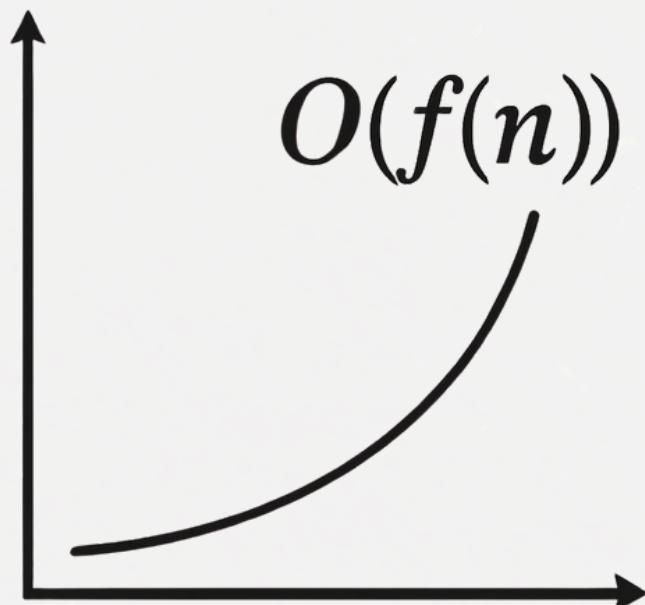
Операции

- push()
- pop()
- front()



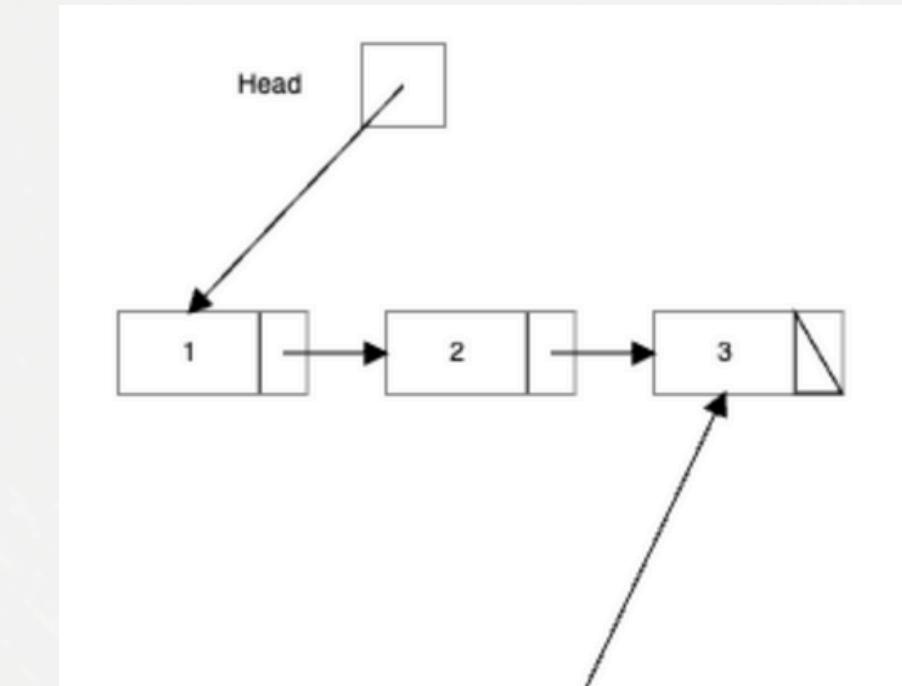
Сложность (Big O)

- $O(1)$ за всички операции



Визуализация на Стек

- [Visualisation Website](#)



Опашка (Queue)

First in, First Out (FIFO)

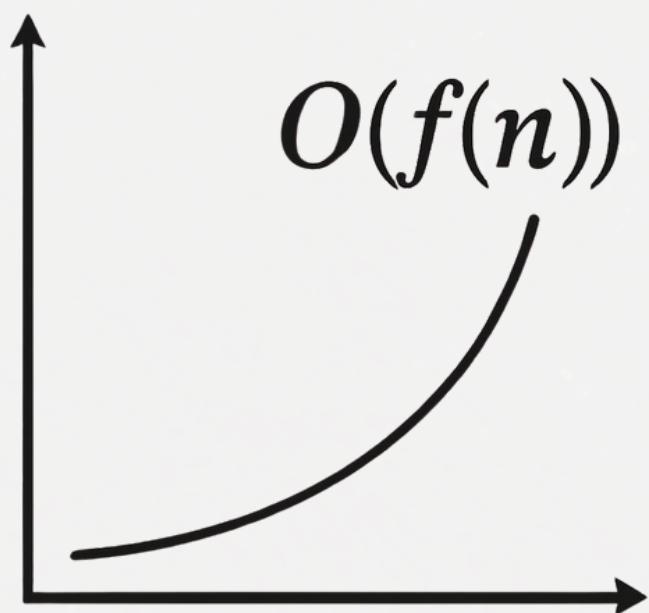
Операции

- push()
- pop()
- top()



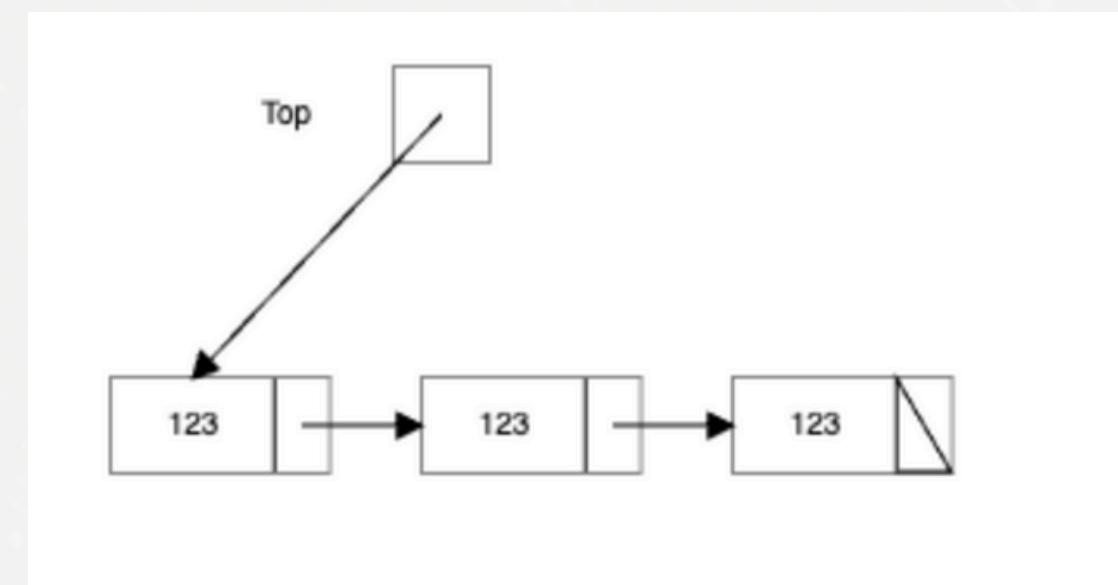
Сложность (Big O)

- $O(1)$ за всички операции



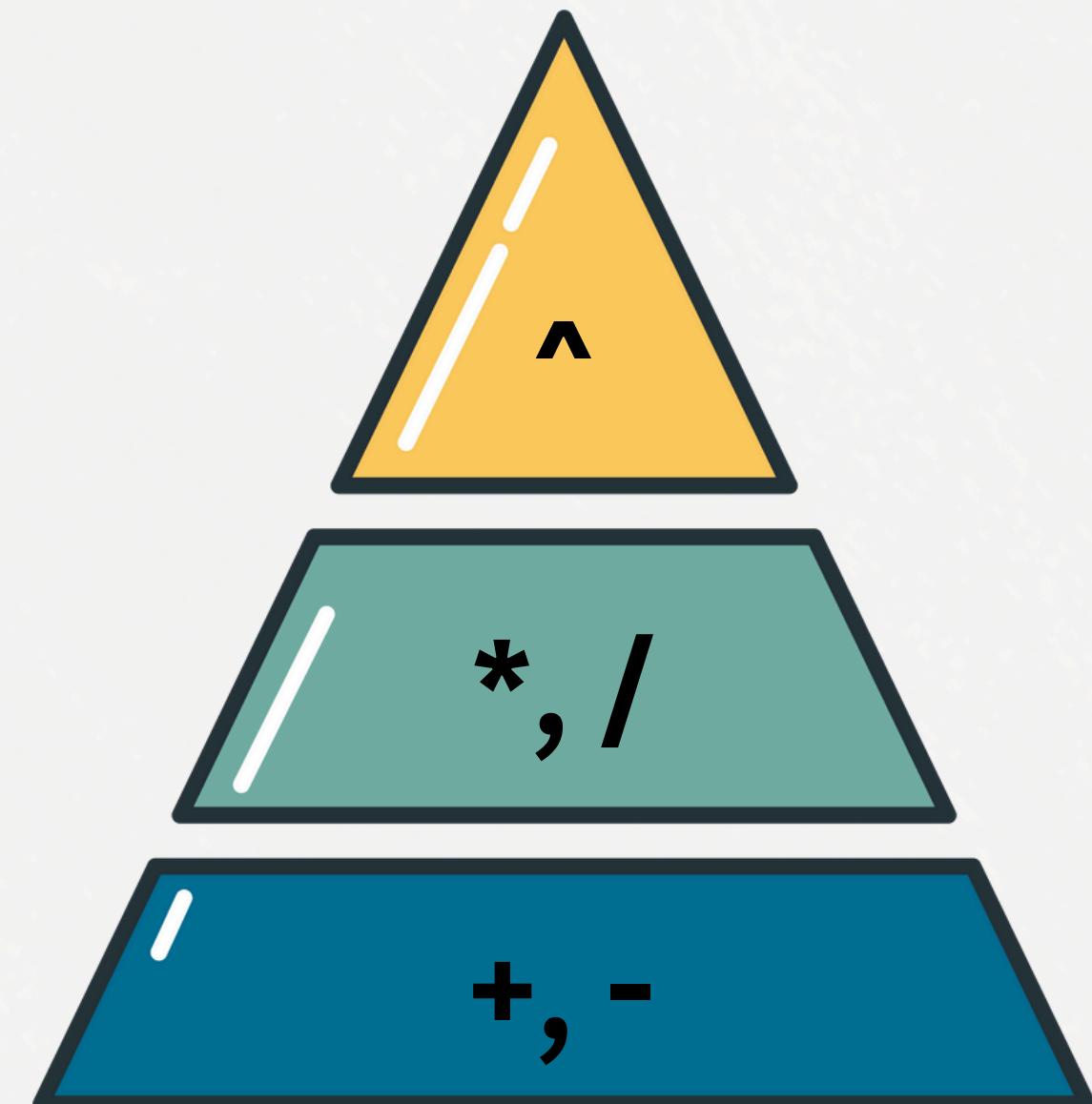
Визуализация на Стек

- [Visualisation Website](#)

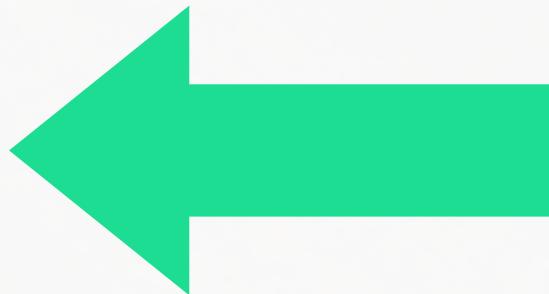


Приоритет на операциите

Оператор	Приоритет	Пример
\wedge	3 (най-висок)	2^3
$*$, $/$	2	$4 * 5$
$+$, $-$	1 (най-нисък)	$2 + 3$

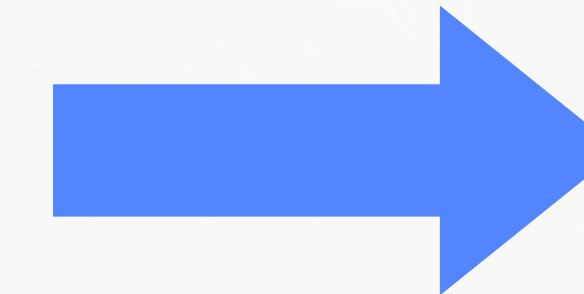
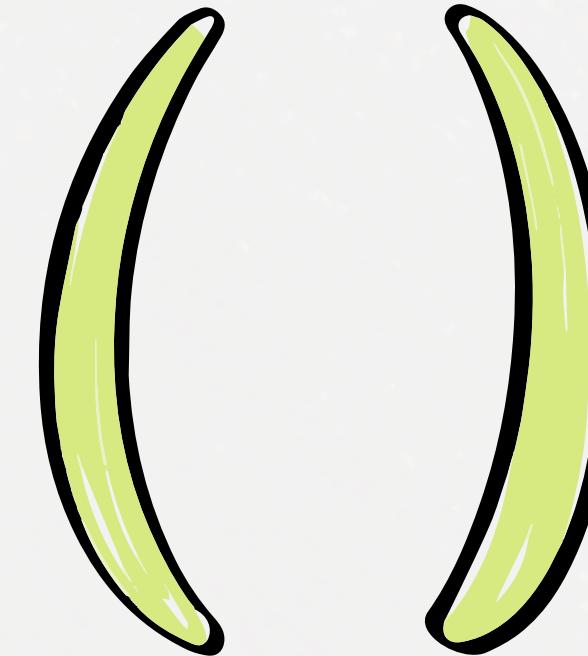


— Асоциативност на операциите



Лява

$$10 - 3 - 2 = (10 - 3) - 2 = 5$$

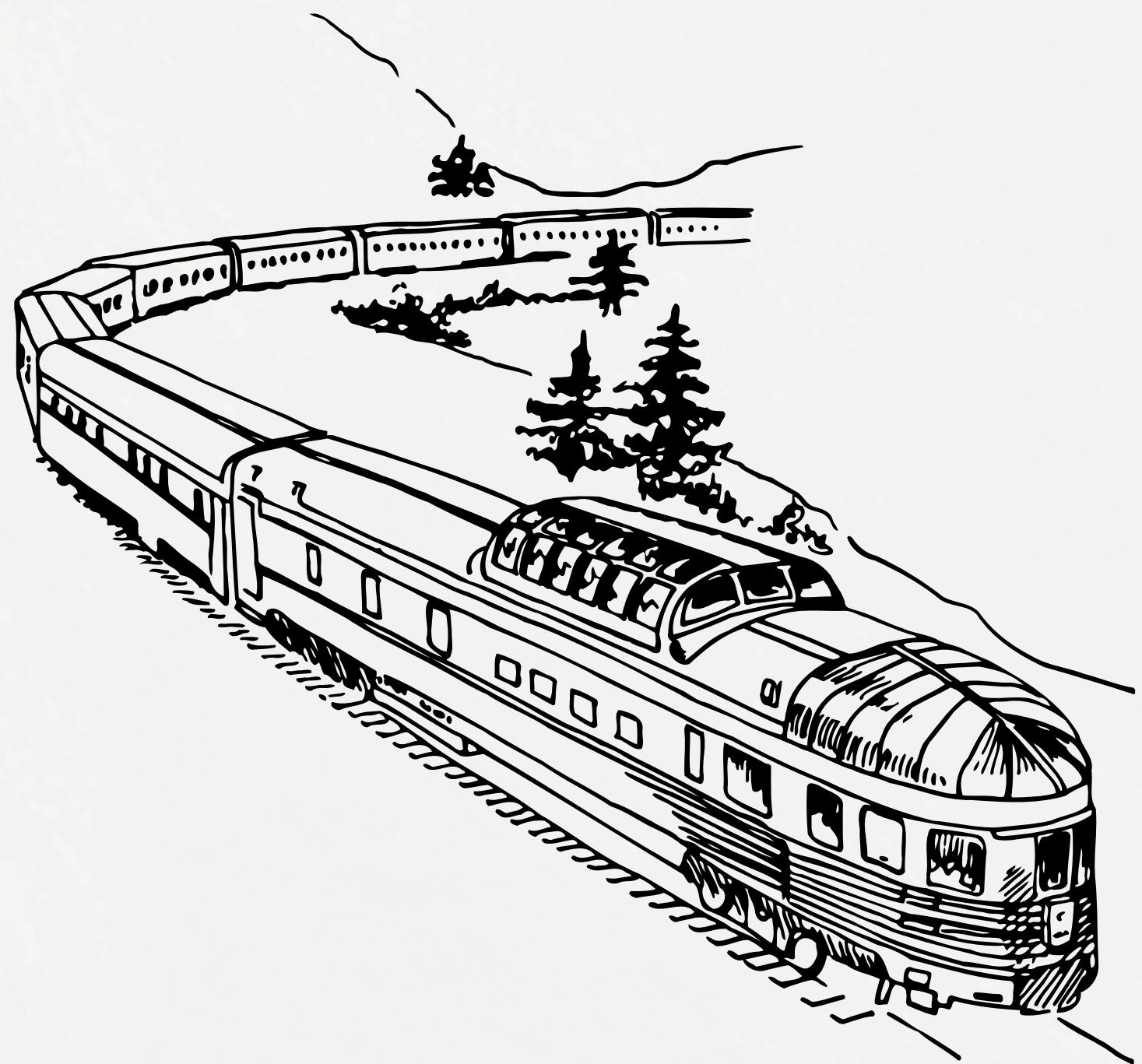


Дясна

$$2^3^2 = 2^{(3^2)} = 64$$

⚠ НЕ $2^3^2 = (2^3)^2 = 64$

Shunting Yard

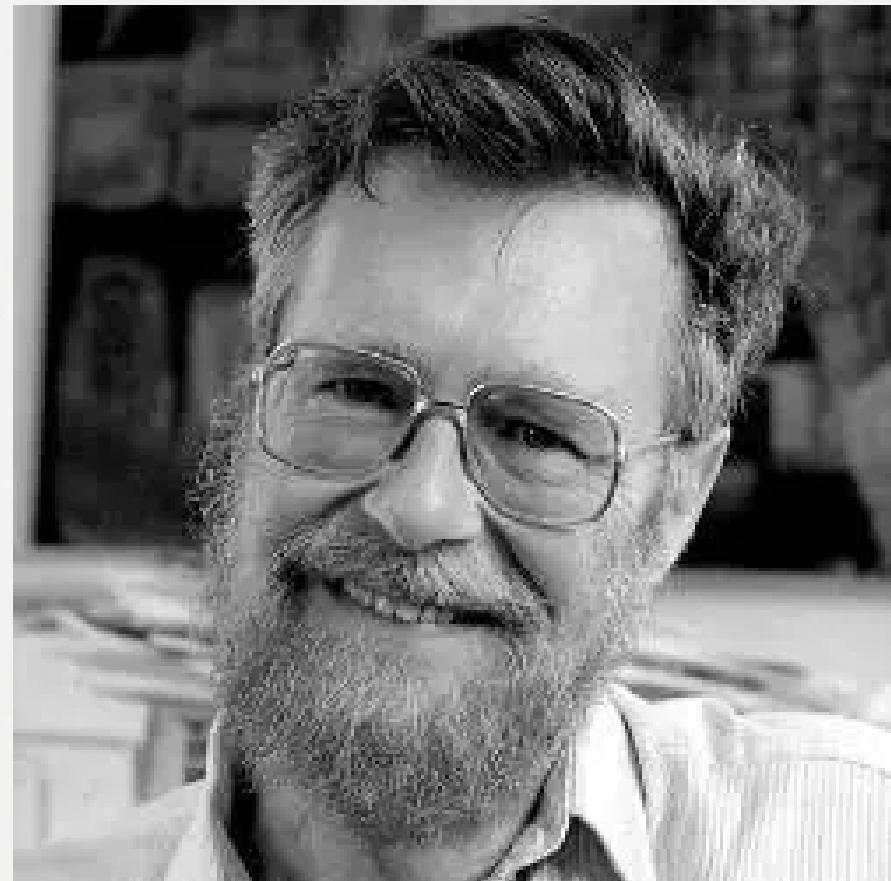


История и Концепция

Shunting Yard алгоритъм

История

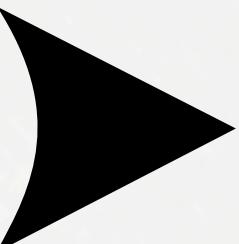
- Edsger Dijkstra, 1961
- Цел: Конвертира Инфикс \rightarrow Обратна полска нотация
- Защо? ОПН е лесен за оценка от компютър



Инфикс Нотация



Shunting Yard



Обратна Полска
Нотация

Структура на Алгоритъма

Shunting Yard алгоритъм



Стек за оператори
(временно)



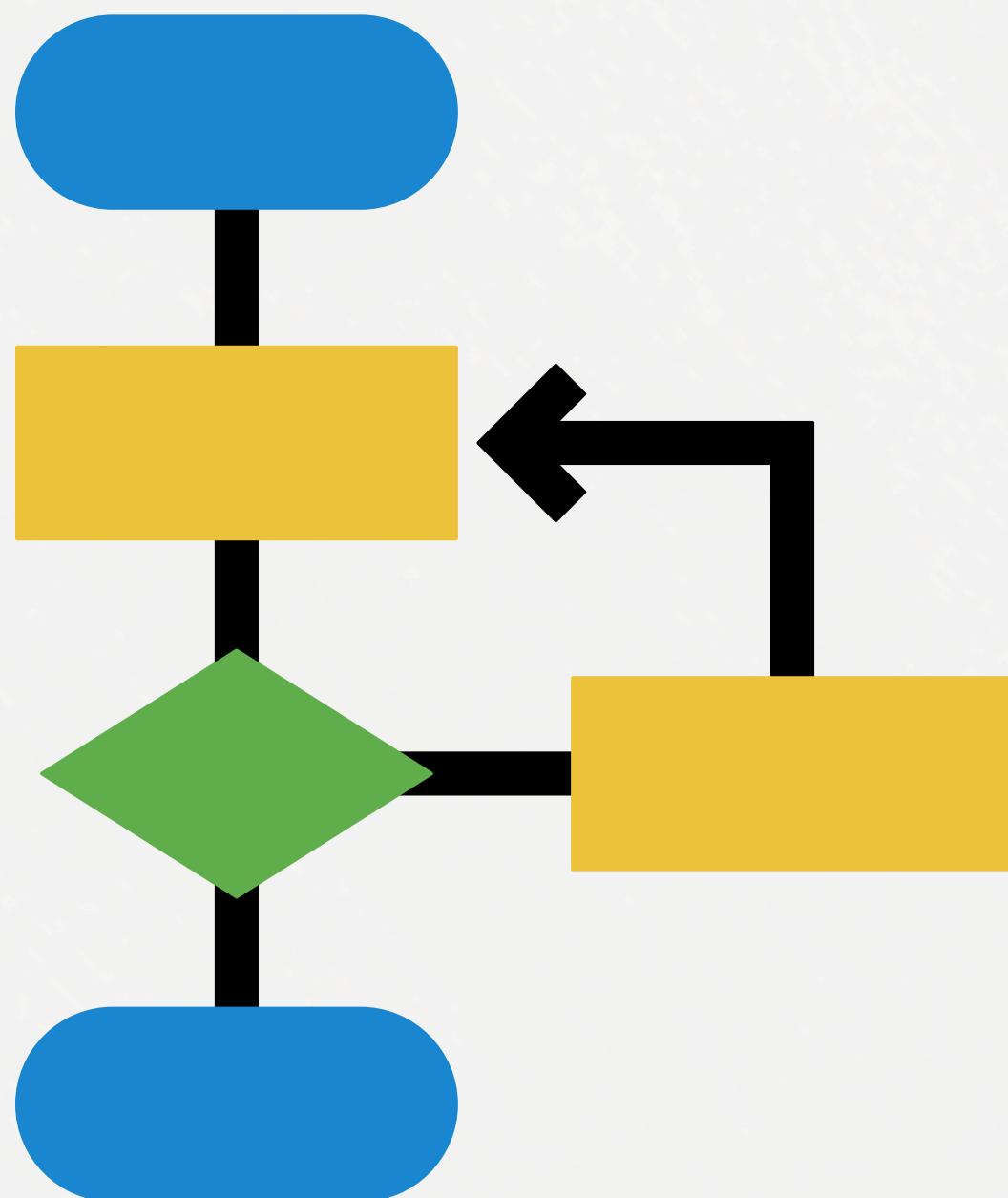
Опашка за изхода
(результат)

Правила на Алгоритъма

Shunting Yard алгоритъм

Правила

1. Четем един елемент от входа
2. Ако е число
 - а. Пренасяме го директно към изхода
3. Ако е оператор (+, -, *, /)
 - а. Докато на върха на стека има оператор с по-висок приоритет или (равен приоритет и има лява асоциативност):
 - i. Изваждаме го от стека
 - ii. Слагаме го в изходната опашка
 - б. След това слагаме новия оператор в стека
4. Ако е лява скоба (
 - а. Слагаме я в стека
5. Ако е дясна скоба)
 - а. Изваждаме от стека и слагаме в изходната опашка, докато не стигнем лявата скоба
 - б. Махаме лявата скоба (но НЕ я слагаме в изхода)



Пример за Изпълнение

Shunting Yard алгоритъм Визуализация

Пример: $3 + 4 * 2$

Токен	Стек	Изход (Опашка)
3	[]	[3]
+	[+]	[3]
4	[+]	[3, 4]
*	[*, +]	[3, 4]
2	[*, +]	[3, 4, 2]
край	[]	[3, 4, 2, *, +]

Пример за Изпълнение

Shunting Yard алгоритъм Визуализация

Пример: $3 * 4 + 2$

Токен	Стек	Изход (Опашка)
3	[]	[3]
*	[*]	[3]
4	[*]	[3, 4]
+	[+]	[3, 4, *]
2	[+]	[3, 4, *, 2]
край	[]	[3, 4, *, 2, +]

Пример за Изпълнение

Shunting Yard алгоритъм Визуализация

Пример: $3 * 4 / 2 + 2$

Токен	Стек	Изход (Опашка)
3	[]	[3]
*	[*]	[3]
4	[*]	[3, 4]
/	[/, *]	[3, 4]
2	[/, *]	[3, 4, 2]
+	[+]	[3, 4, 2, /, *, 2]
2	[+]	[3, 4, 2, /, *, 2]
край	[]	[3, 4, 2, /, *, 2, +]

Правила на Алгоритъма за Изпълнение

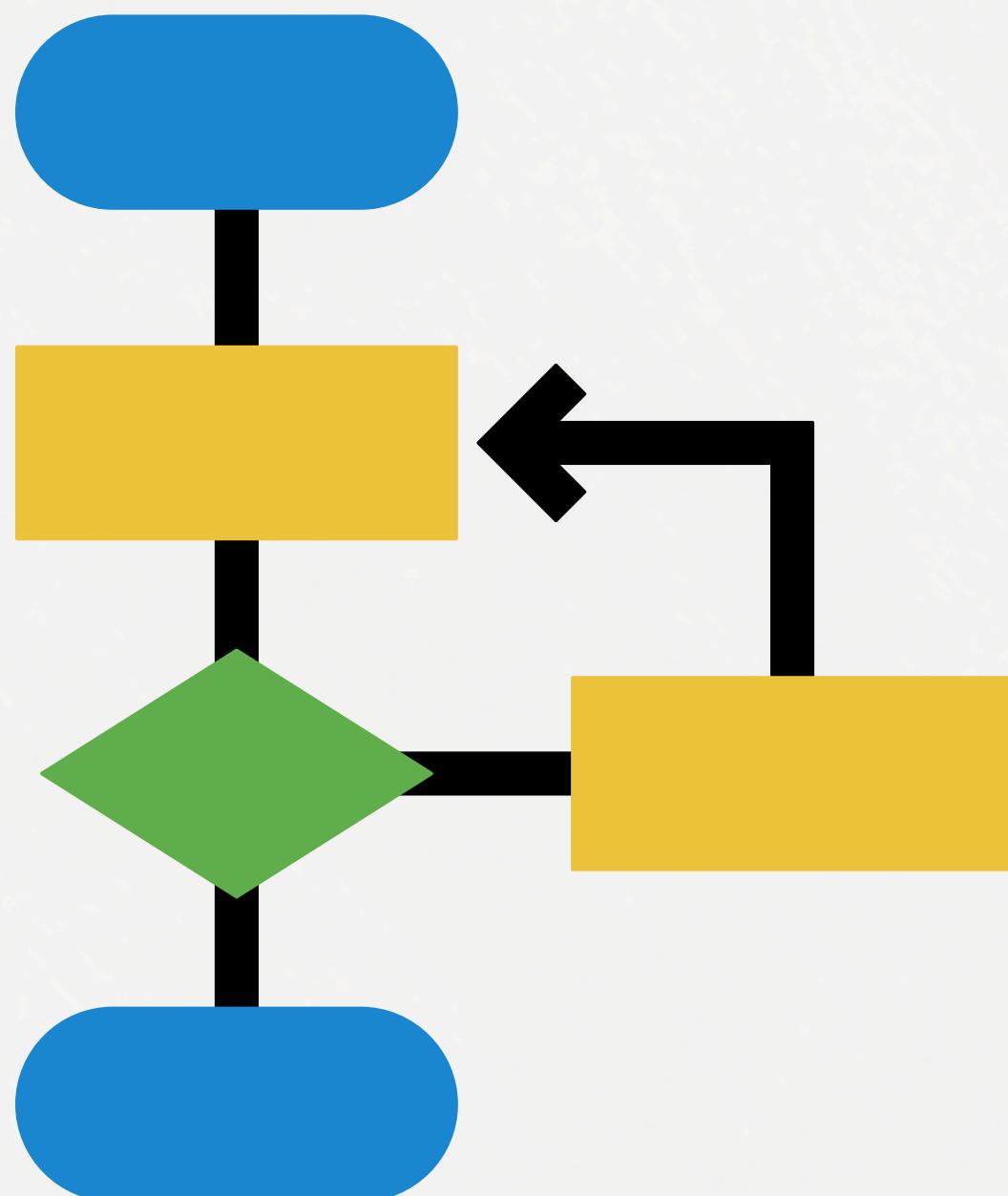
Изпълняване на Обратен Полски Запис

Правила

1. Четем елемент от входа
2. Ако е число
 - a. Слагаме го в стека
3. Ако е оператор (+, -, *, /)
 - a. Изваждаме две числа от стека (внимание на реда!)
 - b. Изпълняваме операцията
 - c. Слагаме резултата обратно в стека
4. В края
 - a. В стека остава само един елемент - това е крайният резултат

Структура

- Стек



Пример за Изпълнение

Пресмятаме Обратен Полски Запис

Пример: 3, 4, 2, *, +

Токен	Стек
3	[3]
4	[3, 4]
2	[3, 4, 2]
*	[3, 8]
+	[11]
край	[11]

1. pop 2 числа
2. изпълняване на операцията
3. push

Пример за Изпълнение

Shunting Yard алгоритъм

Пример: $(4 + 3 * 20) / (10 + 3)$

Токен	Стек	Изход (Опашка)
([()	[]
4	[()	[4]
+	[+, ()]	[4]
3	[+, ()]	[4, 3]
*	[*, +, ()]	[4, 3]
20	[*, +, ()]	[4, 3, 20]
)	[]	[4, 3, 20, *, +]
/	[/]	[4, 3, 20, *, +]

Токен	Стек	Изход (Опашка)
/	[/]	[4, 3, 20, *, +]
([(, /)]	[4, 3, 20, *, +]
10	[(, /)]	[4, 3, 20, *, +, 10]
+	[+, (, /)]	[4, 3, 20, *, +, 10]
3	[+, (, /)]	[4, 3, 20, *, +, 10, 3]
)	[/]	[4, 3, 20, *, +, 10, 3, +]
	[/]	[4, 3, 20, *, +, 10, 3, +, /]
Край	[]	[4, 3, 20, *, +, 10, 3, +, /]

— Какво може да се обърка?

Грешка при:

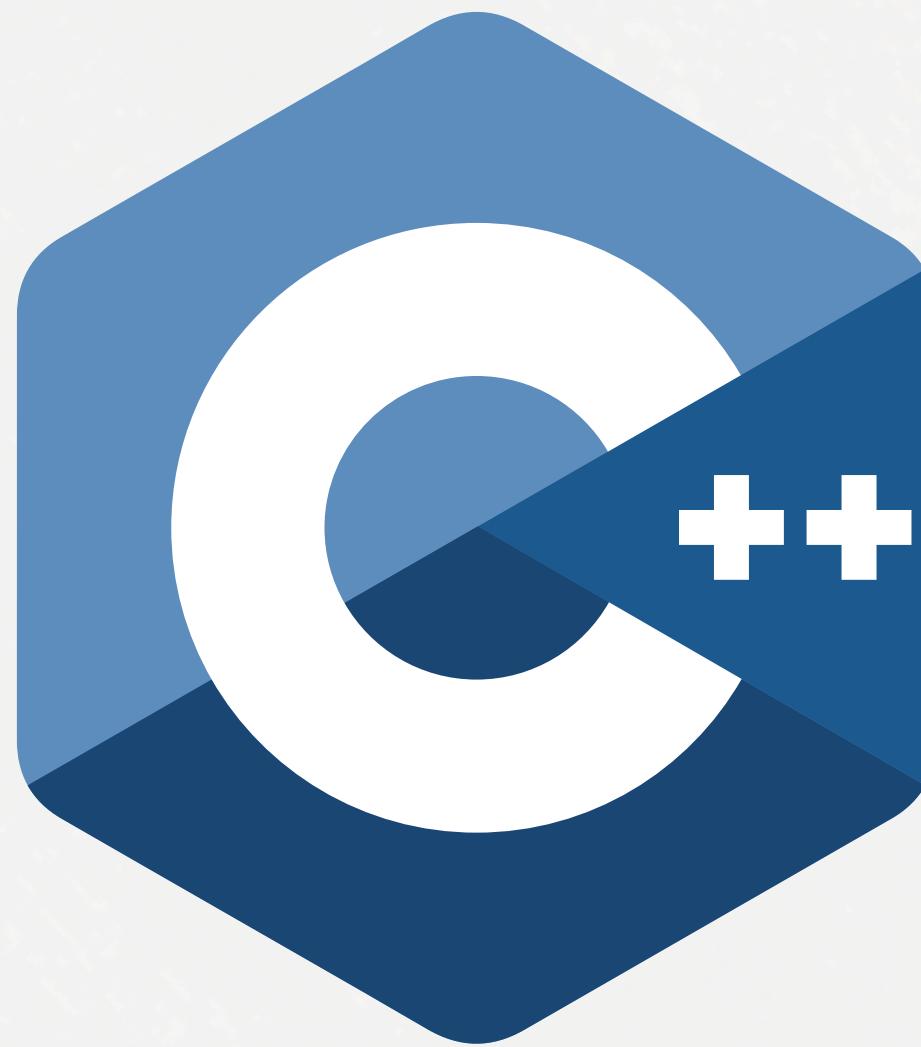
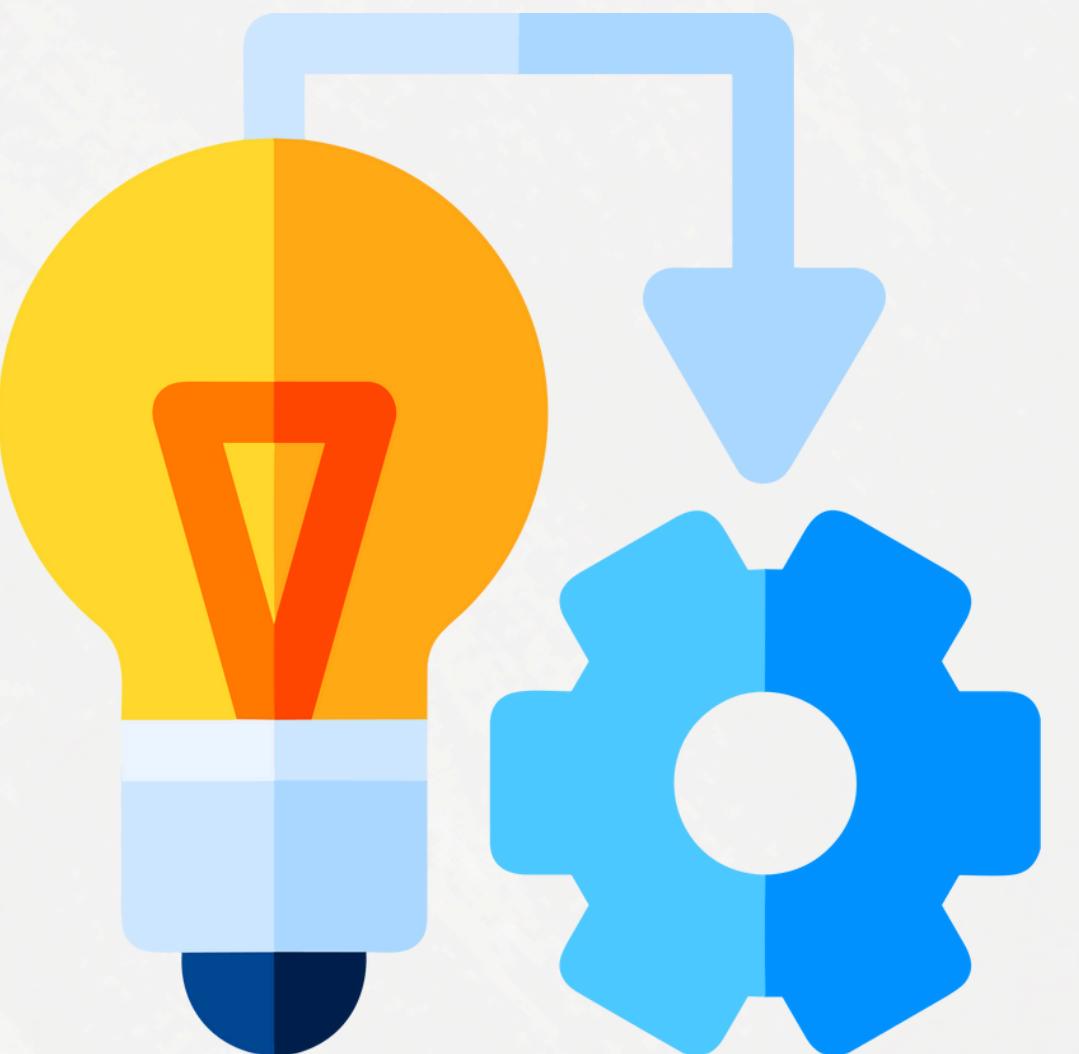
- Недостатъчно операнди
- Твърде много операнди
- Деление на нула

Грешени изрази

- [1, 2, +, +]
- [1, 0, /]
- [1, 2, 3, +]



— C++ Имплементация



C++ Имплементация

Компоненти:

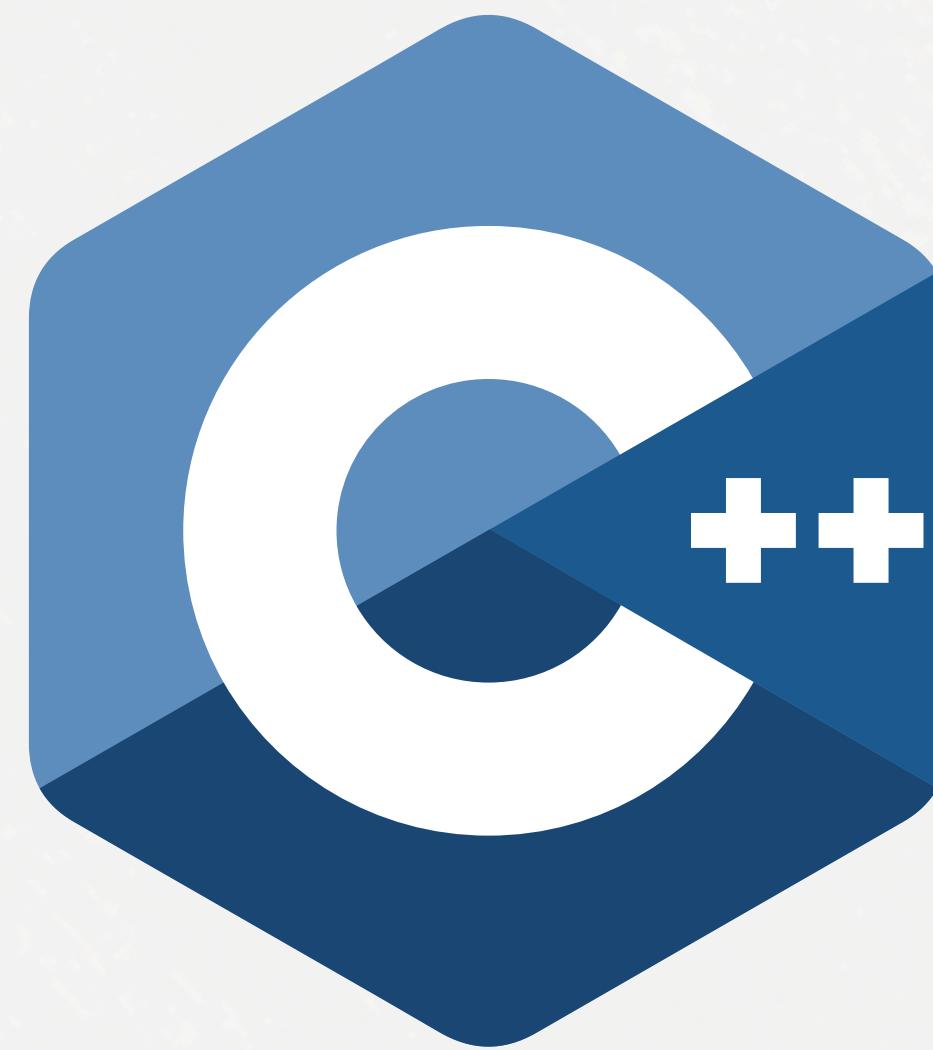
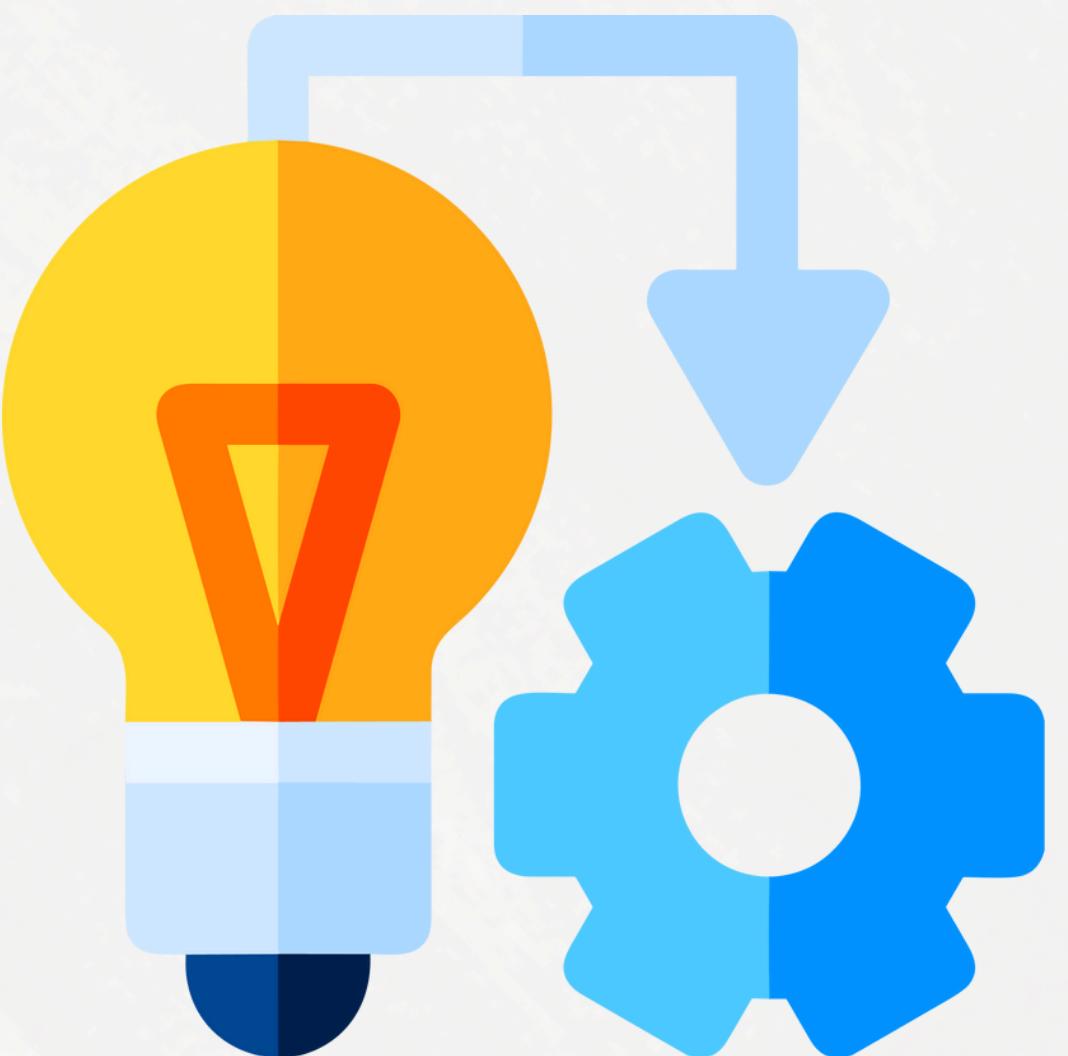
- Стек - #include <stack>
- Опашка - #include <queue>

Имплементация на алгоритма

- while, for
- if, else



Напреднали Теми



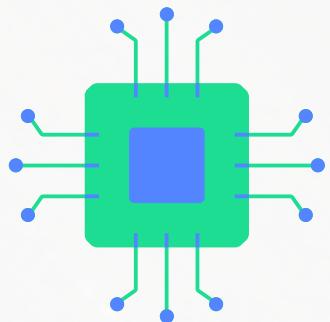
Напреднали Теми

Разширяване на имплементацията

- Унарен минус
 - $2 * -5$
- Разширяване с функции
 - $\sin(x)$, \sqrt{x}
- Разширяване с променливи
 - $x + 1 * y$



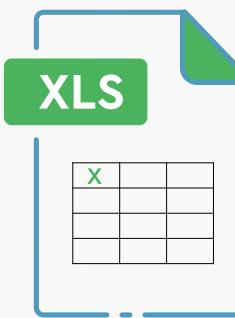
Къде се използва постфиксна нотация в реалния свят?



Компилатори
`if (a > b + 1) ...`



Калкулатори
(всички научни калк.)



Ексел формули
`(A1 + B2 * C3)`

Благодаря

ЗА ВНИМАНИЕТО

Александър Айтov

ФМИ СДП 2025