

Стек

Трифон Трифонов

Структури от данни и програмиране, спец. Компютърни науки, 2 поток, 2024/25 г.

10–17 октомври 2024 г.

Тази презентация е достъпна под лиценза Creative Commons Признание-Некомерсиално-Споделяне на споделеното 4.0 Международен



"Stack of Rocks, Cattle Point, San Juan Island" by Ryan Harvey
CC BY SA-2.0

АТД: стек

Хомогенна линейна структура с организация „последен влязъл — пръв излязъл“ (LIFO)

Операции

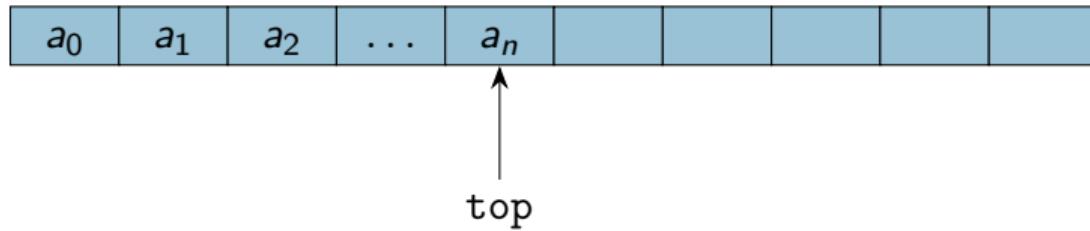
- `create()` — създаване на празен стек
- `empty()` — проверка за празнота на стек
- `push(x)` — включване на елемент на стек
- `pop()` — изключване на елемент от стек
- `peek()` — последен елемент на стека

АТД: стек

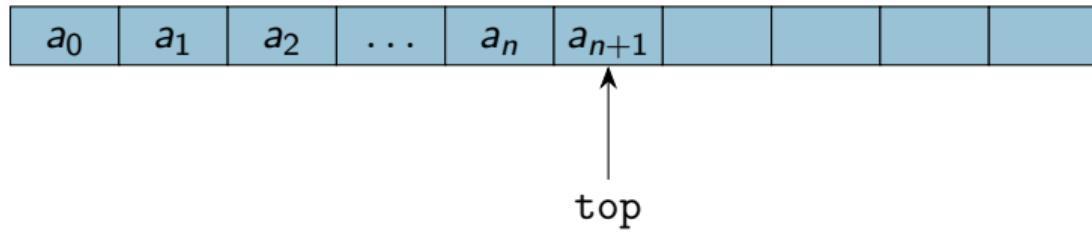
Свойства на операциите

- `create().empty() = true`
- `s.push(x).empty() = false`
- `create().peek(), create().pop()` — **грешка**
- `s.push(x).peek() = x`
- `s.push(x).pop() = s`

Последователно представяне на стек

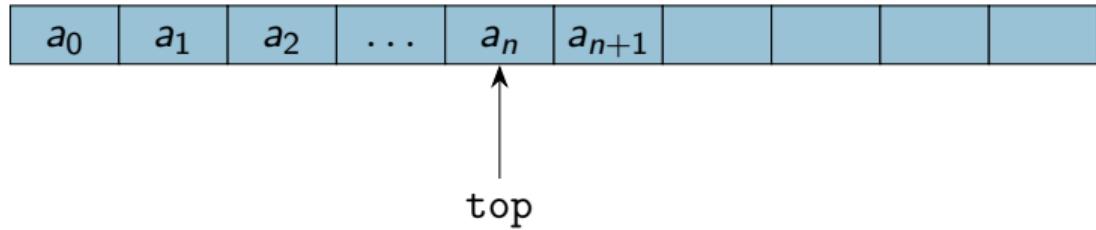


Последователно представяне на стек



- включване на елемент (push)

Последователно представяне на стек



- включване на елемент (push)
- изключване на елемент (pop)

Ограничения на последователния стек

- Нашата реализация изисква предварително да зададем горна граница на броя на елементите в стека!

Ограничения на последователния стек

- Нашата реализация изисква предварително да зададем горна граница на броя на елементите в стека!
- Ако стекът се препълни, програмата няма да може да продължи да работи... въпреки че компютърът има много налична свободна памет

Ограничения на последователния стек

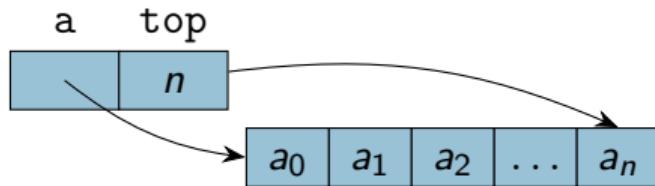
- Нашата реализация изисква предварително да зададем горна граница на броя на елементите в стека!
- Ако стекът се препълни, програмата няма да може да продължи да работи... въпреки че компютърът има много налична свободна памет
- Дали е възможно стекът да се „разширява“ при нужда?

Разширяващ се стек

- Обектът няма да съдържа целия масив
- Ще се пази указател към масив в динамичната памет

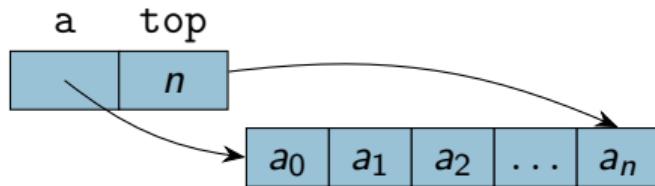
Разширяващ се стек

- Обектът няма да съдържа целия масив
- Ще се пази указател към масив в динамичната памет



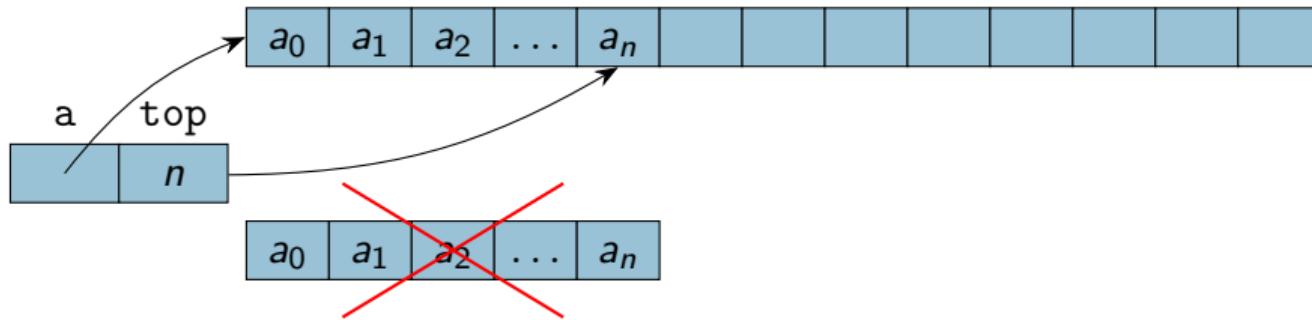
Разширяващ се стек

- Обектът няма да съдържа целия масив
- Ще се пази указател към масив в динамичната памет
- При нужда стекът ще се разширява



Разширяващ се стек

- Обектът няма да съдържа целия масив
- Ще се пази указател към масив в динамичната памет
- При нужда стекът ще се разширява



Ограничения на разширяващия се стек

- При разширяване трябва да се копират всички съществуващи данни!

Ограничения на разширяващия се стек

- При разширяване трябва да се копират всички съществуващи данни!
- Операцията push обикновено е бърза, но ако се случи да правим разширение, може да е доста по-бавна

Ограничения на разширяващия се стек

- При разширяване трябва да се копират всички съществуващи данни!
- Операцията `push` обикновено е бърза, но ако се случи да правим разширение, може да е доста по-бавна
- Ако стекът се пълни рядко, то в по-голямата част от живота му паметта няма да се използва

Ограничения на разширяващия се стек

- При разширяване трябва да се копират всички съществуващи данни!
- Операцията `push` обикновено е бърза, но ако се случи да правим разширение, може да е доста по-бавна
- Ако стекът се пълни рядко, то в по-голямата част от живота му паметта няма да се използва
- Дали може да се направи стек, при който:

Ограничения на разширяващия се стек

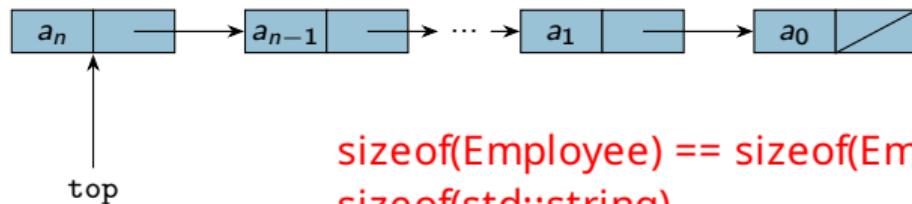
- При разширяване трябва да се копират всички съществуващи данни!
- Операцията `push` обикновено е бърза, но ако се случи да правим разширение, може да е доста по-бавна
- Ако стекът се пълни рядко, то в по-голямата част от живота му паметта няма да се използва
- Дали може да се направи стек, при който:
 - не се налага копиране на памет

Ограничения на разширяващия се стек

- При разширяване трябва да се копират всички съществуващи данни!
- Операцията `push` обикновено е бърза, но ако се случи да правим разширение, може да е доста по-бавна
- Ако стекът се пълни рядко, то в по-голямата част от живота му паметта няма да се използва
- Дали може да се направи стек, при който:
 - не се налага копиране на памет
 - не се държи излишна памет

Свързано представяне на стек

Представяме стека като „верига“ от двойни кутии



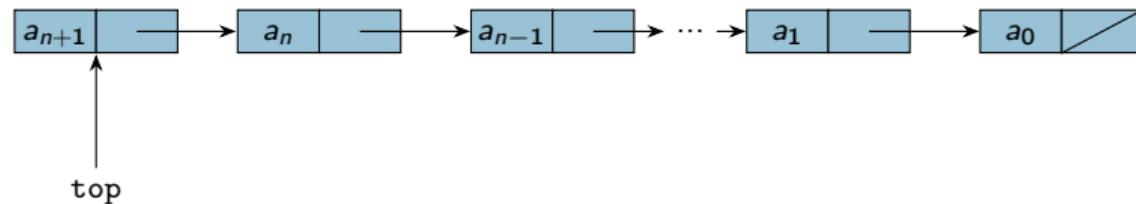
`sizeof(Employee) == sizeof(Employee) +
sizeof(std::string)`

```
template <typename T>
struct StackElement {
    T data;
    StackElement* next;
};
```

```
struct Employee {
    std::string name;
    Employee boss;
};
```

Свързано представяне на стек

Представяме стека като „верига“ от двойни кутии

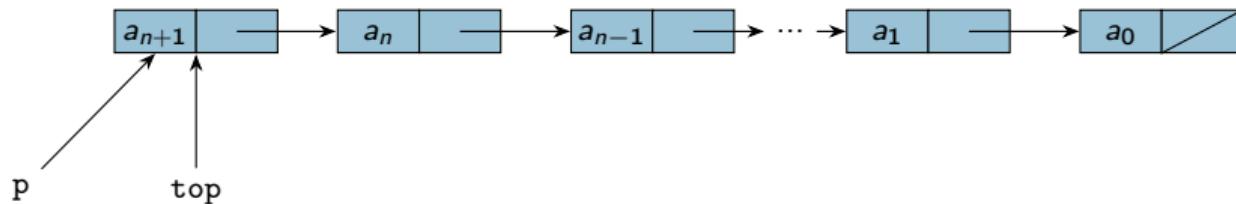


```
template <typename T>
struct StackElement {
    T data;
    StackElement* next;
};
```

- включване на елемент (push)

Свързано представяне на стек

Представяме стека като „верига“ от двойни кутии

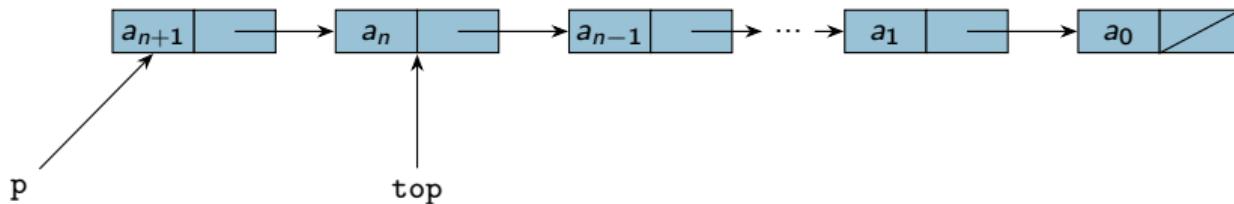


```
template <typename T>
struct StackElement {
    T data;
    StackElement* next;
};
```

- включване на елемент (push)
 - изключване на елемент (pop)

Свързано представяне на стек

Представяме стека като „верига“ от двойни кутии



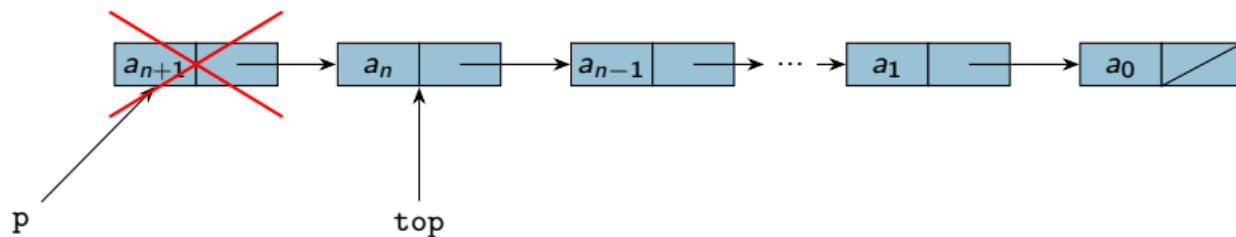
```

template <typename T>
struct StackElement {
    T data;
    StackElement* next;
};
  
```

- включване на елемент (push)
- изключване на елемент (pop)

Свързано представяне на стек

Представяме стека като „верига“ от двойни кутии



```
template <typename T>
struct StackElement {
    T data;
    StackElement* next;
};
```

- включване на елемент (push)
- изключване на елемент (pop)



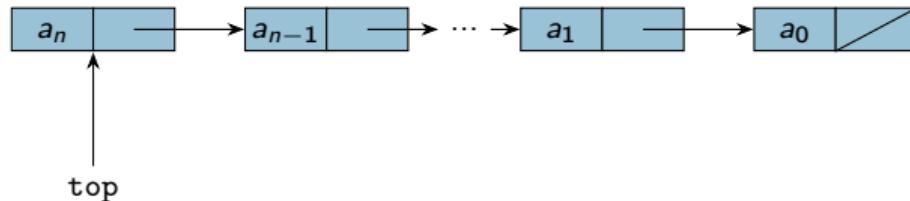
toCopy toCopy



hasCopied

Свързано представяне на стек

Представяме стека като „верига“ от двойни кутии



```

template <typename T>
struct StackElement {
    T data;
    StackElement* next;
};
  
```

- включване на елемент (push)
- изключване на елемент (pop)

Ограничения на свързания стек

- За всеки елемент се изразходва 2-3 пъти повече памет

Ограничения на свързания стек

- За всеки елемент се изразходва 2-3 пъти повече памет
- Често се заделят и освобождават малки парчета памет

Ограничения на свързания стек

- За всеки елемент се изразходва 2-3 пъти повече памет
- Често се заделят и освобождават малки парчета памет
- Какво се случва при копиране на стек?

Ограничения на свързания стек

- За всеки елемент се изразходва 2-3 пъти повече памет
- Често се заделят и освобождават малки парчета памет
- Какво се случва при копиране на стек?
 - `LinkedStack<int> s2 = s1; s1.pop(); s2.pop(); s2.push(10);`

Ограничения на свързания стек

- За всеки елемент се изразходва 2-3 пъти повече памет
- Често се заделят и освобождават малки парчета памет
- Какво се случва при копиране на стек?
 - `LinkedStack<int> s2 = s1; s1.pop(); s2.pop(); s2.push(10);`
- Какво се случва при унищожаване на стек?
`for(int i = 0; i < 1E8; i++) { LinkedStack<int> s; }`

Ограничения на свързания стек

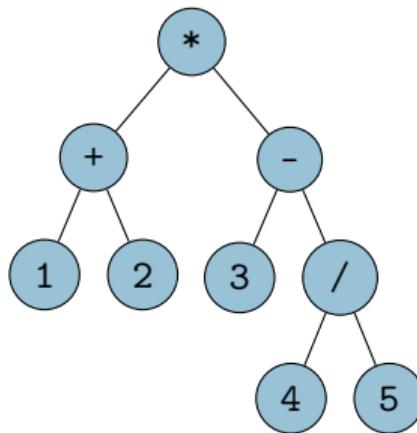
- За всеки елемент се изразходва 2-3 пъти повече памет
- Често се заделят и освобождават малки парчета памет
- Какво се случва при копиране на стек?
 - `LinkedStack<int> s2 = s1; s1.pop(); s2.pop(); s2.push(10);`
- Какво се случва при унищожаване на стек?

```
for(int i = 0; i < 1E8; i++) { LinkedStack<int> s; .... }
```

```
for(int i = 0; i < 1E8; i++) {
    LinkedStack<int>* s = new LinkedStack<int>;
    ....
    delete s;
}
```

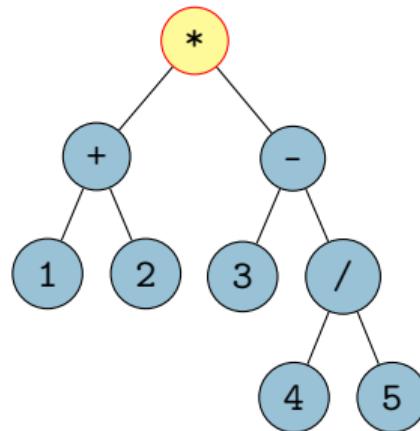
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



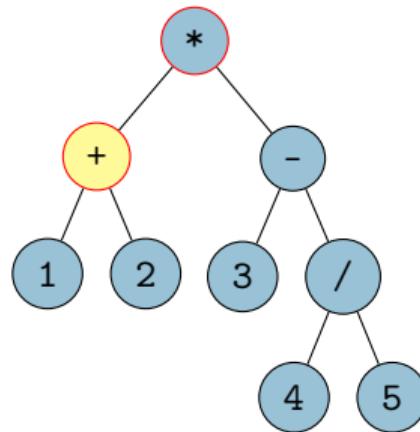
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
12+345/-*



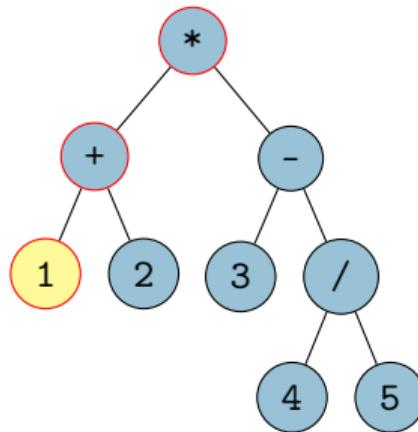
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



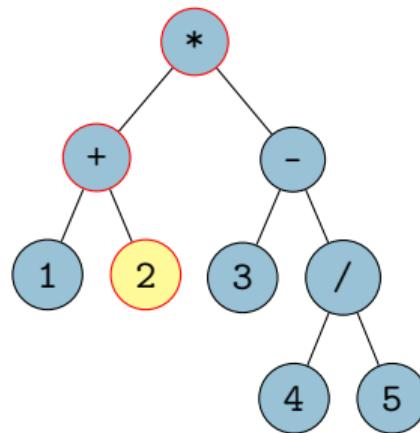
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



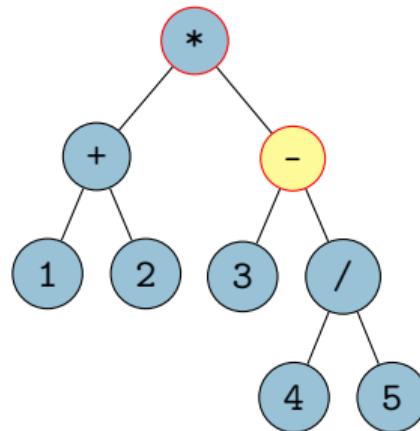
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



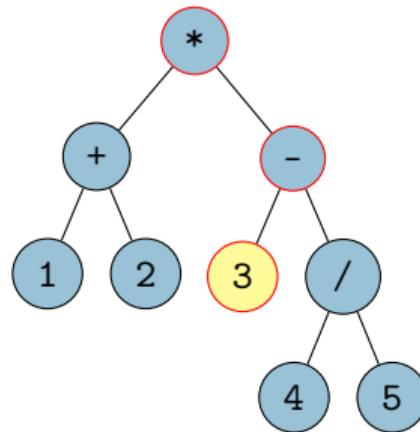
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



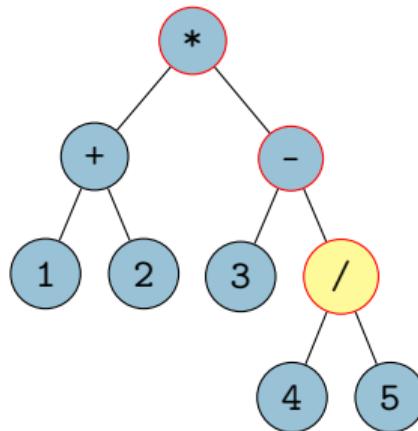
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



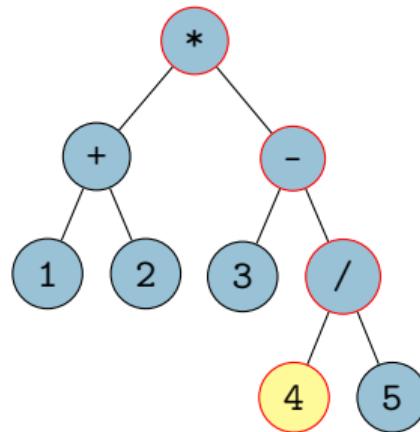
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



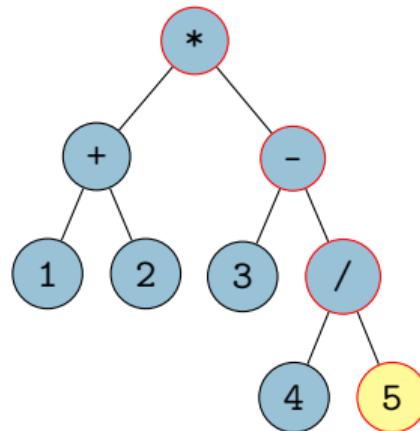
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



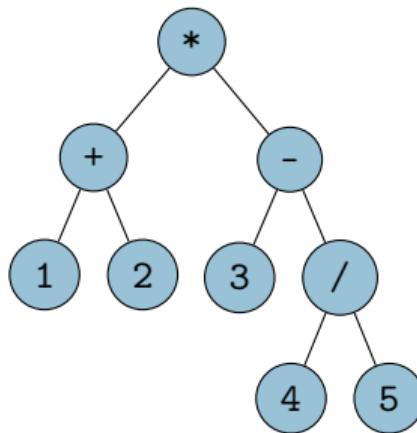
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



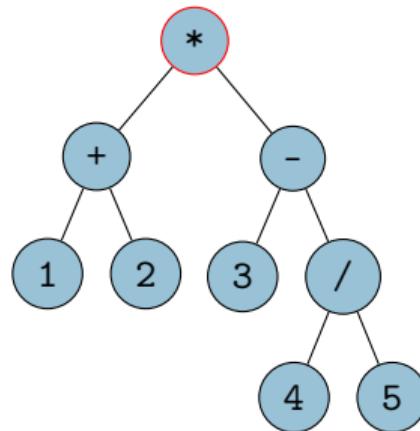
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



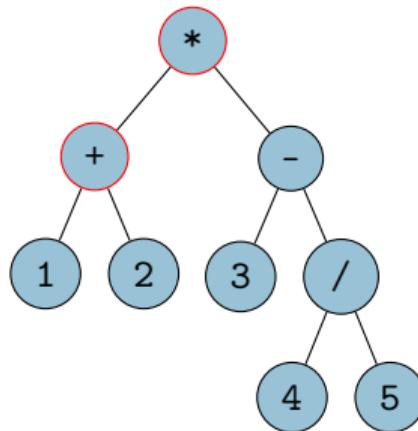
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



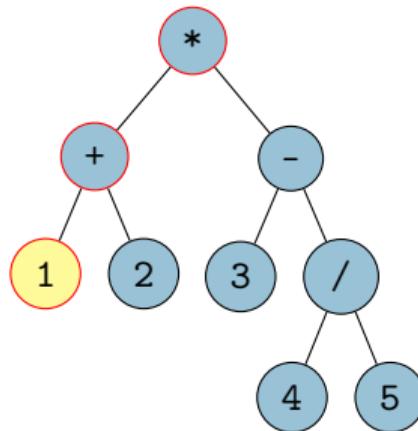
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



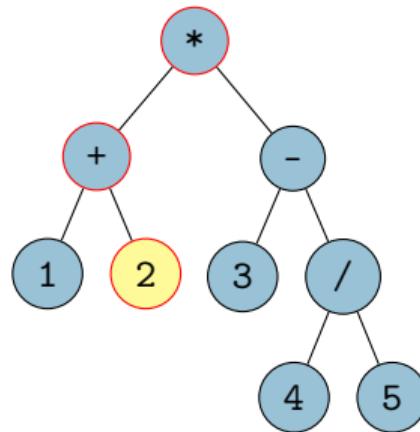
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



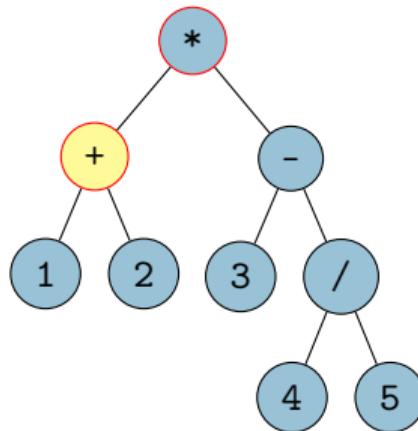
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



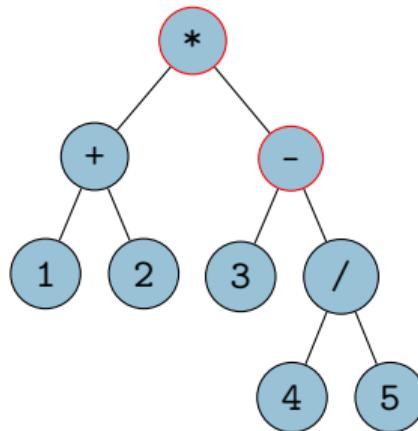
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



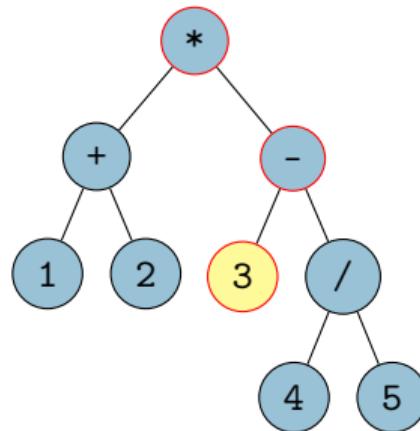
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



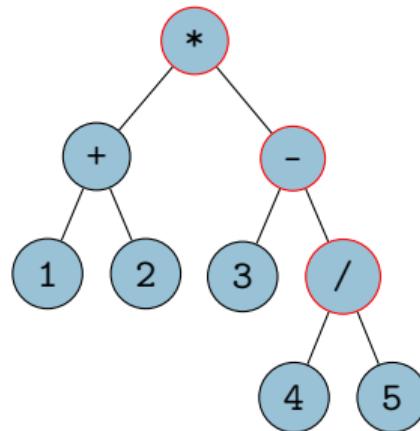
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



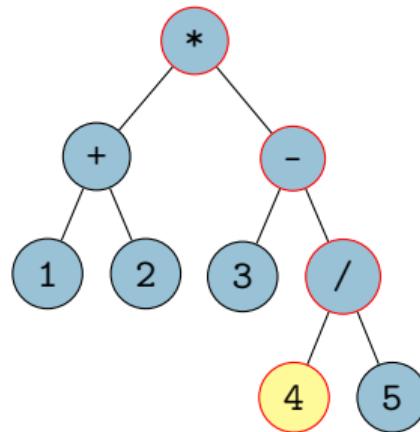
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



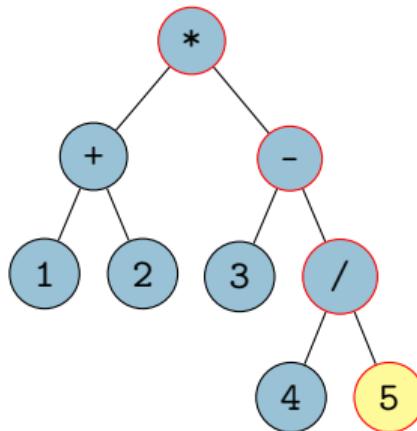
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



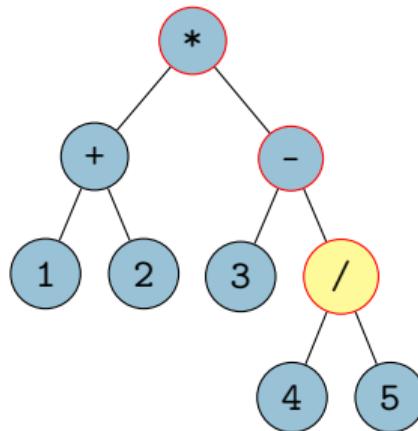
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



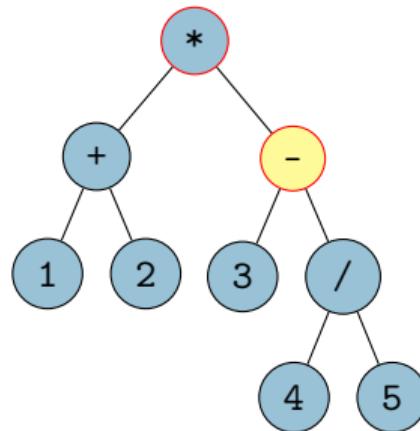
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



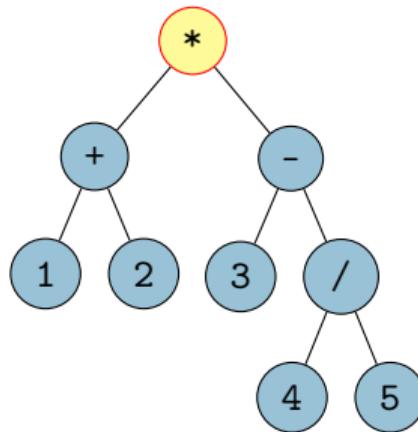
Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



Обратен полски запис

- инфиксен запис:
 $(1+2)*(3-4/5)$
- префиксен (полски) запис:
 $*+12-3/45$
- постфиксен (обратен полски) запис
 $12+345/-*$



Пресмятане на израз в обратен полски запис



Алгоритъм за пресмятане на обратен полски запис

За всеки пореден символ с от израза в обратен полски запис:

Алгоритъм за пресмятане на обратен полски запис

За всеки пореден символ с от израза в обратен полски запис:

- Ако с е цифра, добавяме стойността ѝ в стека за резултати

Алгоритъм за пресмятане на обратен полски запис

За всеки пореден символ с от израза в обратен полски запис:

- Ако с е цифра, добавяме стойността ѝ в стека за резултати
- Ако с е (двуместна) операция:

Алгоритъм за пресмятане на обратен полски запис

За всеки пореден символ с от израза в обратен полски запис:

- Ако с е цифра, добавяме стойността ѝ в стека за резултати
- Ако с е (двуместна) операция:
 - изваждаме най-горните два елемента от стека

Алгоритъм за пресмятане на обратен полски запис

За всеки пореден символ с от израза в обратен полски запис:

- Ако с е цифра, добавяме стойността ѝ в стека за резултати
- Ако с е (двуместна) операция:
 - изваждаме най-горните два елемента от стека
 - прилагаме операцията над тях

Алгоритъм за пресмятане на обратен полски запис

За всеки пореден символ с от израза в обратен полски запис:

- Ако с е цифра, добавяме стойността ѝ в стека за резултати
- Ако с е (двуместна) операция:
 - изваждаме най-горните два елемента от стека
 - прилагаме операцията над тях
 - добавяме резултата в стека

Алгоритъм за пресмятане на обратен полски запис

За всеки пореден символ с от израза в обратен полски запис:

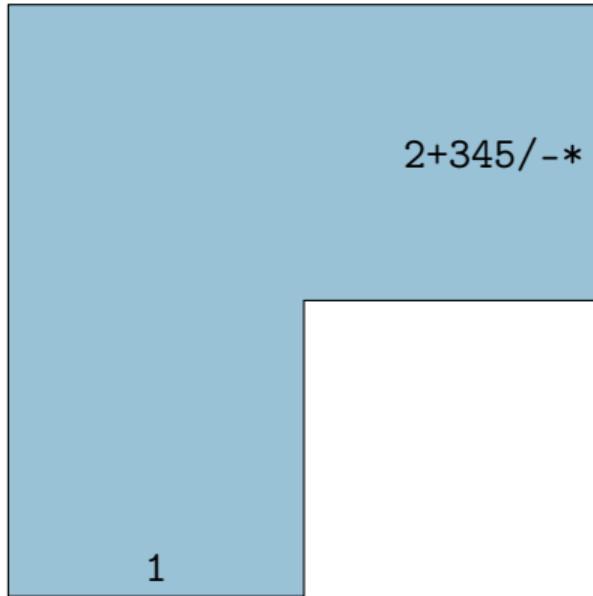
- Ако с е цифра, добавяме стойността ѝ в стека за резултати
- Ако с е (двуместна) операция:
 - изваждаме най-горните два елемента от стека
 - прилагаме операцията над тях
 - добавяме резултата в стека

В стека остава единствен елемент: крайният резултат.

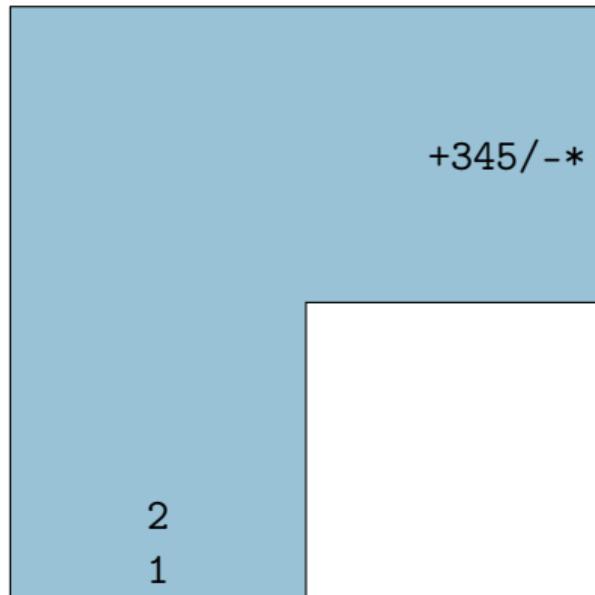
Пример: Пресмятане на израз в обратен полски запис

12+345/-*

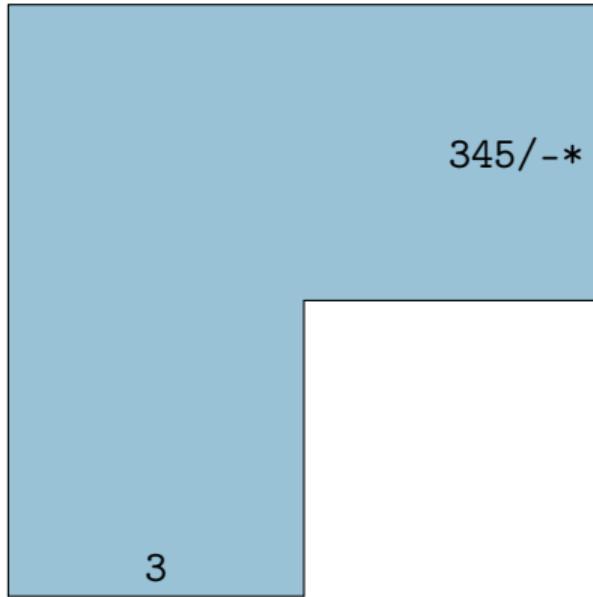
Пример: Пресмятане на израз в обратен полски запис



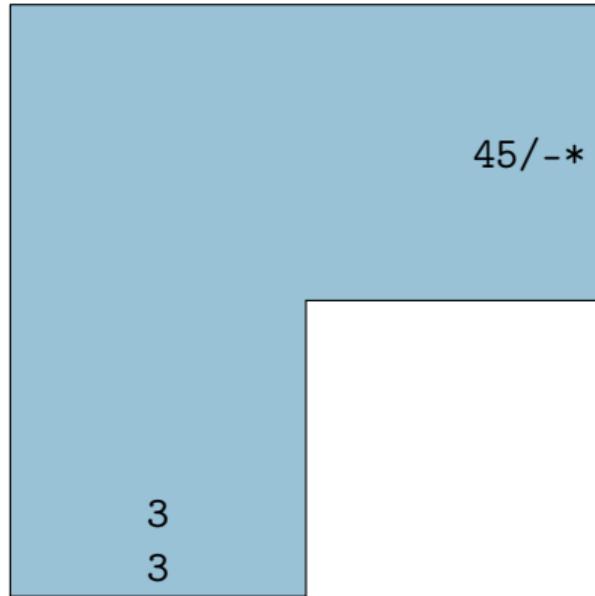
Пример: Пресмятане на израз в обратен полски запис



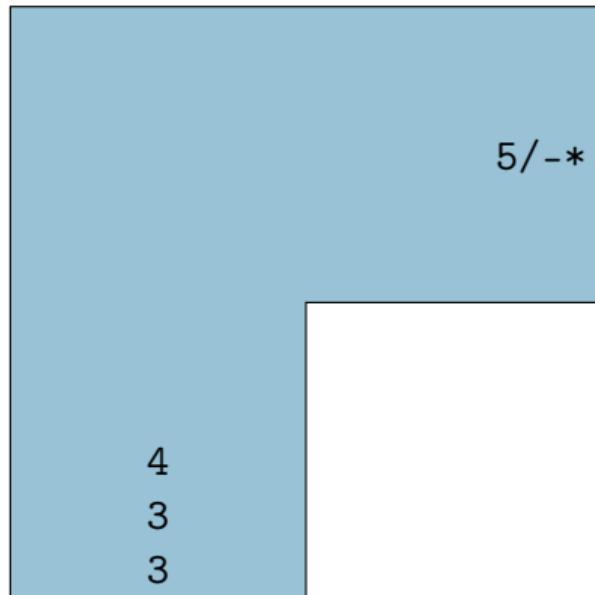
Пример: Пресмятане на израз в обратен полски запис



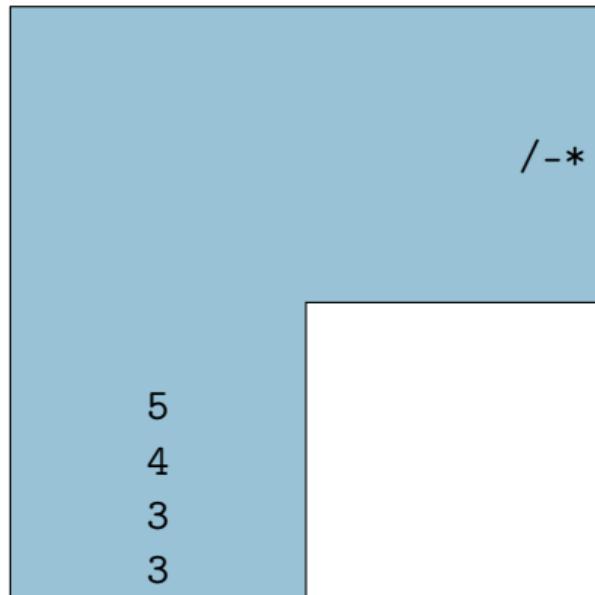
Пример: Пресмятане на израз в обратен полски запис



Пример: Пресмятане на израз в обратен полски запис



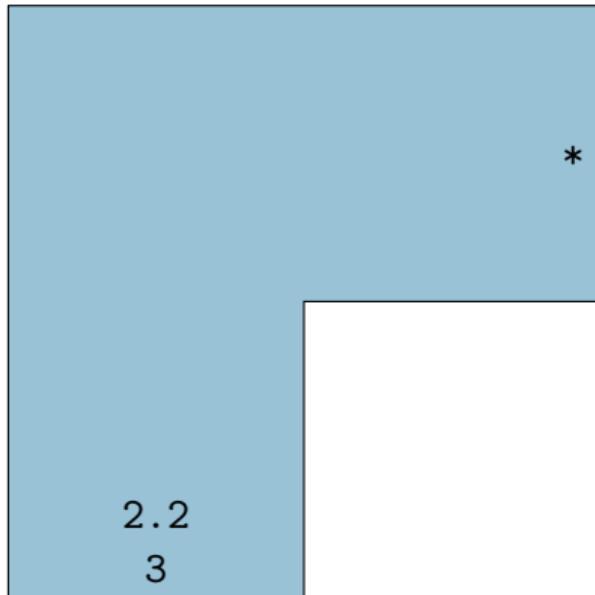
Пример: Пресмятане на израз в обратен полски запис



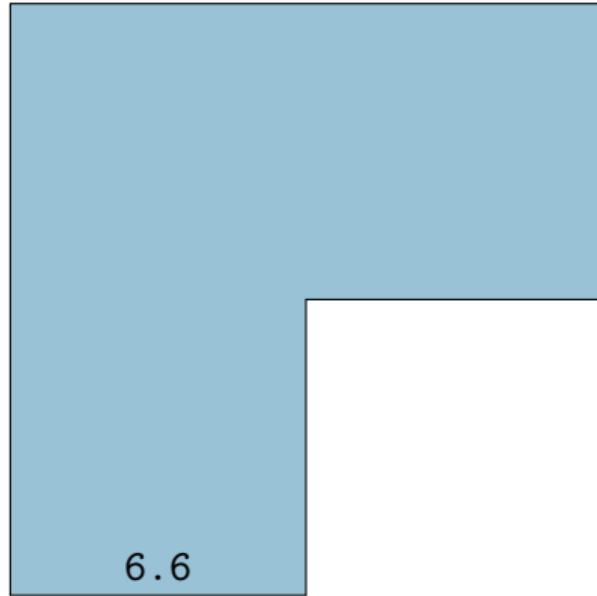
Пример: Пресмятане на израз в обратен полски запис



Пример: Пресмятане на израз в обратен полски запис



Пример: Пресмятане на израз в обратен полски запис



Преобразуване в обратен полски запис



Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в крайния резултат

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в крайния резултат
- Ако с е отваряща скоба, поставяме я в стека с операции

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в крайния резултат
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция, поставяме я в стека с операции

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в крайния резултат
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция, поставяме я в стека с операции
- Ако с е затваряща скоба, последователно изваждаме от стека и записваме в резултата всички операции до достигане на отваряща скоба

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в крайния резултат
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция, поставяме я в стека с операции
- Ако с е затваряща скоба, последователно изваждаме от стека и записваме в резултата всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека и записваме в резултата всички останали операции.

Пример: Преобразуване в обратен полски запис

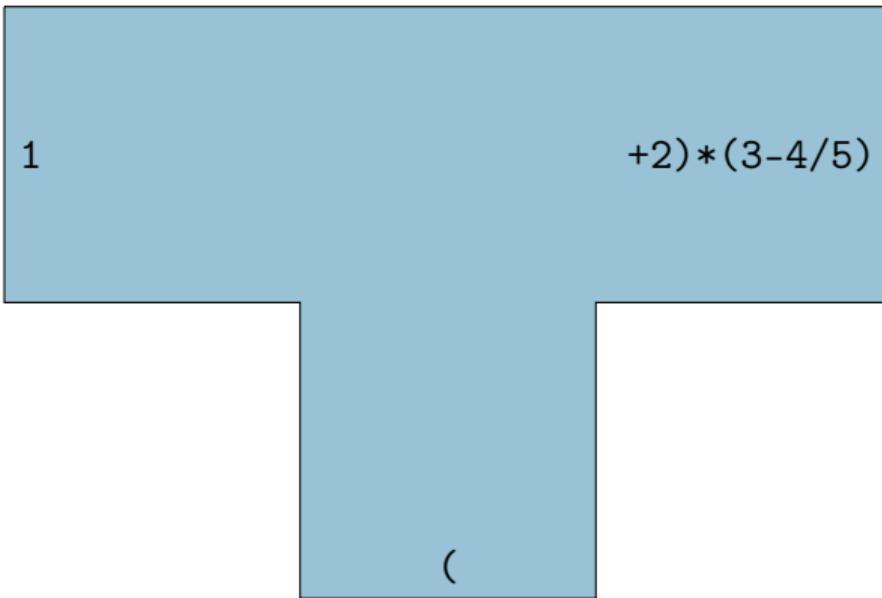
$$(1+2)*(3-4/5)$$

Пример: Преобразуване в обратен полски запис

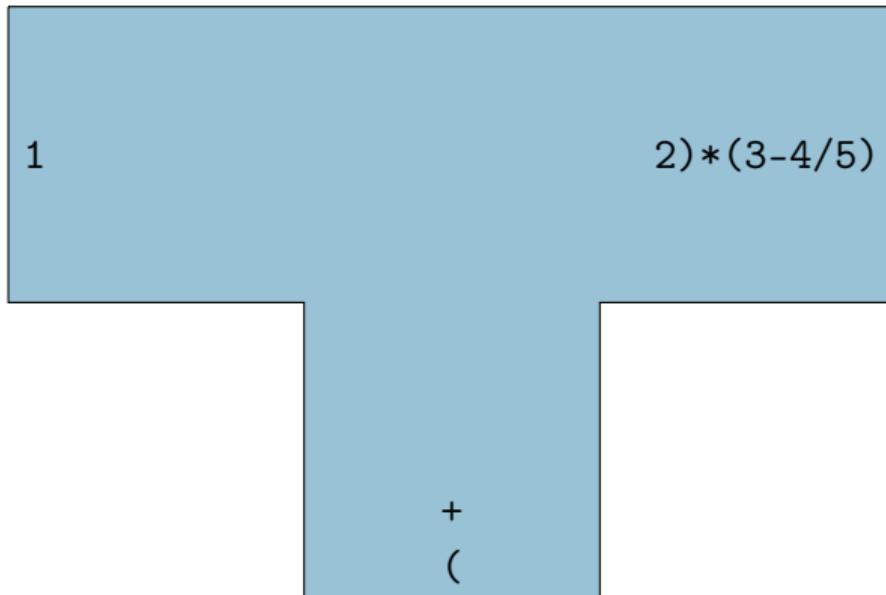
 $1+2)*(3-4/5)$

()

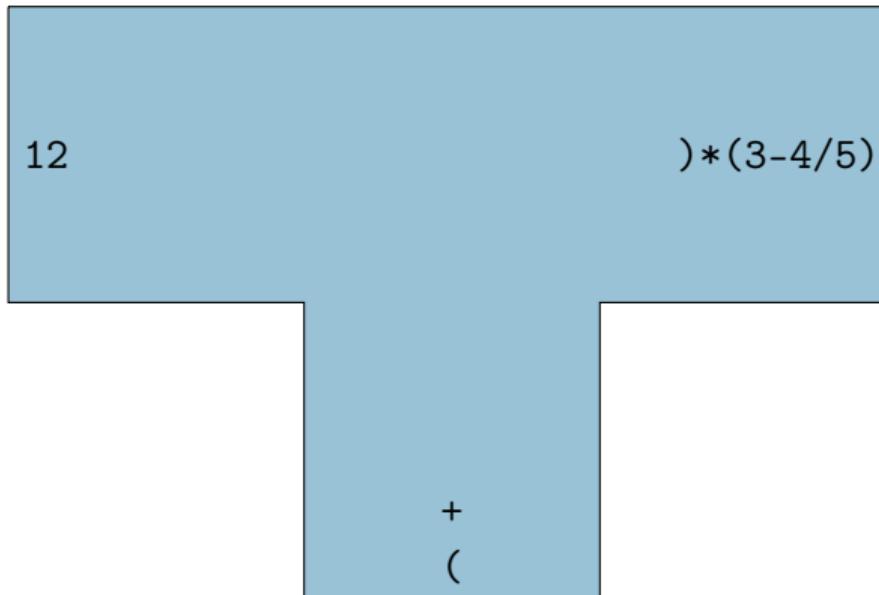
Пример: Преобразуване в обратен полски запис



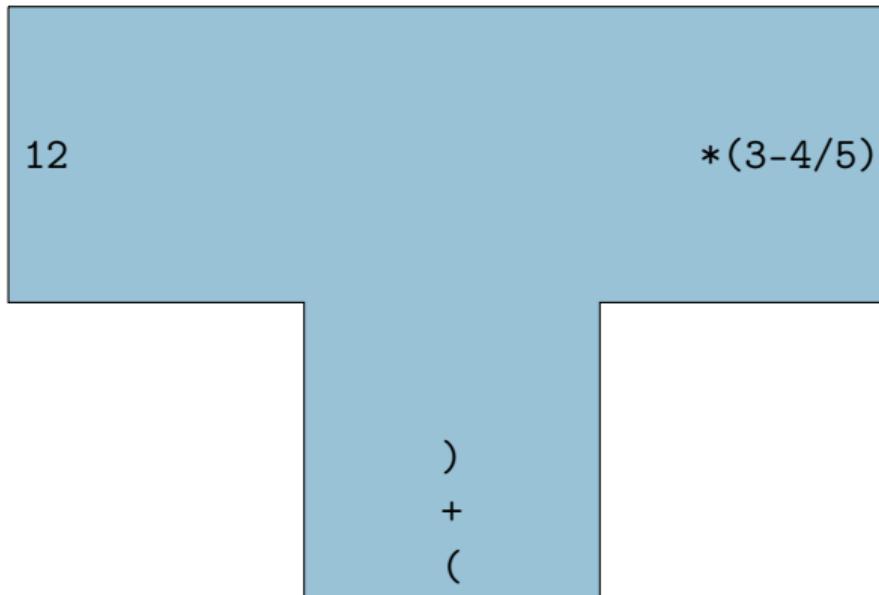
Пример: Преобразуване в обратен полски запис



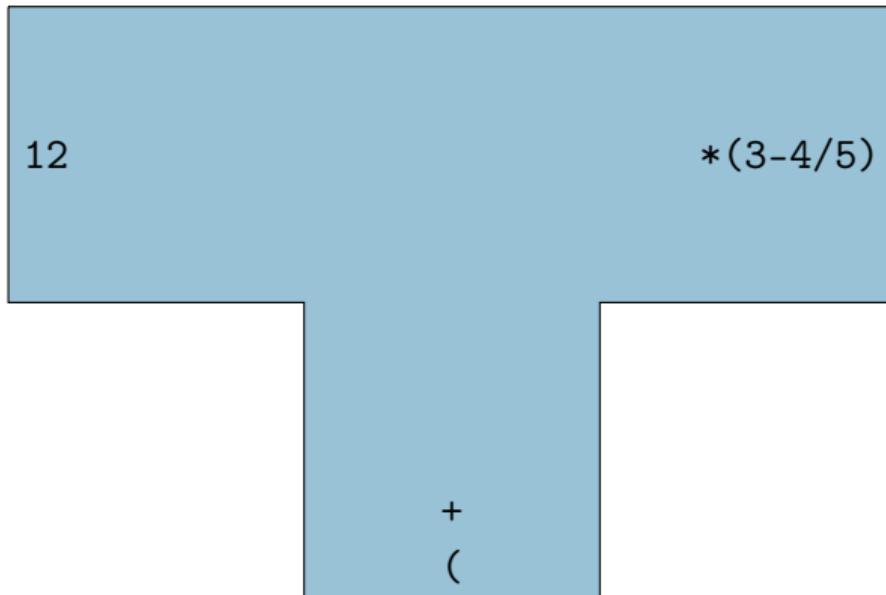
Пример: Преобразуване в обратен полски запис



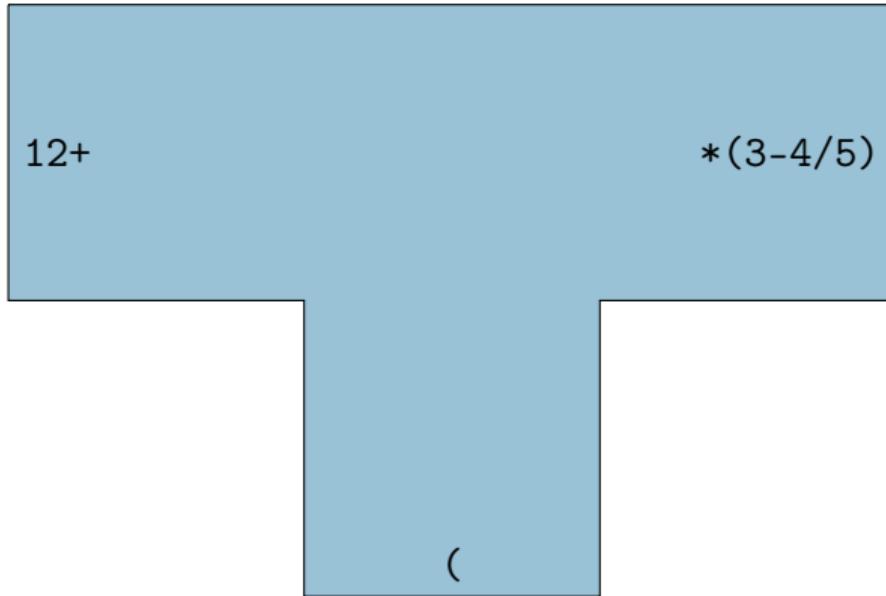
Пример: Преобразуване в обратен полски запис



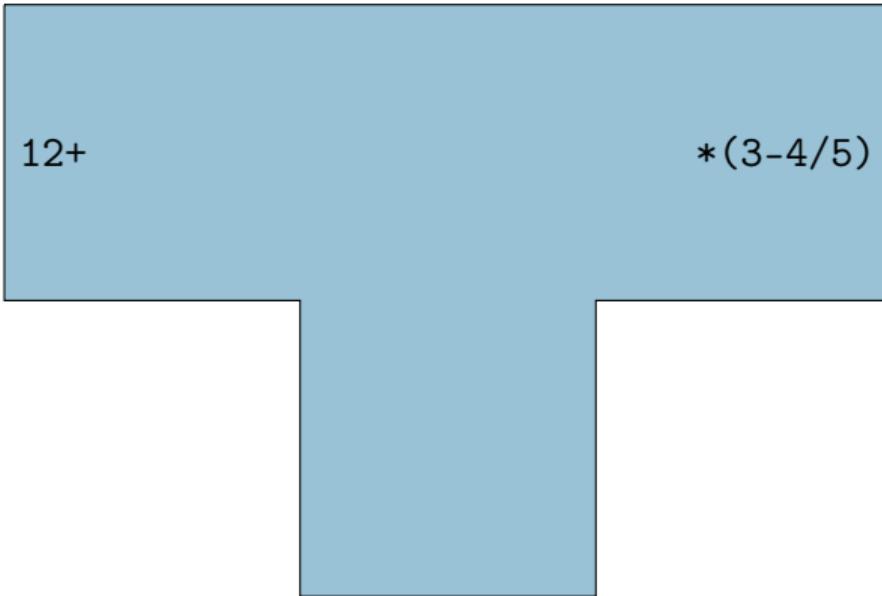
Пример: Преобразуване в обратен полски запис



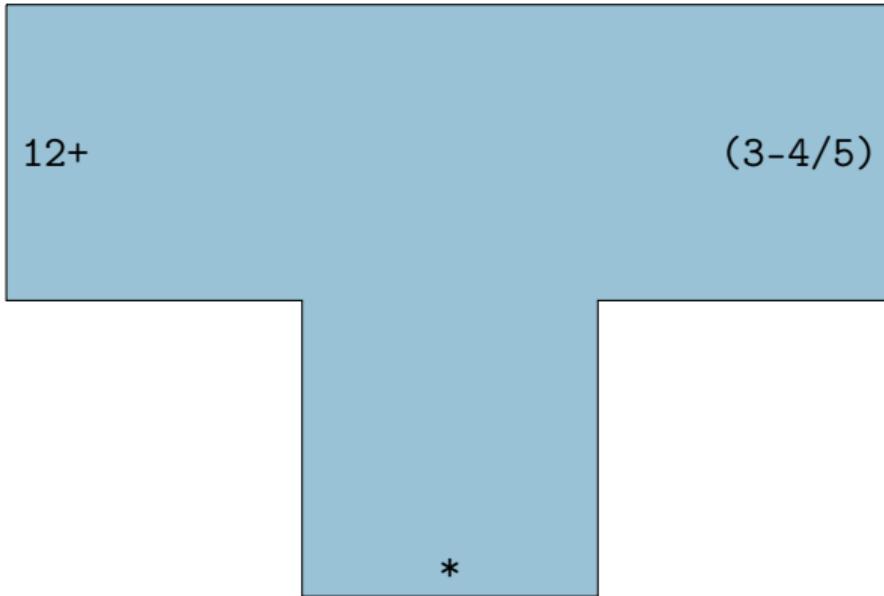
Пример: Преобразуване в обратен полски запис



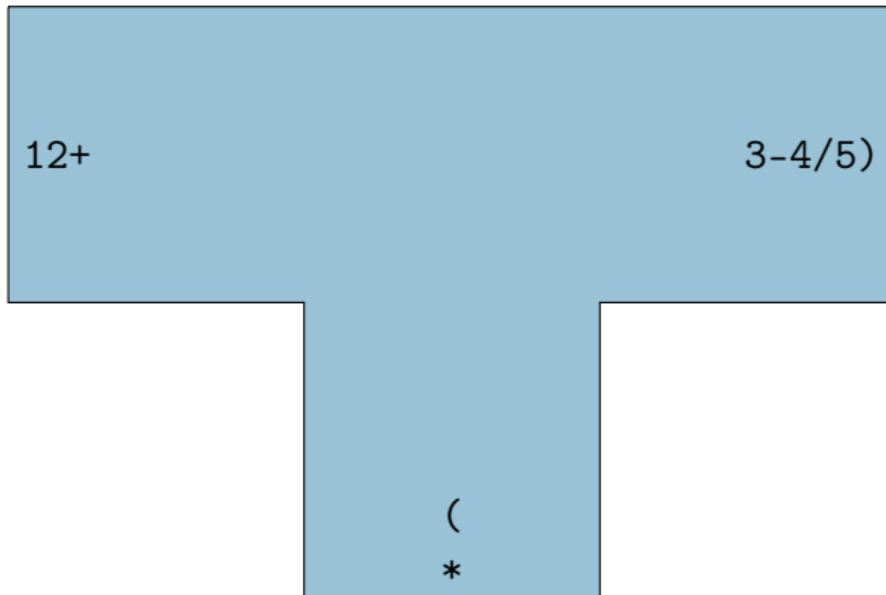
Пример: Преобразуване в обратен полски запис



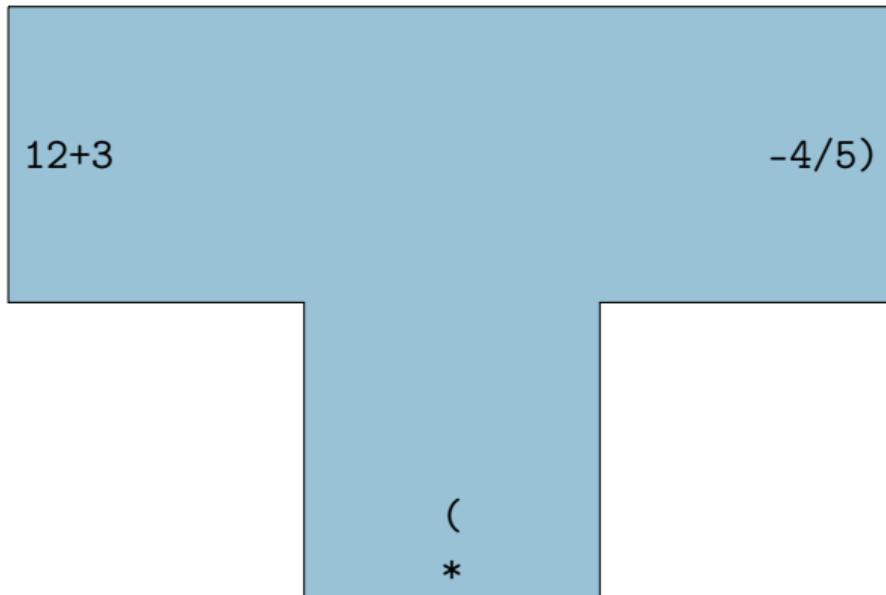
Пример: Преобразуване в обратен полски запис



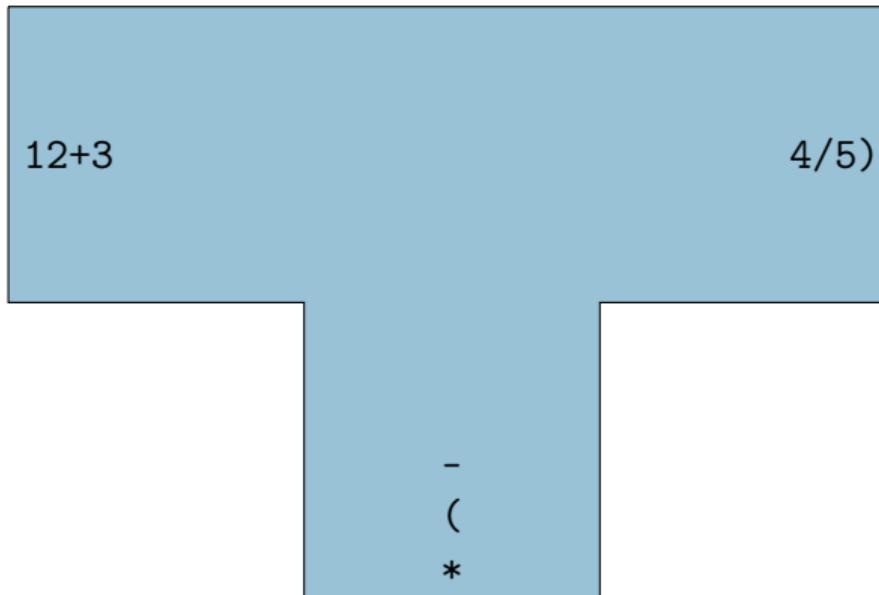
Пример: Преобразуване в обратен полски запис



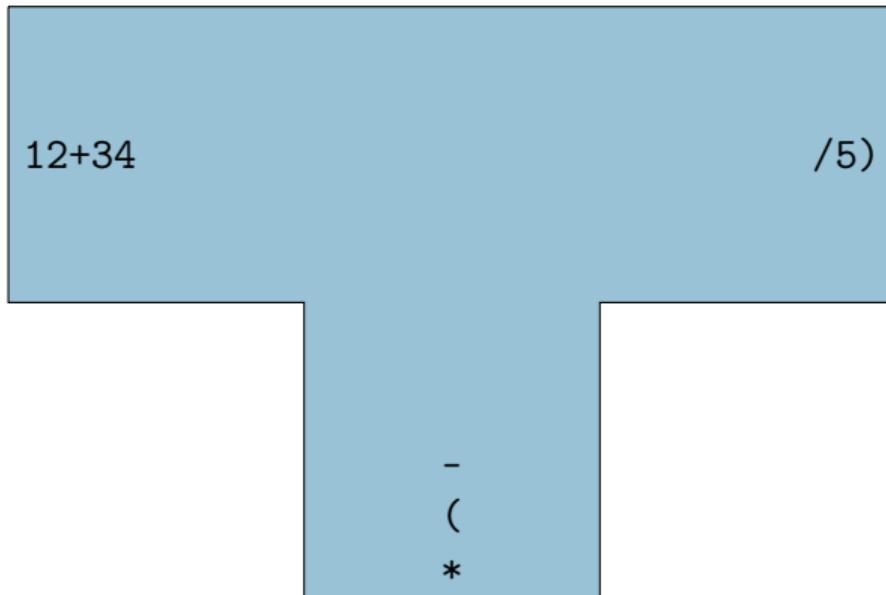
Пример: Преобразуване в обратен полски запис



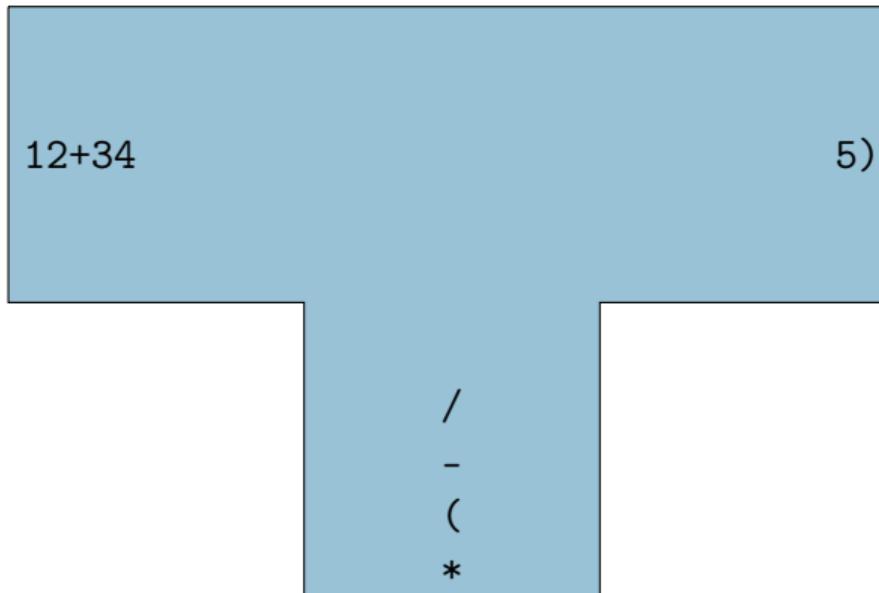
Пример: Преобразуване в обратен полски запис



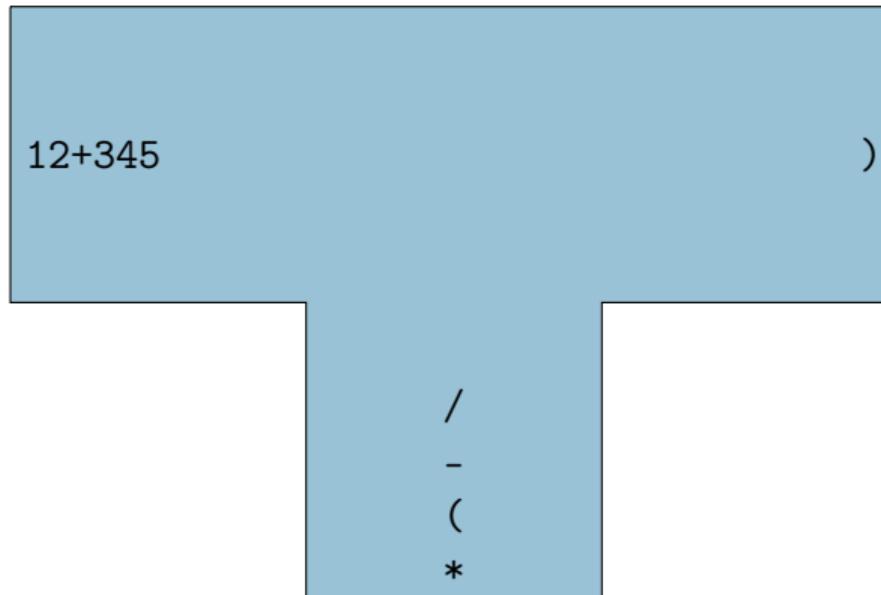
Пример: Преобразуване в обратен полски запис



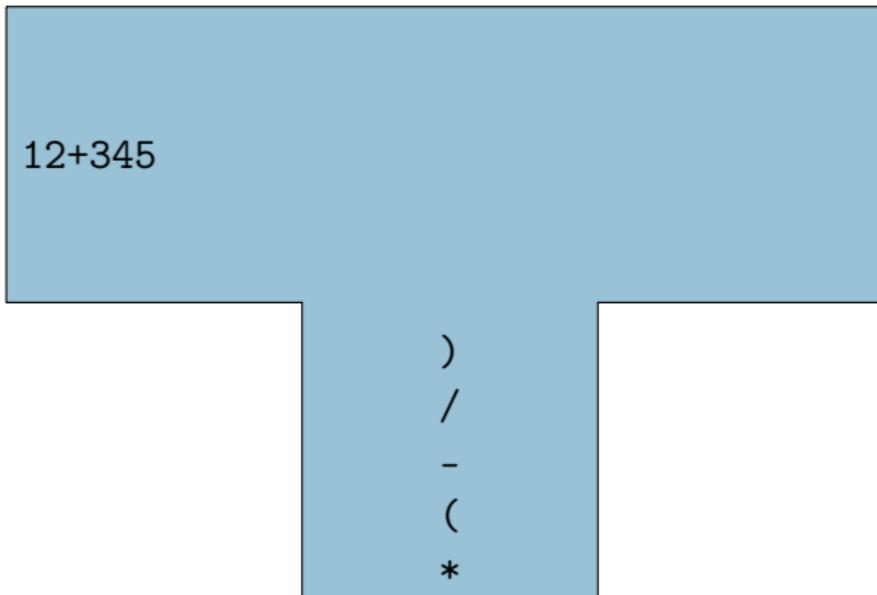
Пример: Преобразуване в обратен полски запис



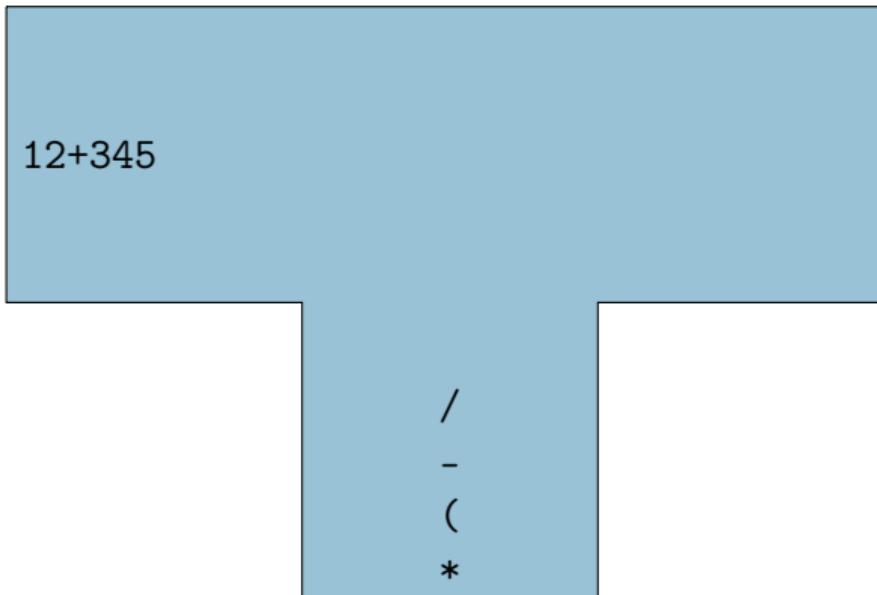
Пример: Преобразуване в обратен полски запис



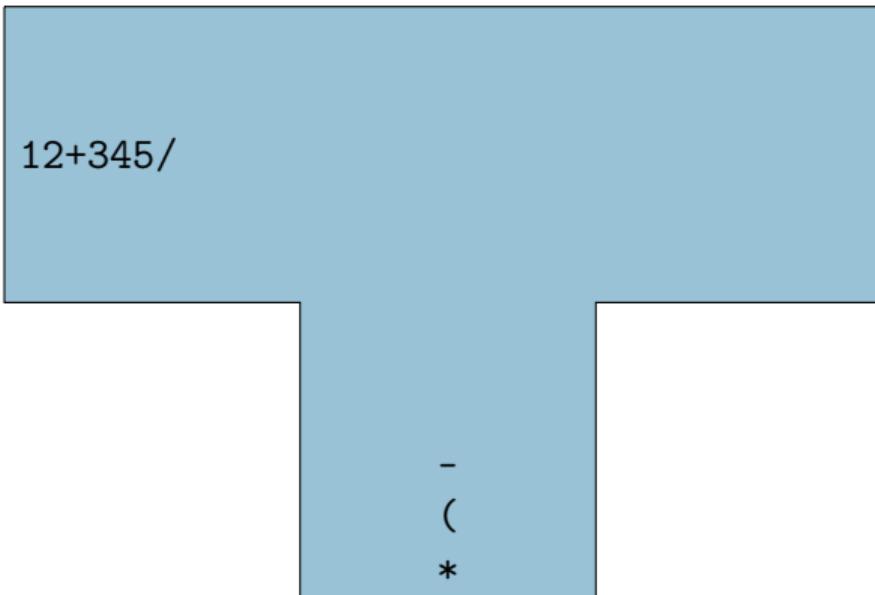
Пример: Преобразуване в обратен полски запис



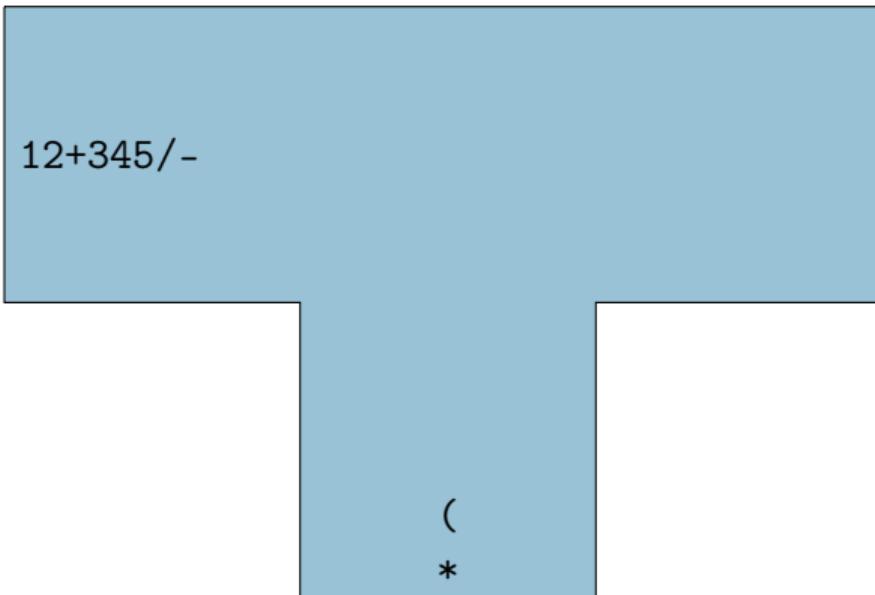
Пример: Преобразуване в обратен полски запис



Пример: Преобразуване в обратен полски запис



Пример: Преобразуване в обратен полски запис



Пример: Преобразуване в обратен полски запис

12+345/-

*

Пример: Преобразуване в обратен полски запис

12+345/-*

Приоритети на операциите

Какъв е обратния полски запис на израза $(1+2)*(3/4-5)$?

- Ако следваме алгоритъма, описан по-рано, резултатът би трявало да е $12+345-/*...$

Приоритети на операциите

Какъв е обратния полски запис на израза $(1+2)*(3/4-5)$?

- Ако следваме алгоритъма, описан по-рано, резултатът би трявало да е $12+345-/*\dots$
- ... но резултатът от този израз е **-9**, а не **-12.75!**

Приоритети на операциите

Какъв е обратния полски запис на израза $(1+2)*(3/4-5)$?

- Ако следваме алгоритъма, описан по-рано, резултатът би трявало да е $12+345-/*\dots$
- ... но резултатът от този израз е **-9**, а не **-12.75**!
- На обратния полски запис $12+345-/*$ съответства инфиксен запис $(1+2)*(3/(4-5))$!

Приоритети на операциите

Какъв е обратния полски запис на израза $(1+2)*(3/4-5)$?

- Ако следваме алгоритъма, описан по-рано, резултатът би трявало да е $12+345-/*\dots$
- ... но резултатът от този израз е **-9**, а не **-12.75**!
- На обратния полски запис $12+345-/*$ съответства инфиксен запис $(1+2)*(3/(4-5))!$
- Не сме взели предвид по-високия приоритет на операцията /

Приоритети на операциите

Какъв е обратния полски запис на израза $(1+2)*(3/4-5)$?

- Ако следваме алгоритъма, описан по-рано, резултатът би трявало да е $12+345-/*\dots$
- ... но резултатът от този израз е **-9**, а не **-12.75**!
- На обратния полски запис $12+345-/*$ съответства инфиксен запис $(1+2)*(3/(4-5))!$
- **Не сме взели предвид по-високия приоритет на операцията /**
- Правилният обратен полски запис на $(1+2)*(3/4-5)$ е $12+34/5-*.$

Приоритети на операциите

Какъв е обратния полски запис на израза $(1+2)*(3/4-5)$?

- Ако следваме алгоритъма, описан по-рано, резултатът би трявало да е $12+345-/*\dots$
- ... но резултатът от този израз е **-9**, а не **-12.75**!
- На обратния полски запис $12+345-/*$ съответства инфиксен запис $(1+2)*(3/(4-5))!$
- Не сме взели предвид по-високия приоритет на операцията /
- Правилният обратен полски запис на $(1+2)*(3/4-5)$ е $12+34/5-*.$
- Необходимо е да променим алгоритъма, така че да взема предвид приоритета на операциите!

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в крайния резултат
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция, **поставяме я в стека с операции**
- Ако с е затваряща скоба, последователно изваждаме от стека и записваме в резултата всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека и записваме в резултата всички останали операции.

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в крайния резултат
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция:
 - изваждаме от стека и записваме в резултата всички операции с приоритет по-висок или равен на с
 - поставяме с в стека
- Ако с е затваряща скоба, последователно изваждаме от стека и записваме в резултата всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека и запистваме в резултата всички останали операции.

Пример 2: Преобразуване в обратен полски запис

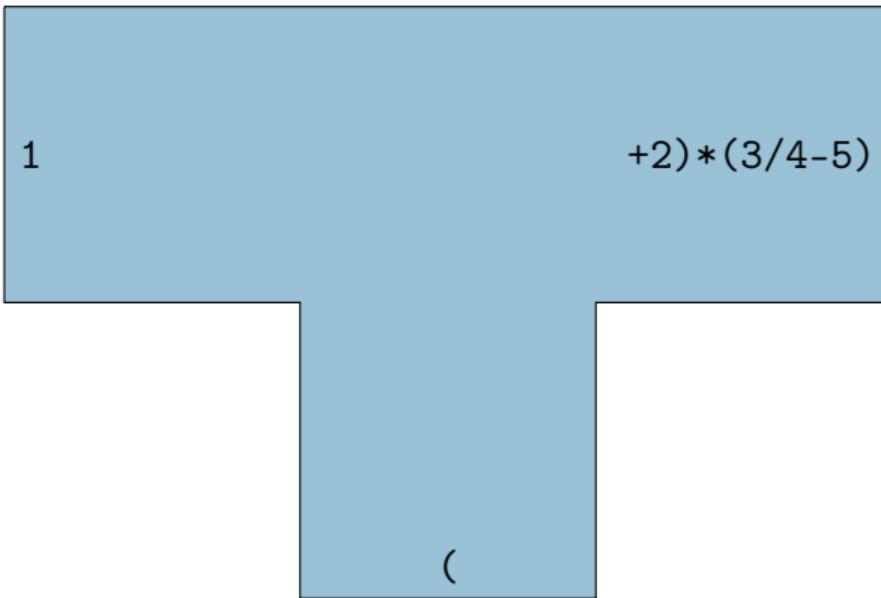
$$(1+2)*(3/4-5)$$

Пример 2: Преобразуване в обратен полски запис

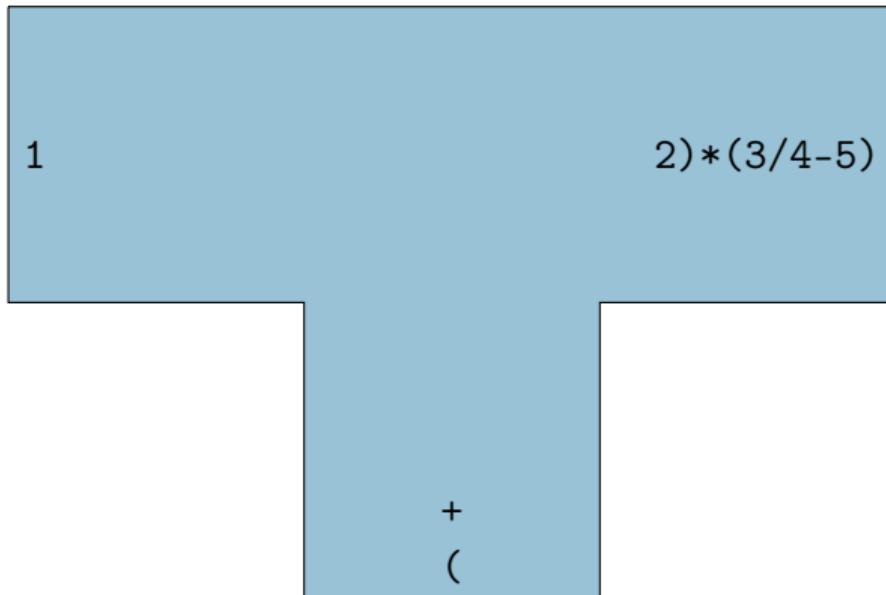
 $1+2)*(3/4-5)$

()

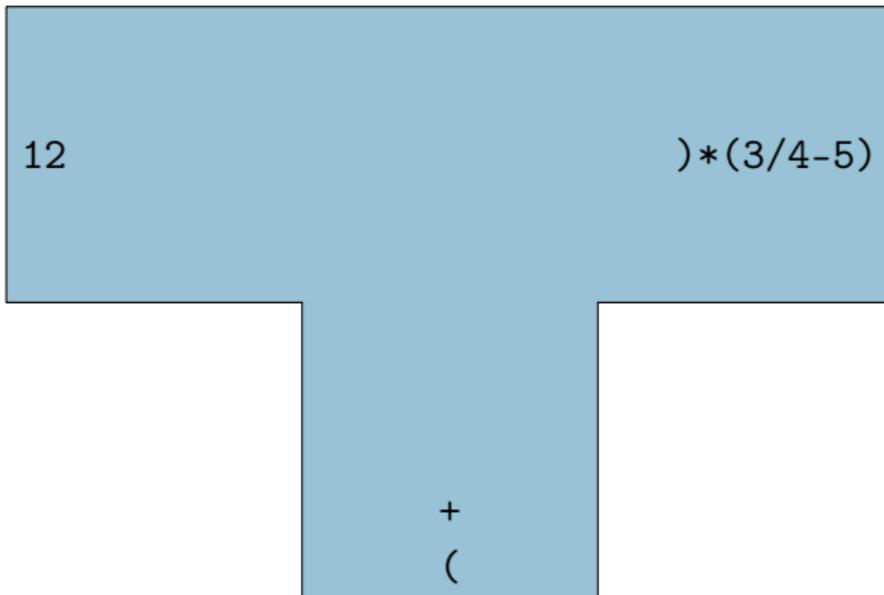
Пример 2: Преобразуване в обратен полски запис



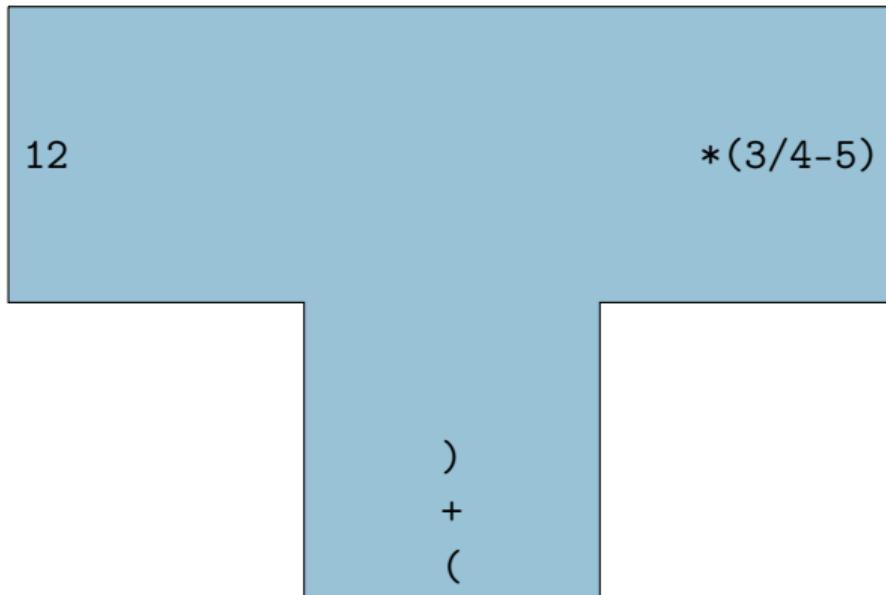
Пример 2: Преобразуване в обратен полски запис



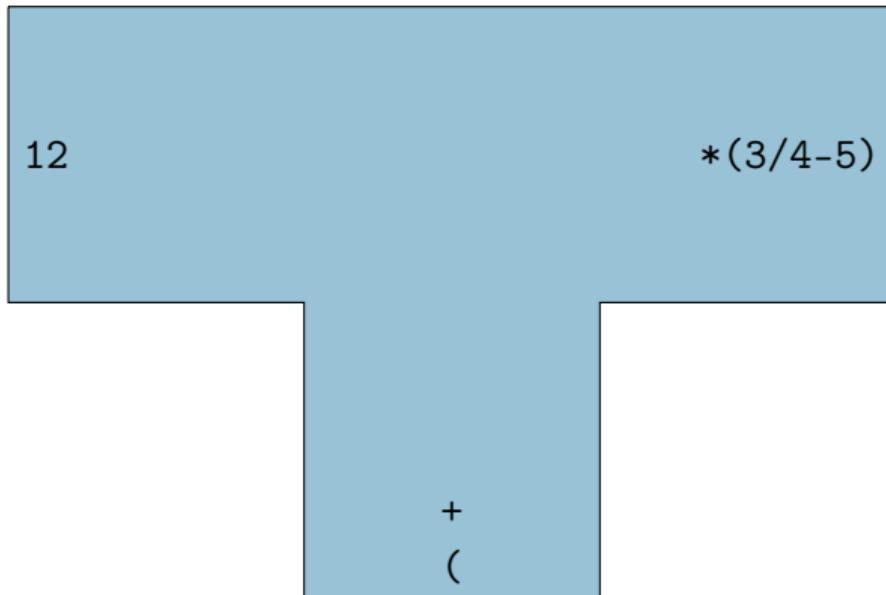
Пример 2: Преобразуване в обратен полски запис



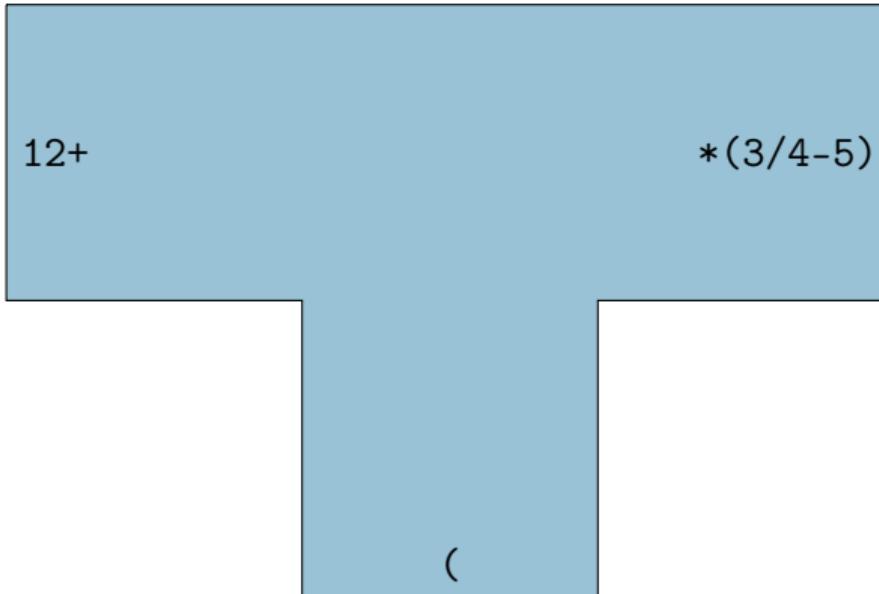
Пример 2: Преобразуване в обратен полски запис



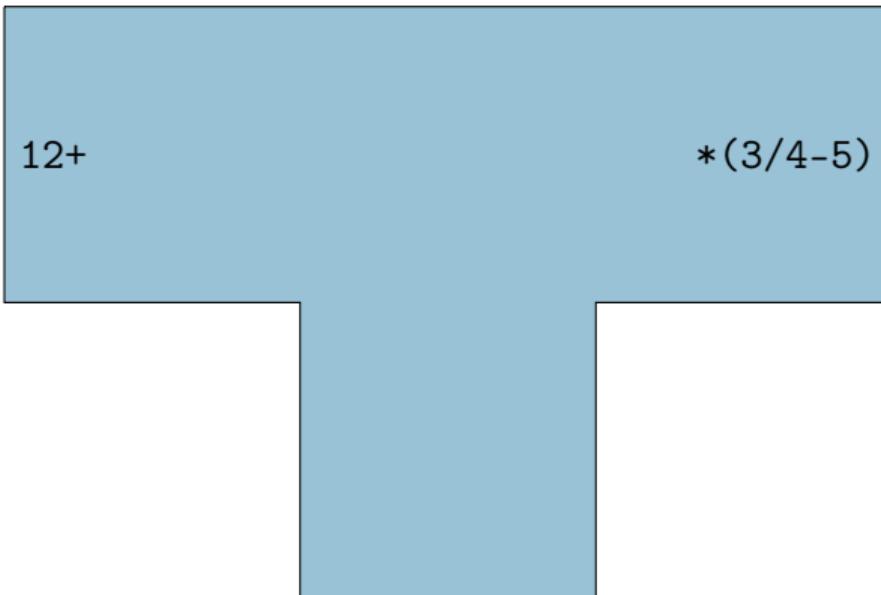
Пример 2: Преобразуване в обратен полски запис



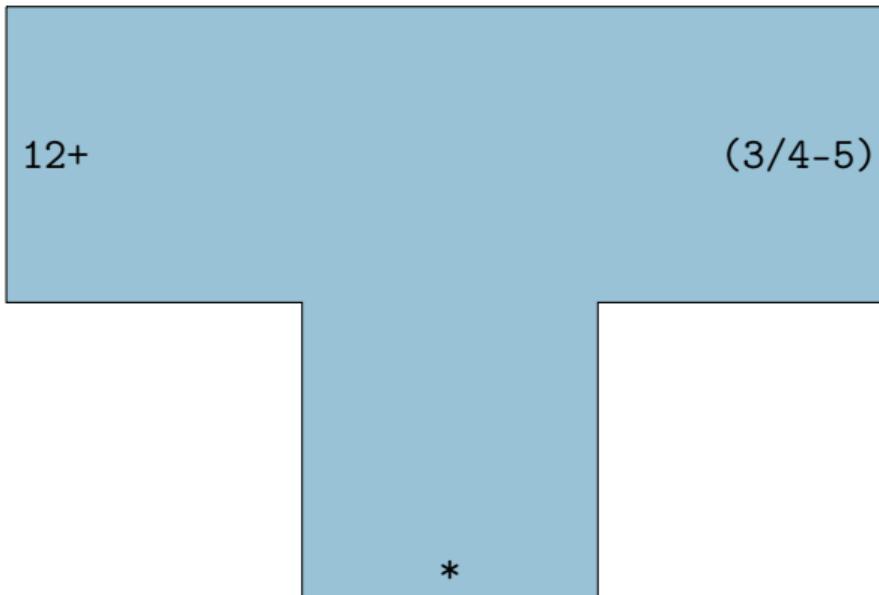
Пример 2: Преобразуване в обратен полски запис



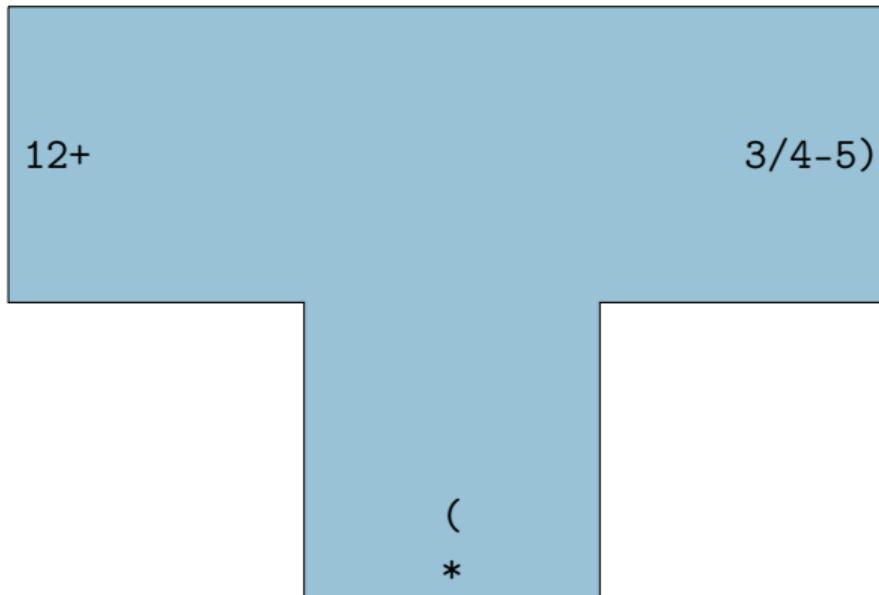
Пример 2: Преобразуване в обратен полски запис



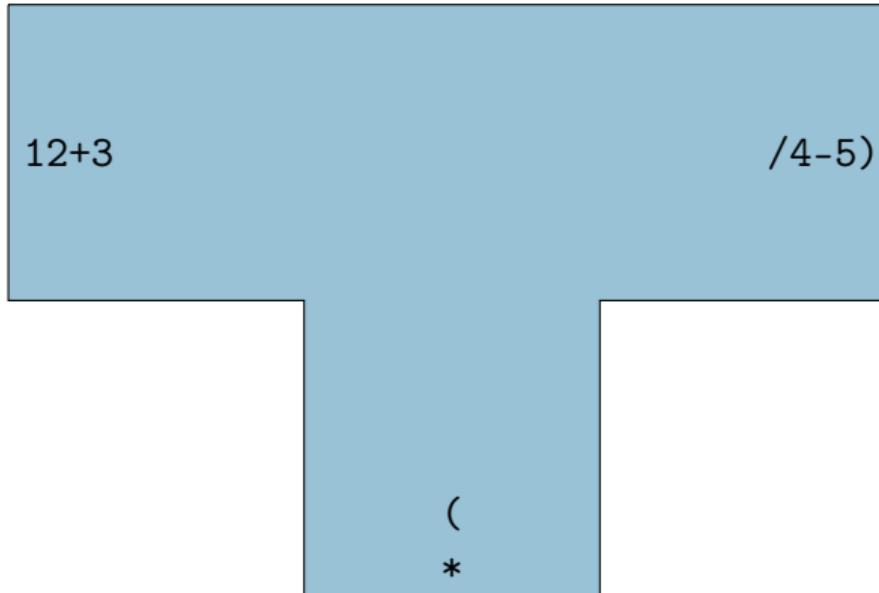
Пример 2: Преобразуване в обратен полски запис



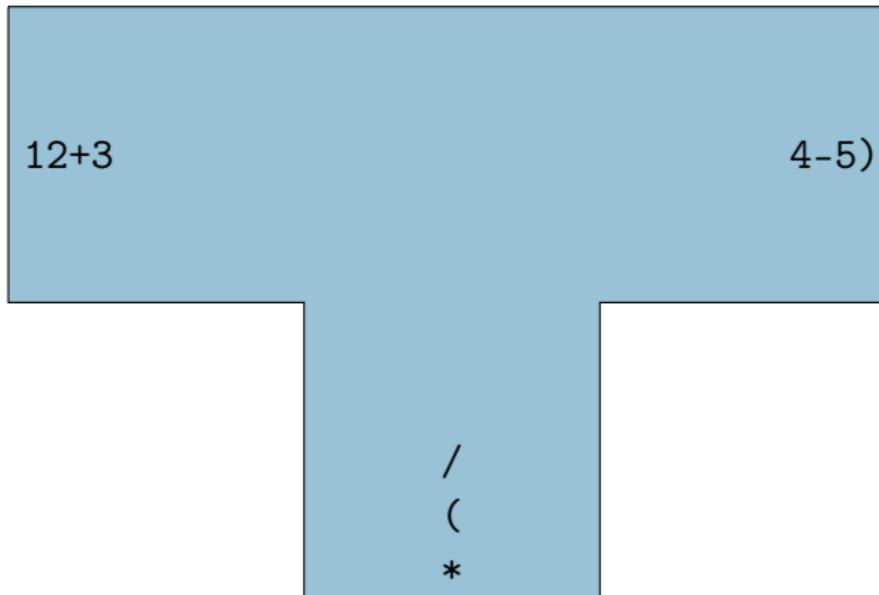
Пример 2: Преобразуване в обратен полски запис



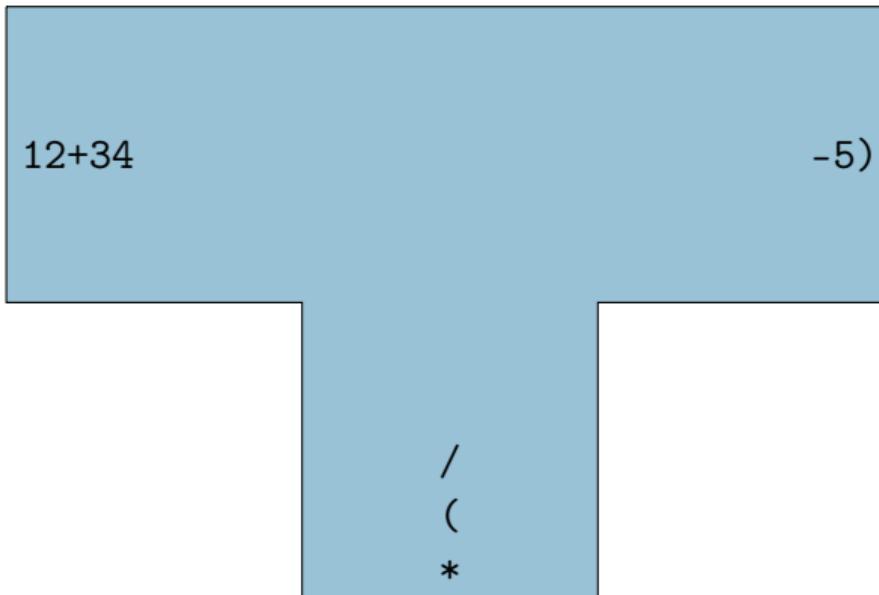
Пример 2: Преобразуване в обратен полски запис



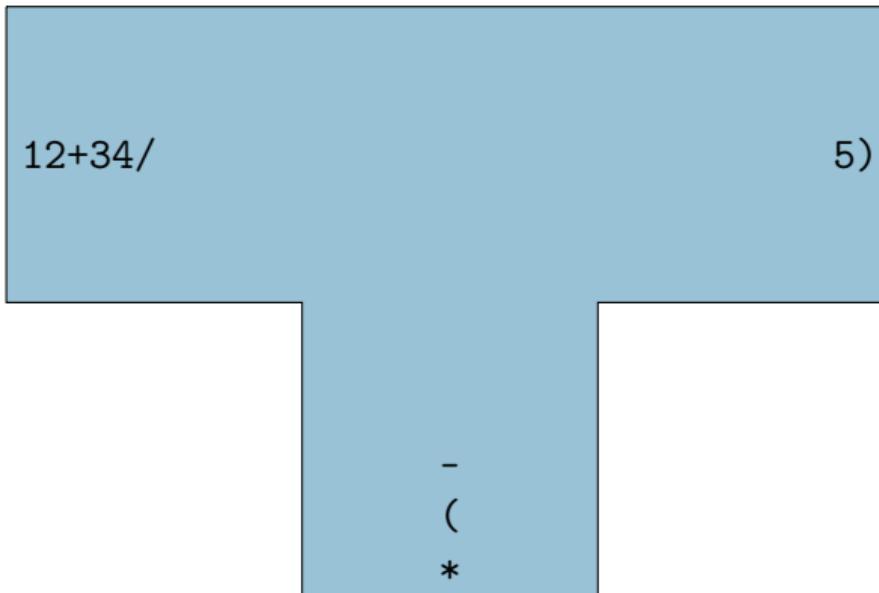
Пример 2: Преобразуване в обратен полски запис



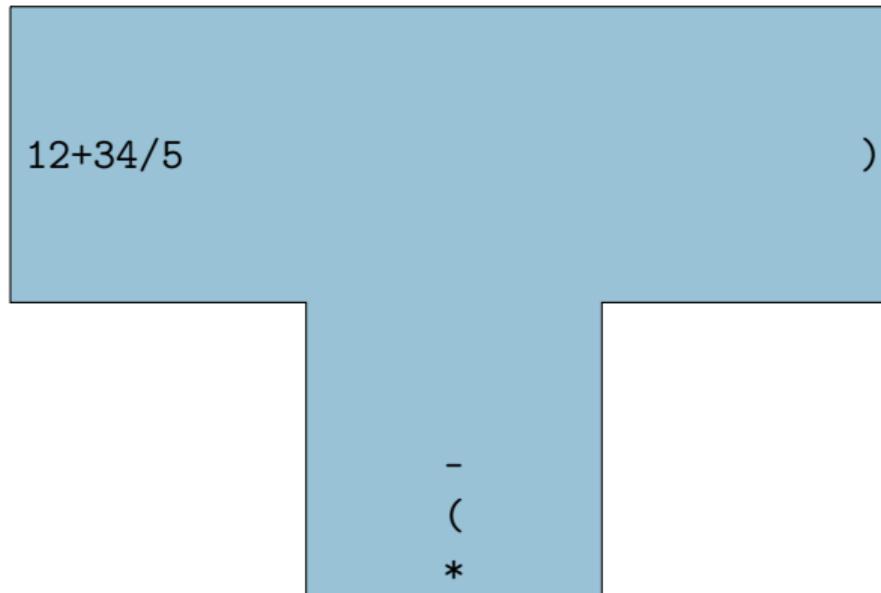
Пример 2: Преобразуване в обратен полски запис



Пример 2: Преобразуване в обратен полски запис



Пример 2: Преобразуване в обратен полски запис



Пример 2: Преобразуване в обратен полски запис

12+34/5

)

-

(

*

Пример 2: Преобразуване в обратен полски запис

12+34/5

-

(

*

Пример 2: Преобразуване в обратен полски запис

12+34/5-

(

*

Пример 2: Преобразуване в обратен полски запис

12+34/5-

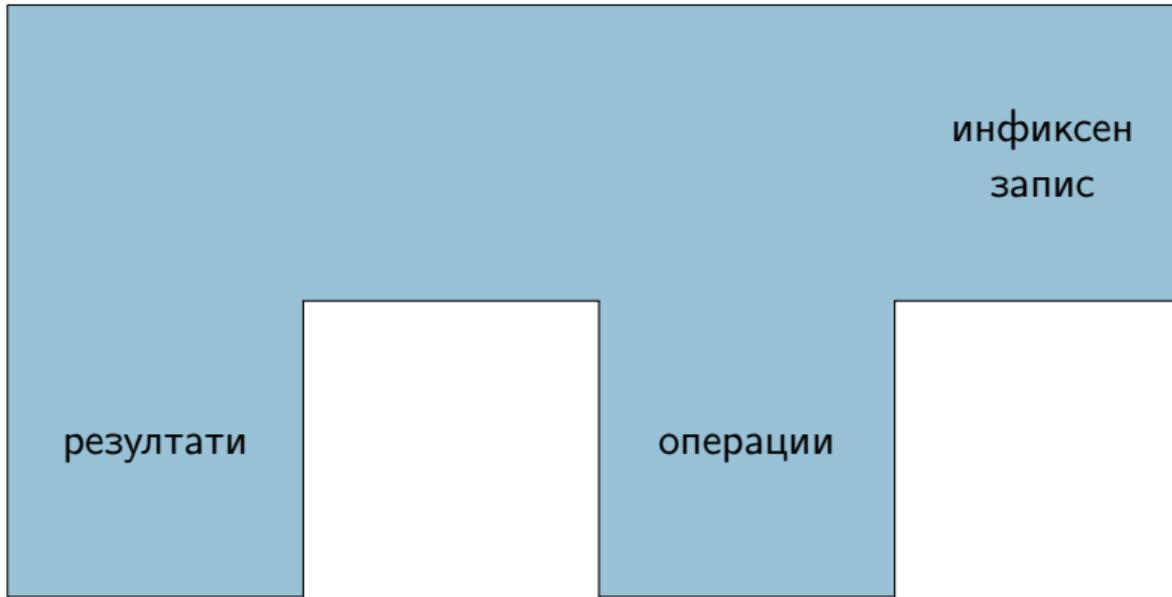
*

Пример 2: Преобразуване в обратен полски запис

12+34/5-*

Директно пресмятане на израз

Ако комбинираме двета алгоритъма, можем да пресмятаме инфиксен израз директно, без използването на обратен полски запис:



Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в **крайния резултат**
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция:
 - изваждаме от стека и записваме в резултата всички операции с приоритет по-висок или равен на с
 - поставяме с в стека
- Ако с е затваряща скоба, последователно изваждаме от стека и записваме в резултата всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека и запистваме в резултата всички останали операции.

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в стека с резултати
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция:
 - изваждаме от стека и записваме в резултата всички операции с приоритет по-висок или равен на с
 - поставяме с в стека
- Ако с е затваряща скоба, последователно изваждаме от стека и записваме в резултата всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека и запистваме в резултата всички останали операции.

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в стека с резултати
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция:
 - изваждаме от стека **и записваме в резултата** всички операции с приоритет по-висок или равен на с
 - поставяме с в стека
- Ако с е затваряща скоба, последователно изваждаме от стека **и записваме в резултата** всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека **и записваме в резултата** всички останали операции.

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

- Ако с е цифра, прехвърляме я в стека с резултати
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция:
 - изваждаме от стека и прилагаме всички операции с приоритет по-висок или равен на с
 - поставяме с в стека
- Ако с е затваряща скоба, последователно изваждаме от стека и прилагаме всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека и прилагаме всички останали операции.

Алгоритъм на разпределителната гара (shunting yard)

За преобразуване от инфиксен в обратен полски запис, за всеки пореден символ с:

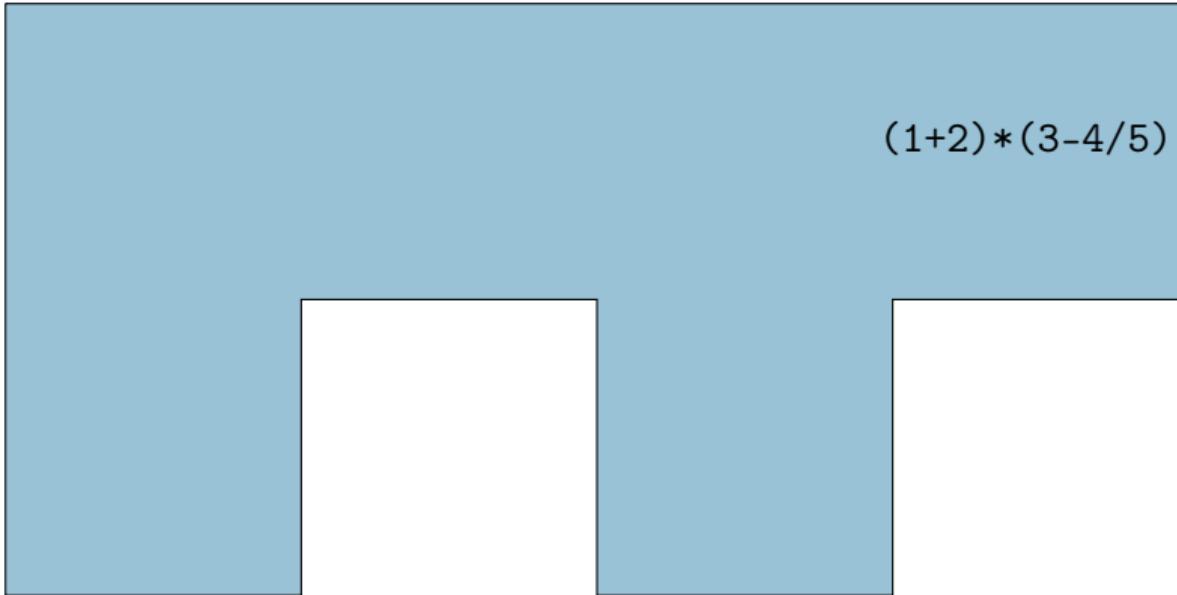
- Ако с е цифра, прехвърляме я в стека с резултати
- Ако с е отваряща скоба, поставяме я в стека с операции
- Ако с е операция:
 - изваждаме от стека и прилагаме всички операции с приоритет по-висок или равен на с
 - поставяме с в стека
- Ако с е затваряща скоба, последователно изваждаме от стека и прилагаме всички операции до достигане на отваряща скоба

След приключване на входния низ, последователно изваждаме от стека и прилагаме всички останали операции.

Крайният резултат е единственият елемент в стека с резултати.

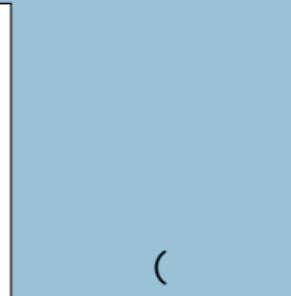
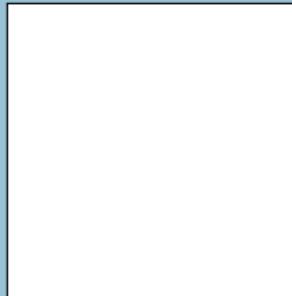
Пример: Директно пресмятане на израз

$$(1+2)*(3-4/5)$$

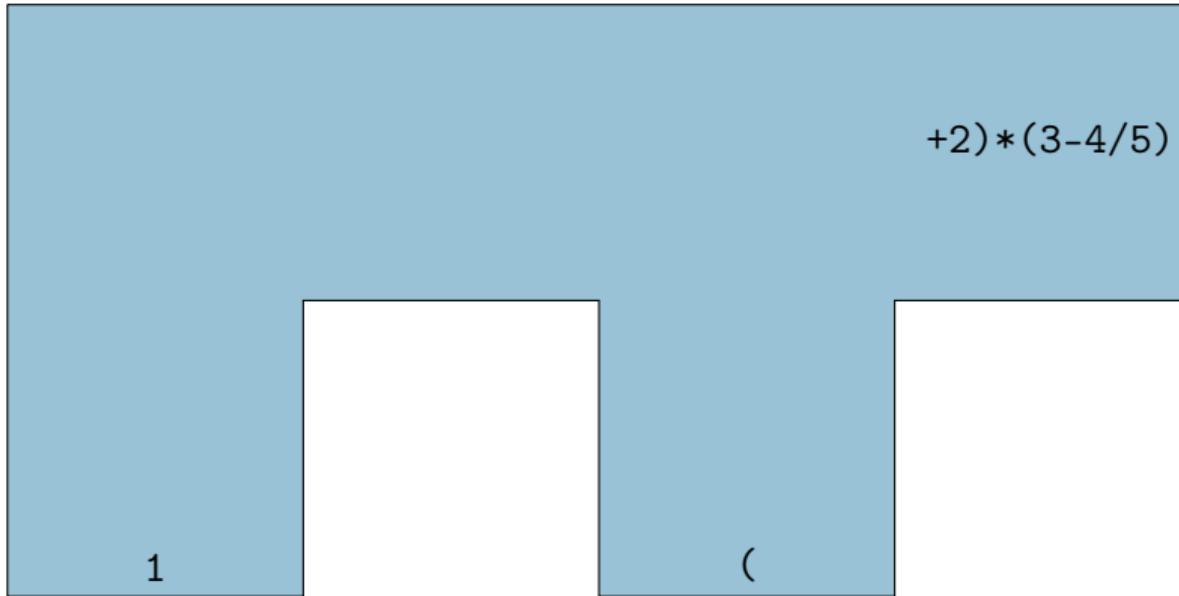


Пример: Директно пресмятане на израз

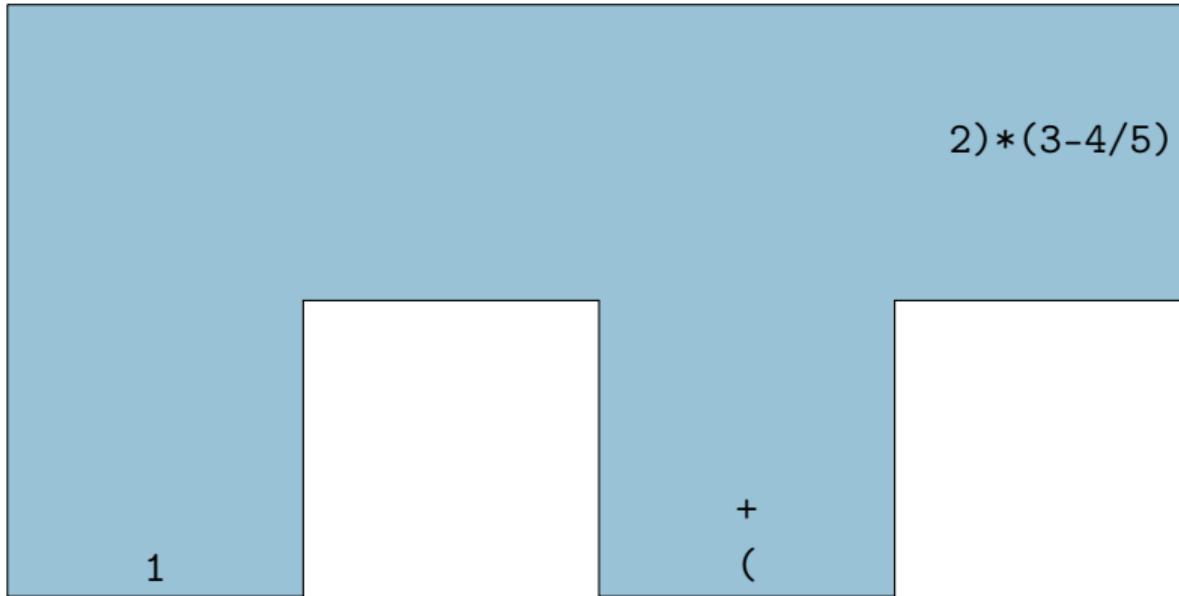
$$1+2)*(3-4/5)$$



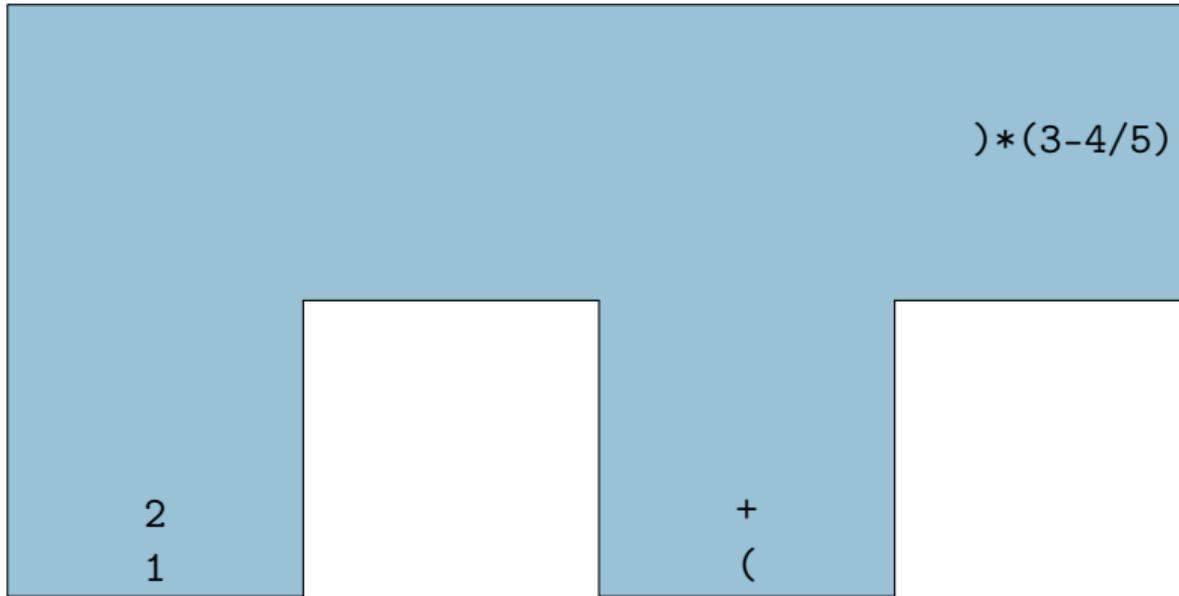
Пример: Директно пресмятане на израз



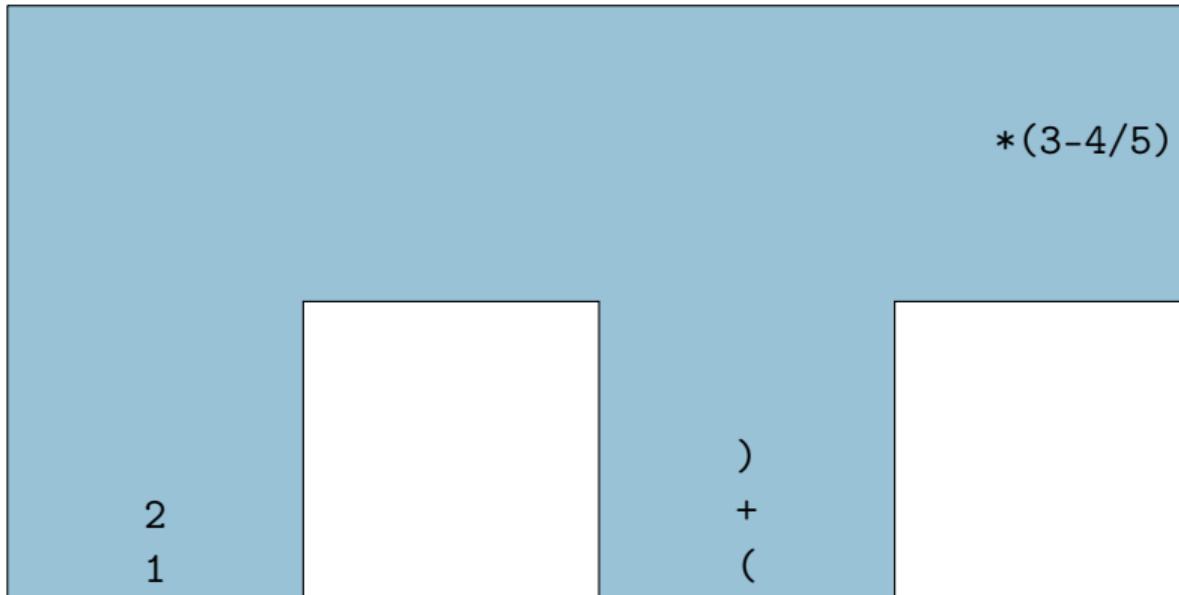
Пример: Директно пресмятане на израз



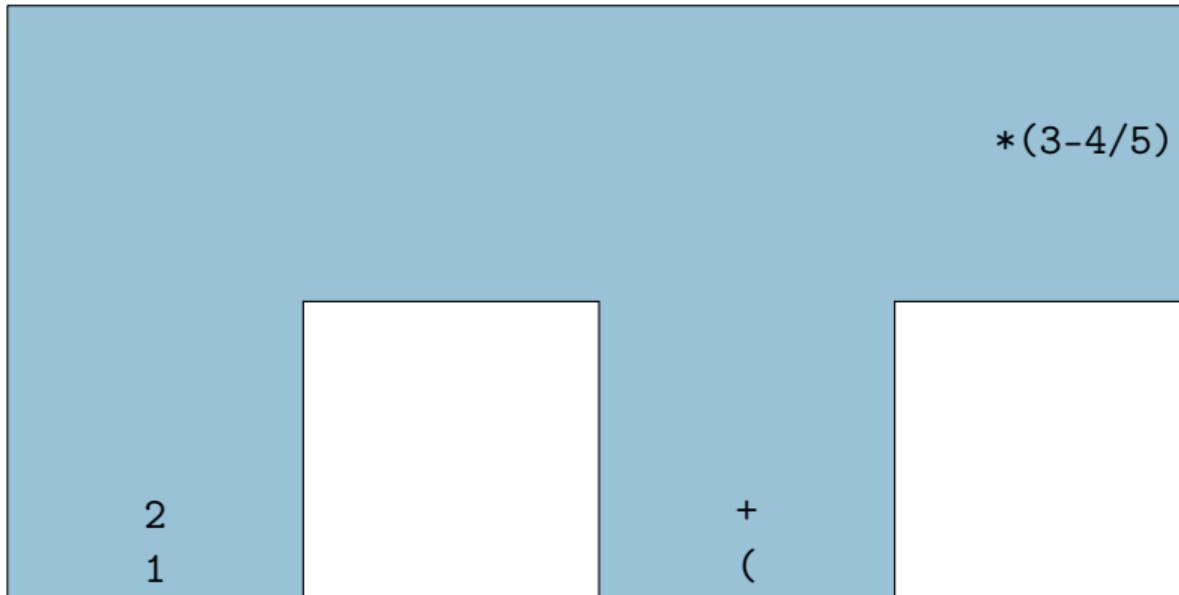
Пример: Директно пресмятане на израз



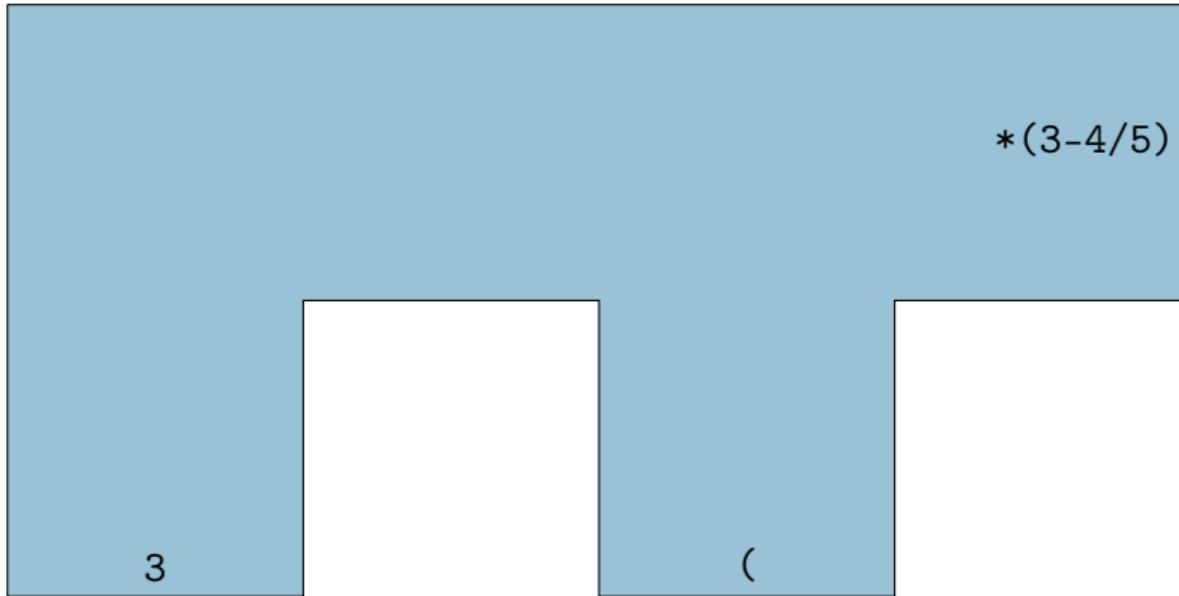
Пример: Директно пресмятане на израз



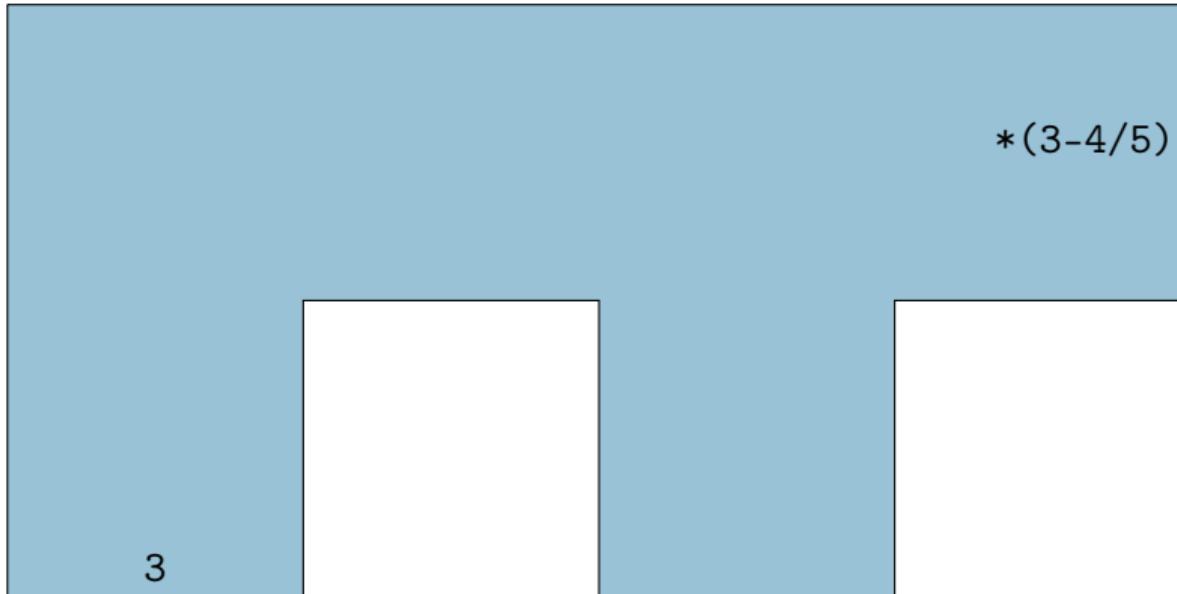
Пример: Директно пресмятане на израз



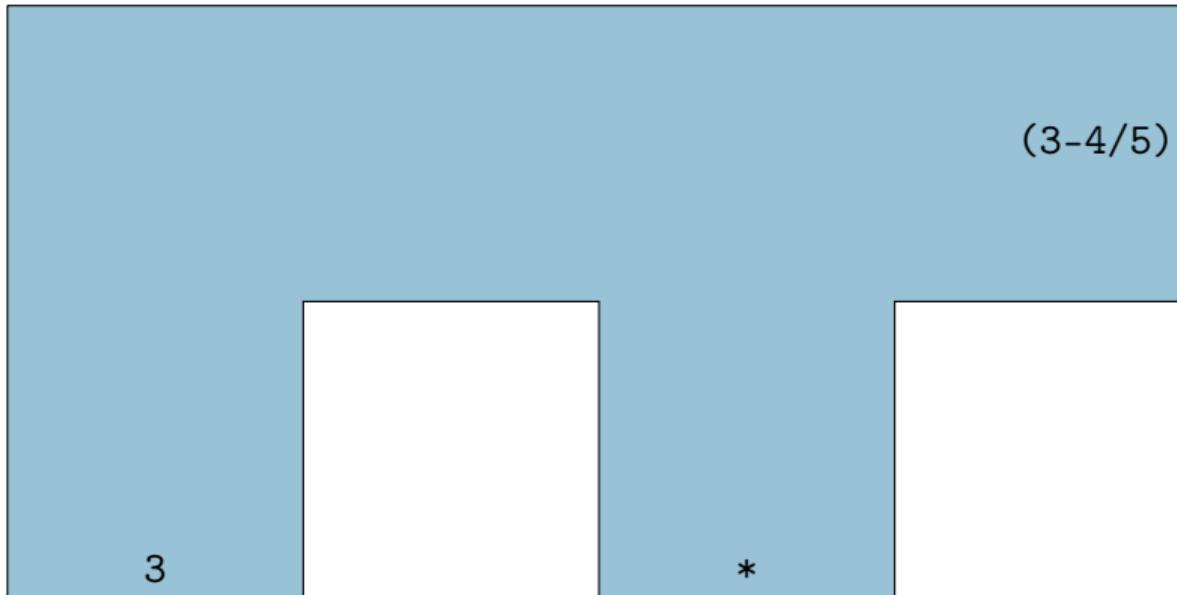
Пример: Директно пресмятане на израз



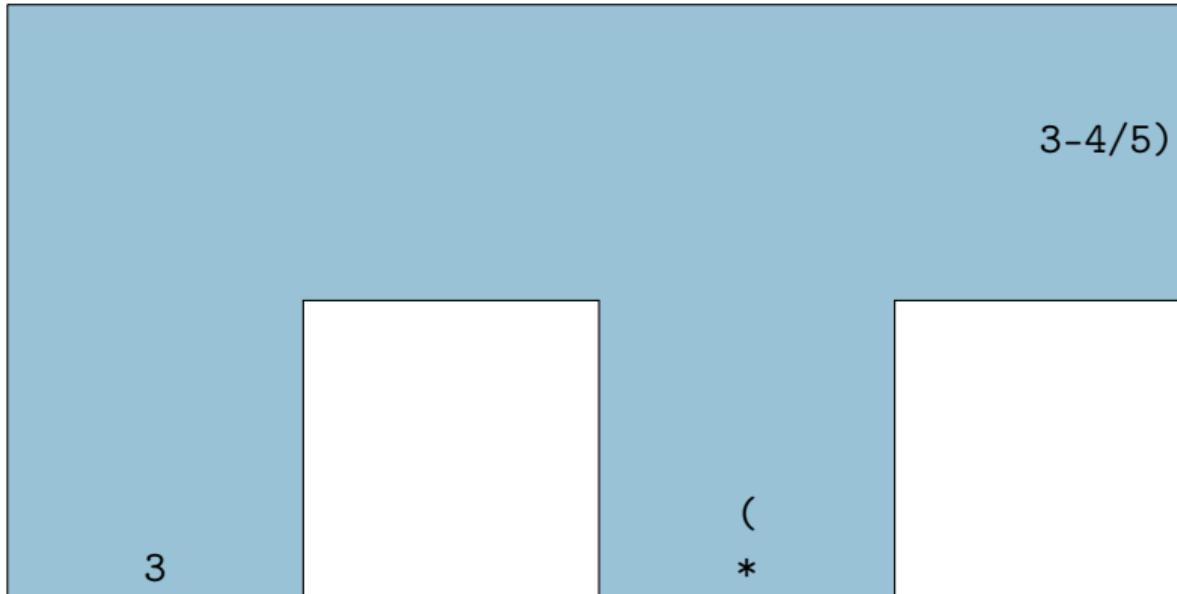
Пример: Директно пресмятане на израз



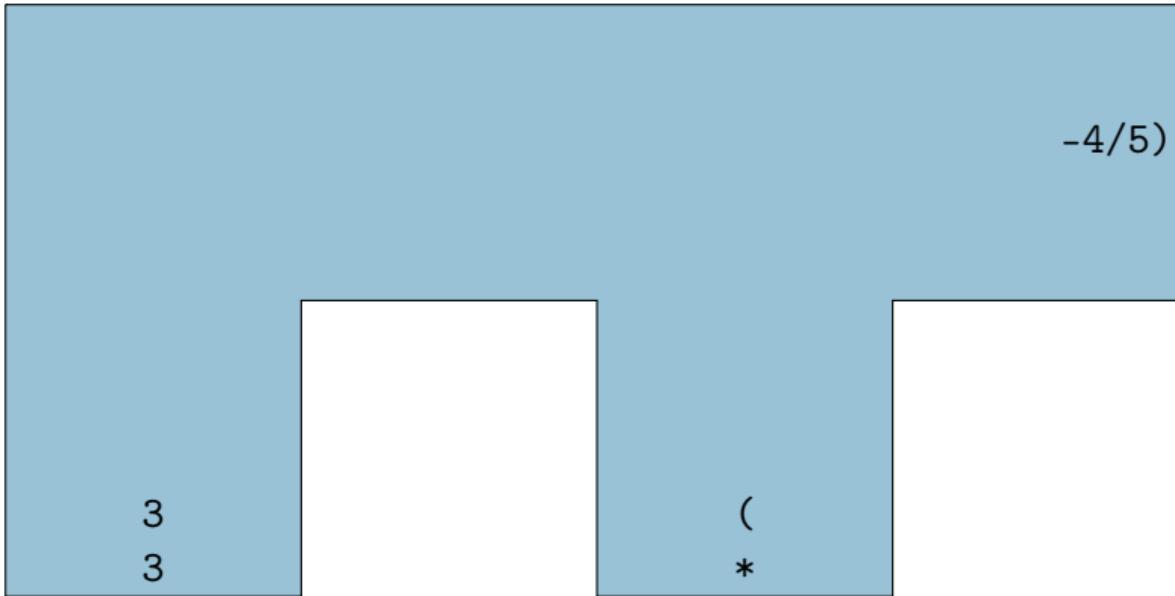
Пример: Директно пресмятане на израз



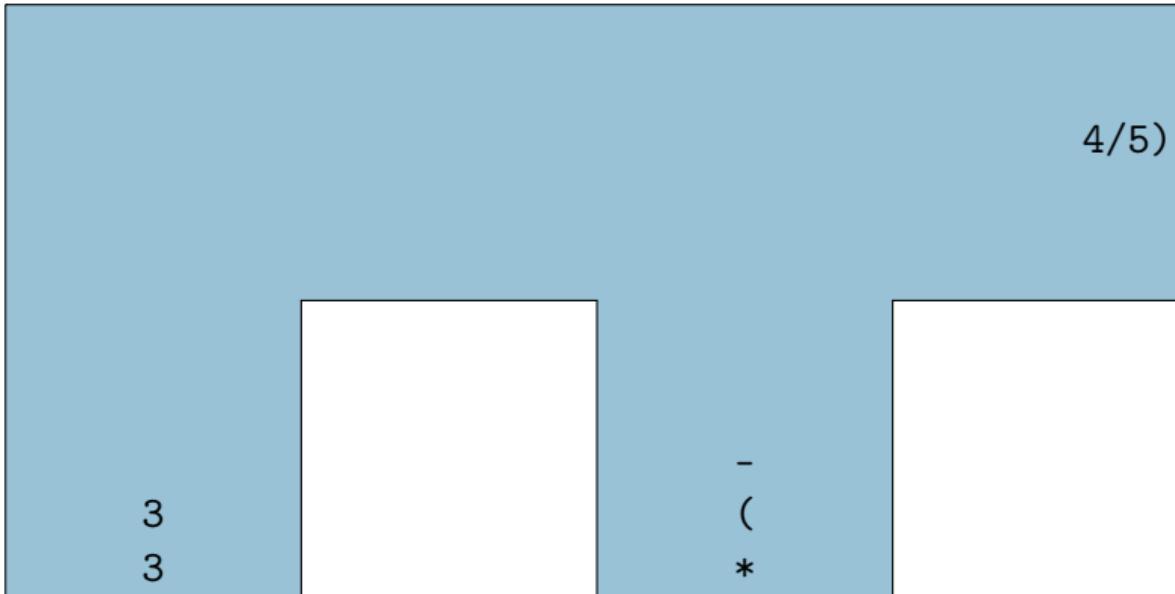
Пример: Директно пресмятане на израз



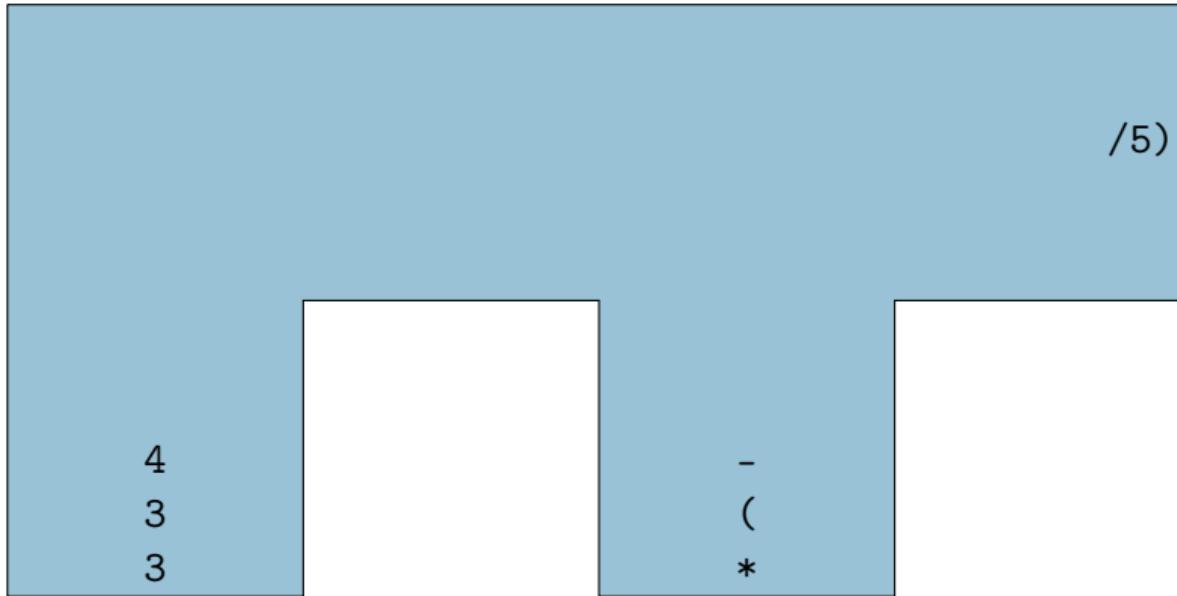
Пример: Директно пресмятане на израз



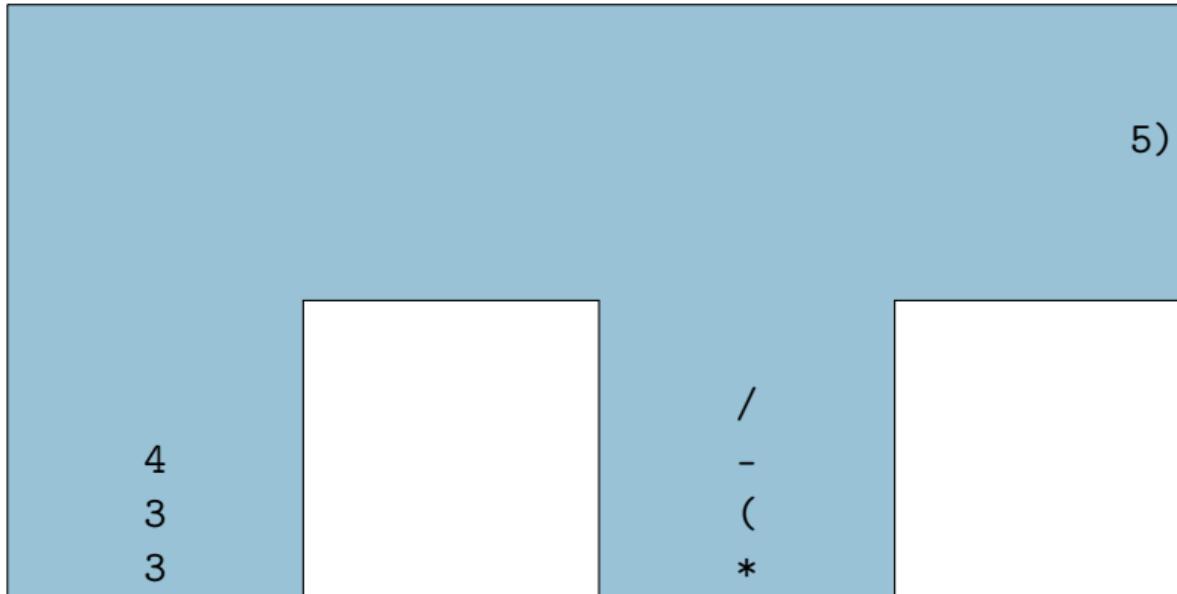
Пример: Директно пресмятане на израз



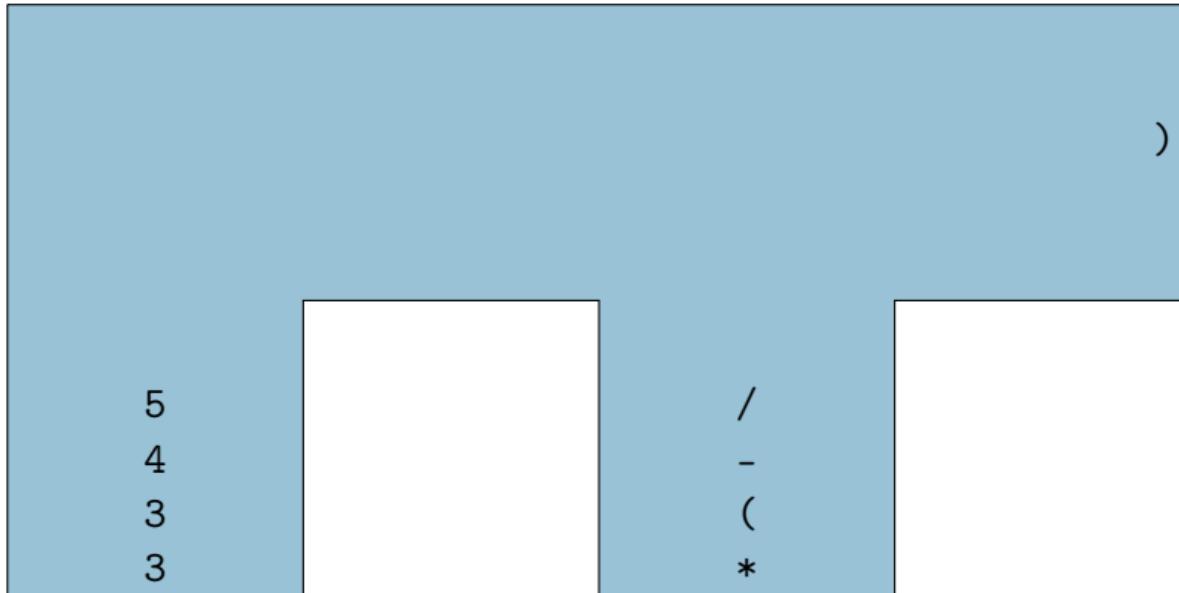
Пример: Директно пресмятане на израз



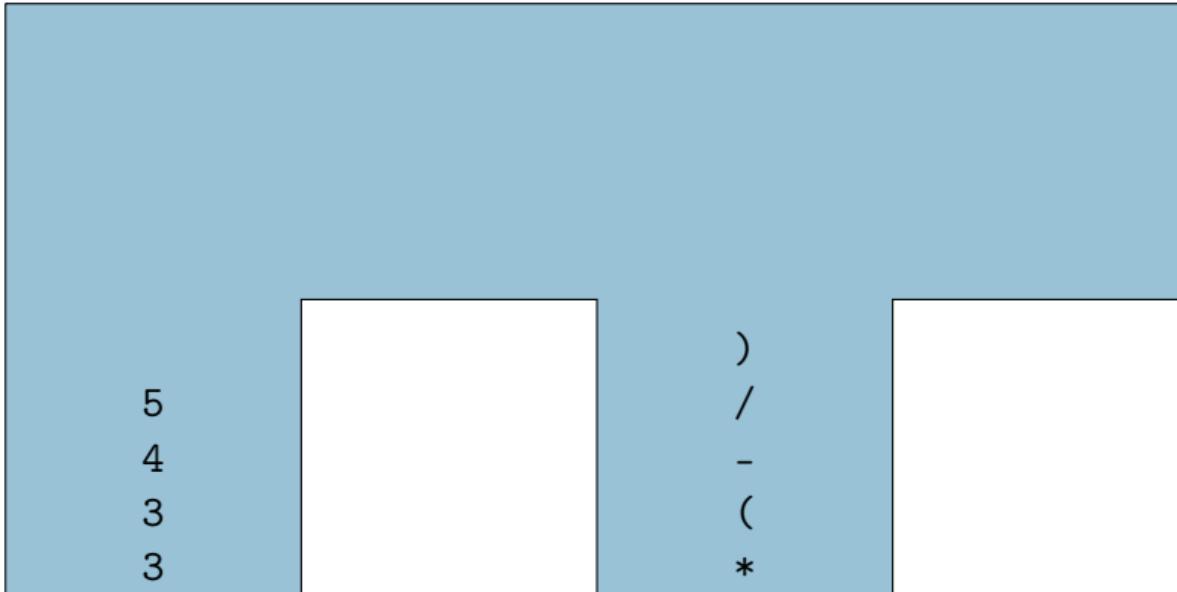
Пример: Директно пресмятане на израз



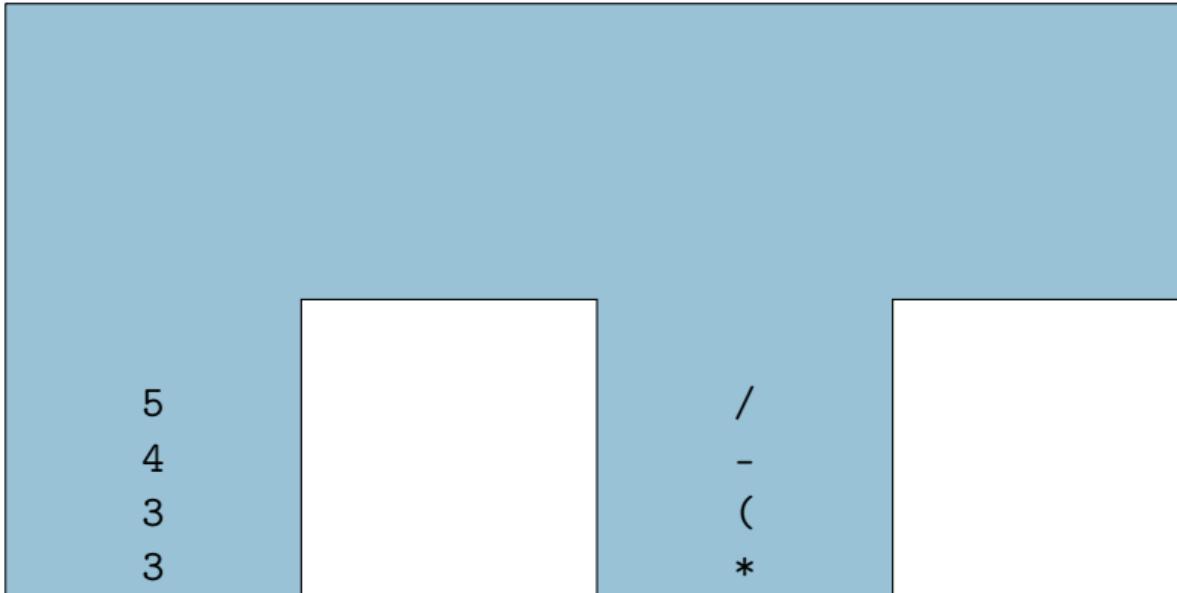
Пример: Директно пресмятане на израз



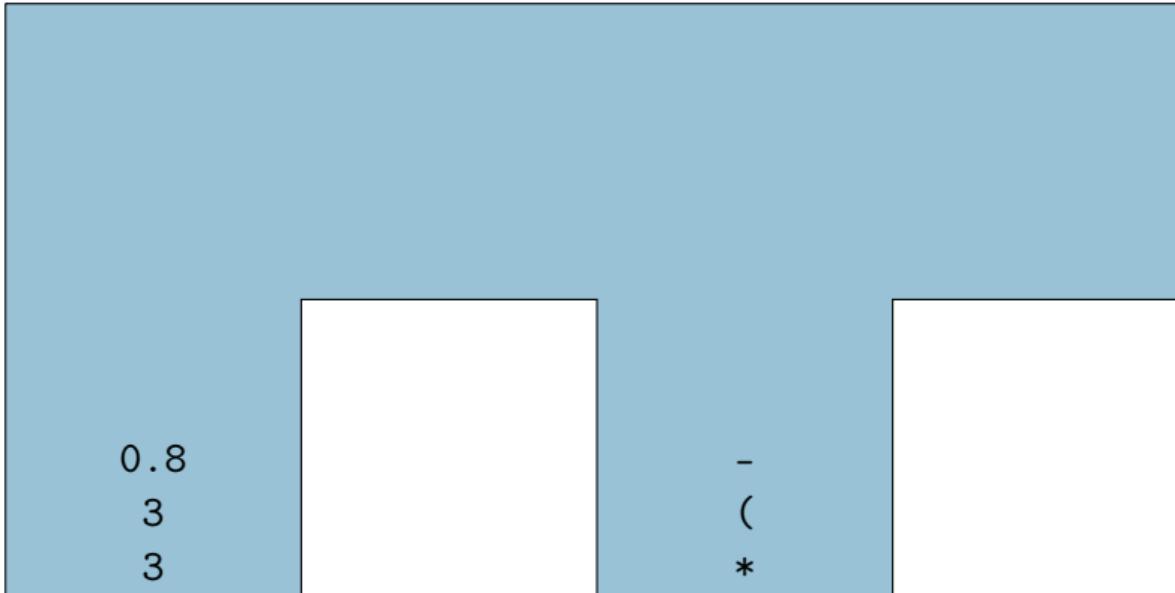
Пример: Директно пресмятане на израз



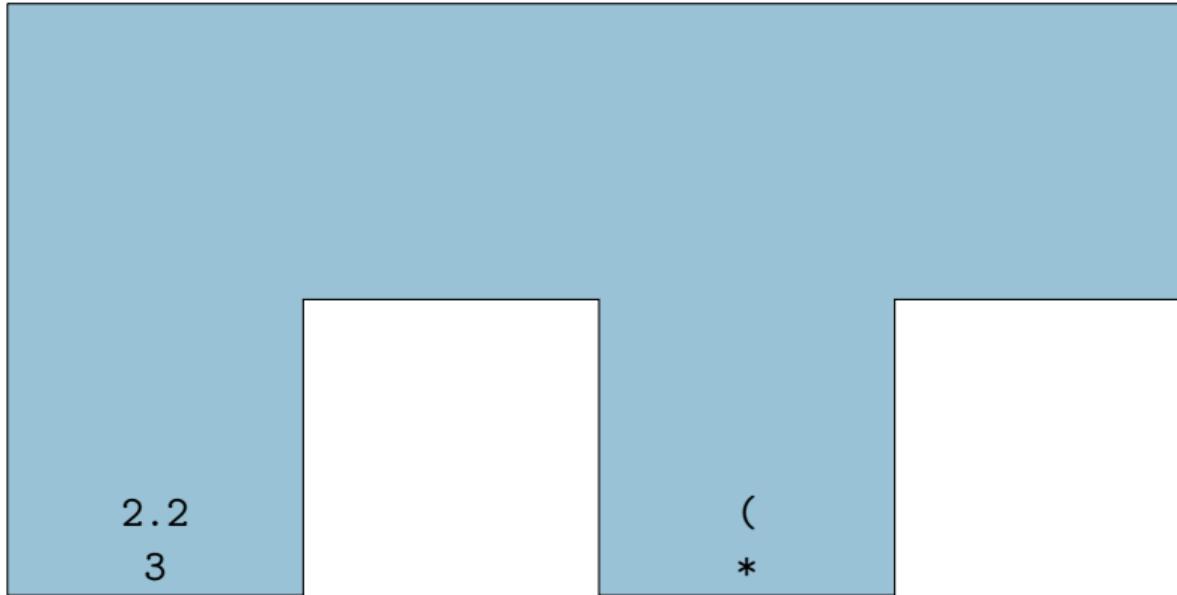
Пример: Директно пресмятане на израз



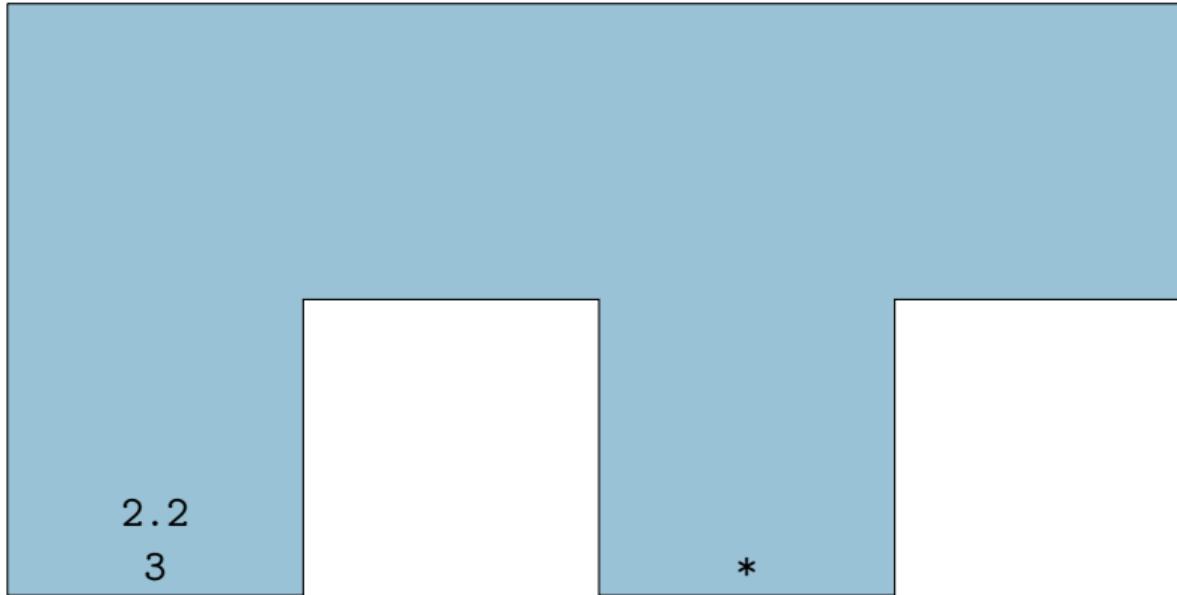
Пример: Директно пресмятане на израз



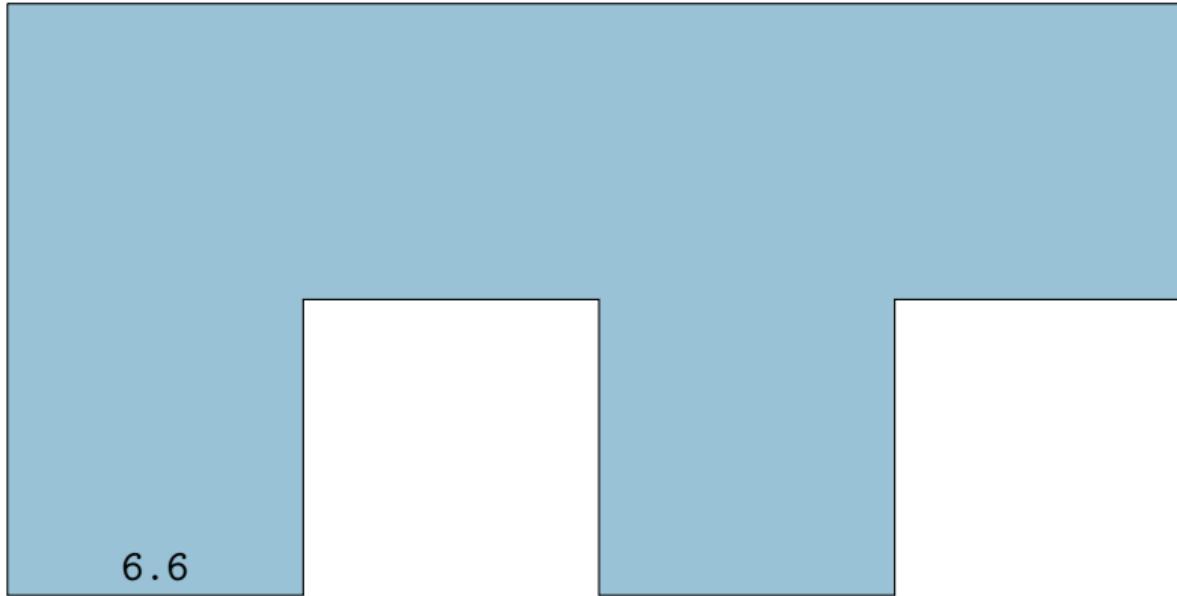
Пример: Директно пресмятане на израз



Пример: Директно пресмятане на израз

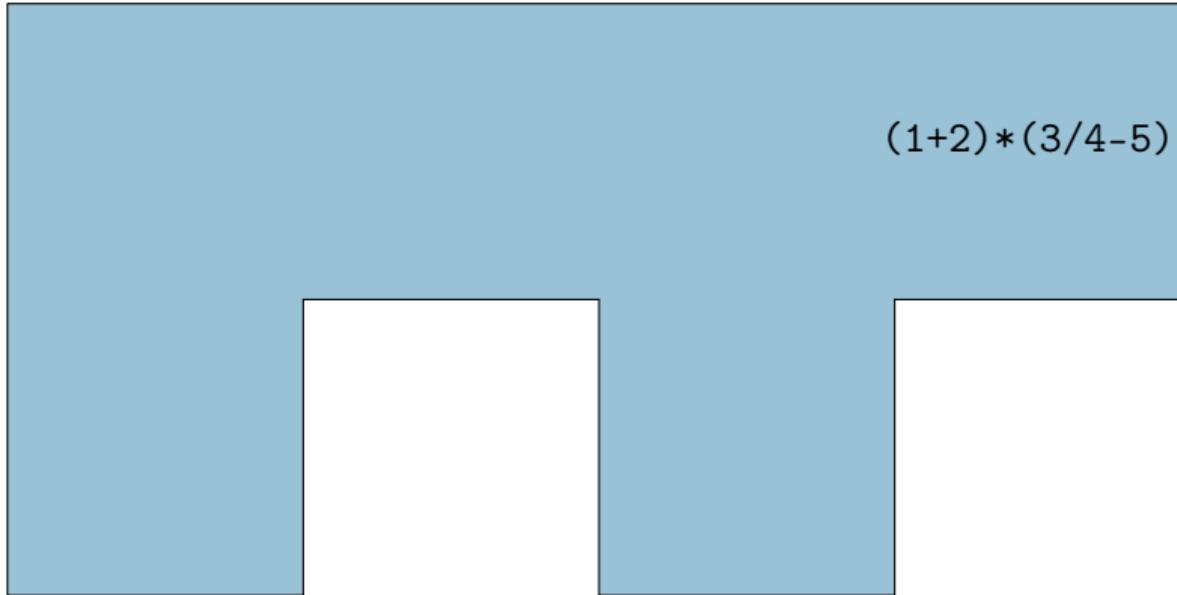


Пример: Директно пресмятане на израз

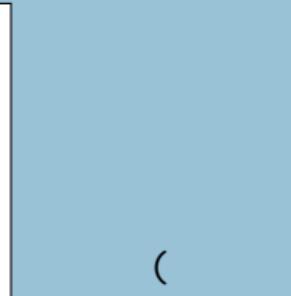
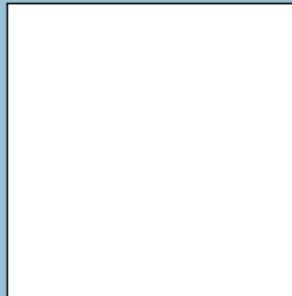


Пример 2: Директно пресмятане на израз

$$(1+2)*(3/4-5)$$



Пример 2: Директно пресмятане на израз

 $1+2)*(3/4-5)$ 

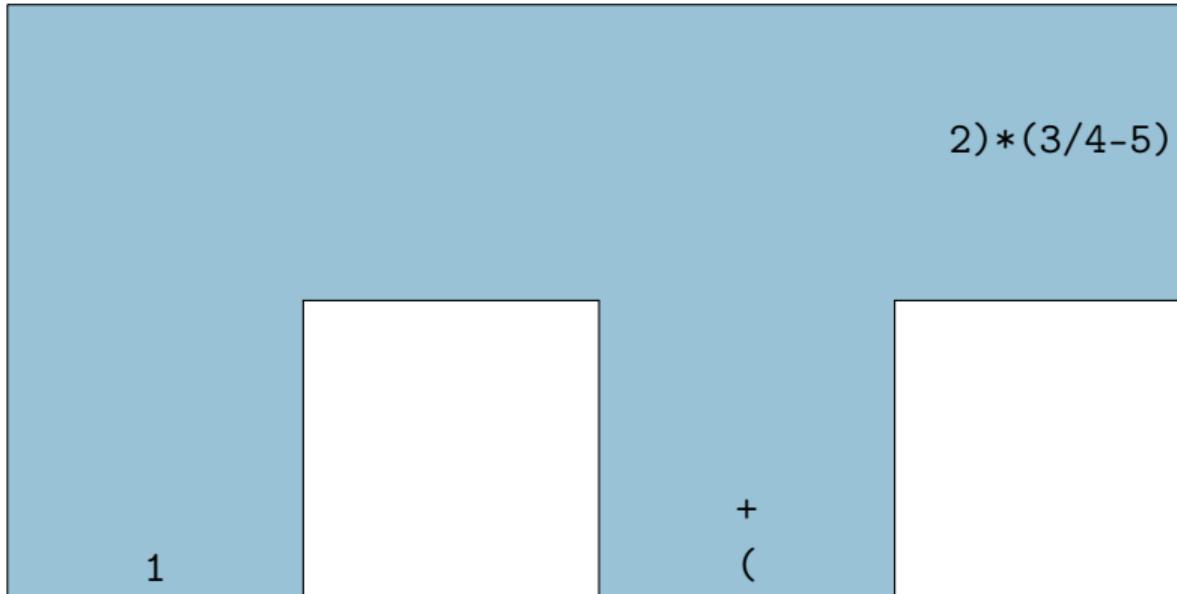
Пример 2: Директно пресмятане на израз

 $+2)*(3/4-5)$

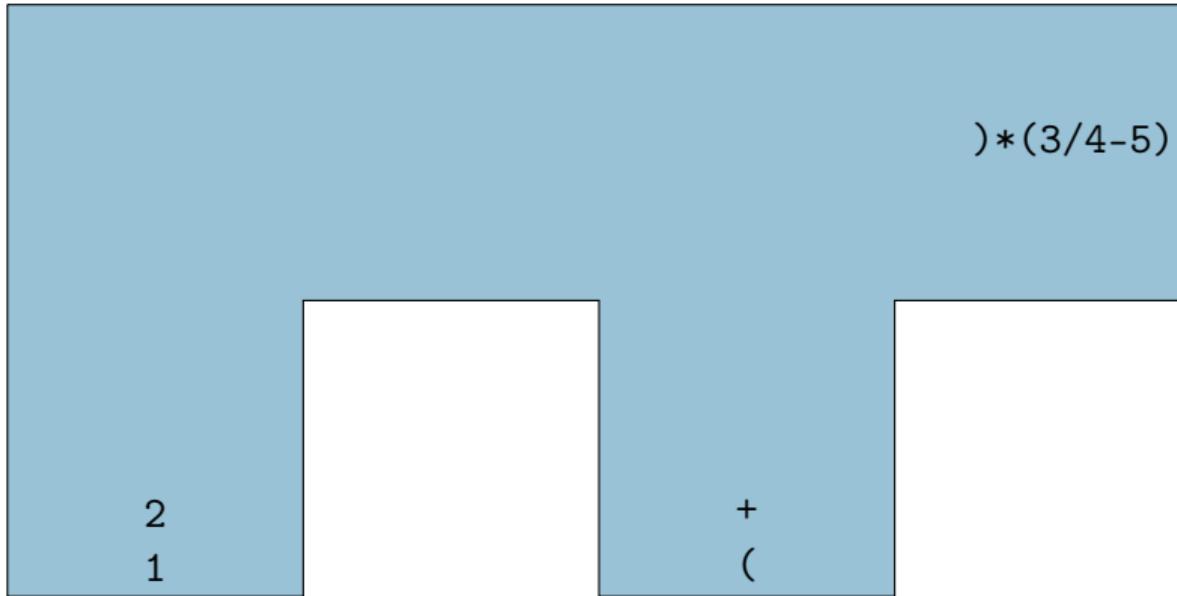
1

()

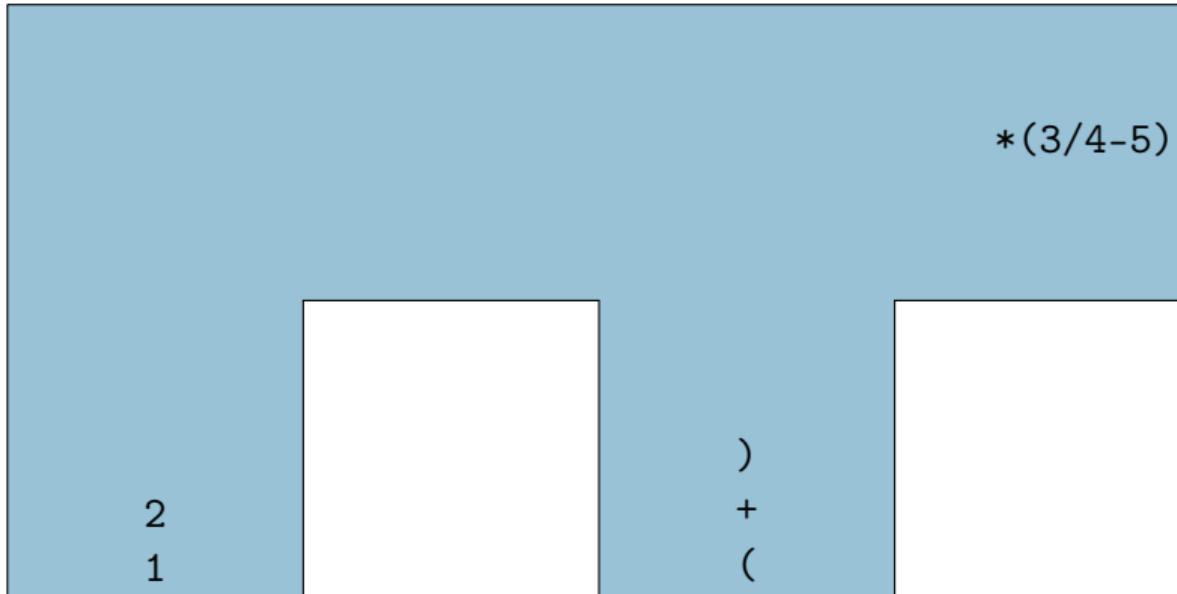
Пример 2: Директно пресмятане на израз



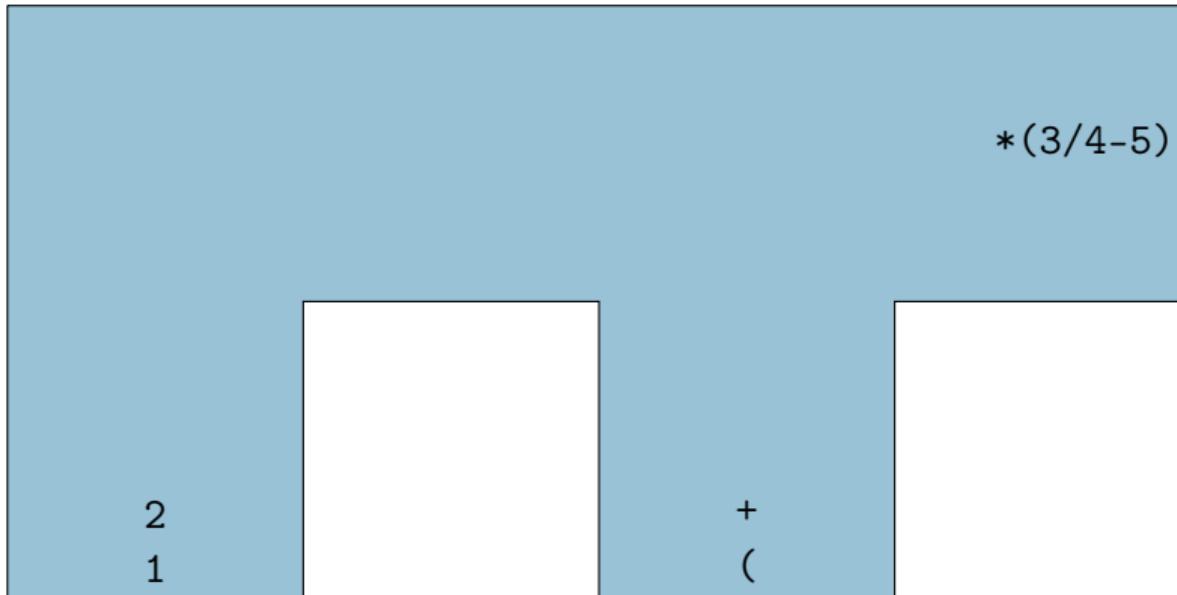
Пример 2: Директно пресмятане на израз



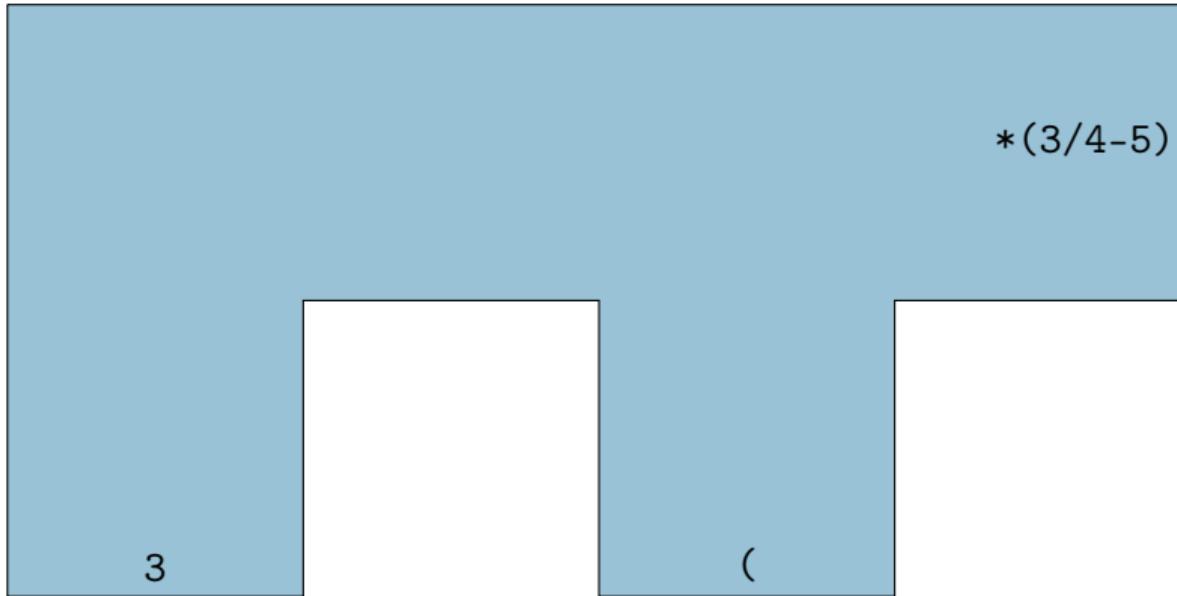
Пример 2: Директно пресмятане на израз



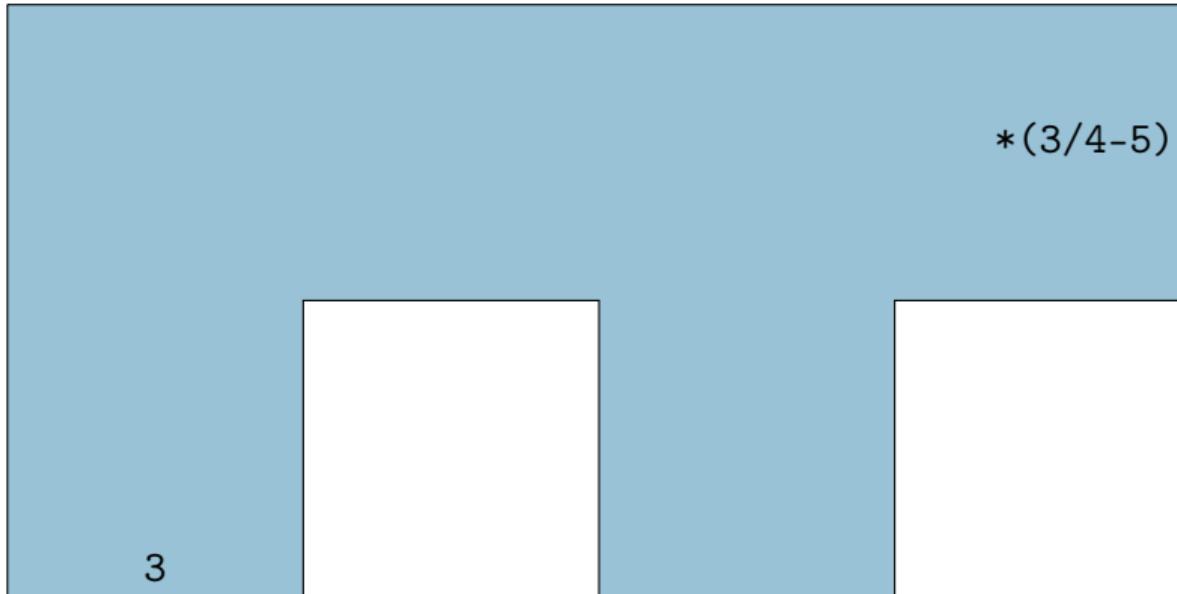
Пример 2: Директно пресмятане на израз



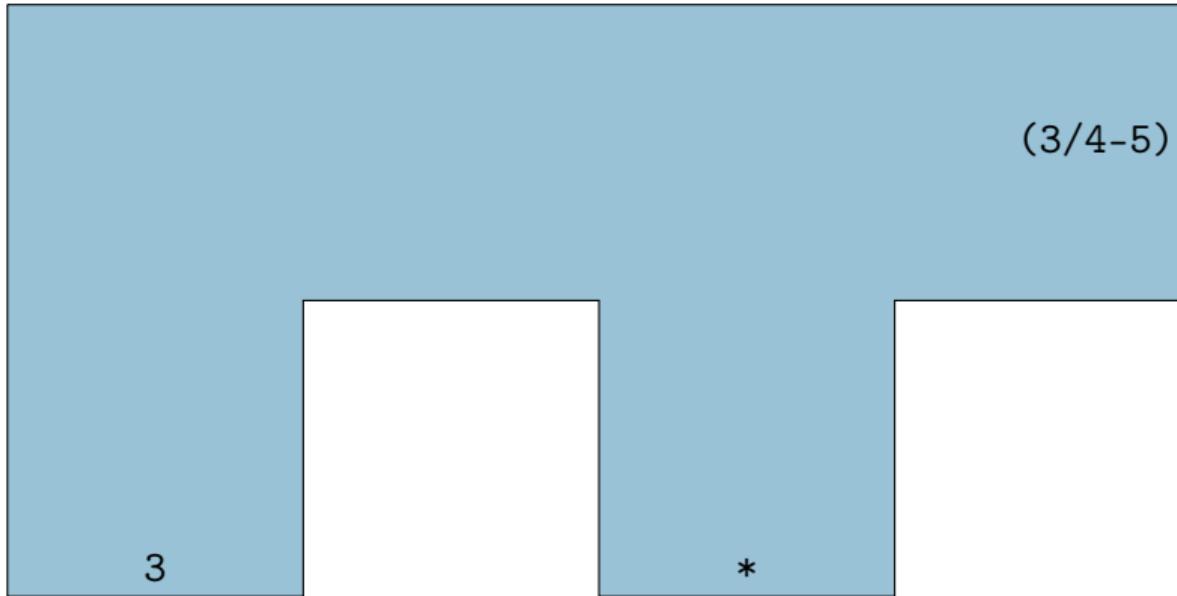
Пример 2: Директно пресмятане на израз



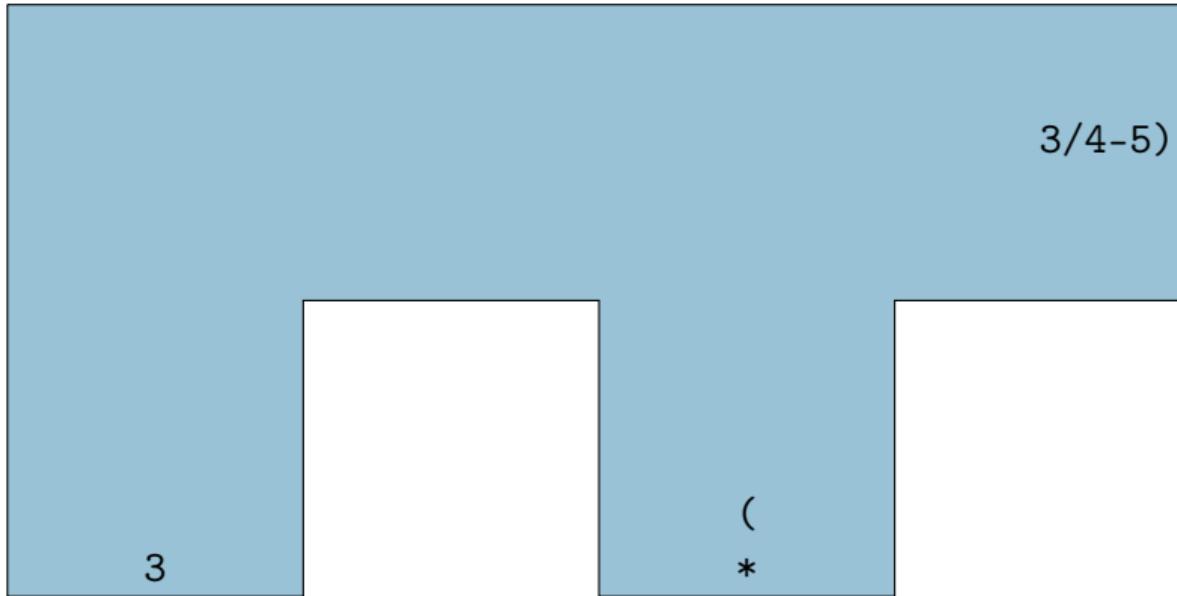
Пример 2: Директно пресмятане на израз



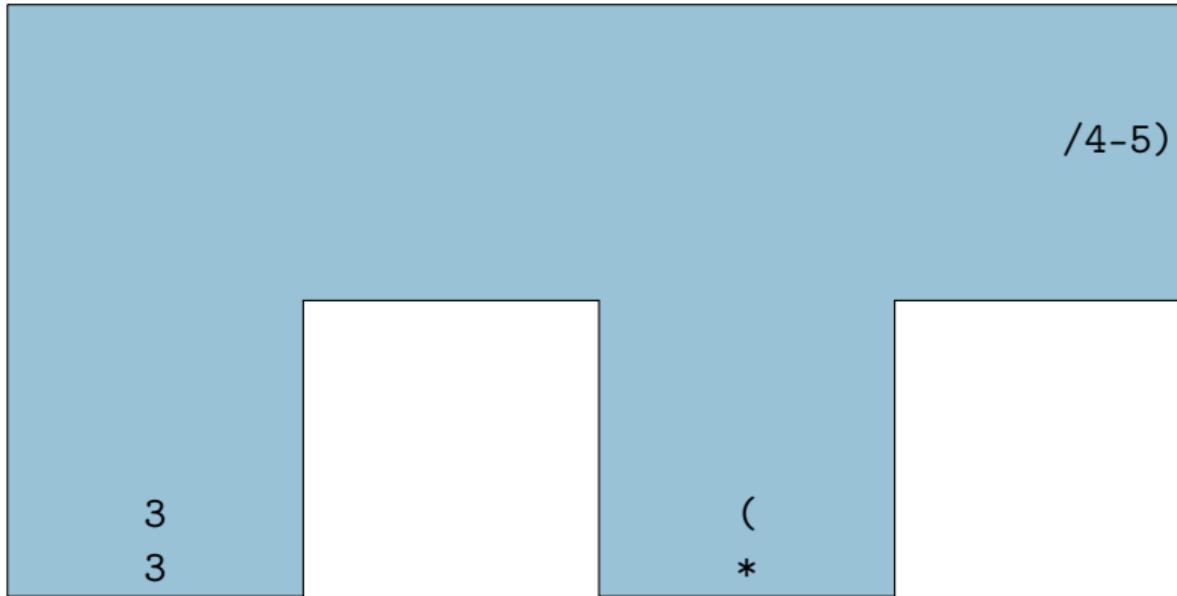
Пример 2: Директно пресмятане на израз



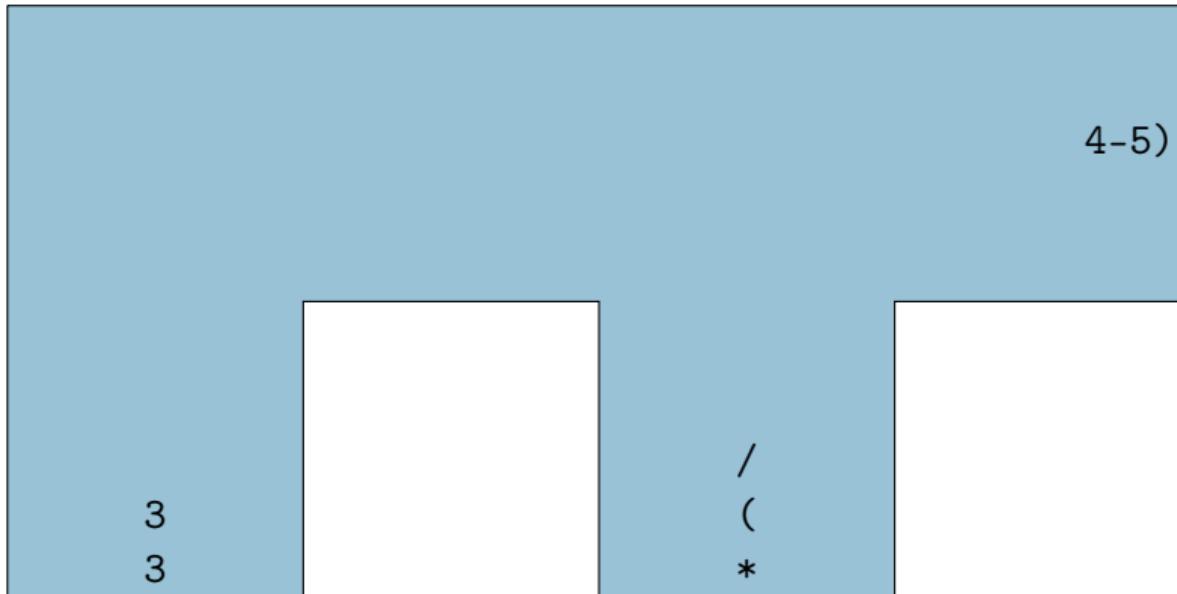
Пример 2: Директно пресмятане на израз



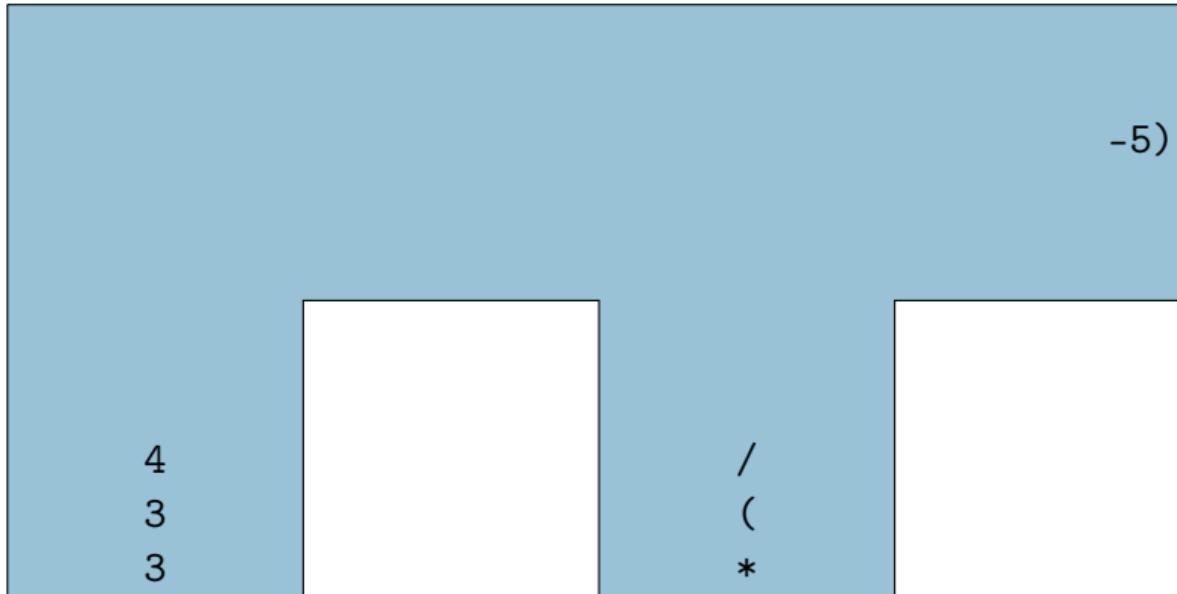
Пример 2: Директно пресмятане на израз



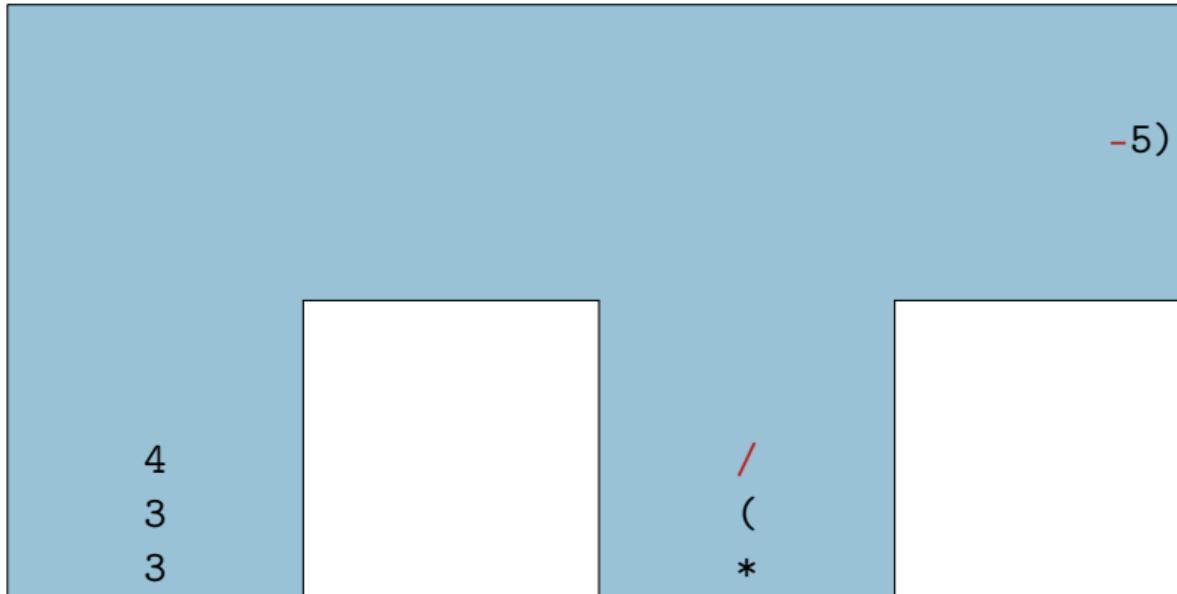
Пример 2: Директно пресмятане на израз



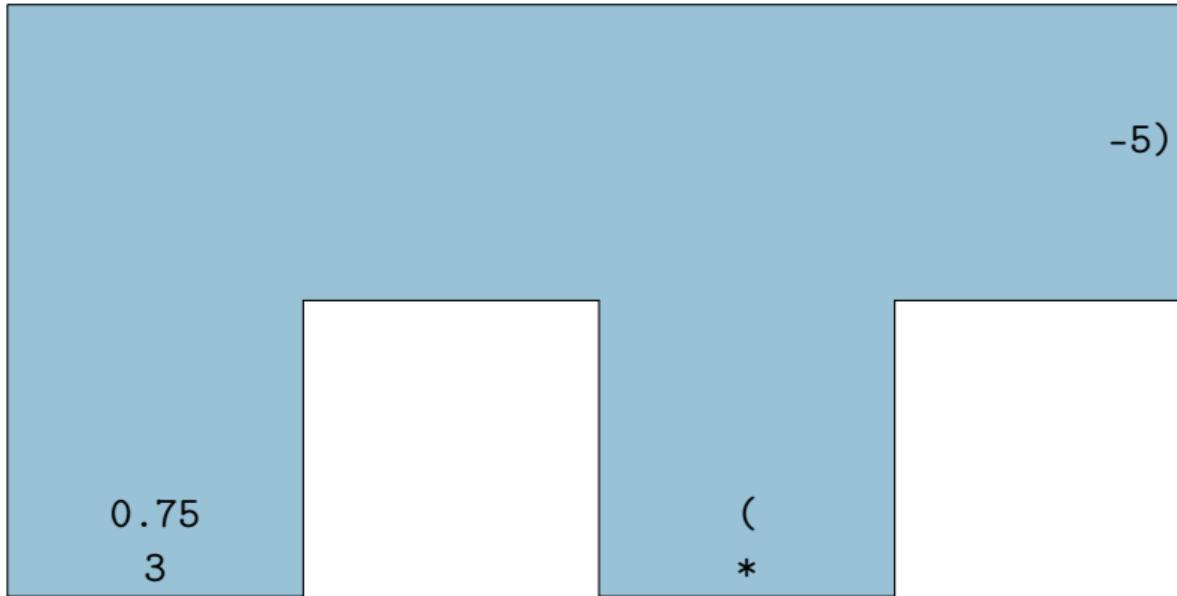
Пример 2: Директно пресмятане на израз



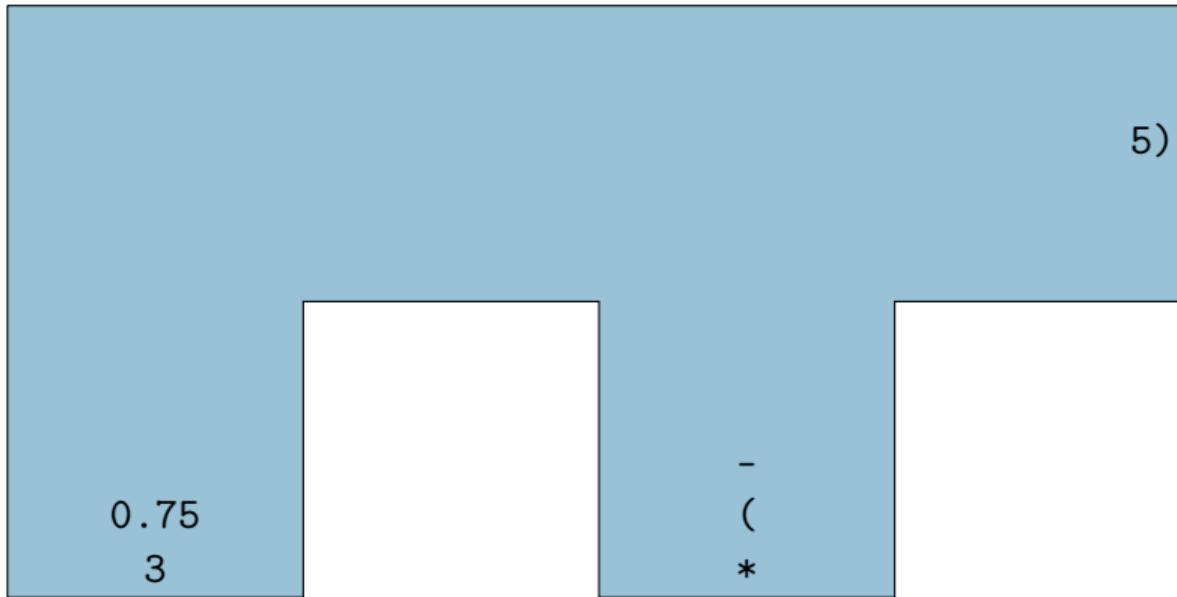
Пример 2: Директно пресмятане на израз



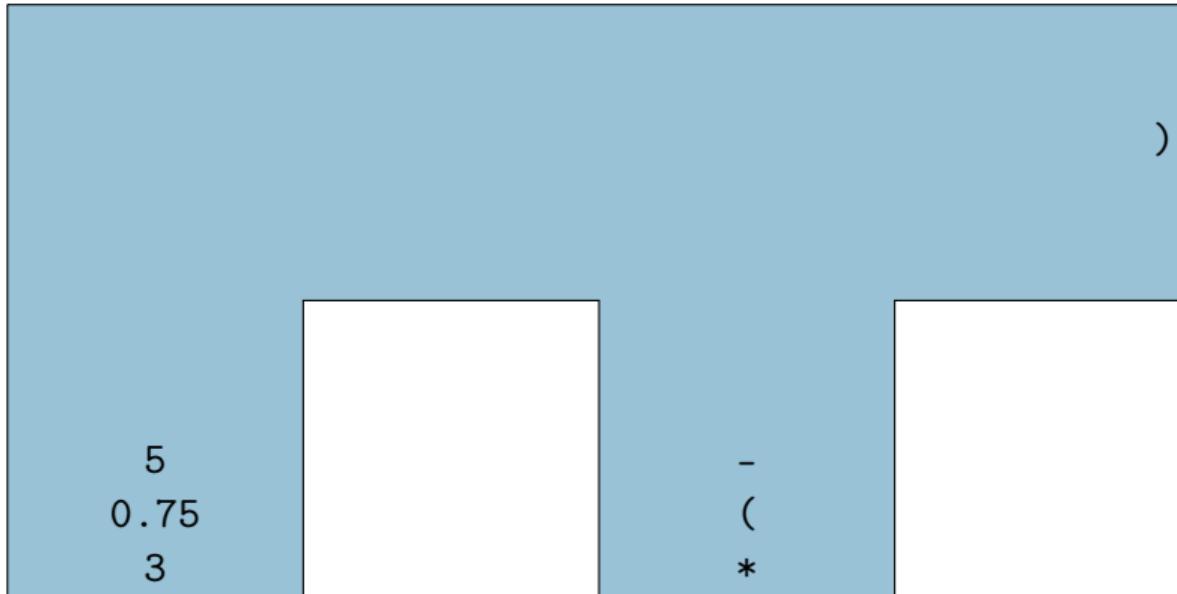
Пример 2: Директно пресмятане на израз



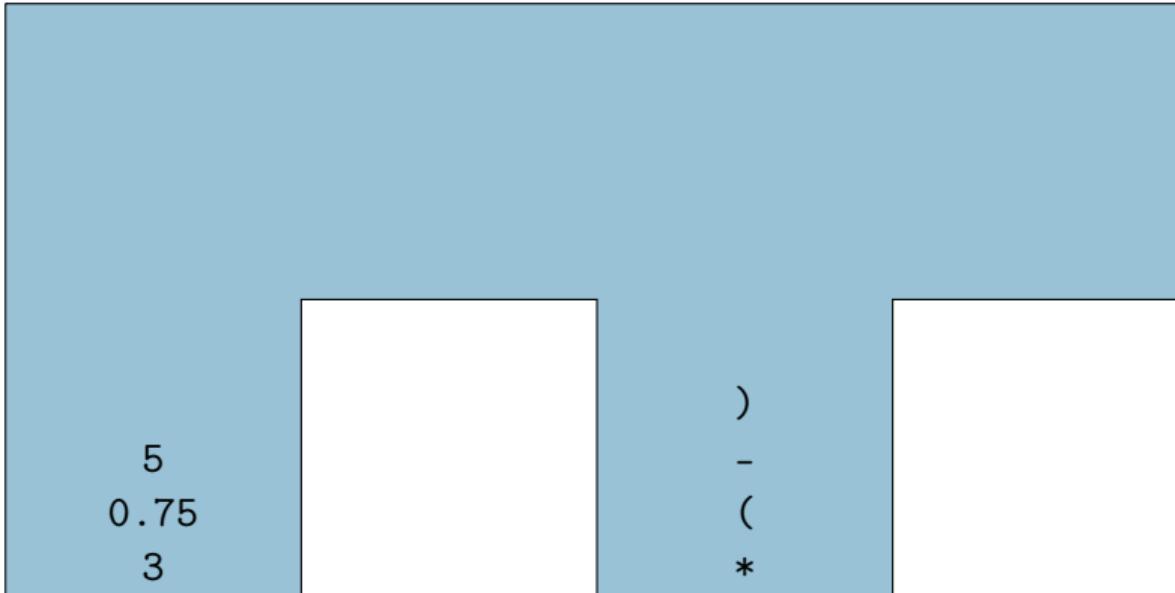
Пример 2: Директно пресмятане на израз



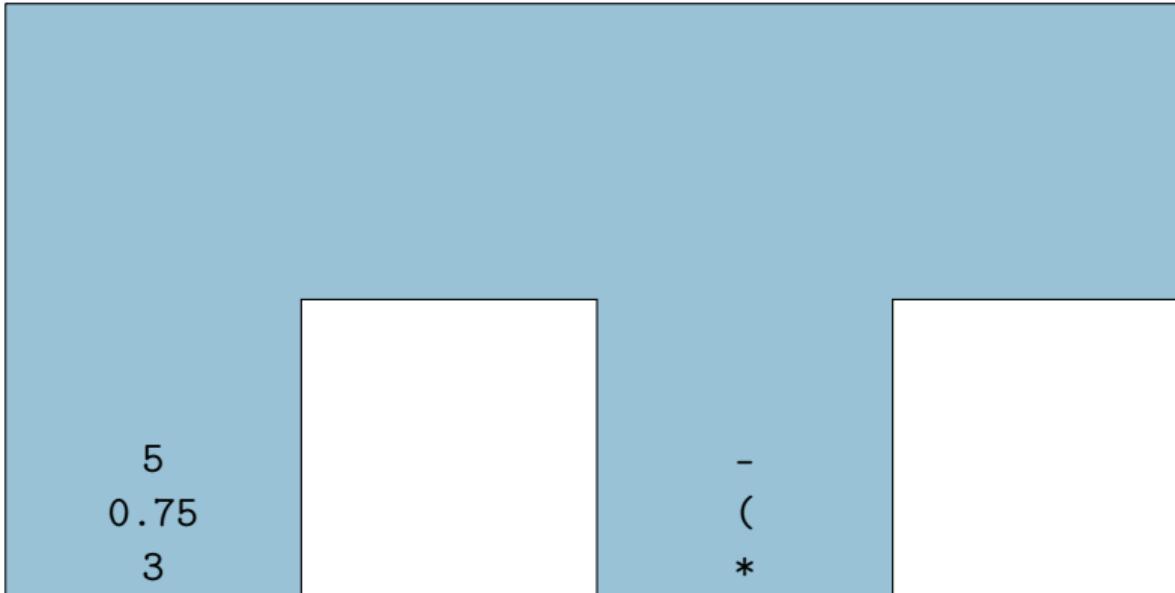
Пример 2: Директно пресмятане на израз



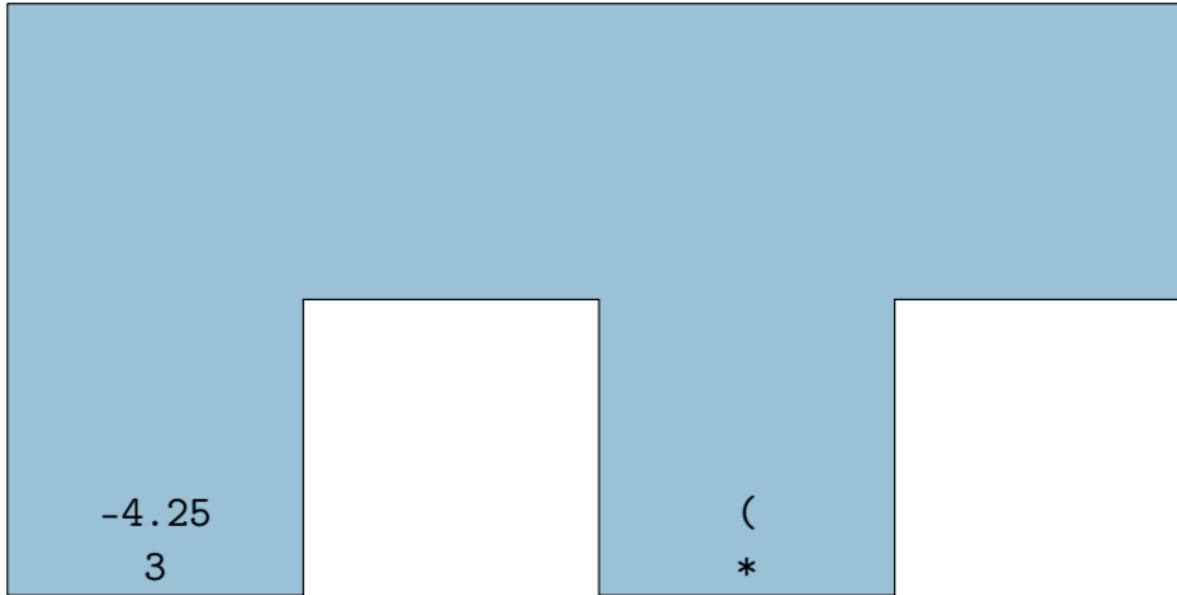
Пример 2: Директно пресмятане на израз



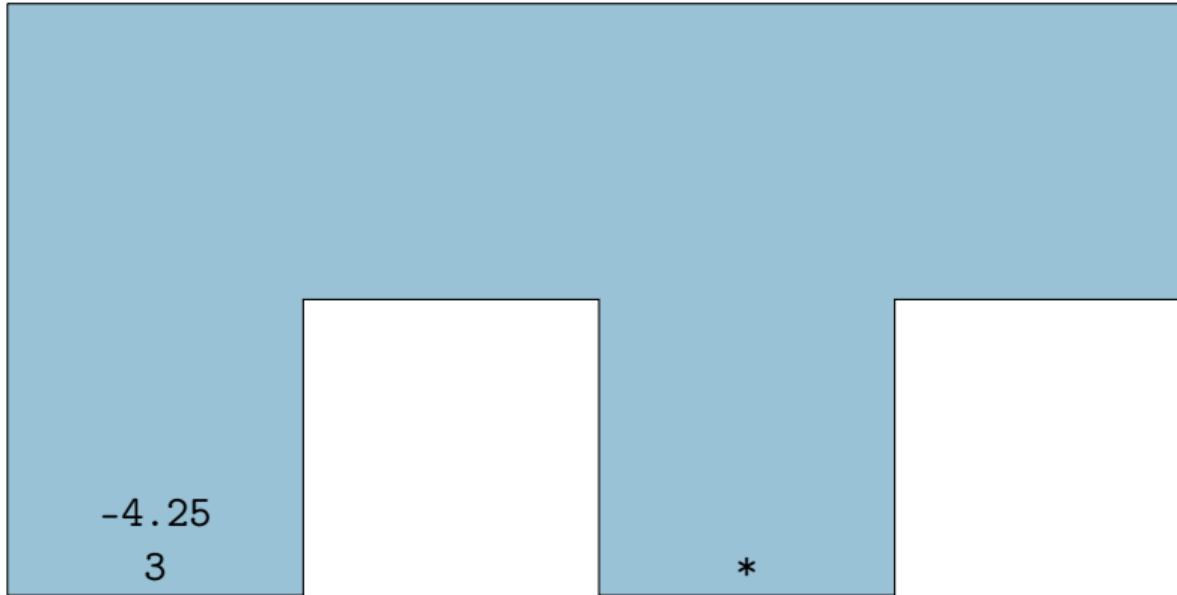
Пример 2: Директно пресмятане на израз



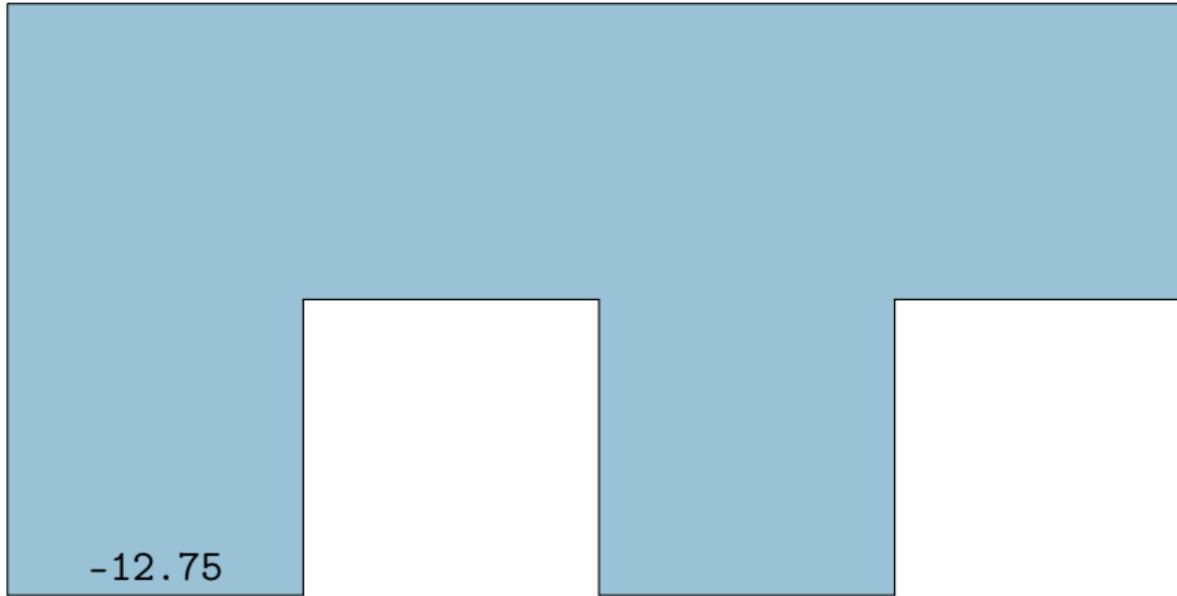
Пример 2: Директно пресмятане на израз



Пример 2: Директно пресмятане на израз



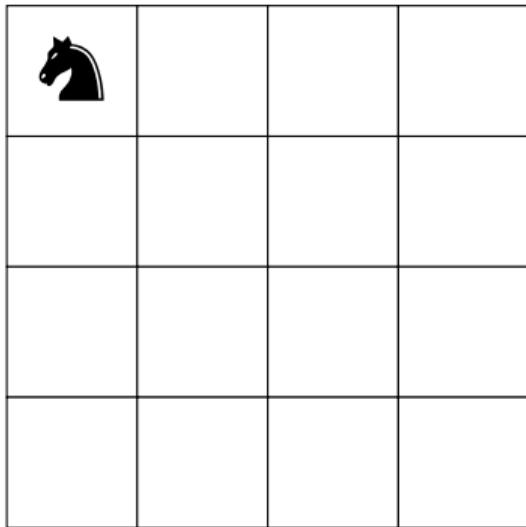
Пример 2: Директно пресмятане на израз



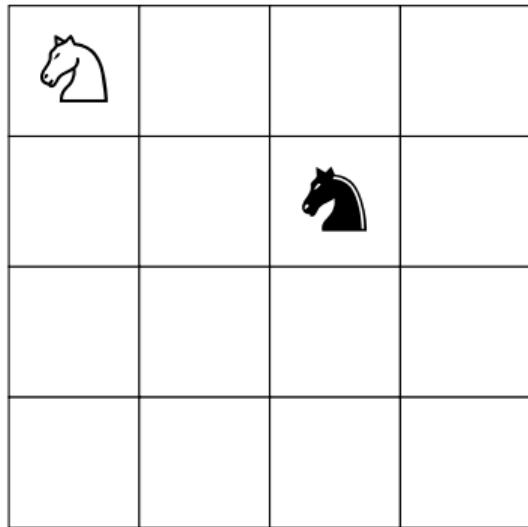
Симулиране на рекурсия

- Стекова рамка
 - при извикване на функция
 - при рекурсия
- Стек вместо стекова рамка
- Пример: ход на коня

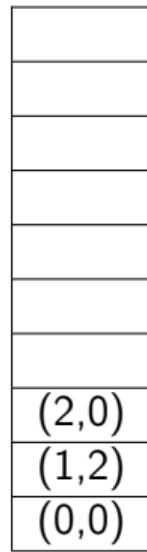
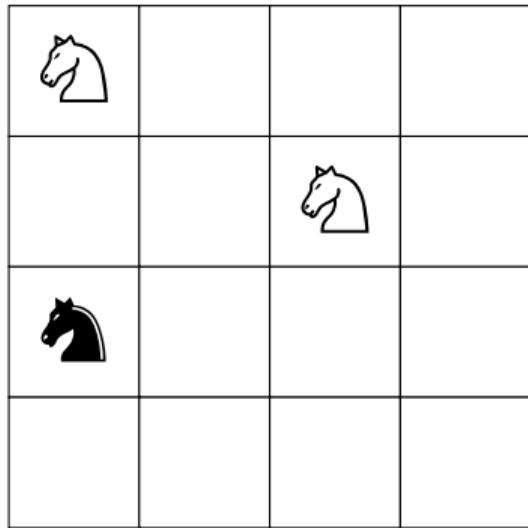
Пример: ход на коня



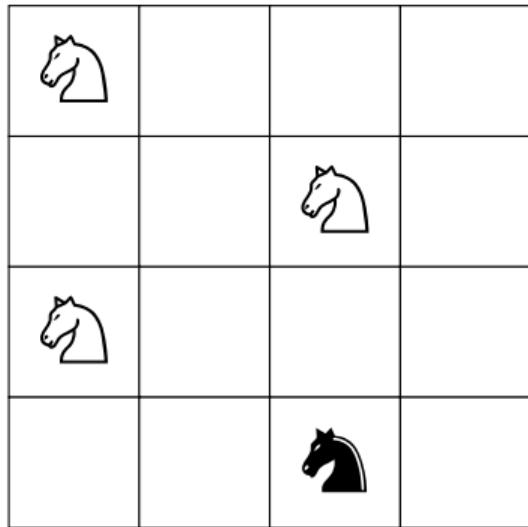
Пример: ход на коня



Пример: ход на коня

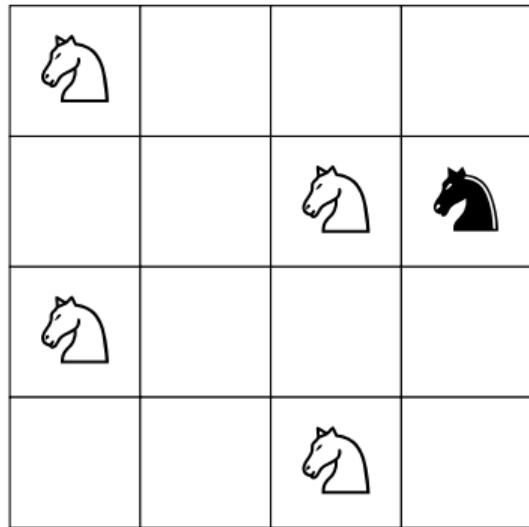


Пример: ход на коня



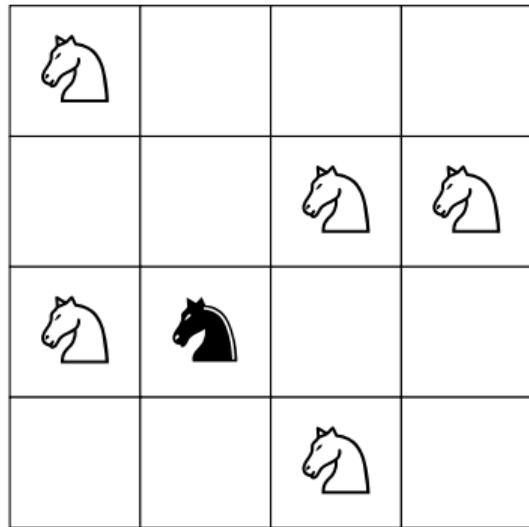
(3,2)
(2,0)
(1,2)
(0,0)

Пример: ход на коня



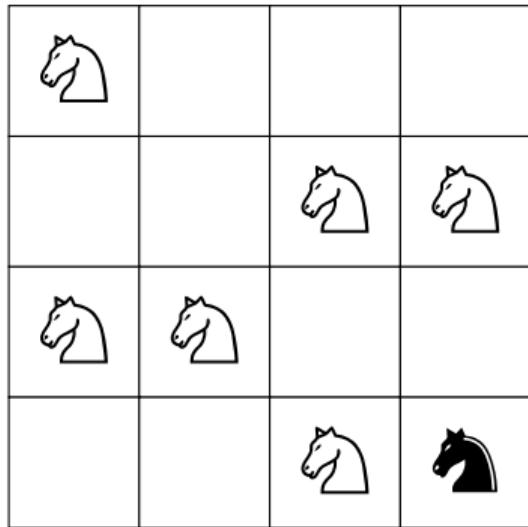
(1,3)
(3,2)
(2,0)
(1,2)
(0,0)

Пример: ход на коня



(2,1)
(1,3)
(3,2)
(2,0)
(1,2)
(0,0)

Пример: ход на коня



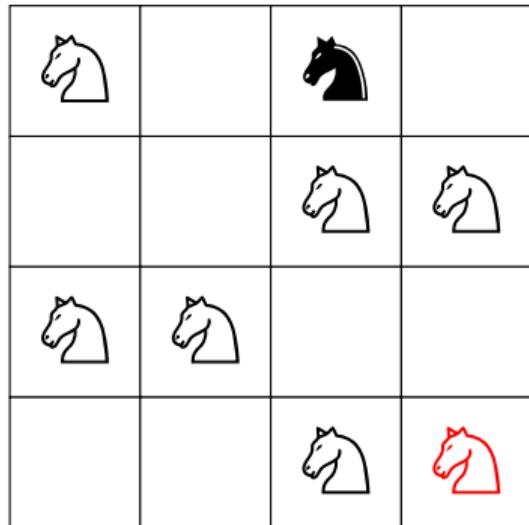
(3,3)
(2,1)
(1,3)
(3,2)
(2,0)
(1,2)
(0,0)

Пример: ход на коня



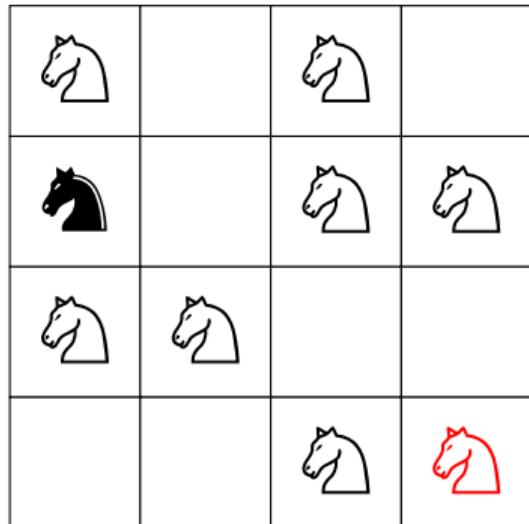
(2,1)
(1,3)
(3,2)
(2,0)
(1,2)
(0,0)

Пример: ход на коня



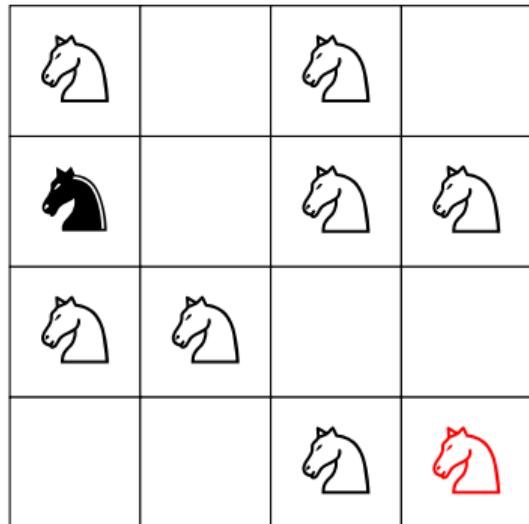
(0,2)
(2,1)
(1,3)
(3,2)
(2,0)
(1,2)
(0,0)

Пример: ход на коня



(1,0)
(0,2)
(2,1)
(1,3)
(3,2)
(2,0)
(1,2)
(0,1)
(3,1)
(1,0)
(0,2)
(2,1)

Пример: ход на коня



(1,0)
(0,2)
(2,1)
(0,1)
(1,3)
(3,2)
(0,1)
(2,0)
(3,1) (3,3)
(1,2)
(2,1)
(0,0)

std::stack<T>

- `stack()` — създаване на празен стек
- `empty()` — проверка за празнота на стек
- `push(x)` — включване на елемент на стек
- `pop()` — изключване на елемент от стек
- `top()` — последен елемент на стека
- `size()` — дължина на стека
- `==, !=, <, >, <=, >=` — лексикографско сравнение на два стека