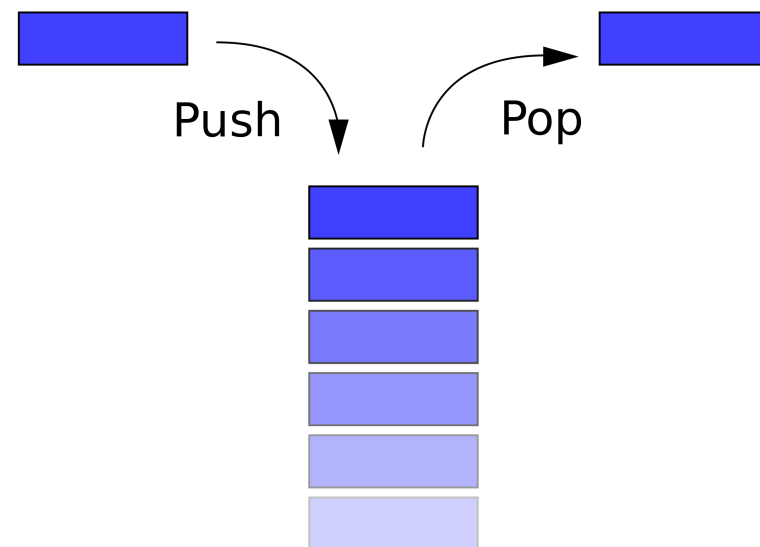
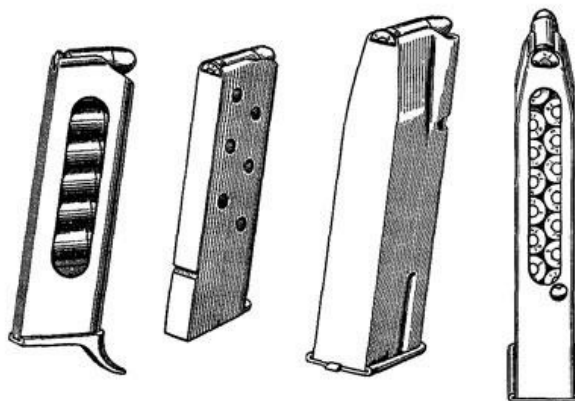
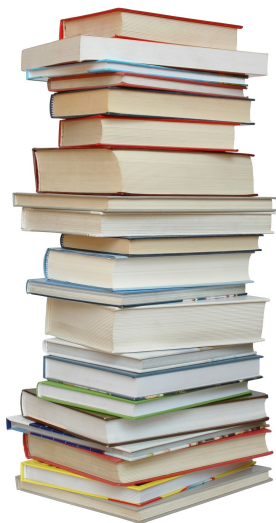
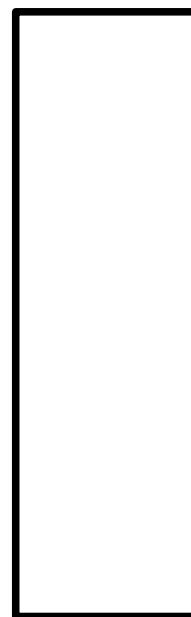
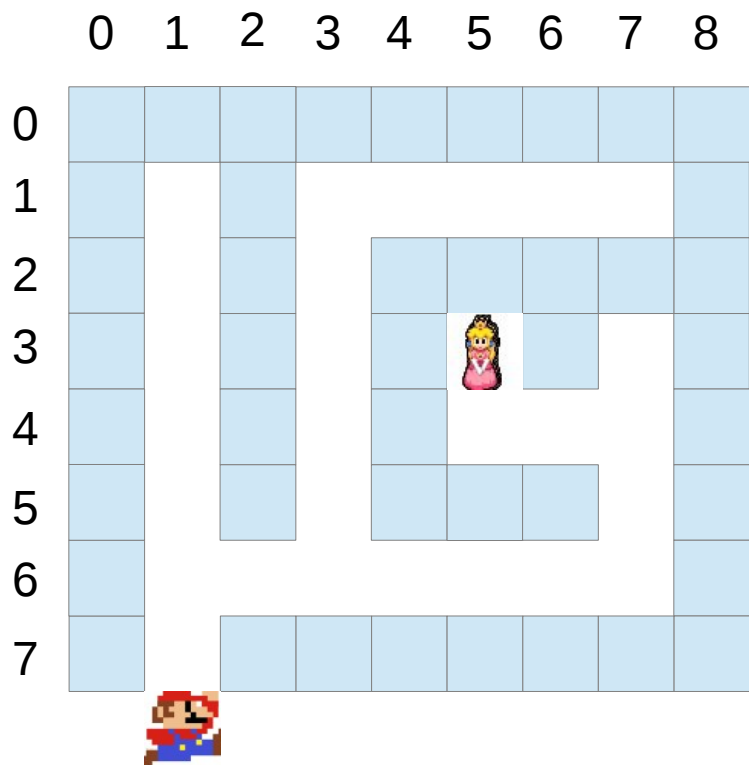


Стек

# Стек

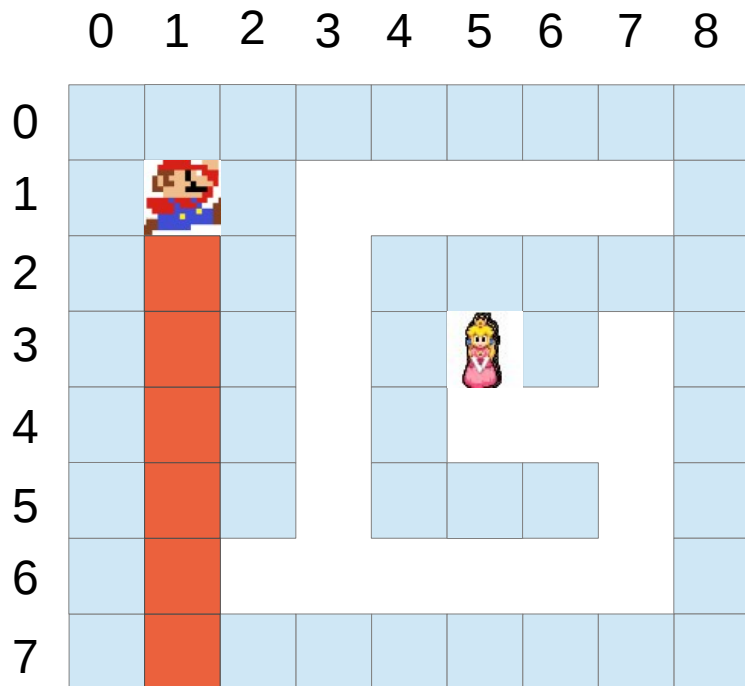


# Применение стека



# Применение стека

Будем запоминать пройденный путь в стеке



1, 1
2, 1
3, 1
4, 1
5, 1
6, 1
7, 1

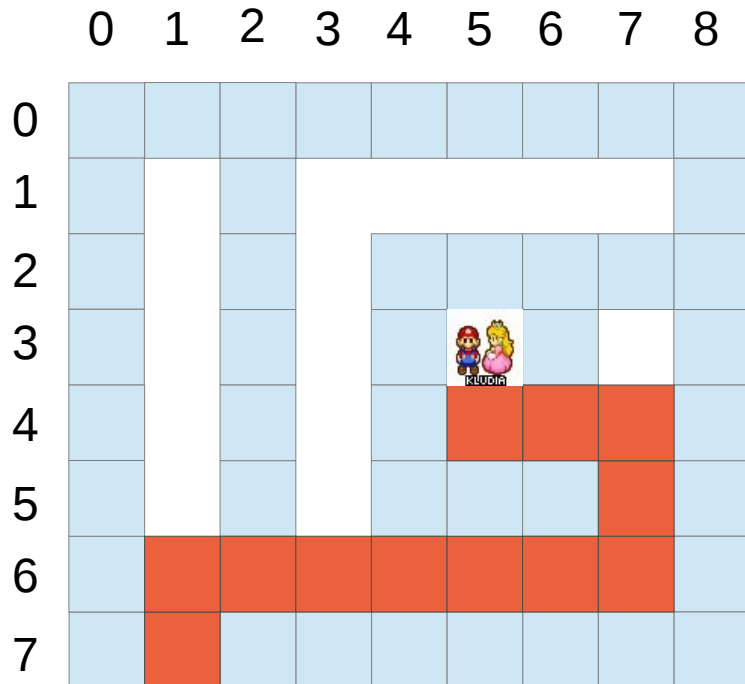
# Применение стека

	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									
5									
6									
7									

1, 7
1, 6
1, 5
.
.
.
.
6, 3
6, 2
6, 1
7, 1

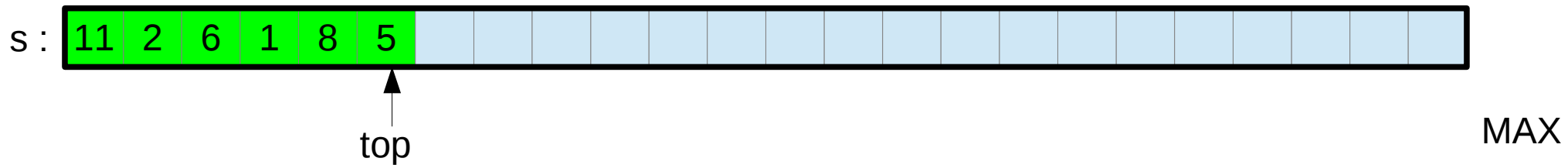
# Применение стека

Теперь выбраться из лабиринта не проблема!



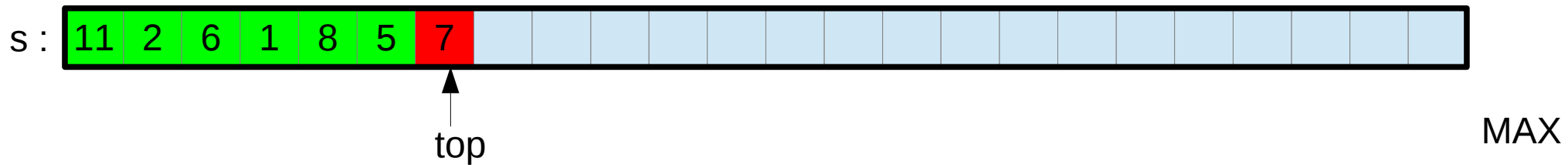
3, 5
4, 5
4, 6
.
.
.
6, 3
6, 2
6, 1
7, 1

# Стек на основе массива



```
const int MAX = 24;  
  
int s[MAX];  
  
int top = -1; // изначально top никуда не указывает  
  
bool empty()  
{  
    return (top == -1);  
}
```

# Добавление элемента



```
void push(int x)
{
    top++; // top = top + 1
    s[top] = x;
}
```





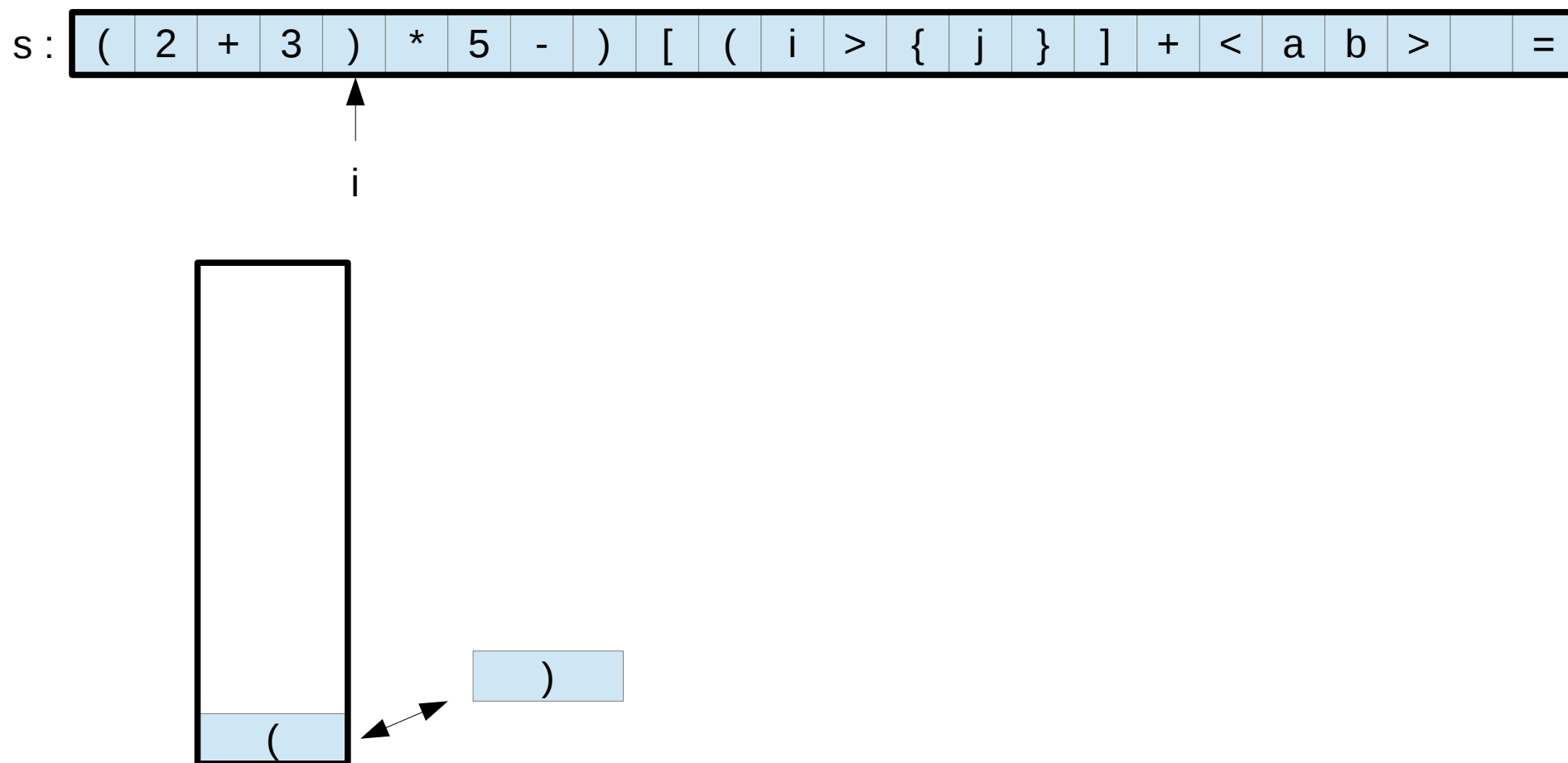
# Проверка скобок

s: ( 2 + 3 ) \* 5 - ) [ ( i > { j } ] + < a b > =

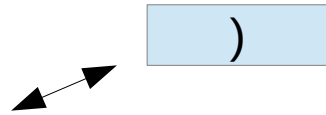
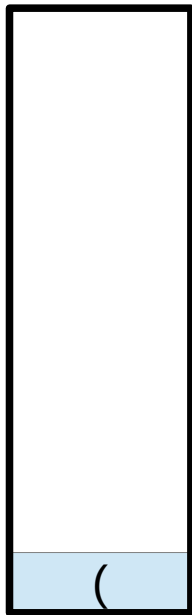
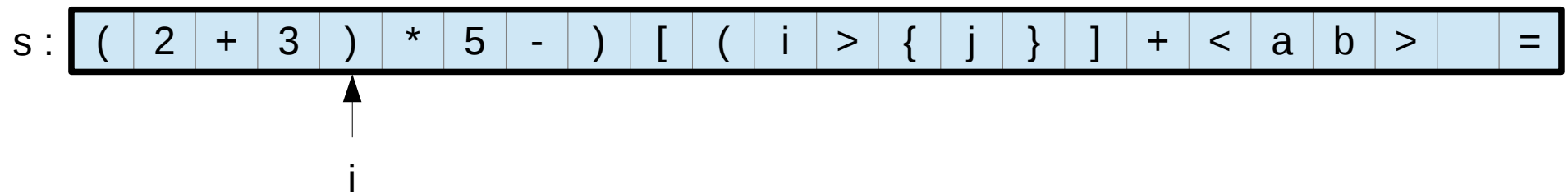
↑  
i

(

# Проверка скобок



# Проверка скобок

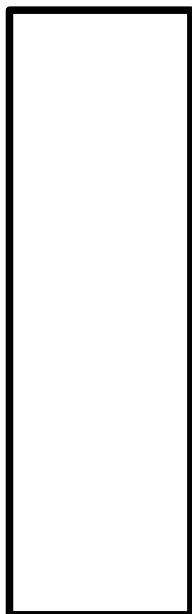


```
if (pair(s[i]) == s[i])  
{  
    pop();  
}
```

# Проверка скобок

s : ( 2 + 3 ) \* 5 - ) [ ( i > { j } ] + < a b > =

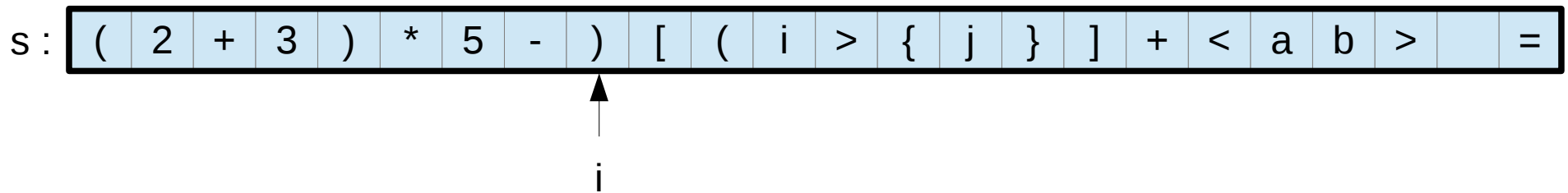
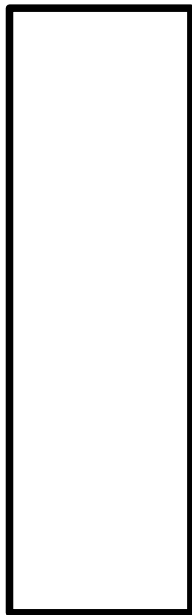
i



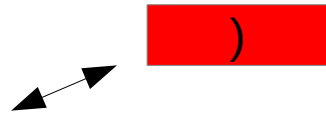
# Проверка скобок

s: ( 2 + 3 ) \* 5 - ) [ ( i > { j } ] + < a b > =

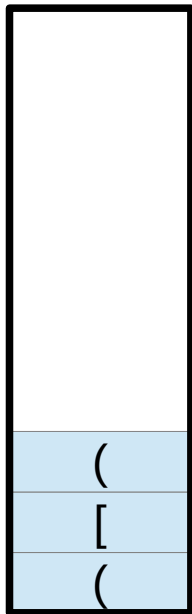
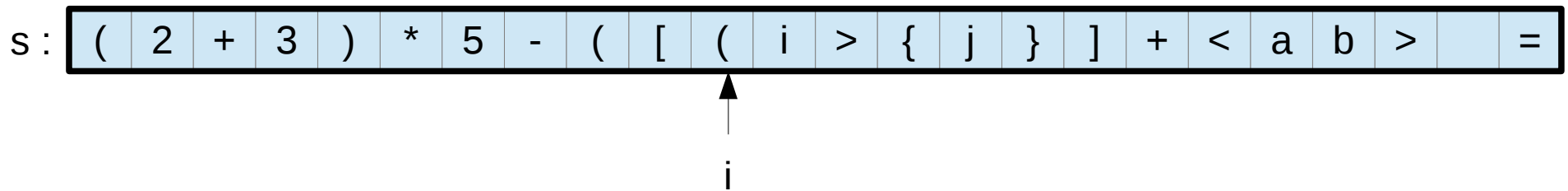
i

A horizontal array of 21 cells containing the characters: '(', '2', '+', '3', ')', '\*', '5', '-', ')', '[', '(', 'i', '>', '{', 'j', '}', ']', '+', '<', 'a', 'b', '>', '='. A vertical arrow labeled 'i' points to the 9th cell, which contains the closing parenthesis ')' of the first sub-expression.

```
if (empty())  
{  
    return false;  
}
```



# Проверка скобок

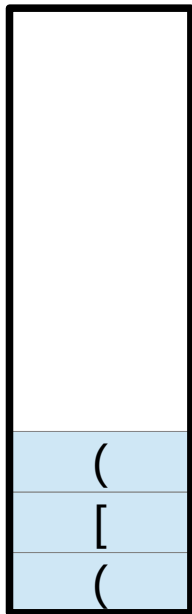


```
if (s[i] – открывающая скобка)
{
    push(s[i]);
}
```

# Проверка скобок

s: ( 2 + 3 ) \* 5 - ( [ ( i > { j } ] + < a b > =

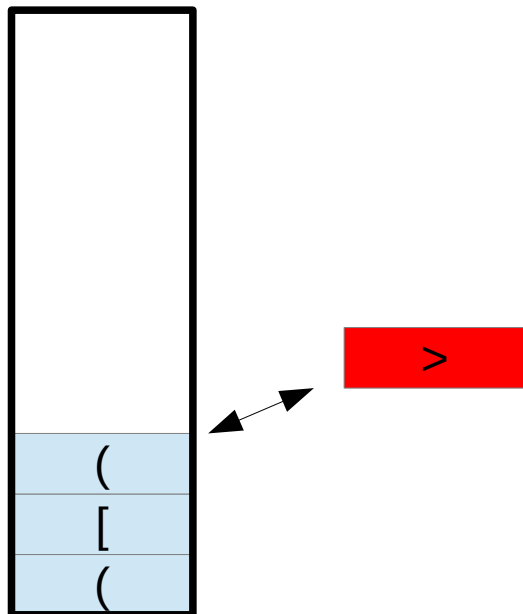
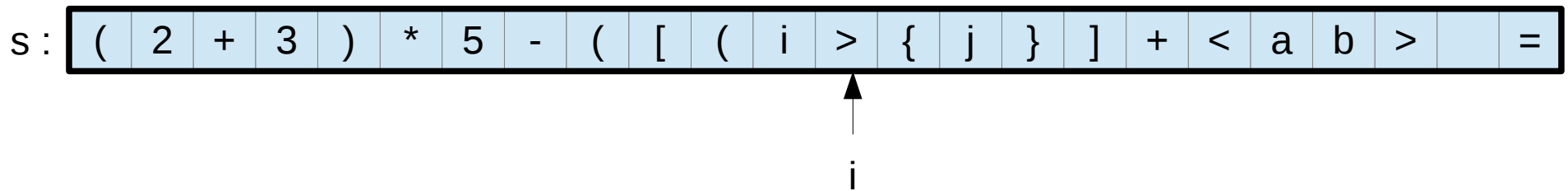
↑  
i



```
if (s[i] – открывающаяся скобка)
{
    push(s[i]);
}
```

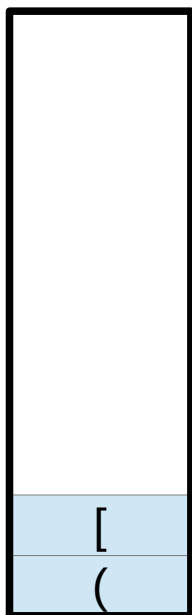
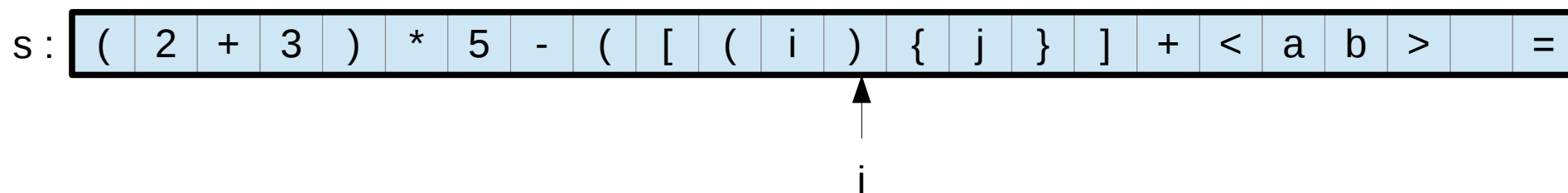


# Проверка скобок



```
if (pair(s[i]) != s[i])  
{  
    return false;  
}
```

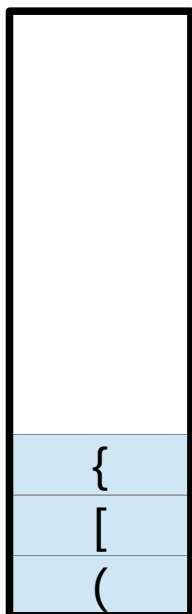
# Проверка скобок



# Проверка скобок

s : ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > =

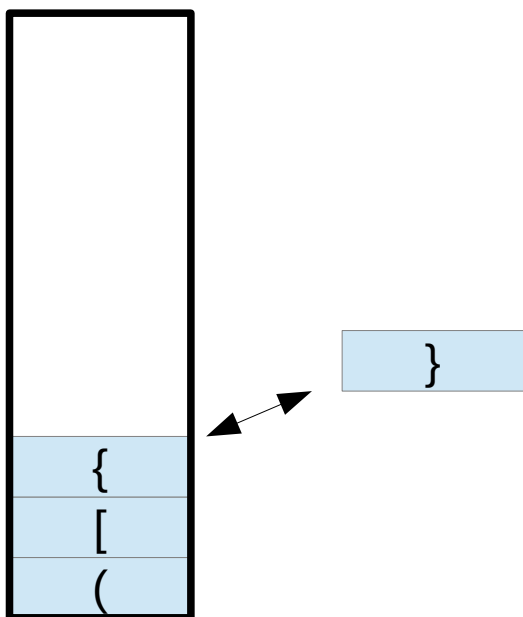
↑  
i



# Проверка скобок

s: ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > =

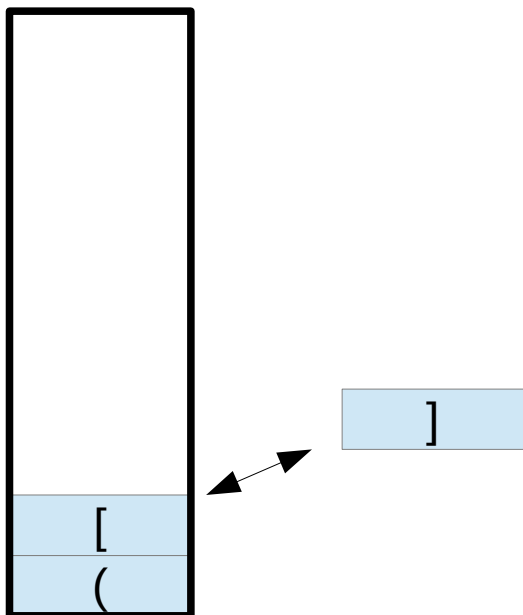
i



# Проверка скобок

s: ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > =

↑  
i

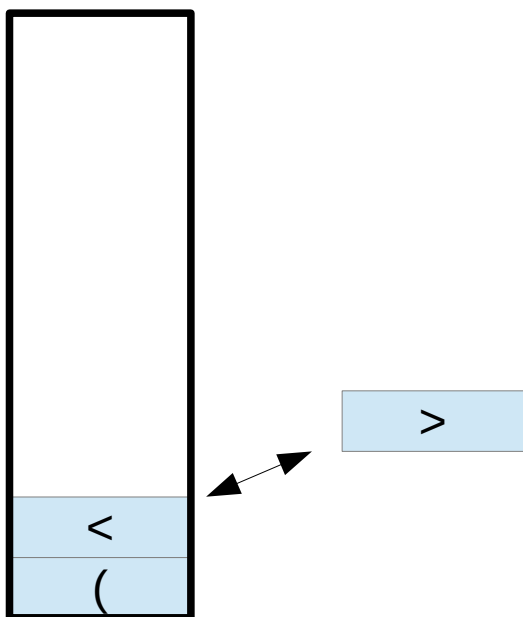




# Проверка скобок

s: ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > =

i



# Проверка скобок

s: ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > =

i

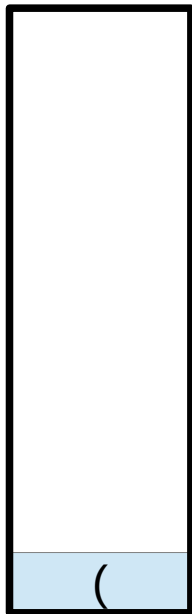
(



# Проверка скобок

s: ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > =

↑  
i



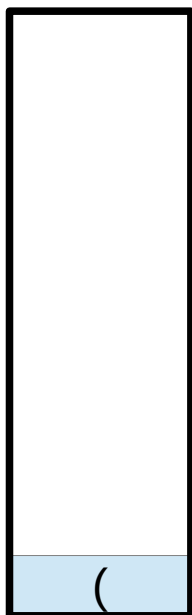
// после просмотра строки

```
if ( !empty() )  
{  
    return false;  
}
```

# Проверка скобок

str : ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > ) =

i



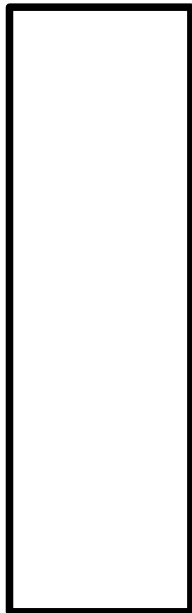
)

pop();

# Проверка скобок

s: ( 2 + 3 ) \* 5 - ( [ ( i ) { j } ] + < a b > ) =

↑  
i



// после просмотра строки

```
if ( !empty() )  
{  
    return false;  
}
```

// если стек пустой

```
return true;
```