**Скобочные последовательности. Следующая и предыдущая последовательность**

**Построение следующей скобочной последовательности**

Общий подход к построению следующего в лексикографическом порядке комбинаторного объекта универсален для разных видов объектов:

1. Найти самый правый элемент комбинаторного объекта, который можно увеличить, не модифицируя предыдущие элементы.
2. Увеличить его на минимально возможную величину.
3. Все последующие элементы сделать минимально возможными.

Для построение следующей скобочной последовательности алгоритм выглядит следующим образом:

1. Идти с конца подсчитывая количество открывающих скобок (op) и количество закрывающих скобок (cl), пока не будет встречена открывающая скобка, для которой значение op < cl (включая эту скобку.
2. Найденная открывающая скобка заменяется на закрывающую.
3. Последующие элементы делаются минимально возможными в лексикографическом порядке: сначала вставляется op открывающих скобок, затем все оставшиеся скобки до конца последовательности делаются закрывающими.

Пример реализации алгоритма. Функция возвращает true, если была построена следующая скобочная последовательность, или false, если переданная скобочная последовательность была максимальной.

**bool next(string & S)**

**{**

**int op = 0, cl = 0, i;**

**for (i = S.size() - 1; i >= 0; --i)**

**{**

**if (S[i] == '(')**

**++op;**

**else**

**++cl;**

**if (S[i] == '(' && op < cl)**

**break;**

**}**

**if (i == -1)**

**return false;**

**S[i] = ')';**

**++i;**

**while (op > 0)**

**{**

**S[i] = '(';**

**++i;**

**--op;**

**}**

**while (i < S.size())**

**{**

**S[i] = ')';**

**++i;**

**}**

**return true;**

**}**

**Построение предыдущей скобочной последовательности**

Идея алгоритма построения предыдущей скобочной последовательности аналогичная.

1. Найдем самую правую открывающую скобку. Затем будем двигаться к началу последовательности, пока не найдем закрывающую скобку. Подсчитываем число открывающих и закрывающих скобок (op и cl).

2. Заменим закрывающую скобку на открывающую. При этом значение cl уменьшается на 1.

3. Следующие символы нужно сделать максимально большими в лексикографическом порядке. Поэтому будем сначала ставить закрывающие скобки, уменьшая значение cl, пока значения op и cl не сравняются. После этого будем ставить скобки парами: "()()()"

Пример реализации алгоритма:

**bool prev (string & S)**

**{**

**int op = 0, cl = 0, i;**

**for (i = S.size() - 1; i >= 0; --i)**

**{**

**if (S[i] == '(')**

**++op;**

**else**

**++cl;**

**if (S[i] == '(')**

**break;**

**}**

**--i;**

**for (; i >= 0; --i)**

**{**

**if (S[i] == '(')**

**++op;**

**else**

**++cl;**

**if (S[i] == ')')**

**break;**

**}**

**if (i < 0)**

**return false;**

**S[i] = '(';**

**--op;**

**++i;**

**while (op != cl)**

**{**

**S[i] = ')';**

**--cl;**

**++i;**

**}**

**while (i < S.size())**

**{**

**S[i] = '(';**

**S[i + 1] = ')';**

**i += 2;**

**}**

**return true;**

**}**