# Упражнения: Генериране на комбинации

## Генериране на комбинации без повторение

Да се напише програма, която въвежда число и генерира всички възможни комбинации без повторение от **k** клас от неговите цифри.

### Пример

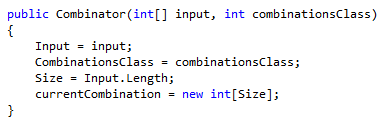
|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 12345  3 | 1 2 3  1 2 4  1 2 5  1 3 4  1 3 5  1 4 5  2 3 4  2 3 5  2 4 5  3 4 5 |

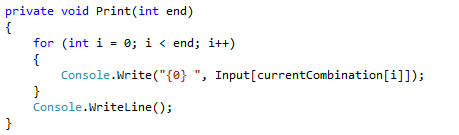
### Решение

Алгоритъмът за генериране на комбинации е удобно да се реализира чрез **backtracking** схемата. Ще използваме помощен масив - **int[] currentCombination**, в който ще пазим какво число сме поставили на позиция **i**.

За целите на задачата ще приемем, че вече сме раздробили числото към масив **int[] input**, в който всяка цифра се намира на отделен елемент.

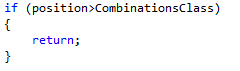
Този масив ще подадем като параметър към **конструктора** на класа **Combinator**, освен него трябва да подадем и желания клас за генериране на комбинация:



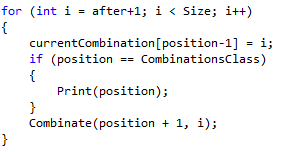
Нека да създадем метод за извеждане до дадена позиция, който изглежда така:  


Сега нека да реализираме според **backtracking** схемата и Combinate(int **position**, int **after**) метод, който ще се опитва да разположи число (от after+1 нататък) върху индекс **position.**

Ако позицията се окаже по-голяма на **CombinationClass,** то ние сме приключили с разполагането на елементи и трябва да прекратим рекурсията.



В противен случай, обхождаме всичките възможни кандидати и разполагаме съответния кандидат на позиция **position-1** в **currentCombination.** След това извикваме метода рекурсивно за следващата позиция и указваме, че следващия кандидат трябва да седи след текущия, т.е. следващите кандидати трябва да са по-големи от стойността на **i**:



След конструирането на класа **Combinator**, може да го ползвате така:



## Генериране на комбинации с повторение

Да се напише програма, която въвежда число и генерира всички възможни комбинации с повторение от **k** клас от неговите цифри.

### Пример

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 12345  3 | 1 1 1  1 1 2  1 1 3  1 1 4  1 1 5  1 2 2  1 2 3  1 2 4  1 2 5  1 3 3  1 3 4  1 3 5  1 4 4  1 4 5  1 5 5  2 2 2  2 2 3  2 2 4  2 2 5  2 3 3  2 3 4  2 3 5  2 4 4  2 4 5  2 5 5  3 3 3  3 3 4  3 3 5  3 4 4  3 4 5  3 5 5  4 4 4  4 4 5  4 5 5  5 5 5 |

### Решение

Решението се състои в леко модифициране на предния код. За да допускаме повторения, трябва цикълът да започва от **after**, вместо **after+1**. Също така при извикването на метода трябва да използваме за параметри 1 и 0, вместо 1 и -1.