Немного изменим задачу, рассмотренную на прошлых лекциях.

Генерация перестановок
Вывести все последовательности чисел от 1 до n, в которых каждое число встречается ровно один раз. Такие последовательности называются *перестановками*.

Например, при $n = 3$ существует 6 различных перестановок. Ниже они все приведены в лексикографическом порядке.

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

Упражнение 1.3.1
Какие из данных последовательностей являются перестановками?
1 2 3 4 5
3 2 4 1 6
3 2 3 1 4
4 2 5 3 1

Представим, что мы решаем задачу о генерации перестановок при помощи функции rec, которую мы использовали для решения предыдущей задачи (см. рис. 1).

Переменная m в цикле заменена на n, потому что в данном случае количество чисел, которые мы можем использовать, равно n — количеству элементов в перестановке.

for (**int** i = 1; i <= n; i++)

Приведённая функция будет использовать одни и те же числа несколько раз. Нужно запретить ей это делать. Для этой цели мы создадим вектор used (англ. «использованный») с элементами логического типа bool:

vector<**bool**> used;

```
void rec(int idx)
{
    if (idx == n)
    {
        out();
        return;
    }
    for (int i = 1; i <= n; i++)
    {
        a[idx] = i;
        rec(idx + 1);
    }
}
```

Рис. 1. Перебор последовательностей с повторениями

Будем постепенно заполнять вектор a, как в предыдущей задаче, и при этом отмечать в векторе used, какие числа уже использованы. Значение used[i] = true, если число i использовано, и used[i] = false в противном случае — если число i пока свободно. В функции main инициализируем переменную used вектором размера $(n + 1)$, все значения которого — ложные.

used = vector<**bool**>(n + 1, **false**);

Размер вектора $(n + 1)$, потому что числа в перестановке будут от 1 до n, и нам будет нужен n-й элемент вектора used.

В функцию rec нужно будет внести изменения, отмеченые на рис. 2.

```
void rec(int idx)
{
    if (idx == n)
    {
        out();
        return;
    }
    for (int i = 1; i <= n; i++)
    {
        if (used[i])
            continue;
        a[idx] = i;
        used[i] = true;
        rec(idx + 1);
        used[i] = false;
    }
}
```

Рис. 2. Решение задачи о генерации перестановок

Когда мы в цикле перебираем, какое число i поставить в текущую позицию, нужно проверить, что это число i не было использовано ранее. Если used[i] = true, то нам нельзя использовать это значение, и мы переходим на следующую итерацию цикла.

if (used[i])
    continue;

Если число i не использовано, то оно помещается в ячейку a[idx]:

a[idx] = i;

После этого число i стало использованным, и мы отмечаем это в векторе used:

used[i] = **true**;

И делаем рекурсивный переход.

rec(idx + 1);

После этого нужно обязательно освободить число i, чтобы его можно было использовать в дальнейшем.

used[i] = **false**;

Разберём работу новой функции rec на примере $n = 2$. Нужно перебрать все перестановки из двух элементов. Вначале мы заходим в rec с idx = 0 и вектором used, заполненным ложными значениями. Очевидно, idx $\neq n$. Условие не выполняется:

if (idx == n)

Переходим к циклу. Сначала $i = 1$.

for (**int** i = 1; i <= n; i++)

Значение $i = 1$ пока не используется в векторе a, и used[1] = false. Значит, условие не выполняется.

if (used[i])

Помещаем значение 1 в текущую позицию в векторе a.

a[idx] = i;

Пометим в векторе used, что число 1 использовано.

used[i] = **true**;

Выполняем рекурсивный переход.

rec(idx + 1);

Мы оказываемся на следующем уровне рекурсии с idx = 1. Условие снова не выполняется.

if (idx == n)

Переходим к циклу, $i = 1$.

for (**int** i = 1; i <= n; i++)

Теперь число 1 использовано, мы уже поместили его в вектор a, поэтому условие выполнится.

if (used[i])

Будет выполнена команда

continue;

Мы перейдём на следующую итерацию цикла с $i = 2$.

for (**int** i = 1; i <= n; i++)

Число 2 пока не используется, поэтому used[2] = false.

if (used[i])

Помещаем число 2 в a[idx], то есть в a[2].

a[idx] = i;

Помечаем в векторе used, что число 2 использовано.

used[i] = **true**;

Выполняем переход на следующий уровень рекурсии.

rec(idx + 1);

Теперь idx = 2, и условие выхода из рекурсии выполняется.

if (idx == n)

Выводим полученную последовательность «1 2».

out();

И выходим из функции rec.

return;

Мы возвращаемся на предыдущий уровень рекурсии, где idx = 1, $i = 2$. Освобождаем число 2: присваиваем used[2] значение false.

used[i] = **false**;

При этом число 2 остаётся в векторе a, но это рудимент с предыдущих шагов, и можно считать, что его там нет.

Цикл выполняться больше не будет, потому что i стало больше n.

for (**int** i = 1; i <= n; i++)

Мы спускаемся на предыдущий уровень рекурсии, где idx = 0, $i = 1$. Освобождаем число 1:

used[i] = **false**;

Остановимся на этом. На следующей итерации цикла в ячейку a[0] будет помещено число 2. Затем мы перейдём на следующей уровень рекурсии и поместим в массив a число 1. После этого выведем последовательность «2 1». Собственно, существует только две перестановки из двух элементов: «1 2» и «2 1», мы их обе вывели.

Таким образом, мы немного модифицировали решение предыдущей задачи о переборе последовательностей с повторяющимися числами: добавили массив used, и научились генерировать перестановки.