3. Generarea combinărilor

Se citesc două numere naturale n şi c, cu $1 \leq c \leq n$. Se cere să se tipărească toate combinările de n elemente luate cate c, în ordine lexicografică. **De exemplu**, pentru n=3 şi c=2 avem combinările : 1 2, 1 3, 2 3 .

Definiţie: Fie **A** o mulţime cu **n** elemente. Combinările de **n** elemente luate câte **c** sunt toate submulţimile ale lui **A** care au fiecare câte **c** elemente.

Reprezentarea soluţiei: - fiecare componentă are valori în mulţimea {1,2,….,n}
-soluţia are **c** componente

Condiţia de validare: - componentele unei soluţii sunt diferite (condiţia de la permutări)
- componentele unei soluţii le scriem în ordine crescătoare

Din cele două condiţii rămâne doar condiţia ca soluţia să aibă componentele în ordine crescătoare. Pentru soluţia parţială $(x_1,x_2,...,x_{k-1})$ avem $x_1 < x_2 <...< x_{k-1}$. Pentru a trece la soluţia următoare $(x_1,x_2,...,x_{k-1},x_k)$ mai trebuie să verificăm doar condiţia: $x_{k-1} < x_k$.

Numărul combinărilor: $C_n^c = \frac{n!}{n!\cdot(n-c)!}$

Varianta iterativă

```
#include<iostream>
using namespace std;
int x[30];
int n,c,nr;

void tipar()
{
    int i;
    for(i=1;i<=c;i++)
     cout<<x[i]<<' ';
    cout<<'\n';
    nr++;
}

int valid(int k)
{
    int i;
    if(x[k]>x[k-1])return 1;
    return 0;
}

void back()
{
    int i,k=1;
    x[1]=0;
    while(k>0)//stiva nu este vida
    {
    while(x[k]<n)
    {
        x[k]=x[k]+1;
        if(valid(k))
            if(k==c) tipar();
            else{k=k+1;x[k]=0;}
    }
    k=k-1;
    }
}

int main()
{   x[0]=0; //valoarea inițială
    cin>>n;
    cin>>c;
    back();
    cout<<nr;
}
```

Varianta recursivă

```
#include<iostream>
using namespace std;
int x[30];
int n,c,nr;

void tipar()
{
    int i;
    for(i=1;i<=c;i++)
     cout<<x[i]<<' ';
    cout<<'\n';
    nr++;
}

int valid(int k)
{
    int i;
    if(x[k]>x[k-1])return 1;
    return 0;
}

void back(int k)
{
    int i;
    if(k==c+1)tipar();
    else
        for(i=1;i<=n;i++)
        {
            x[k]=i;
            if(valid(k)) back(k+1);
        }
}

int main()
{   x[0]=0; //valoarea inițială
    cin>>n;
    cin>>c;
    back(1);
    cout<<nr;
}
```