

Rekurzija

kombinacija n števil od 0 ... k-1

n = 3, k = 2

0	0	0
0	0	1
0	1	0
.	.	.
1	1	1

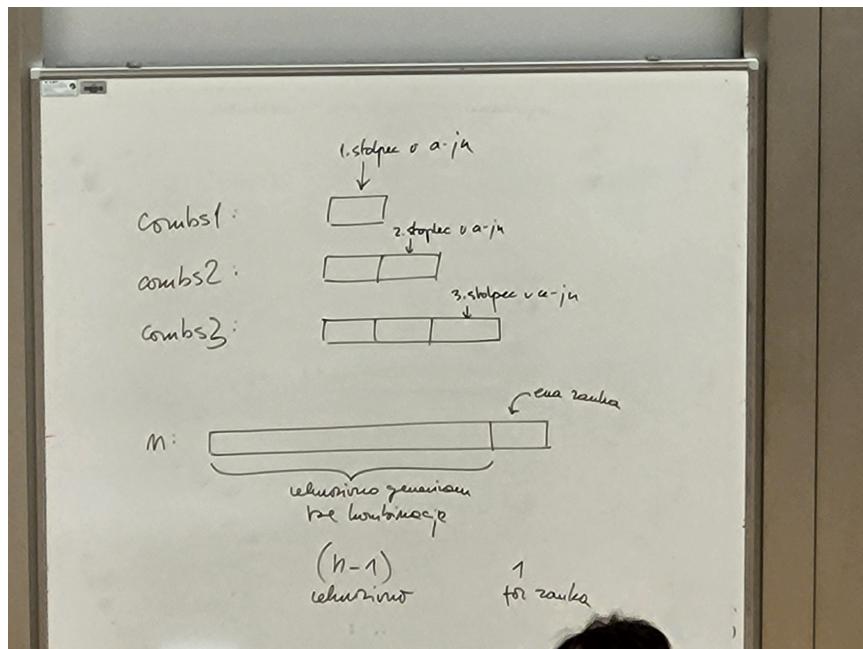
k^n vrstic

```
// n = 1
void combs1(int *a, int k){
    for(int j = 0; j < k; j++){
        a[0] = j;
        //print(a, 1);
        combs0(a, k);    // samo izpis a-ja (kombinacije)
    }
}

// n = 2
void combs2(int *a, int k){
    for(int j = 0; j < k; j++){
        a[1] = j; combs1(a, k);
    }
}

// n = 3
void combs3(int *a, int k){
    for(int j = 0; j < k; j++){
        a[2] = j; combs2(a, k);
    }
}

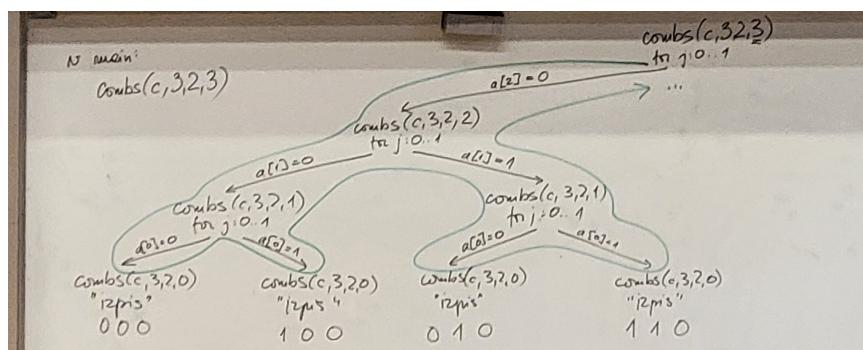
// n = 4
void combs4(int *a, int k){
    for(int j = 0; j < k; j++){
        a[3] = j; combs3(a, k);
    }
}
```



```

void combs(int *a, int n, int k, int f){
    if(f == 0){ // ustavitevni pogoj:
        for(int i = 0; i < n; i++){
            printf("%2d ", a[i]);
        }
        printf("\n");
        return;
    }
    // korak rekurzije:
    for(int j = 0; j < k; j++){
        a[f - 1] = j;
        combs(a, n, k, f-1);
    }
    return;
}
...
int c[100];
combs(c, 7, 3, 7); // 7 stolpcev, od 0 do 2, opravi 7 zank (ko je f = 0 samo izpis)

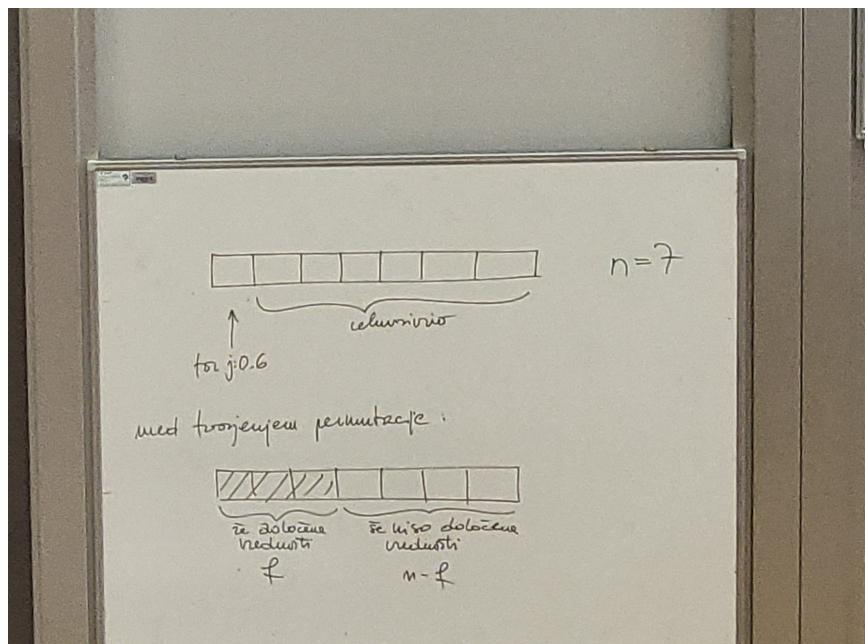
```



Permutacije n števil od 0 do n-1

```
// klic: perms(c, 7, 0)
void perms1(int *a, int n; int f){
    if(f == n){
        // izpiši a
        return;
    }

    for(int j = 0; j < n; j++){
        bool valid = true;
        for(int i = 0; i < f; i++){
            valid = valid && (a[i] != j);
        }
        if(valid){
            a[f] = j;
            perms(a, n, f+1);
        }
    }
}
```

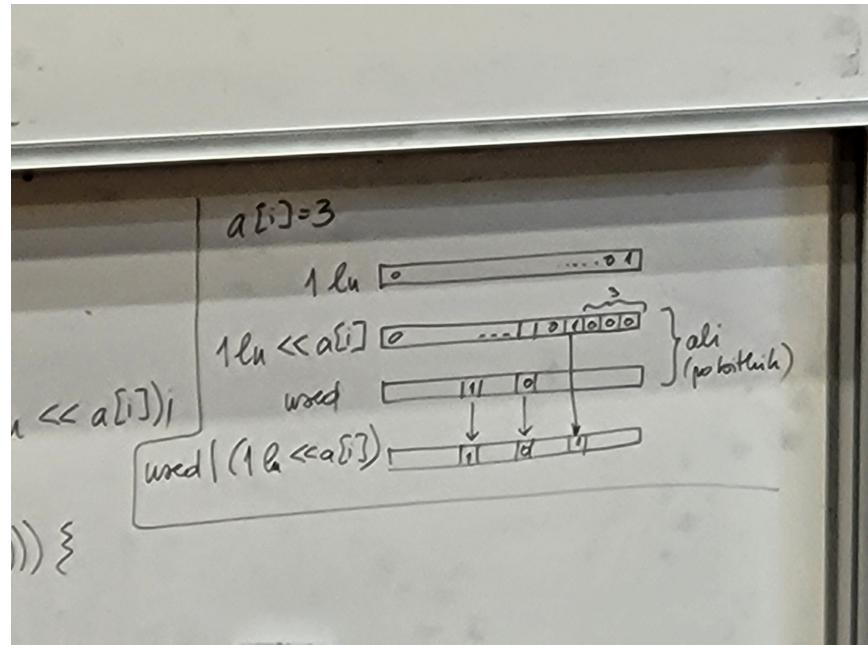


```
void perms2(int *a, int n, int f){
    if(f == n){
        // izpiši a
        return;
    }

    long unsigned used = 0lu;
    for(int i = 0; i < f; i++){
        used = used | (1lu << a[i]);
    }
    for(int j = 0; j < f; j++){
        if(!(used & (1lu << j))){
            a[f] = j;
            perms2(a, n, f+1);
        }
    }
}
```

```

        }
    }
}
```



```

// tabelo globalno ali kot dodatni parameter
// klic: perms3(c, 7, 0, 0lu)
void perms3(int *a, int n, int f, long unsigned used){
    if(f == n){
        // izpiši a
        return;
    }

    for(int j = 0; j < n; j++){
        if(!(used & (1lu << j))){
            a[f] = j;
            perms3(a, n, f+1, used | (1lu << j));
        }
    }
}

//long unsigned used = 0lu;
void perms4(int *a, int n, int f){
    if(f == n){
        // izpiši a
        return;
    }

    static long unsigned used = 0lu;
    for(int j = 0; j < n; j++){
        if(!(used & (1lu << j))){
            a[f] = j;
            used = used | (1lu << j);
            perms4(a, n, f+1);
        }
    }
}
```

```
        used = used & (~(11u << j)); // ugasnemo bit
    }
}

// iterativno...
```

```
int c[100];
// c <-- {0, 1, 2, 3, 4, 5}      n = 6
// zamenjam 0 in 0
// 0, permutiramo: 1, 2, 3, 4, 5
// vrnem 0 na pravo mesto
// zamenjam 0 in 1
// 1, permutiramo: 0, 2, 3, 4, 5
// vrnem 1 na pravo mesto
// zamenjam 0 in 2
// 2, permutiramo: 1, 0, 3, 4, 5
// vrnem 2 na pravo mesto
// ...
```

// vrne število izpisanih vrstic...