Nella maggioranza dei problemi che si affrontano con i calcolatori l'esponenzialità è legata all'impiego, a volte mascherato, di due strutture combinatorie di base: l'insieme delle 2^n configurazioni di n bit, e l'insieme delle $n!$ permutazioni di n elementi. Vediamo come tali strutture possano essere costruite algoritmicamente e quindi utilizzate. Dato un vettore A di lunghezza n, il seguente algoritmo `Configurazioni` costruisce in A le 2^n possibili configurazioni binarie (disposizioni con ripetizione dei due elementi 0 e 1 in gruppi di n). Per ogni configurazione l'algoritmo stabilisce se essa possiede determinate proprietà mediante una procedura `Controllo` che specificheremo nel seguito trattando di problemi concreti. Il calcolo è avviato con la chiamata iniziale `Configurazioni(`$A, 1$`)` che innesca la costruzione di tutte le configurazioni binarie dalla $00\ldots0$ alla $11\ldots1$, facendo variare le cifre a partire dall'ultima.

`Procedure Configurazioni(`A, k`):`
    for $i \leftarrow 0$ **to** 1 **do**
        $A[k] \leftarrow i$;
        if $k = n$ **then** `Controllo(`A`)`
                else `Configurazioni(`$A, k+1$`)`.

Un altro problema combinatorio di base è quello della costruzione di permutazioni. Dato un insieme di n elementi contenuti in un vettore P, l'algoritmo seguente costruisce in P tutte le $n!$ permutazioni di tali elementi. Come nel caso precedente, per ogni permutazione l'algoritmo stabilisce se essa possiede determinate proprietà mediante una procedura `Controllo`.

`Procedure Permutazioni(`P, k`):`
    if $k = n$ **then** `Controllo(`P`)`
            else for $i \leftarrow k$ **to** n **do**
                scambia $P[k] \leftrightarrow P[i]$;
                `Permutazioni(`$P, k+1$`)`;
                scambia $P[k] \leftrightarrow P[i]$.

Il calcolo è avviato con la chiamata iniziale `Permutazioni(`$P, 1$`)`. L'algoritmo è basato sull'osservazione che le permutazioni di n elementi possono essere divise in gruppi, ponendo in ciascun gruppo quelle che iniziano con il primo, il secondo,$\ldots$, l'n-esimo elemento ciascuno seguito dalle permutazioni degli altri $n-1$. Nel primo gruppo troviamo $P[1]$ seguito da tutte le permutazioni di $P[2], \ldots, P[n]$; nel secondo troviamo $P[2]$ seguito da tutte le permutazioni di $P[1], P[3], \ldots, P[n]$ e così via. Questa definizione è induttiva e la correttezza dell'algoritmo si può quindi dimostrare per induzione: occorre solo notare che la seconda operazione di scambio tra $P[k]$ e $P[i]$ è necessaria per ripristinare l'ordine degli elementi dopo la costruzione ricorsiva delle permutazioni degli elementi che seguono il primo. Più difficile è capire il funzionamento dell'algoritmo e individuare l'ordine in cui vengono costruite le permutazioni. Invitiamo il lettore a rifletterci un momento simulando a mano il comportamento dell'algoritmo sull'insieme $\{1, 2, 3\}$: otterrà nell'ordine le permutazioni: 1,2,3 - 1,3,2 - 2,1,3 - 2,3,1 - 3,2,1 - 3,1,2, e con l'ultima operazione di scambio ripristinerà l'ordine iniziale 1,2,3 degli elementi.