Árvores múltiplas top-down

Desenvolva um programa que implemente uma árvore Top-Down

- Implementar a função insere(int chave)
- Implementar a função busca(int chave)
- Implementar um percurso em árvore top-down
- Inserir 5000 elementos em uma árvore top-down
- Contar o número de nós de uma árvore múltipla

Códigos auxiliares

```
const int ORDEM = 200;            // Número máximo de filhos
const int MAX_CHAVES = ORDEM-1;   // Número máximo de chaves
//const int MIN_OCUP = (ORDEM-1)/2;   // Ocupaçao máxima de um nó p/árvore B
const int TOTAL = 10000;

struct no {
   int dado[ORDEM-1];
   struct no *filhos[ORDEM];
   struct no *pai;
   int n_chaves;
};
struct no *raiz=NULL;
// Encontra uma chave no nó apontado por atual e retorna a posição dela no nó
// Se a chave nó existir, retorna a posicao onde a chave deveria ser inserida.
int encontra_chave(struct no *atual, int chave) {
   int i=0;
   while (i<atual->n_chaves && chave>atual->dado[i])
      i++;
            return i;
}
// Encontra o nó folha onde uma chave deve ser inserida e retorna um ponteiro para ele
```

```
// A posicao que a chave terá no no folha retornada por referencia na variavel posicao
struct no *encontra_no(int chave, int &posicao) {
    struct no *atual, *anterior;
    anterior=NULL;
    atual=raiz;
    // Percorre a arvore com dois ponteiros, ate achar uma subarvore nula.
    // Quando atual for nulo, retorna o valor do ponteiro anterior que estara
    // no nivel anterior.
    while (atual!=NULL) {
        posicao=encontra_chave(atual,chave);
        anterior = atual;
        atual = atual->filhos[posicao];
    }
    return anterior;
}
// Insere uma nova chave em um no folha que possui espaco
void insere_folha(struct no *atual, int chave) {
    int i;
    for (i=atual->n_chaves ; i>0 && chave<atual->dado[i-1] ; i--)
        atual->dado[i]=atual->dado[i-1];
    atual->dado[i] = chave;
    atual->n_chaves++;
}
// Insere uma nova chave em um arvore multipla top-down
int insere(int chave) {
    struct no *novo,*atual;
    int posicao;
    if (raiz==NULL) {
        raiz=cria_no(chave);
        return 1;
    }
    // Encontra no folha onde a nova chave será inserida
    atual = encontra_no(chave,posicao);
```

```
    // Existe espaco no no folha, insere nova chave
    if (atual->n_chaves < MAX_CHAVES) {
        insere_folha(atual,chave);
        return 1;
    }
    // Nao existe espaco no no folha, cria novo no
    novo=cria_no(chave);
    atual->filhos[posicao]=novo;
    return 1;
}
int conta(struct no *atual) {
    int i, total;
    if (atual!=NULL) {
        total=1;
        for (i=0; i<atual->n_chaves+1; i++) {
            total = total + conta(atual->filhos[i]);
        };
        return total;
    };
    return 0;
}
```