user: sakura

第23题——从上到下打印二叉树 I

从上到下打印出二叉树的每个节点，同一层的节点按照从左到右的顺序打印。

例如:

> 例如:
> 给定二叉树: [3, 9, 20, null, null, 15, 7],
>
> ```
>   3
>  / \
> 9  20
>   /  \
>  15   7
> ```
>
> 返回:
>
> ```
> [3,9,20,15,7]
> ```

提示：节点总数 <= 1000

数组模拟队列

使用一个数组来模拟队列从而达到广度优先遍历的目的。

- 判空
- 根节点入队
- 进入while循环：条件是front<rear

```
1.队头元素出队

2.将节点的值存入数组arr

3.判断：如果有左孩子，则左孩子入队

4.判断：如果有右孩子，则右孩子入队
```

eg:

```
3入队。队列：3。

3出队的同时，有左孩子，则左孩子入队，有右孩子，右孩子入队。队列：9，20。
```

9出队。无左右孩子。队列：20。

20出队。有左孩子，入队，有右孩子，入队。队列：15,7。

15出队。无孩子。队列：7。

7出队。无孩子。队列为空。

得到数组　arr [3,9,20,15,7]

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* levelOrder(struct TreeNode* root, int* returnSize){
    struct TreeNode* Queue[1010];
    int front =-1,rear=-1;
    int i=0;//用来记录数组长度
    struct TreeNode *p;//工作指针
    int* arr=(int *)malloc(sizeof(int)*1010);

    if(root==NULL)
    {//如果二叉树为空，则直接返回
        (*returnSize)=0;
        return root;
    }
    Queue[++rear]=root;//根节点先入队
    while(front<rear)
    {
        p=Queue[++front];//队头元素出队
        arr[i++]=p->val;
        if(p->left)//如果当前节点有左孩子，那就入队
            {Queue[++rear]=p->left;}
        if(p->right)//如果当前节点有右孩子，那就入队
            {Queue[++rear]=p->right;}
    }
    *returnSize=i;//将数组长度返回给size
return arr;
}
```