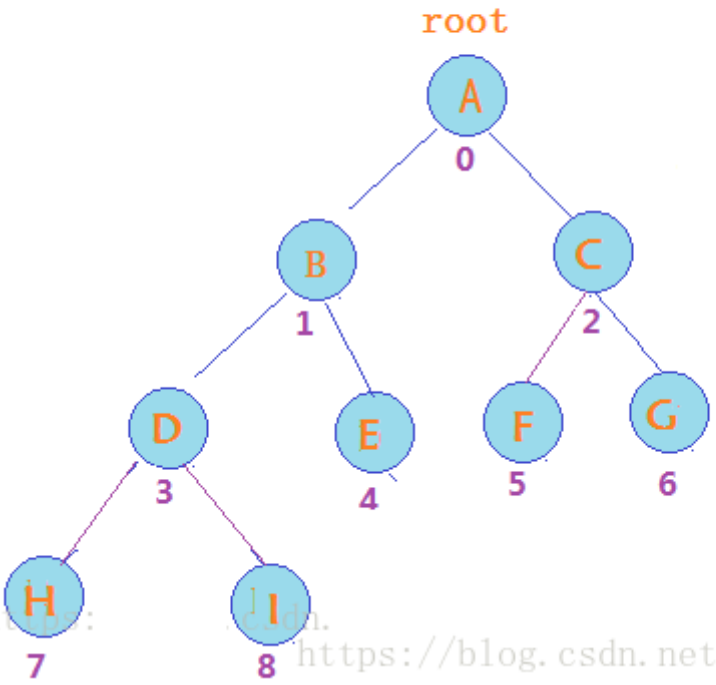
完全二叉树的三种遍历（二）--- 顺序存储

在上篇文章中已经贴了二叉树三种遍历的递归及非递归的实现代码，那这篇文章是干什么的呢？嗯，虽然不论是二叉树的顺序存储还是二叉树的链式存储，三种遍历的思路都是一样的，但是在写代码的时候仍然还要注意边界，所以决定也贴出来吧，好查找呗^O^

来，继续上篇文章的树，二叉树的顺序存储就是在数组中存储。在代码上看起来就是一个数组，我们之所以说它是二叉树，意味着，在这篇文章中，逻辑意义上我们将它看成一颗树！！你会注意到上篇文章的树不是一个完全二叉树，所以在数组中存储起来很浪费空间，所以本篇说的都是完全二叉树的顺序存储哦！

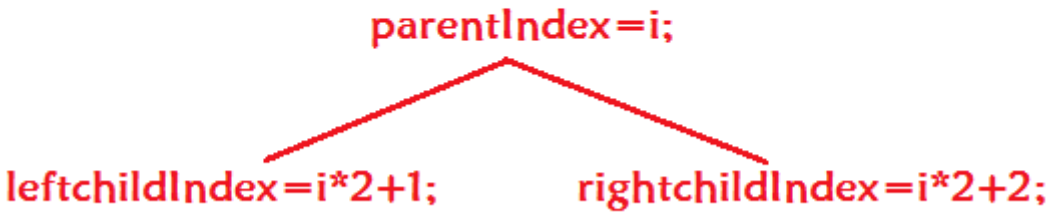
如下：有一个完全二叉树：



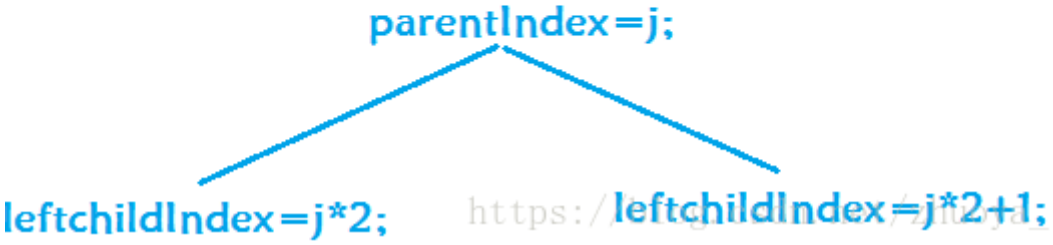
一颗树，我们将它按层从上往下，每层从左往右依次存到数组中去。当然你也可以从数组的1号下标开始存储喔~，我就直接从0号下标开始存储咯！

根结点与左右孩子索引的关系如下！！！

从数组的0号下标开始存储时
逻辑意义上根结点和左、
右孩子的索引关系：



从数组的1号下标开始存储时
逻辑意义上根结点和左、
右孩子的索引关系：




```
1. /*****************************************************************/
2. //先序遍历的递归
3. void PreTravel1(char *arr,int parentindex,int len)
4. {
5.     if(parentindex<=len-1)
6.     {
7.             cout<<arr[parentindex]<<" ";
8.             PreTravel1(arr,parentindex*2+1,len);
9.             PreTravel1(arr,parentindex*2+2,len);
10.     }
11. }
12. 
13. //先序遍历的非递归
14. void PreTravel2(char* arr,int parentindex,int len)
15. {
16.     //1
```

```
17.  //stack<char> st;
18.  
19.  //while(1)
20.  //{
21.  //    if(st.empty() && parentindex>len-1)
22.  //            break;
23.  //    if(parentindex<=len-1 )
24.  //    {
25.  //            st.push(parentindex);
26.  //            cout<<arr[parentindex]<<" ";
27.  //            parentindex=parentindex*2+1;
28.  //    }
29.  //    else
30.  //    {
31.  //            parentindex=st.top();
32.  //            parentindex=parentindex*2+2;
33.  //            st.pop();
34.  //    }
35.  //}
36.  //1使用队列
37.  //deque<char> que;
38.  //que.push_back(parentindex);
39.  //while(!que.empty())
40.  //{
41.  //    parentindex=que.front();que.pop_front();
42.  //    cout<<arr[parentindex]<<" ";
43.  //    if(parentindex*2+1<=len)
44.  //            que.push_back(parentindex*2+2);
45.  //    if(parentindex*2+2<=len)
46.  //            que.push_back(parentindex*2+1);
47.  //}
48.  
49.  //2使用栈
50.   stack<char> st;
51.   while(!st.empty() || parentindex<=len-1)
```

```
52.  {
53.          while(parentindex<=len-1)
54.          {
55.                  st.push(parentindex);
56.                  cout<<arr[parentindex]<<" ";
57.                  parentindex=parentindex*2+1;
58.          }
59.          parentindex=st.top();
60.          parentindex=parentindex*2+2;
61.          st.pop();
62.   }
63. }
64. void Pre(char *arr,int len)
65. {
66.   if(arr==NULL || len<=0)
67.          return ;
68.   PreTravel1(arr,0,len);
69.   //PreTravel2(arr,0,len);
70. }
71. /*****************************************************************/
72. //中序遍历的递归实现
73. {
74.   if(parentindex<=len-1)
75.   {
76.          InTravel1(arr,parentindex*2+1,len);
77.          cout<<arr[parentindex]<<" ";
78.          InTravel1(arr,parentindex*2+2,len);
79.   }
80. }
81. //中序遍历的非递归实现
82. void InTravel2(char* arr,int parentindex,int len)
83. {
84.   stack<char> st;
85.   while(!st.empty() || parentindex<=len-1)
86.   {
```

```
87.         while(parentindex<=len-1)
88.         {
89.                 st.push(parentindex);
90.                 parentindex=parentindex*2+1;
91.         }
92.         parentindex=st.top();
93.         cout<<arr[parentindex]<<" ";
94.         st.pop();
95.   }
96. }
97. void In(char *arr,int len)
98. {
99.         return ;
100.        InTravel1(arr,0,len);
101.        //InTravel2(arr,0,len);
102. }
103. /*****************************************************************/
104. //后序遍历的递归实现
105. void PostTravel1(char *arr,int parentindex,int len)
106. {
107.        if(parentindex<=len-1)
108.        {
109.                PostTravel1(arr,parentindex*2+1,len);
110.                PostTravel1(arr,parentindex*2+2,len);
111.                cout<<arr[parentindex]<<" ";
112.        }
113. }
114. //后序遍历的非递归实现
115. void PostTravel2(char* arr,int parentindex,int len)
116. {
117.        stack<char> st;
118.        int tag=-1;
119.        while(!st.empty() || parentindex<=len-1)
120.        {
121.                while(parentindex<=len-1)
```

```
122.             {
123.                 st.push(parentindex);
124.                 parentindex=parentindex*2+1;
125.             }
126.             parentindex=st.top();st.pop();
127.             if(parentindex*2+2==tag || parentindex*2+2>len-1)
128.             {
129.                 cout<<arr[parentindex]<<" ";
130.                 tag=parentindex;
131.                 parentindex=len;
132.             }
133.             //parentindex=parentindex*2+2;
134.             else
135.             {
136.                 st.push(parentindex);
137.                 parentindex=parentindex*2+2;
138.             }
139.         }
140. }
141. 
142. void Post(char *arr,int len)
143. {
144.     if(arr==NULL && len<=0)
145.         return ;
146.     PostTravel1(arr,0,len);
147.     //PostTravel2(arr,0,len);
148. }
149. /*****************************************************************/
150. int main()
151. {
152.     char arr[]={'A','B','C','D','E','F','G','H'};
153.     int len=sizeof(arr)/sizeof(arr[0]);
154. 
155.     cout<<"PreTravel: ";
156.     Pre(arr,len);
```

```
157.         cout<<endl;
158.
159.         cout<<"InTravel: ";
160.         In(arr,len);
161.         cout<<endl;
162.
163.         cout<<"PostTravel: ";
164.         Post(arr,len);
165.         cout<<endl;
166.         return 0;
167. }
```

```
PreTravel: A B D H E C F G
InTravel: H D B E A F C G
PostTravel: H D E B F G C A
请按任意键继续. . .
```