基础作业

1 实现链表的查找、修改，画图演示链表的删除过程

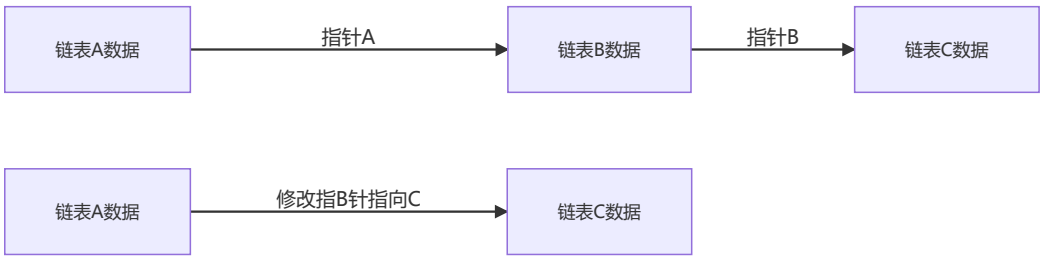
- 链表的查找

```
void findList(pStudent_t phead,int num)
{
    if(phead==NULL)
    {
        cout<<"the list is empty"<<endl;
    }
    else
    {
        while(phead)
        if(phead->num=val){
            cout<<"find the val"<<endl;
            break;
        }
        else{
            phead=phead->next;
        }
        if(phead==NULL)
        {
            cout<<"don't find val"<<endl;
        }
    }
}
```

- 链表的修改

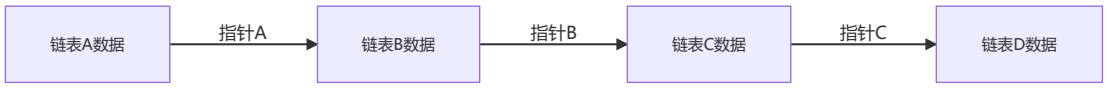
```
void modifyList(pStudent_t phead,int val,float score)
{
    if(phead==NULL)
    {
        cout<<"the list is empty"<<endl;
    }
    else
    {
        while(phead)
        if(phead->num=val){
            cout<<"find the val"<<endl;
            phear->data=score;
            break;
        }
        else{
            phead=phead->next;
        }
        if(phead==NULL)
        {
            cout<<"don't find val"<<endl;
        }
    }
}
```

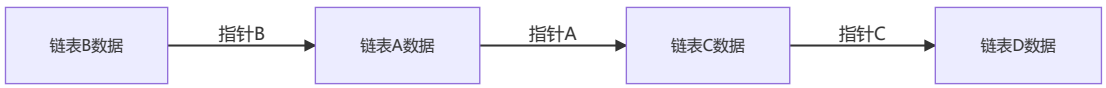
- 链表的删除



  - 在链表删除的过程中有一下集中特殊情况
    - 删除的恰好是第一个数据：则须重新定义头指针指向被删除的next
    - ```
      (*pphead)=pCur->next;
      ```
    - 删除的恰好是尾指针：则须重新定义尾指针指向被删除的前一个数据的数据段。
    - ```
      *pptail=pPre;
      ```
    - 以上还要注意删除完还需判断是否需要修改尾指针。（删除之后仅剩一个元素的情况下，还要修改尾指针的指向）

2 画图说明链表的逆置过程



- 

- 修改头指针，让头指针指向`B->data`
- 修改B的指针，让他指向`A->data`
- 修改A的指针，让它指向`C->data`
- 不断重复，直到`A->next=ptail`即A的next指针指向尾指针

3 栈的特点是什么？需要设计哪些接口？如何实现链式栈和顺序栈？

- 栈的特点：先进后出||后进先出
- 需要设计哪些接口：
  - 初始化栈/弹栈/压栈/返回栈顶元素/查找栈中元素/返回栈长度
- 如何实现链式栈和顺序栈？
  - 链式栈：用链表实现
  - 顺序栈：用数组实现

4 队列的的特点是什么？需要设计哪些接口？如何实现链式队列？

- 队列的的特点：先进先出（FIFO）
- 需要设计哪些接口：
  - 初始化/进队/出队
- 如何实现链式队列
  - 双向链表实现

5 循环队列的首下标、尾下标是多少？容量是多少？实现循环队列的入队和出队

- 设大小为N的循环队列
  - 循环队列的首下标：0
  - 尾下标为n-1
  - 容量为n-1
- 循环队列的入队
  - 队满条件`(queue->rear + 1) % Maxsize == queue->front`即尾指针+1等于头指针
  - ```
    void enQueue(SqQueue_t* queue, ElemType x)
    {
        if ((queue->rear + 1) % Maxsize == queue->front)
        { //队满则无法进队
            cout << "queue is full" << endl;
            return;
        }
        queue->dada[queue->rear] = x;
        queue->rear = (queue->rear + 1) % Maxsize;
        //移动尾指针
    }
    ```
- 实现循环队列的出队
  - 队空条件`queue->front = queue->rear`即头指针指向尾指针
  - ```
    void deQueue(SqQueue_t* queue, ElemType*x)
    {
        if (queue->front = queue->rear) //队空则无法出队
        {
            cout << "queue is empty" << endl;
            return;
        }
        *x = queue->dada[queue->front] ;
        queue->front = (queue->front + 1) % Maxsize;
        //移动头指针

    }
    ```

7 二叉树的特点是什么？如何

- 度最多为2的数
- 声明一个二叉树结点
  - ```
    void bulidBinaryTree(pNode_t* treeroot, pQueue_t*, pQueue_t*,
    ElemType)
    {
        pNode_t treeNew = (pNode_t)calloc(1, sizeof(Node_t));
    ```
    ```

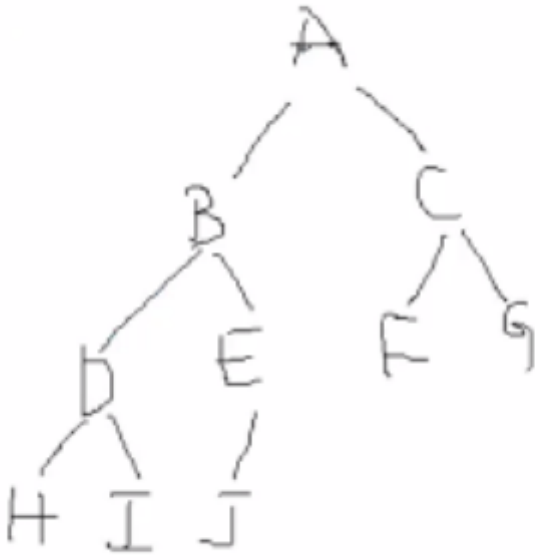
```
    pQueue_t queNew = (pQueue_t)calloc(1, sizeof(Queue_t));
    pQueue_t queCur = *qHead;
    treeNew->c = val;
    queNew->insertPos = treeNew;
    if (NULL == *treeroot) //如果根节点为空
    {
        *treeroot = treeNew;
        *qHead = queNew;
        *qTail = queNew;
    }
    else
    {
        (*qTail)->pNext = queNew;
        *qTail = queNew;
        if (NULL==(*qHead)->insertPos->pleft)
        {//非空则优先填左孩子
            (*qHead)->insertPos->pleft = treeNew;
        }
        else if (NULL==(*qHead)->insertPos->pright)
        {//若左孩子满则填右孩子
            queCur->insertPos->pright = treeNew;
            *qHead = queCur->pNext;
            free(queCur);
            queCur = NULL;
        }
    }
}
```

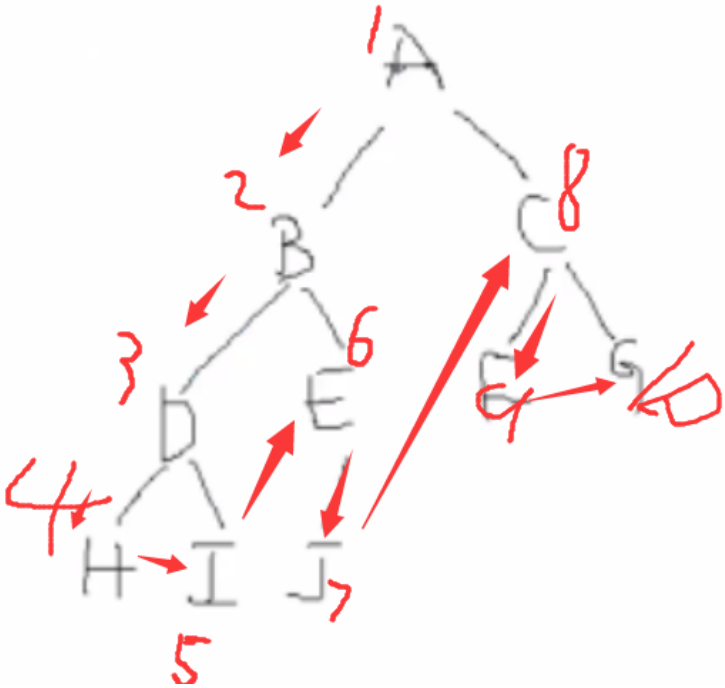
8 画图说明什么是二叉树的层次遍历?

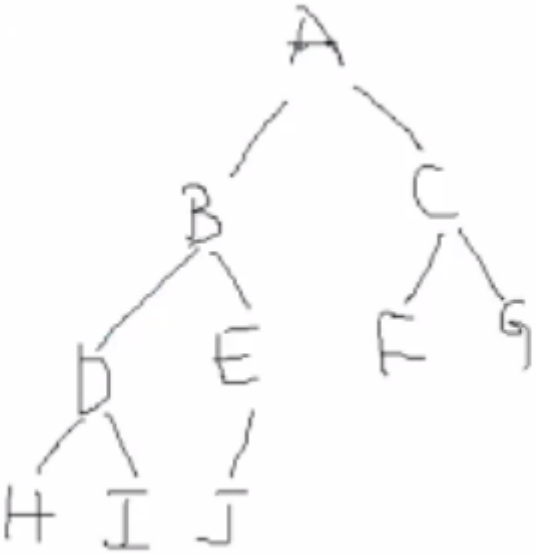
- 每层按照顺序填入队列中，然后依次输出队列

9 画图说明什么是二叉树的先序、中序和后序遍历

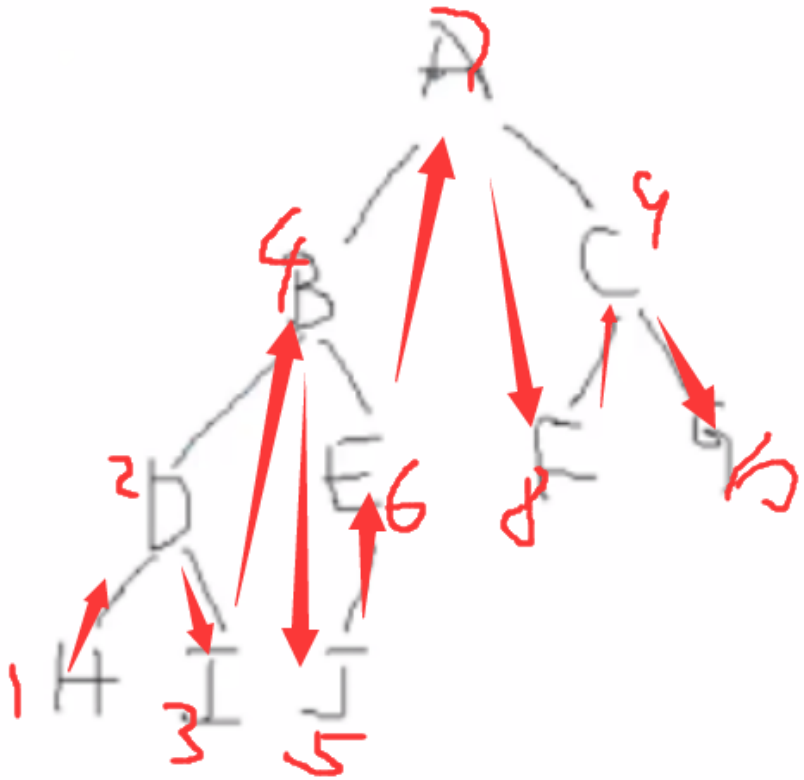
- 先序遍历

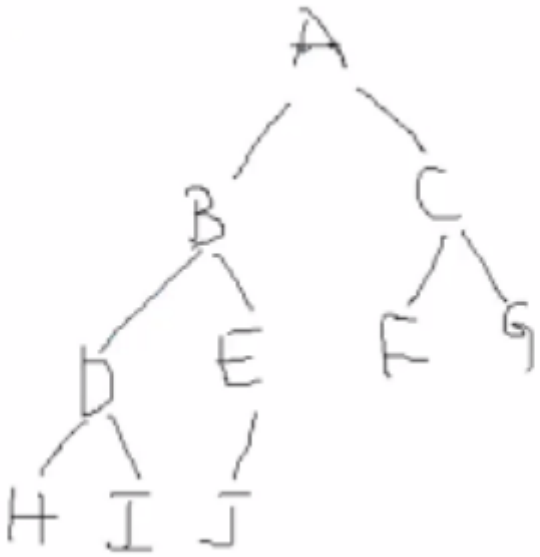


- 先序遍历结果为：ABDHIE JCFG 流程图如下
  - 

- 中序遍历
  - 

  - 中序编列结果为 HDIBJEAFCG ~~将上图一脚踩扁 按照顺序输出~~
  - 流程图如下

- 

- 后序遍历
  - 

    - 后序遍历结果为 HIDJEBFGCA 流程图如下

- 
A
10
B
6
C
9
D
2
E
5
F
7
G
8
H
1
I
3
J
4