测试作业

1 删除指定的某个结点。

- 头文件

```
#include<iostream>
#include<cstdio>
#include<string>
#include<cmath>
using namespace std;
typedef struct myList
{
    int num;
    char name[20];
    float cham;
    struct myList* pNext;
}List_t,*pList_t;
void ListDelete(pList_t* ppHead, List_t** ppTail, int deleteNum);
void print(pList_t p);
pList_t currentList(pList_t p);
void getLast4th(pList_t head);、

```

```

int main()
{
    List_t sArr[5] = { { 1001,"lilei",98.8 },{1002,"hanmeimei",99.5},{1007,"leelee",77},{1024,"otto",59}
        ,{7777,"clearlove",43.96} };
    pList_t p[5];
    pList_t phead = p[0];
    pList_t ptail = p[4];
    for (int i = 0; i < 5; i++)
    {
        p[i] = &sArr[i];
    }
    int deleteNum;
    cin >> deleteNum;
    print(phead);
    ListDelete( &phead, &ptail, deleteNum);
    cout << "--------------------" << endl;
    print(phead);
    return 0;
}
void ListDelete(pList_t* ppHead, List_t** ppTail, int deleteNum)
{
    pList_t pCur = *ppHead;
    pList_t pPre = *ppHead;
    if (pCur == NULL)
    {
        cout << "真的一滴都没有了" << endl;
        return;
    }
    else if(deleteNum==pCur->num)
    {//删除的位置恰好为头节点
        *ppHead = pCur->pNext;
        if (NULL==*ppHead) //如果恰好只有一个元素则清空链表的值
        {
            *ppTail == NULL;
        }
    }
    else
    {
        while (pCur)
        {
            if (deleteNum == pCur->num)
            {
                pPre->pNext = pCur->pNext;
                cout << "成功删除" << deleteNum << endl;
                break;
            }
            pPre = pCur;//后指针等于前指针
            pCur = pCur->pNext;//前指针往前走一格
        }
        if (NULL == pCur)
        {
            cout << "这个" << deleteNum << "真没有" << endl;
            return;
        }
        if (pCur == *ppTail)
        {//如果删除的是未节点
            *ppTail = pPre;
        }
    }
    free(pCur);
    pCur = NULL;
}
void print(pList_t p)
{
    pList_t pCur = p;
    while (pCur)
    {
        printf("%3d %s %3.2d", pCur->num, pCur->name, pCur->cham);
        pCur = pCur->pNext;
    }
}
```

2 将两个有序链表合并成一个有序链表。

-

3 将一个链表逆置。如：1->2 ->3 ->4 ->5 ->NULL，输出: 5 -> 4 -> 3 ->2 ->1 -> NULL；

```
//我是弟弟我用栈了
//头文件同第一题
```

```
pList_t currentList(pList_t phead)
{
    pList_t newList;
    stack<pList_t> stack1;
    while (phead)
    {
        stack1.push(phead);
        phead = phead->pNext;
    }
    newList = phead;
    while (!stack1.empty)
    {
        phead->pNext = stack1.pop;
        phead = phead->pNext;
        stack1.pop;
    }
    phead->pNext = NULL;
    return newList;
}
```

4 找出链表的倒数第四个节点

```
//头文件同第一题
void getLast4th(pList_t head)
{
    pList_t fast = head;
    pList_t low = head;
    int step = 4;
    while (step)
    {
        fast = fast->pNext;
        step--;
    }
    if (fast == NULL)
    {
        cout << "链表长度小于四还求啥倒数第四啊" << endl;
    }
    else
    {
        while (fast)
        {   //齐头并进
            fast = fast->pNext;
            low = low->pNext;
        }
    }
    cout << "想不到吧" << low->num << "就是倒数第四个" << endl;
}
```

5 找出链表的中间节点

6 判断单链表是否有环

-
  ```
  bool getRing(pList_t head)
  {
      pList_t fast = head;
      pList_t low = head;
      int step = 2;
      while (step)
      {
          fast = fast->pNext;
          step--;
      }
      if (fast == NULL)
      {
          cout << "两个以下不会有环，别找了" << endl;
      }
      else
      {
          while (low) //如果没环则比对到慢指针的时候跳出循环
          {
              if (low == fast) return 1;
              else
              {
                  fast = fast->pNext;
                  low = low->pNext;
              }
          }
          return 0;
      }
  }
  ```

7 判断两个链表是否相交，如果相交，计算交点

8 实现链式栈

-
  ```

  ```

-

9 实现循环队列

-
  ```
  #include<cstdio>
  #include<cstring>
  #include <iostream>
  #define Maxsize 5
  using namespace std;
  typedef int ElemType;
  typedef struct
  {
      ElemType dada[Maxsize];
      int front, rear;
  }SqQueue_t;
  void initQueue(SqQueue_t*);
  int main()
  {
      ElemType e;
      SqQueue_t Q;
      initQueue(&Q);
  ```

```c
}
void initQueue(SqQueue_t*queue)
{
    queue->front = queue->rear = 0;
}  //初始化队列
```

-

10 实现二叉树层次建树