**#include<stdio.h>**

**#include<stdlib.h>**

**struct node \*create1();//创建链表，表头法**

**struct node\*create2();//创建链表，表尾法**

**struct node\* insert1(struct node \*head,int num,int score);//插入结点，到表头**

**struct node\* insert2(struct node \*head,int num,int score);//插入结点，到表尾**

**struct node\* insert3(struct node \*head,int num,int score);//插入结点到有序链表中**

**struct node\* del(struct node \*head,int num); // 根据学号，删除结点**

**void print(struct node\* head); //输出结点**

**struct node**

**{**

**int num;**

**int score;**

**struct node \*next;**

**};**

**int main()**

**{**

**struct node \*head;**

**int num,score;**

**head=create2(); //调用函数，创建链表**

**printf("输出链表：\n");**

**print(head); //调用函数，输出链表**

**printf("enter num，score:\n");**

**scanf("%d%d",&num,&score);**

**head=insert2(head,num,score); //调用函数，在原有的链表中插入新结点**

**printf("输出新链表：\n");**

**print(head); //输出链表**

**return 0;**

**}**

**struct node \*create1()**

**{**

**struct node \*head=NULL,\*p;**

**int n;**

**printf("enter n: ");**

**scanf("%d",&n);**

**while(n>0)**

**{**

**p=(struct node \*)malloc(sizeof(struct node));**

**scanf("%d%d",&p->num,&p->score);**

**p->next=head;**

**head=p;**

**n--;**

**}**

**return head;}**

**void print(struct node\* head)**

**{**

**struct node \*p;**

**for(p=head;p!=NULL;p=p->next)**

**printf("%d,%d\n",p->num,p->score);**

**}**

**struct node\*create2()**

**{**

**struct node \*p,\*head=NULL,\*tail;**

**int n;**

**printf("enter n: ");**

**scanf("%d",&n);**

**while(n>0)**

**{**

**p=(struct node\*) malloc(sizeof(struct node));**

**scanf("%d%d",&p->num,&p->score);**

**p->next=NULL;**

**if(head==NULL)**

**{**

**head=p;**

**}**

**else**

**tail->next=p;**

**tail=p;**

**n--;**

**}**

**return head;**

**}**

**struct node\* insert1(struct node \*head,int num,int score) //将新结点插入到链表头**

**{**

**struct node \*p;**

**p=(struct node\*)malloc(sizeof(struct node));**

**p->num=num;**

**p->score=score;**

**p->next=head;**

**head=p;**

**return head;**

**}**

**struct node\* insert2(struct node \*head,int num,int score) //将新结点插入到链表尾**

**{**

**struct node \*p,\*q;**

**p=(struct node\*)malloc(sizeof(struct node));**

**p->num=num;**

**p->score=score;**

**p->next=NULL;**

**if(head==NULL)**

**head=p;**

**else**

**{**

**for(q=head;q->next!=NULL;q=q->next) ;**

**q->next=p;**

**}**

**return head;**

**}**

**struct node\* insert3(struct node \*head,int num,int score) //在有序链表中，插入新结点**

**{**

**struct node \*p,\*p1,\*p2;**

**p=(struct node\*)malloc(sizeof(struct node));**

**p->num=num;**

**p->score=score;**

**if(head==NULL) //原来的链表为空，将新结点插入到表头**

**{**

**p->next=head;**

**head=p;**

**return head;**

**}**

**if(num<head->num) //新结点的学号，比头结点的学号小，将新结点插入到表头**

**{**

**p->next=head;**

**head=p;**

**return head;**

**}**

**p1=head;**

**p2=head->next; //从第2个结点开始，按照学号，找到插入位置**

**while(p2!=NULL)**

**{**

**if(num<p2->num) //如果新结点的学号，比P2指向的结点的学号小，新结点插入到P2前**

**{**

**p->next=p2;**

**p1->next=p;**

**return head;**

**}**

**else //否则指针后移一个结点**

**{**

**p1=p2;**

**p2=p2->next;**

**}**

**}**

**/\* 遍历整个链表没找到合适的位置，说明新结点的学号比所有结点都大**

**这时，p2==NULL,p1表示p2的前一个结点，p1指向尾结点 \*/**

**p1->next=p;**

**p->next=NULL;**

**return head;**

**}**

**struct node\* del(struct node \*head,int num) //按照学号，删除链表中的某个结点**

**{**

**struct node \*p1,\*p2,\*p;**

**if(head==NULL)**

**return head;**

**if(head->num==num)**

**{**

**p=head;**

**head=head->next;**

**free(p);**

**return head;**

**}**

**p1=head;**

**p2=head->next;**

**while(p2!=NULL)**

**{**

**if(p2->num==num)**

**{**

**p1->next=p2->next;**

**free(p2);**

**return head;**

**}**

**else**

**{**

**p1=p2;**

**p2=p2->next;**

**}**

**}**

**printf("not found!\n");**

**return head;**

**}**