

Homework10

王世烜 PB20151796

2022/11/23

结构体和链表

1、

用结构体数组实现学生成绩表。

说明：

结构体类型定义为：

```
struct student{
    int stunum; //学号
    char name[20]; //姓名
    float examscore; //考试成绩
    float labscore; //实验成绩
    float totalmark; //总评成绩
};
```

1. 在主函数中定义结构体数组，`struct student stutable[10];` 输入如下十个学生的成绩数据，每个学生信息包括 学号、姓名、考试成绩，实验成绩。同时计算每个学生的总评成绩（= 考试成绩 * 60% + 实验成绩 * 40%）并保存至每个结构体的 totalmark。

输入格式如下：

```
71250 李霞 95 82
69753 李友友 88 86
12254 东方亮 87 88
61256 张男 73 85
30258 孙杰 25 88
11260 柯以乐 82 76
33262 谢涛 91 85
29263 叶林 80 75
22483 陈翔 80 76
71525 王子 71 88
```

[注] 数据输入的方式可以采用输入文件重定向

2. 在主函数中定义一个结构体指针数组，`struct student *parray[10];` 使其每一个指针指向上述结构体数组中的一个元素；按总评成绩从高到低的顺序，对指针数组 parray 进行排序，数组 stutable 保持不变，可避免结构体数组元素之间的交换移动。按总评成绩从高到低的顺序输出排序之后的全部学生成绩。

源码

```

#include <stdio.h>

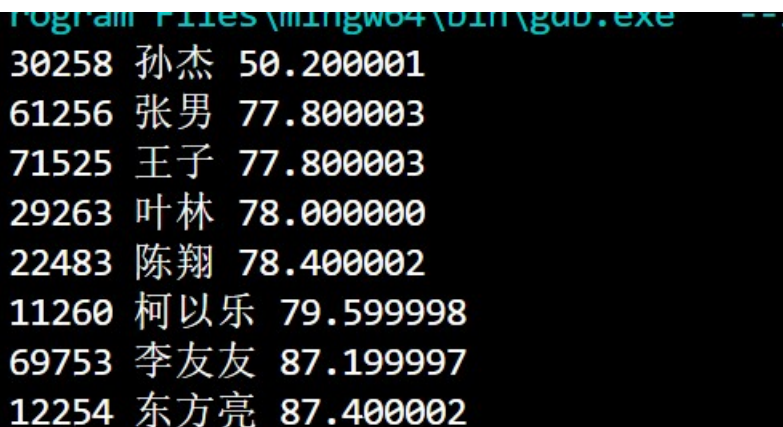
struct student
{
    int stunum;        //学号
    char name[20];     //姓名
    float examscore;   //考试成绩
    float labscore;    //实验成绩
    float totalmark;   //总评成绩
};

void Sort(struct student *parray[], int n) // 插入排序
{
    int i, j;
    struct student *temp;
    for (i = 0; i < n; i++)
    {
        temp = parray[i];
        for (j = i - 1; j >= 0 && parray[j]->totalmark > temp->totalmark; j--)
        {
            parray[j + 1] = parray[j];
        }
        parray[j + 1] = temp;
    }
    return;
}

int main()
{
    struct student student[10];
    struct student *parray[10];
    freopen("score.txt", "r", stdin);
    for (int i = 0; i < 10; i++)
    {
        scanf("%d %s %f %f", &student[i].stunum, student[i].name, &student[i].examscore, &student[i].labscore);
        student[i].totalmark = 0.6 * student[i].examscore + 0.4 * student[i].labscore;
        // printf("%d %s %f %f %f\n", student[i].stunum, student[i].name,
        // student[i].examscore, student[i].labscore, student[i].totalmark);
        parray[i] = student + i;
    }
    Sort(parray, 10);
    for (int i = 0; i < 10; i++)
    {
        // printf("%d %s %f %f %f\n", parray[i]->stunum, parray[i]->name, parray[i]->examscore,
        // parray[i]->labscore, parray[i]->totalmark);
        printf("%d %s %f\n", parray[i]->stunum, parray[i]->name, parray[i]->totalmark);
    }
    return 0;
}

```

运行结果



```

Program Files\mingw64\bin\gdb.exe --
30258 孙杰 50.200001
61256 张男 77.800003
71525 王子 77.800003
29263 叶林 78.000000
22483 陈翔 78.400002
11260 柯以乐 79.599998
69753 李友友 87.199997
12254 东方亮 87.400002

```

```
33262 谢涛 88.599998
71250 李霞 89.800003
PS C:\wsd\vscode>
```

其中，`score.txt` 内容如下：

```
71250 李霞 95 82
69753 李友友 88 86
12254 东方亮 87 88
61256 张男 73 85
30258 孙杰 25 88
11260 柯以乐 82 76
33262 谢涛 91 85
29263 叶林 80 75
22483 陈翔 80 76
71525 王子 71 88
```

实验报告

结构体指针的使用。

2、

用链表实现学生成绩表管理。

接上题，对结构体类型定义增加一个指针成员。

1. 结构体类型定义修改为：

```
struct student{
int stunum; //学号
char name[20]; //姓名
float examscore; //考试成绩
float labscore; //实验成绩
float totalmark; //总评成绩
struct student * next; //下一个结点
};
```

- 编写函数实现建立链表：`struct student * create(int n)`，`n` 是学生人数。函数中输入 `n` 个学生的信息，同时计算总评成绩，按照总评成绩从高到低的方式形成有序链表。返回链表头指针。
- 编写函数 `struct student * delete(struct student * head, int stunum)`，将学号为 `stunum` 的结点删除；返回链表的头指针。
- 编写函数 `struct student * insert(struct student * head)`，插入一个新的结点到链表中，并保持按总评成绩从高到低有序。返回链表的头指针。
- 在主函数中分别调用上述函数，建立链表的 10 个学生数据同第一题。删除结点时的测试数据可以是现有的学号、也可以是不存在的学号—函数应输出提示未找到并返回原有头指针。新增结点时数据为任意与现有结点不同的值。在主函数中输出每次函数调用后的链表内容。

源码

```

#include <stdio.h>
#include <stdlib.h>

struct student
{
    int stunum;           //学号
    char name[20];        //姓名
    float examscore;      //考试成绩
    float labscore;       //实验成绩
    float totalmark;      //总评成绩
    struct student *next; //下一个结点
};

struct student *Sort(struct student *head) // 冒泡排序
{
    int count = 0;
    struct student *temp = head->next;
    struct student *last = head; // 指向上一个结点
    while (temp)
    {
        temp = temp->next;
        count++;
    }
    for (int i = count - 1; i > 0; --i)
    {
        last = head;
        temp = head->next;
        for (int j = 0; j < i; j++, last = last->next)
        {
            if (temp->totalmark > temp->next->totalmark)
            {
                last->next = temp->next;
                temp->next = temp->next->next;
                last->next->next = temp;
            }
            else
            {
                temp = temp->next; // 若未发生交换，则正常继续遍历
            }
        }
    }
    return head;
}

struct student *create(int n)
{
    struct student *head, *p1, *p2;
    head = (struct student *)malloc(sizeof(struct student));
    head->next = NULL;
    p1 = head;
    for (int i = 0; i < n; i++)
    {
        p2 = (struct student *)malloc(sizeof(struct student));
        p2->next = NULL;
        scanf("%d %s %f %f", &p2->stunum, p2->name, &p2->examscore, &p2->labscore);
        p2->totalmark = 0.6 * p2->examscore + 0.4 * p2->labscore;
        p1->next = p2;
        p1 = p2;
    }
    head = Sort(head);
    return head;
}

struct student *delete(struct student *head, int stunum)
{
    struct student *temp = head->next, *last = head;

```

```

int flag = 0; // 判断是否找到要删除的结点
while (temp)
{
    if (temp->stunum == stunum)
    {
        last->next = temp->next;
        free(temp); // 释放内存
        temp = last->next;
        flag = 1;
        continue; // 若删除成功那么temp与last均不必更新
    }
    last = last->next;
    temp = temp->next;
}
if (!flag)
{
    printf("未找到要删除的学号! \n");
}

return head;
}

struct student *insert(struct student *head)
{
    struct student *p1;
    printf("请输入待插入的学生信息: \n");
    p1 = (struct student *)malloc(sizeof(struct student));
    scanf("%d %s %f %f", &p1->stunum, p1->name, &p1->examscore, &p1->labscore);
    p1->totalmark = 0.6 * p1->examscore + 0.4 * p1->labscore;
    p1->next = head->next;
    head->next = p1;
    head = Sort(head); // 维护有序性质
    printf("插入成功: \n");
    return head;
}

void Print(struct student *head)
{
    struct student *temp;
    temp = head->next;
    while (temp)
    {
        printf("%d %s %f\n", temp->stunum, temp->name, temp->totalmark);
        temp = temp->next;
    }
    return;
}

int main()
{
    struct student *head;
    int stunum = 11111;
    freopen("score.txt", "r", stdin);
    printf("创建链表: \n");
    head = create(10);
    Print(head);
    printf("插入结点: \n");
    head = insert(head);
    Print(head);
    printf("删除结点, 学号为 %d : \n",stunum);
    head = delete(head, stunum);
    Print(head);
    printf("删除结点, 学号为 111 (不存在): \n");
    head = delete(head, 111);
    Print(head);
    return 0;
}

```

运行结果

```
gdb -i student -Microsoft-MS-Engine
rogram Files\mingw64\bin\gdb.exe
创建链表:
30258 孙杰 50.200001
61256 张男 77.800003
71525 王子 77.800003
29263 叶林 78.000000
22483 陈翔 78.400002
11260 柯以乐 79.599998
69753 李友友 87.199997
12254 东方亮 87.400002
33262 谢涛 88.599998
71250 李霞 89.800003
插入结点:
请输入待插入的学生信息:
插入成功:
30258 孙杰 50.200001
61256 张男 77.800003
71525 王子 77.800003
29263 叶林 78.000000
22483 陈翔 78.400002
11260 柯以乐 79.599998
69753 李友友 87.199997
12254 东方亮 87.400002
33262 谢涛 88.599998
71250 李霞 89.800003
11111 插入 100.000000
删除结点, 学号为 11111 :
30258 孙杰 50.200001
61256 张男 77.800003
71525 王子 77.800003
29263 叶林 78.000000
22483 陈翔 78.400002
11260 柯以乐 79.599998
69753 李友友 87.199997
12254 东方亮 87.400002
33262 谢涛 88.599998
71250 李霞 89.800003
删除结点, 学号为 111 (不存在):
未找到要删除的学号!
30258 孙杰 50.200001
61256 张男 77.800003
71525 王子 77.800003
29263 叶林 78.000000
22483 陈翔 78.400002
11260 柯以乐 79.599998
69753 李友友 87.199997
12254 东方亮 87.400002
33262 谢涛 88.599998
71250 李霞 89.800003
PS C:\wsd\vscode>
```

由于使用重定向，故在 `score.txt` 末尾添加要插入的表项，内容如下：

```
71250 李霞 95 82
69753 李友友 88 86
12254 东方亮 87 88
61256 张男 73 85
30258 孙杰 25 88
11260 柯以乐 82 76
33262 谢涛 91 85
29263 叶林 80 75
22483 陈翔 80 76
71525 王子 71 88
11111 插入 100 100
```

实验报告

练习插入、删除等链表操作，需要对于指针的熟练运用。