**源代码**

#include<stdio.h>

#include<string.h>

#include<math.h>

#include "malloc.h"

#include <stdlib.h>

#define MAX\_LEN 10 /\* 字符串最大长度 \*/

#define STU\_NUM 30 /\* 最多的学生人数 \*/

#define COURSE\_NUM 6 /\* 最多的考试科目数 \*/

#define LEN sizeof(struct Student)

typedef struct Student

{

long num; /\* 每个学生的学号 \*/

char name[MAX\_LEN]; /\* 每个学生的姓名 \*/

float score[COURSE\_NUM]; /\* 每个学生COURSE\_NUM门功课的成绩 \*/

float sum; /\* 每个学生的总成绩 \*/

float aver; /\* 每个学生的平均成绩 \*/

struct Student \*next;

}STU;

int Menu(void); //创建菜单

void Print(STU \*head, int n, int m); //打印函数

void AverSumofEveryStudent(STU \*head, int n, int m); //计算每门课程的总分和平均分

void AverSumofEveryCourse(STU \*head, int n, int m); //计算每个学生的总分和平均分

STU \*SortbyScore(STU \*head, int n); //按每个学生的总分由高到低排出名次表

STU \*Creat(int n, int m); //创建链表并录入信息

STU \*Creat1(int n, int m);

STU \*SortbyScore1(STU \*head, int n); //按每个学生的总分由低到高排出名次表

STU \*SortbyNum(STU \*head); //按学号由小到大排出成绩表

STU \*SortbyName(STU \*head, int n); //按姓名的字典顺序排出成绩表

void SearchbyNum(STU \*head, int n, int m); //按学号查询学生排名及其考试成绩

void SearchbyName(STU \*head, int n, int m); //按姓名查询学生排名及其考试成绩

void StatisticAnalysis(STU \*head, int n, int m); //按类别及比例输出

void WritetoFile(STU \*head, int n, int m); //将每个学生的纪录信息写入文件

STU \*ReadfromFile(STU \*head, int \*n, int \*m); //从文件中读出每个学生的纪录信息并显示

int main()

{

int n, m;

int i;

STU \*head; //定义头节点

head = (STU \*)malloc(LEN);

while (1)

{

i = Menu();

if (i == 1)

{

system("cls"); //清屏

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\tInput student number(n<30):\n"); //输入学生数

printf("\t\t\t");

scanf("%d", &n);

printf("\t\t\tInput course number(m<=6):\n");

printf("\t\t\t");

scanf("%d", &m);

}

switch (i)

{

case 1:

printf("\t\t\tInput student's ID, name and score:\n");

head = Creat(n, m);

system("cls"); //清屏

break;

case 2:

system("cls"); //清屏

AverSumofEveryStudent(head, n, m);

break;

case 3:

system("cls"); //清屏

AverSumofEveryCourse(head, n, m);

break;

case 4:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\tSort in ascending order by score:\n");

head = SortbyScore(head, n);

Print(head, n, m);

break;

case 5:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\tSort in ascending order by score:\n");

head = SortbyScore1(head, n);

Print(head, n, m);

break;

case 6:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\tSort in ascending order by number:\n");

head = SortbyNum(head);

Print(head, n, m);

break;

case 7:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\tSort in dictionary order by name:\n");

head = SortbyName(head, n);

Print(head, n, m);

break;

case 8:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\tInput the number you want to search:\n");

SearchbyNum(head, n, m);

break;

case 9:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\tInput the name you want to search:\n");

SearchbyName(head, n, m);

break;

case 10:

system("cls"); //清屏

StatisticAnalysis(head, n, m);

break;

case 11:

system("cls"); //清屏

printf("\n\n\n");

Print(head, n, m);

break;

case 12:

system("cls"); //清屏

WritetoFile(head, n, m);

break;

case 13:

system("cls"); //清屏

head = ReadfromFile(head, &n, &m);

break;

case 0:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\tEnd of program!\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

return 0;

default:

system("cls"); //清屏

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\tInput error!\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

return 0;

}

// 创建菜单

int Menu(void)

{

int i;

system("title 学生成绩管理系统V7.0 By wyxiang");

printf("\n\n");

printf("\t\t\t 学生成绩管理系统V7.0\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\* Management for Students' scores \*\n");

printf("\t\t\t\* 1. Input record \*\n");

printf("\t\t\t\* 2. Caculate total and average score of every course \*\n");

printf("\t\t\t\* 3. Caculate total and average score of every student \*\n");

printf("\t\t\t\* 4. Sort in descending order by score \*\n");

printf("\t\t\t\* 5. Sort in ascending order by score \*\n");

printf("\t\t\t\* 6. Sort in ascending order by number \*\n");

printf("\t\t\t\* 7. Sort in dictionary order by name \*\n");

printf("\t\t\t\* 8. Search by number \*\n");

printf("\t\t\t\* 9. Search by name \*\n");

printf("\t\t\t\* 10. Statistic analysis \*\n");

printf("\t\t\t\* 11. List record \*\n");

printf("\t\t\t\* 12. Write to a file \*\n");

printf("\t\t\t\* 13. Read from a file \*\n");

printf("\t\t\t\* 0. Exit \*\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n");

printf("\n");

printf("\n");

printf("\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Please Input your choice:");

scanf("%d", &i);

return i;

}

// 创建链表

STU \*Creat(int n, int m) {

STU \*head;

STU \*p1, \*p2;

int i, j;

// system("cls");

for (i = 1; i<n + 1; i++)

{

p1 = (STU \*)malloc(LEN);

printf("\t\t\t");

scanf("%ld", &p1->num);

printf("\t\t\t");

scanf("%s", p1->name);

for (j = 0; j<m; j++)

{

printf("\t\t\t");

scanf("%f", &p1->score[j]);

}

p1->next = NULL;

if (i == 1)

{

head = p2 = p1;

}

else

{

p2->next = p1;

p2 = p1;

}

}

return(head);

}

STU \*Creat1(int n, int m)

{

STU \*head;

STU \*p1, \*p2;

int i, j;

// system("cls");

for (i = 1; i<n + 1; i++)

{

p1 = (STU \*)malloc(LEN);

p1->next = NULL;

if (i == 1)

{

head = p2 = p1;

}

else

{

p2->next = p1;

p2 = p1;

}

}

return(head);

}

void AverSumofEveryStudent(STU \*head, int n, int m)

{

STU \*p;

int i, j = 1;

char ch;

float sum;

p = head;

if (head != NULL)

{

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

for (i = 0; i<m; i++)

{

p = head;

sum = 0.0;

do {

sum += p->score[i];

p = p->next;

} while (p != NULL);

printf("\t\t\tcourse %d: sum=%.0f , aver=%.0f\n", j, sum, sum / n);

j++;

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

void AverSumofEveryCourse(STU \*head, int n, int m)

{

STU \*p;

int i, j;

p = head;

if (head != NULL)

{

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

for (i = 0; i<n; i++)

{

p->sum = 0.0;

for (j = 0; j<m; j++)

{

p->sum += p->score[j];

}

p->aver = p->sum / m;

printf("\t\t\tstudent %d: sum=%.0f , aver=%.0f\n", i + 1, p->sum, p->aver);

p = p->next;

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

STU \*SortbyScore(STU \*head, int n)

{

STU \*endpt; //控制循环比较

STU \*p; //临时指针变量

STU \*p1, \*p2;

p1 = (STU \*)malloc(LEN);

p1->next = head; //注意理解：我们增加一个节点，放在第一个节点的前面，主要是为了便于比较。因为第一个节点没有前驱，我们不能交换地址

head = p1; //让head指向p1节点，排序完成后，我们再把p1节点释放掉

for (endpt = NULL; endpt != head; endpt = p)

{

for (p = p1 = head; p1->next->next != endpt; p1 = p1->next)

{

if (p1->next->sum < p1->next->next->sum) //如果前面的节点键值比后面节点的键值大，则交换

{

p2 = p1->next->next;

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

p2->next = p1->next;

p1->next = p2;

p = p1->next->next;

}

}

}

p1 = head; //把p1的信息去掉

head = head->next; //让head指向排序后的第一个节点

free(p1); //释放p1

p1 = NULL; //p1置为NULL，保证不产生“野指针”，即地址不确定的指针变量

return head;

}

STU \*SortbyScore1(STU \*head, int n)

{

STU \*endpt; //控制循环比较

STU \*p; //临时指针变量

STU \*p1, \*p2;

p1 = (STU \*)malloc(LEN);

p1->next = head; //注意理解：我们增加一个节点，放在第一个节点的前面，主要是为了便于比较。因为第一个节点没有前驱，我们不能交换地址

head = p1; //让head指向p1节点，排序完成后，我们再把p1节点释放掉

for (endpt = NULL; endpt != head; endpt = p)

{

for (p = p1 = head; p1->next->next != endpt; p1 = p1->next)

{

if (p1->next->sum > p1->next->next->sum) //如果前面的节点键值比后面节点的键值大，则交换

{

p2 = p1->next->next;

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

p2->next = p1->next;

p1->next = p2;

p = p1->next->next;

}

}

}

p1 = head; //把p1的信息去掉

head = head->next; //让head指向排序后的第一个节点

free(p1); //释放p1

p1 = NULL; //p1置为NULL，保证不产生“野指针”，即地址不确定的指针变量

return head;

}

STU \*SortbyNum(STU \*head)

{

STU \*first; //为原链表剩下用于直接插入排序的节点头指针

STU \*t; //临时指针变量：插入节点

STU \*p, \*q; //临时指针变量

first = head->next; //原链表剩下用于直接插入排序的节点链表：可根据图12来理解

head->next = NULL; //只含有一个节点的链表的有序链表：可根据图11来理解

while (first != NULL) //遍历剩下无序的链表

{

//注意：这里for语句就是体现直接插入排序思想的地方

for (t = first, q = head; ((q != NULL) && (q->num < t->num)); p = q, q = q->next); //无序节点在有序链表中找插入的位置

first = first->next; //无序链表中的节点离开，以便它插入到有序链表中

if (q == head) //插在第一个节点之前

{

head = t;

}

else //p是q的前驱

{

p->next = t;

}

t->next = q; //完成插入动作

//first = first->next;

}

return head;

}

STU \*SortbyName(STU \*head, int n)

{

STU \*endpt; //控制循环比较

STU \*p; //临时指针变量

STU \*p1, \*p2;

p1 = (STU \*)malloc(LEN);

p1->next = head; //注意理解：我们增加一个节点，放在第一个节点的前面，主要是为了便于比较。因为第一个节点没有前驱，我们不能交换地址

head = p1; //让head指向p1节点，排序完成后，我们再把p1节点释放掉

for (endpt = NULL; endpt != head; endpt = p)

{

for (p = p1 = head; p1->next->next != endpt; p1 = p1->next)

{

if (strcmp(p1->next->name, p1->next->next->name)>0) //如果前面的节点键值比后面节点的键值大，则交换

{

p2 = p1->next->next;

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

p2->next = p1->next;

p1->next = p2; //结合第4点理解

p = p1->next->next; //结合第6点理解

}

}

}

p1 = head; //把p1的信息去掉

head = head->next; //让head指向排序后的第一个节点

free(p1); //释放p1

p1 = NULL; //p1置为NULL，保证不产生“野指针”，即地址不确定的指针变量

return head;

}

void Print(STU \*head, int n, int m)

{

STU \*p;

int i;

p = head;

if (head != NULL) //只要不是空链表，就输出链表中所有节点

{

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

do

{

printf("\t\t\t%ld\t%s\t", p->num, p->name);

for (i = 0; i<m; i++)

{

printf("%.0f\t", p->score[i]);

}

printf("%.0f\t%.0f\n", p->sum, p->sum / m);

p = p->next; //移到下一个节点

} while (p != NULL);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

void SearchbyNum(STU \*head, int n, int m)

{

long num;

int i;

int flag = 1;

printf("\t\t\t");

scanf("%ld", &num);

STU \*p;

p = head;

if (head != NULL)

{

do {

if (p->num == num)

{

printf("\t\t\t");

printf("%ld\t%s\t", p->num, p->name);

for (i = 0; i<m; i++)

{

printf("%.0f\t", p->score[i]);

}

printf("%.0f\t%.0f\n", p->sum, p->sum / m);

flag = 0;

}

p = p->next;

} while (p != NULL);

if (flag)

{

printf("\t\t\t");

printf("Not found!\n");

}

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

void SearchbyName(STU \*head, int n, int m)

{

char name[MAX\_LEN];

int i;

int flag = 1;

printf("\t\t\t");

scanf("%s", name);

STU \*p;

p = head;

if (head != NULL)

{

do {

if (strcmp(name, p->name) == 0)

{

printf("\t\t\t");

printf("%ld\t%s\t", p->num, p->name);

for (i = 0; i<m; i++)

{

printf("%.0f\t", p->score[i]);

}

printf("%.0f\t%.0f\n", p->sum, p->sum / m);

flag = 0;

}

p = p->next;

} while (p != NULL);

if (flag)

{

printf("\t\t\t");

printf("Not found!\n");

}

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

void StatisticAnalysis(STU \*head, int n, int m)

{

int a[6];

STU \*p;

p = head;

int i, j;

for (i = 0; i<m; i++)

{

p = head; // 不要忘了初始化，不然会炸

for (j = 0; j<6; j++) // 初始化

{

a[j] = 0;

}

do {

if (p->score[i]<60)

{

a[0]++;

}

else if (p->score[i]<70)

{

a[1]++;

}

else if (p->score[i]<80)

{

a[2]++;

}

else if (p->score[i]<90)

{

a[3]++;

}

else if (p->score[i]<100)

{

a[4]++;

}

else

{

a[5]++;

}

p = p->next;

} while (p != NULL);

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\* For course %d:\n", i + 1);

printf("\t\t\t\* <60\t%d\t%.2f%%\n", a[0], 100 \* a[0] / 6.0);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 60, 69, a[1], 100 \* a[1] / 6.0);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 70, 79, a[2], 100 \* a[2] / 6.0);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 80, 89, a[3], 100 \* a[3] / 6.0);

printf("\t\t\t\* %d-%d\t%d\t%.2f%%\n", 90, 99, a[4], 100 \* a[4] / 6.0);

printf("\t\t\t\* %d\t%d\t%.2f%%\n", 100, a[5], 100 \* a[5] / 6.0);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

void WritetoFile(STU \*head, int n, int m)

{

STU \*p;

p = head;

FILE \*fp;

int i, j;

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

if ((fp = fopen("student.txt", "w")) == NULL)

{

printf("\t\t\tFail to open student.txt\n");

return;

}

fprintf(fp, "%d\t%d\n", n, m);

for (i = 0; i<n; i++)

{

fprintf(fp, "%12ld%12s", p->num, p->name);

for (j = 0; j<m; j++)

{

fprintf(fp, "%12.0f", p->score[j]);

}

fprintf(fp, "%12.0f%12.0f\n", p->sum, p->sum / m);

p = p->next;

}

printf("\t\t\tExport Successfully!\n");

fclose(fp);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

/\*

void ReadfromFile(STU1 stu[], int \*n, int \*m)

{

FILE \*fp;

int i, j;

printf("\n\n\n");

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

if ((fp = fopen("student.txt", "r")) == NULL)

{

printf("\t\t\tFail to open student.txt\n");

return;

}

fscanf(fp, "%d\t%d\n", n, m);

for (i = 0; i<\*n; i++)

{

fscanf(fp, "%12ld", &stu[i].num);

fscanf(fp, "%12s", stu[i].name);

for (j = 0; j<\*m; j++)

{

fscanf(fp, "%12f", &stu[i].score[j]);

}

fscanf(fp, "%12f%12f", &stu[i].sum, &stu[i].aver);

}

printf("\t\t\tImport Successfully!\n");

fclose(fp);

for (i = 0; i < \*n; i++)

{

printf("\t\t\t%ld\t%s\t", stu[i].num, stu[i].name);

for (j = 0; j<\*m; j++)

{

printf("%.0f\t", stu[i].score[j]);

}

printf("%.0f\t%.0f\n", stu[i].sum, stu[i].aver);

}

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

\*/

//这里用了直接生成了一个新的链表，确保可以直接读取存好的数据

STU \*ReadfromFile(STU \*head, int \*n, int \*m)

{

STU \*p;

FILE \*fp;

int i, j;

if ((fp = fopen("student.txt", "r")) == NULL)

{

printf("\t\t\tFail to open student.txt\n");

return NULL;

}

fscanf(fp, "%d\t%d\n", n, m);

head = Creat1(\*n, \*m); //创建了一个空链表，并且赋给head

p = head;

for (i = 0; i<\*n; i++)

{

fscanf(fp, "%12ld", &p->num);

fscanf(fp, "%12s", &p->name);

for (j = 0; j<\*m; j++)

{

fscanf(fp, "%12f", &p->score[j]);

}

fscanf(fp, "%12f%12f", &p->sum, &p->aver);

p = p->next;

}

i = \*n;

j = \*m;

printf("\t\t\tImport Successfully!\n");

fclose(fp);

Print(head, i, j);

return head;

}