##### 实验1 链表、文件程序设计

**班别：**2023级计算机类 4班  **学号： 2022463010911**  **姓名： 匡剑衡**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **考核点** | **代码与运行**  **（占70分）** | **页面排版**  **（占10分）** | **总结**  **（占20分）** | **成绩** |
| **计分** |  |  |  |  |

**一、实验须知**

（1）请完成本实验每道题的程序编写，将所编写的代码与运行结果的截图粘贴到**本文档**对应的位置。

（2） 首次实验，请先在本地计算机磁盘上创建一个以自己学号+姓名建立文件夹。

如：202300000001张三 ，然后将该文件夹（202300000001张三）复制到ftp服务器中本课程对应的班级目录下。

（3）实验文件上缴方式：将填写好的本文档提交到ft服务器自建的个人文件夹对应实验目录下。

注：请在文件浏览器中输入：

<ftp://172.28.89.9> 用户名：fengnengshans 密码：空

（4）提交截止时间：按清览系统设置的截止时间之前提交实验报告，否则本次实验无成绩。

**二、实验目的**

1、掌握单向链表的概念和建立方法。

2、掌握单向链表的基本操作。

3、掌握单向链表的编程应用。

4、掌握文件的基本概念。

5、掌握文本的基本操作。

6、掌握文件的编程应用。

**三、 实验内容**

1、程序功能：建立一个单向链表，头指针是list，链表中每个结点包含姓名、基本工资信息，编写max\_list函数查找链表中最高基本工资的职工信息。要求在主函数中建立单向链表（注：当输入基本工资为0时，表示输入结束。），然后调用max\_list函数查找链表中最高基本工资的职工信息，最后输出查找结果。

（本题文件命名: exp1\_1.c或exp1\_1.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

//题目

/\*1）程序功能：建立一个单向链表，头指针是 list，

链表中每个结点包含姓名、基本工资信息，编写 max\_list 函数查找链表中最高基本工资的职工信息。

要求在主函数中建立单向链表（注：当输入基本工资为 0 时，表示输入结束。），

然后调用 max\_list 函数查找链表中最高基本工资的职工信息，最后输出查找结果。\*/

//思路

//虽然说是建立单向列表，但可以使用多一个指针，用于指向插入链表的末尾，方便插入数据

struct ListNode

{

char name[50];

double money;

ListNode \*next;

};

//函数声明

void insert(struct ListNode \*\*pointerToHead, struct ListNode \*\*pointerToLastNode, char \*name, double money);

struct ListNode \*max\_list(struct ListNode \*head);

void printAllNodeInfo(struct ListNode \*Node);

void printNodeInfo(struct ListNode \*node);

int main()

{

double money;

char name[50];

ListNode \*list = NULL;

ListNode \*lastNode = NULL;

printf("请输入职工信息（输入基本工资为0时结束输入）：\n");

while (1)

{

printf("姓名：");

scanf("%s", name);

printf("基本工资：");

scanf("%lf", &money);

if (money == 0.0) break;//直到输入工资0结束输入

insert(&list, &lastNode, name, money);

}

struct ListNode \*maxNode = max\_list(list);

printNodeInfo(maxNode);

//printAllNodeInfo(list);

// 释放链表节点内存

struct ListNode\* current = list;

while (current != NULL)

{

struct ListNode\* temp = current;

current = current->next;

free(temp);

}

return 0;

}

//将节点信息插入链表，尾插法，保持插入的顺序，接受头指针和尾节点指针的指针作为参数，用于快速找到需要插入的节点

void insert(struct ListNode\*\* pointerToHead,struct ListNode\*\* pointerToLastNode, char\* name, double money)

{

struct ListNode \*newNode = (struct ListNode \*)malloc(sizeof(struct ListNode));

strcpy(newNode->name, name);

newNode->money = money;

newNode->next = NULL;

if (\*pointerToHead == NULL)

{

\*pointerToHead = newNode;

\*pointerToLastNode = newNode;

}

else

{

(\*pointerToLastNode)->next = newNode;

\*pointerToLastNode = newNode;

}

}

//找到工资最大的节点

struct ListNode\* max\_list(struct ListNode\* head)

{

struct ListNode \*maxNode = NULL;

struct ListNode \*tempNode = head;

double maxMoney = 0;

while (tempNode!=NULL)

{

if (tempNode->money>maxMoney)

{

maxMoney = tempNode->money;

maxNode = tempNode;

}

tempNode = tempNode->next;

}

maxNode->money = maxMoney;

return maxNode;

}

//输出所有节点的信息

void printAllNodeInfo(struct ListNode\* Node)

{

struct ListNode \*node = Node;

while(node != NULL)

{

printf("姓名：%s\n", node->name);

printf("基本工资：%.2lf\n", node->money);

node = node->next;

}

printf("\n");

}

//输出最大节点信息

void printNodeInfo(struct ListNode\* node)

{

if (node != NULL)

{

printf("最高基本工资的职工信息：\n");

printf("姓名：%s\n", node->name);

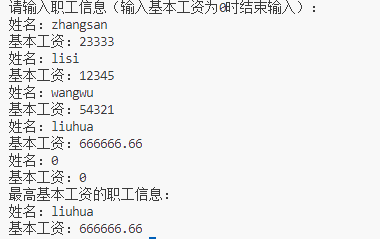
printf("基本工资：%.2lf\n", node->money);

}

else {printf("链表为空或没有有效的基本工资信息。\n");}

}

**【运行结果截图】：**



2、程序功能：建立一个单向链表，头指针是list，链表中每个结点包含姓名、基本工资信息，编写count\_list函数统计链表中超过平均基本工资的人数。要求在主函数中建立单向链表（注：当输入基本工资为0时，表示输入结束。），然后调用count\_list函数统计链表中超过平均基本工资的人数，最后输出统计结果和平均基本工资。（本题文件命名: exp1\_2.c或exp1\_2.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdint.h>

#define \_\_STDC\_FORMAT\_MACROS

#include<inttypes.h>

//题目

/\*2）程序功能：建立一个单向链表，头指针是 list，链表中每个结点包含姓名、基本工资信息，

编写 count\_list 函数统计链表中超过平均基本工资的人数。

要求在主函数中建立单向链表（注：当输入基本工资为 0 时，表示输入结束。），

然后调用 count\_list 函数统计链表中超过平均基本工资的人数，最后输出统计结果和平均基本工资。\*/

//思路

//整体和exp1\_1差不多

struct ListNode

{

char name[50];

double money;

ListNode \*next;

};

//函数声明

void insert(struct ListNode \*\*pointerToHead, struct ListNode \*\*pointerToLastNode, char \*name, double money);

void printAllNodeInfo(struct ListNode \*Node);

uint32\_t count\_list(struct ListNode \*Node, double avg\_money);

double avgMoney(struct ListNode \*Node);

int main()

{

double money;

char name[50];

ListNode \*list = NULL;

ListNode \*lastNode = NULL;

printf("请输入职工信息（输入基本工资为0时结束输入）：\n");

while (1)

{

printf("姓名：");

scanf("%s", name);

printf("基本工资：");

scanf("%lf", &money);

if (money == 0.0) break;//直到输入工资0结束输入

insert(&list, &lastNode, name, money);

}

double avg = avgMoney(list);

uint32\_t count = count\_list(list, avg);

printf("超过平均工资的人数为%" PRIu32 ",平均基本工资为%.2lf\n", count, avg);

//printAllNodeInfo(list);

// 释放链表节点内存

struct ListNode\* current = list;

while (current != NULL)

{

struct ListNode\* temp = current;

current = current->next;

free(temp);

}

return 0;

}

//将节点信息插入链表，尾插法，保持插入的顺序，接受头指针和尾节点指针的指针作为参数，用于快速找到需要插入的节点

void insert(struct ListNode\*\* pointerToHead,struct ListNode\*\* pointerToLastNode, char\* name, double money)

{

struct ListNode \*newNode = (struct ListNode \*)malloc(sizeof(struct ListNode));

strcpy(newNode->name, name);

newNode->money = money;

newNode->next = NULL;

if (\*pointerToHead == NULL)

{

\*pointerToHead = newNode;

\*pointerToLastNode = newNode;

}

else

{

(\*pointerToLastNode)->next = newNode;

\*pointerToLastNode = newNode;

}

}

//输出所有节点的信息

void printAllNodeInfo(struct ListNode\* Node)

{

struct ListNode \*node = Node;

while(node != NULL)

{

printf("姓名：%s\n", node->name);

printf("基本工资：%.2lf\n", node->money);

node = node->next;

}

printf("\n");

}

//统计链表中超过平均工资的人数

uint32\_t count\_list(struct ListNode \*head,double avg\_money)

{

uint32\_t cnt = 0;

struct ListNode \*tempNode = head;

double avg = avg\_money;

while (tempNode!=NULL)

{

if (tempNode->money>avg){

cnt++;

}

tempNode = tempNode->next;

}

return cnt;

}

//计算链表中的平均工资

double avgMoney(struct ListNode \*head)

{

uint32\_t cnt = 0;

double sum = 0;

struct ListNode \*tempNode = head;

while (tempNode!=NULL)

{

cnt++;

sum += tempNode->money;

tempNode = tempNode->next;

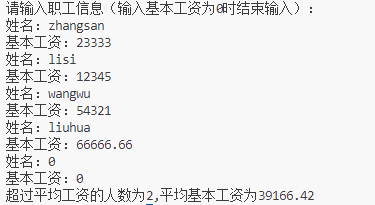
}

double avg = sum / cnt;

return avg;

}

**【运行结果截图】：**



3、程序功能：建立一个单向链表，头指针是list，链表中每个结点包含姓名、基本工资信息，编写del\_list函数删除链表中等于某给定基本工资的所有结点。要求在主函数中先建立单向链表（注：当输入基本工资为0时，表示输入结束。），再输入一个给定基本工资，然后调用del\_list函数删除链表中等于给定基本工资的所有结点，最后输出删除后的链表信息。（本题文件命名: exp1\_3.c或exp1\_3.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdint.h>

#define \_\_STDC\_FORMAT\_MACROS

#include<inttypes.h>

//题目

/\*3）程序功能：建立一个单向链表，头指针是 list，

链表中每个结点包含姓名、基本工资信息，

编写 del\_list 函数删除链表中等于某给定基本工资的所有结点。

要求在主函数中先建立单向链表（注：当输入基本工资为 0 时，表示输入结束。），

再输入一个给定基本工资，

然后调用 del\_list 函数删除链表中等于给定基本工资的所有结点，最后输出删除后的链表信息。\*/

//思路

//删除节点，不过是给节点的值，去删除，可以先找到值返回对应的节点，然后再删除，不过c没有内置map数据结构，否则可以使用map优化查找的时间

//不过还有一个不确定的点就是原链表需不需要保存？如果映射到实际生活的话，认为应该需要保留，因为可能还有恢复操作的需求

//两种都写吧

//如果U\_S 1表示原链表保存，0表示原链表不保存

#define U\_S 1

#define MAX\_NAME\_LEN 50

struct ListNode

{

char name[50];

double money;

ListNode \*next;

};

//函数声明

void insert(struct ListNode \*\*pointerToHead, struct ListNode \*\*pointerToLastNode, char \*name, double money);

void printAllNodeInfo(struct ListNode \*Node);

void del\_list(struct ListNode \*\*pointerToHead, double delMoney);

struct ListNode \*del\_list\_s(struct ListNode \*Node, double delMoney);

void freeList(struct ListNode \*Node);

int main()

{

double money;

char name[MAX\_NAME\_LEN + 1];

ListNode \*list = NULL;

ListNode \*lastNode = NULL;

printf("请输入职工信息（输入基本工资为0时结束输入）：\n");

while (1)

{

printf("姓名：");

scanf("%s", name);

printf("基本工资：");

scanf("%lf", &money);

if (money == 0.0) break;//直到输入工资0结束输入

insert(&list, &lastNode, name, money);

}

//这里其实可以扩展，因为可能不止删除一次

printf("请输入需要删除的节点对应的工资：");

double del\_money;

scanf("%lf", &del\_money);

if (U\_S){

struct ListNode \*newList = del\_list\_s(list, del\_money);

printAllNodeInfo(newList);

freeList(newList);

freeList(list);

}

else{

del\_list(&list, del\_money);

printAllNodeInfo(list);

freeList(list);

}

return 0;

}

//将节点信息插入链表，尾插法，保持插入的顺序，接受头指针和尾节点指针的指针作为参数，用于快速找到需要插入的节点

void insert(struct ListNode\*\* pointerToHead,struct ListNode\*\* pointerToLastNode, char\* name, double money)

{

struct ListNode \*newNode = (struct ListNode \*)malloc(sizeof(struct ListNode));

strcpy(newNode->name, name);

newNode->money = money;

newNode->next = NULL;

if (\*pointerToHead == NULL)

{

\*pointerToHead = newNode;

\*pointerToLastNode = newNode;

}

else

{

(\*pointerToLastNode)->next = newNode;

\*pointerToLastNode = newNode;

}

}

//输出所有节点的信息

void printAllNodeInfo(struct ListNode\* Node)

{

struct ListNode \*node = Node;

while(node != NULL)

{

printf("姓名：%s\n", node->name);

printf("基本工资：%.2lf\n", node->money);

node = node->next;

}

printf("\n");

}

//删除链表中等于delMoney的所有节点,返回新链表的头节点

struct ListNode \*del\_list\_s(struct ListNode \*head, double delMoney)

{

struct ListNode \*newList = NULL;

struct ListNode \*newlastList = NULL;

struct ListNode \*tempNode = head;

while (tempNode!=NULL)

{

if (tempNode->money!=delMoney)

{

insert(&newList, &newlastList, tempNode->name, tempNode->money);

}

tempNode = tempNode->next;

}

return newList;

}

//删除链表中等于delMoney的所有节点

void del\_list(struct ListNode \*\*head, double delMoney)

{

struct ListNode \*curr = \*head;

struct ListNode \*prev = NULL;

// 处理头节点的情况

while (curr != NULL && curr->money == delMoney)

{

struct ListNode \*temp = curr;

curr = curr->next;

free(temp);

\*head = curr;

}

// 处理中间节点和尾节点的情况

prev = curr;

curr = curr->next;

while (curr != NULL)

{

if (curr->money == delMoney)

{

struct ListNode \*temp = curr;

prev->next = curr->next;

curr = curr->next;

free(temp);

} else

{

prev = curr;

curr = curr->next;

}

}

}

void freeList(struct ListNode \*head)

{

// 释放链表节点内存

struct ListNode\* current = head;

while (current != NULL)

{

struct ListNode\* temp = current;

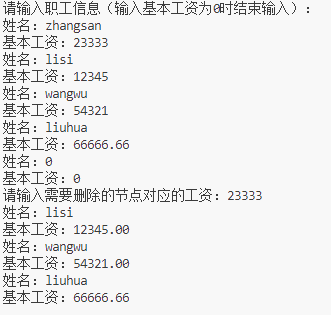
current = current->next;

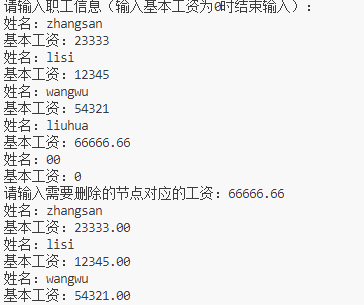
free(temp);

}

}

**【运行结果截图】：**





4、程序功能：输入若干个学生成绩（输入-1为结束标志），建立两个已按升序排序的单向链表，头指针分别为list1、list2，把两个链表拼成一个升序排序的新链表，并输出新链表信息。要求自定义函数，实现将两个链表拼成一个链表，并返回拼组后的新链表。（本题文件命名: exp1\_4.c或exp1\_4.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdint.h>

#define \_\_STDC\_FORMAT\_MACROS

#include<inttypes.h>

//题目

/\*4）程序功能：输入若干个学生成绩（输入-1 为结束标志），

建立两个已按升序排序的单向链表，头指针分别为 list1、list2，

把两个链表拼成一个升序排序的新链表，并输出新链表信息。

要求自定义函数，实现将两个链表拼成一个链表，并返回拼组后的新链表。\*/

//思路

//哦，这个不需要输入学生信息，只要输入成绩。。。？两种都写吧

//既然要写合并的功能，那排序就用归并排序来写吧

#define MAXN 50

typedef struct Student

{

uint32\_t ID;

char name[MAXN + 1];

double score;

struct Student \*next;

}Student;

//函数声明

void freeList(Student \*stu);

Student \*sortList(Student \*head);

Student \*merge(Student \*list1, Student \*list2);

#define ONLY\_SCORE

//用个开关 ONLY\_SCORE 表示只有输入成绩，不定义则表示完整学生信息

#ifdef ONLY\_SCORE

//函数声明

Student \*createNode(double score);

Student \*createList(void);

void printList(Student \*head);

int main()

{

//创建链表

Student \*list1 = NULL;

Student \*list2 = NULL;

printf("请输入 list1 的学生成绩（输入成绩为-1表示结束):\n");

list1 = createList();

printf("请输入 list2 的学生成绩（输入成绩为-1表示结束):\n");

list2 = createList();

//输出链表

printf(" list1 的学生成绩为：\n");

printList(list1);

printf(" list2 的学生成绩为：\n");

printList(list2);

/\*

Student \*temp = merge(list1, list2);

printList(temp);

printf("\n");

\*/

//排序链表

printf("排序后的 list1 的学生成绩为：\n");

sortList(list1);

printList(list1);

printf("排序后的 list2 的学生成绩为：\n");

sortList(list2);

printList(list2);

//拼接链表

printf("拼接 list1 和 list2 的结果为：\n");

Student \*newList = merge(list1, list2);

printList(newList);

//释放链表

freeList(list1);

freeList(list2);

freeList(newList);

return 0;

}

Student \*createNode(double score)

{

Student \*newNode = (Student \*)malloc(sizeof(Student));

newNode->score = score;

newNode->next = NULL;

return newNode;

}

Student \*createList(void)

{

Student \*head = NULL;

Student \*tail = NULL;

double score;

while (1)

{

printf("请输入成绩(-1表示结束)：\n");

scanf("%lf", &score);

if (score==-1){

break;

}

Student \*newNode = createNode(score);

if (head == NULL)

{

head = tail = newNode;

}

else

{

tail->next = newNode;

tail = newNode;

}

}

return head;

}

void printList(Student \*head)

{

Student \*p = head;

while (p!=NULL)

{

printf("%.2lf ", p->score);

p = p->next;

}

printf("\n");

}

#else

//函数声明

Student \*createNode(uint32\_t id, char \*name, double score);

Student \*createList(void);

void printList(Student \*head);

int main()

{

//创建链表

Student \*list1 = NULL;

Student \*list2 = NULL;

printf("请输入 list1 的学生信息（输入学号为0表示结束):\n");

list1 = createList();

printf("请输入 list2 的学生信息（输入学号为0表示结束):\n");

list2 = createList();

//输出链表

printf(" list1 的学生信息为：\n");

printList(list1);

printf(" list2 的学生信息为：\n");

printList(list2);

//排序链表

printf("按成绩排序后的 list1 的学生信息为：\n");

sortList(list1);

printList(list1);

printf("按成绩排序后的 list2 的学生信息为：\n");

sortList(list2);

printList(list2);

//拼接链表

printf("拼接 list1 和 list2 的结果为：\n");

Student \*newList = merge(list1, list2);

printList(newList);

//释放链表

freeList(list1);

freeList(list2);

freeList(newList);

return 0;

}

Student \*createNode(uint32\_t id, char \*name, double score)

{

Student \*newNode = (Student \*)malloc(sizeof(Student));

newNode->ID = id;

strcpy(newNode->name, name);

newNode->score = score;

newNode->next = NULL;

return newNode;

}

Student \*createList(void)

{

Student \*head = NULL;

Student \*tail = NULL;

uint32\_t id;

char name[MAXN + 1];

double score;

while(1)

{

printf("请输入学号(输入0表示结束)：");

scanf("%" SCNu32 "", &id);

if(id==0){

break;

}

printf("请输入姓名：");

scanf("%s", name);

printf("请输入成绩：");

scanf("%lf", &score);

Student \*newNode = createNode(id, name, score);

if (head == NULL)

{

head = tail = newNode;

}

else

{

tail->next = newNode;

tail = newNode;

}

}

return head;

}

void printList(Student \*head)

{

Student \*p = head;

for(p = head;p!=NULL;p = p->next)//遍历

{

printf("学号: %" PRIu32 " 姓名：%s 成绩：%.2lf\n", p->ID, p->name, p->score); // 输出

}

}

#endif

void freeList(Student \*stu)

{

// 释放链表节点内存

struct Student\* current = stu;

while (current != NULL)

{

struct Student\* temp = current;

current = current->next;

free(temp);

}

}

//归并排序

Student \*sortList(Student \*head)

{

Student \*temp = head;

//链表归并排序

if(temp == NULL || temp->next == NULL)return temp;

else

{

//快慢指针找到中间节点

Student \*fast = temp->next, \*slow = temp;

while(fast != NULL && fast->next != NULL)

{

fast = fast->next->next;

slow = slow->next;

}

Student \*mid = slow->next;

slow->next = NULL;

Student \*left = sortList(temp); // 前半段排序

Student \*right = sortList(mid); // 后半段排序

return merge(left,right);

}

}

//升序合并

Student \*merge(Student \*list1, Student \*list2)

{

Student \*head1 = list1;

Student \*head2 = list2;

if(head1 == NULL)return head2;

if(head2 == NULL)return head1;

Student \*res, \*p;

if(head1->score < head2->score)

{

res = head1;

head1 = head1->next;

}

else

{

res = head2;

head2 = head2->next;

}

p = res;

while(head1 != NULL && head2 != NULL)

{

if(head1->score < head2->score)

{

p->next = head1;

head1 = head1->next;

}

else

{

p->next = head2;

head2 = head2->next;

}

p = p->next;

}

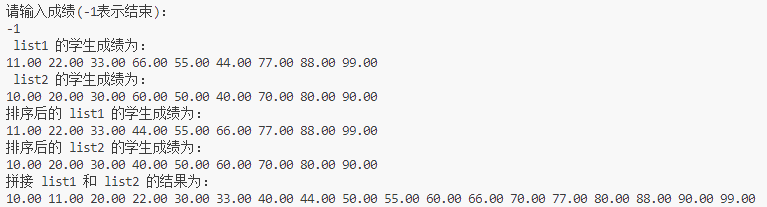
if(head1 != NULL)p->next = head1;

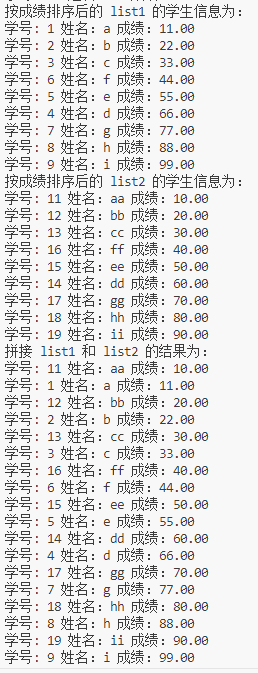
else if(head2 != NULL)p->next = head2;

return res;

}

**【运行结果截图】：**





5、程序功能：已知一个数据文件a.txt中保存了若干个学生的计算机等级考试成绩，包括学号、姓名和成绩。请编程读出显示文件内容并分类统计各等级人数，最后显示统计结果。成绩等级规则：大于等于90-A，大于等于80-B，大于等于70-C，大于等于60-D，60以下-E。（本题文件命名: exp1\_5.c或exp1\_5.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<string.h>

/\*5）程序功能：已知一个数据文件 a.txt 中保存了若干个学生的计算机等级考试成绩，包括学号、姓名和成绩。

请编程读出显示文件内容并分类统计各等级人数，最后显示统计结果。

成绩等级规则：大于等于 90-A，大于等于 80-B，大于等于 70-C，大于等于 60-D，60 以下-E。

\*/

#define MAXN 50

typedef struct Student

{

char id[10];

char name[MAXN + 1];

double score;

struct Student \*next;

}Student;

int main()

{

FILE \*file = fopen("a.txt", "r");

int countA = 0, countB = 0, countC = 0, countD = 0, countE = 0;

Student student;

while (fscanf(file, "%s%s%lf", &student.id, student.name, &student.score) != EOF) {

printf("学号:%s, 姓名:%s, 成绩:%.1lf\n", student.id, student.name, student.score);

if (student.score >= 90) {

countA++;

} else if (student.score >= 80) {

countB++;

} else if (student.score >= 70) {

countC++;

} else if (student.score >= 60) {

countD++;

} else {

countE++;

}

}

fclose(file);

printf("\n统计结果:\n");

printf("A级: %d人\n", countA);

printf("B级: %d人\n", countB);

printf("C级: %d人\n", countC);

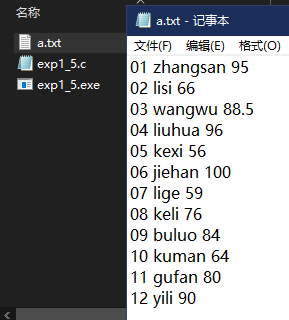
printf("D级: %d人\n", countD);

printf("E级: %d人\n", countE);

return 0;

}

**【运行结果截图】：**





6、程序功能：已知文件 a.txt 存储若干文本数据，编程统计 a.txt 文件的行数、大写字母个数、小写字母个数、数字字符个数及其他字符的个数。要求如下：

① 定义 countFile()函数：统计文本文件的行数、大写字母个数、小写字母个数、数字字符个数及其他字符的个数。

② 定义 main()函数：打开 a.txt 文件，调用 countFile()函数统计文本文件的行数、大写字母个数、小写字母个数、数字字符个数及其他字符的个数，输出统计结果。

（本题文件命名: exp1\_6.c或exp1\_6.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

#include<stdint.h>

#define \_\_STDC\_FORMAT\_MACROS

#include<inttypes.h>

/\*程序功能：已知文件 a.txt 存储若干文本数据，编程统计 a.txt 文件的行数、大写字母个数、小写字母个数、数字字符个数及其他字符的个数。

要求如下：

① 定义 countFile()函数：统计文本文件的行数、大写字母个数、小写字母个数、数字字符个数及其他字符的个数。

② 定义 main()函数：打开 a.txt 文件，调用 countFile()函数统计文本文件的行数、大写字母个数、小写字母个数、数字字符个数及其他字符的个数，输出统计结果。

\*/

#define u32 uint32\_t

void countFile(FILE \*fp, u32 \*lineCount, u32 \*upperCount, u32 \*lowerCount, u32 \*digitCount, u32 \*otherCount) {

u32 ch;

\*lineCount = 0;

\*upperCount = 0;

\*lowerCount = 0;

\*digitCount = 0;

\*otherCount = 0;

while ((ch = fgetc(fp)) != EOF) {

if (ch == '\n') {

(\*lineCount)++;

} else if (isupper(ch)) {

(\*upperCount)++;

} else if (islower(ch)) {

(\*lowerCount)++;

} else if (isdigit(ch)) {

(\*digitCount)++;

} else {

(\*otherCount)++;

}

}

}

int main()

{

FILE \*fp;

u32 lineCount, upperCount, lowerCount, digitCount, otherCount;

fp = fopen("a.txt", "r");

countFile(fp, &lineCount, &upperCount, &lowerCount, &digitCount, &otherCount);

fclose(fp);

printf("统计结果:\n");

printf("行数: %d\n", lineCount);

printf("大写字母个数: %d\n", upperCount);

printf("小写字母个数: %d\n", lowerCount);

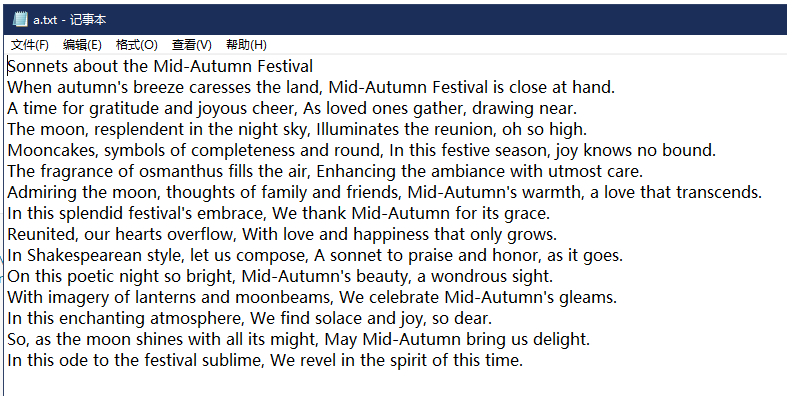
printf("数字字符个数: %d\n", digitCount);

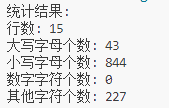
printf("其他字符个数: %d\n", otherCount);

return 0;

}

**【运行结果截图】：**





7、程序功能：编写一个程序，比较两个文本文件a1.txt和a2.txt的内容是否相同，若相同则输出“YES”，否则输出“NO”，并输出两个文件内容首次不同的行号和字符位置。（本题文件命名: exp1\_7.c或exp1\_7.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdint.h>

#define \_\_STDC\_FORMAT\_MACROS

#include<inttypes.h>

/\*

程序功能：编写一个程序，比较两个文本文件a1.txt和a2.txt的内容是否相同，若相同则输出“YES”，否则输出“NO”，并输出两个文件内容首次不同的行号和字符位置。

\*/

#define u32 uint32\_t

#define MAX\_LINE\_LENGTH 2048

Void compareFiles(const char \*file1, const char \*file2) {

FILE \*fp1 = fopen(file1, "r");

FILE \*fp2 = fopen(file2, "r");

char line1[MAX\_LINE\_LENGTH], line2[MAX\_LINE\_LENGTH];

u32 line\_num = 1, char\_pos = 0;

while (fgets(line1, sizeof(line1), fp1) && fgets(line2, sizeof(line2), fp2)) {

char\_pos = 0;

while (line1[char\_pos] == line2[char\_pos]) {

if (line1[char\_pos] == '\0' || line2[char\_pos] == '\0') {

break;

}

char\_pos++;

}

if (line1[char\_pos] != line2[char\_pos]) {

printf("NO\n");

printf("第 %" PRIu32 " 行第 %" PRIu32 " 个字符不同\n", line\_num, char\_pos + 1);

fclose(fp1);

fclose(fp2);

}

line\_num++;

}

if (fgets(line1, sizeof(line1), fp1) || fgets(line2, sizeof(line2), fp2)) {

printf("NO\n");

printf("文件长度不同\n");

} else {

printf("YES\n");

}

fclose(fp1);

fclose(fp2);

}

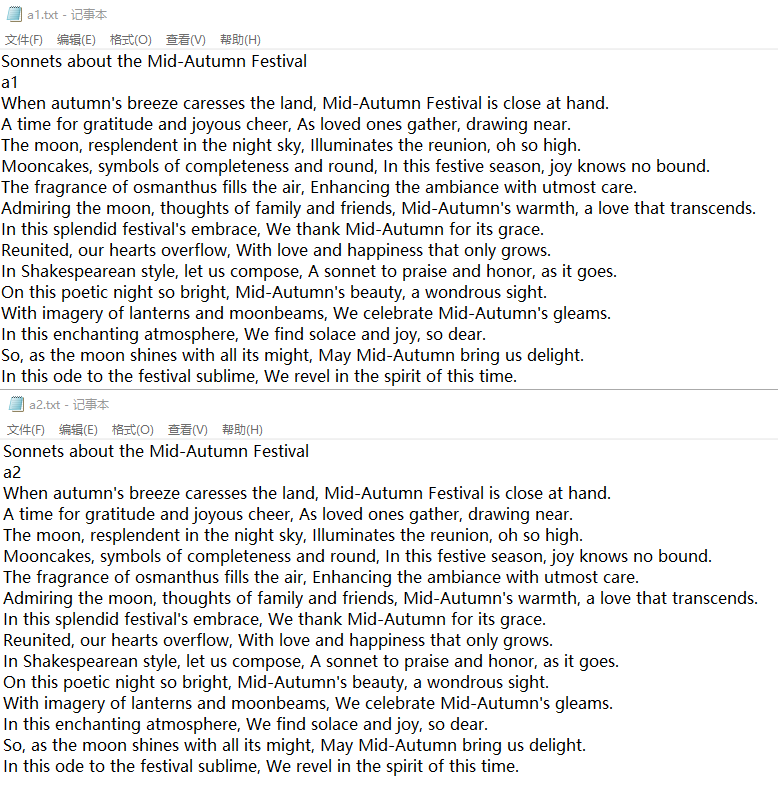
int main() {

compareFiles("a1.txt", "a2.txt");

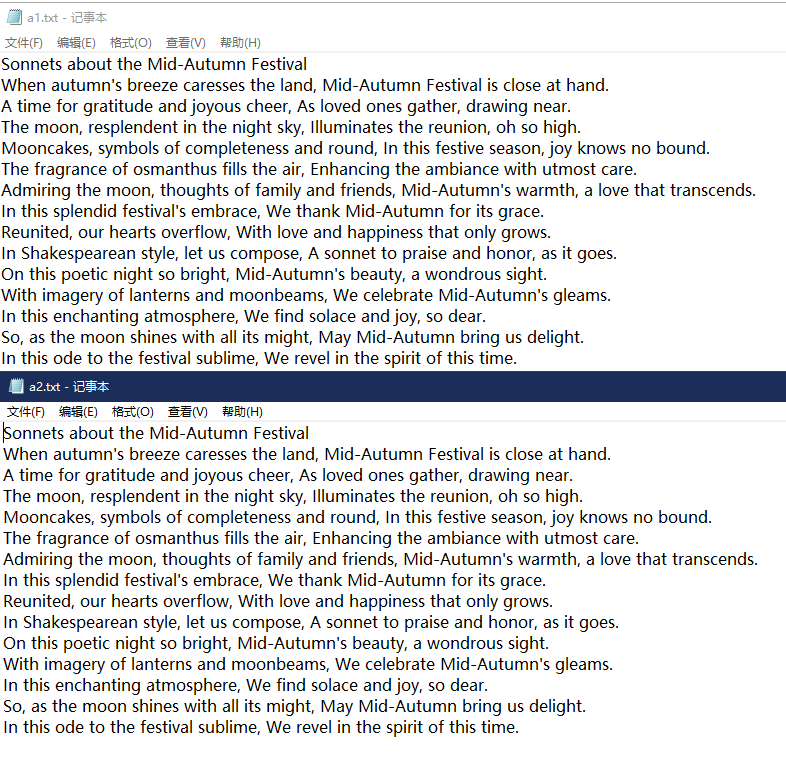
return 0;

}

**【运行结果截图】：**









8、程序功能：从键盘输入若干个学生数据（包括学号、姓名和成绩）保存到文本文件 a.txt 中（以负数成绩表示输入结束），然后再从该文件中读出并显示。

（本题文件命名: exp1\_8.c或exp1\_8.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdint.h>

#define \_\_STDC\_FORMAT\_MACROS

#include<inttypes.h>

/\*

程序功能：从键盘输入若干个学生数据（包括学号、姓名和成绩）保存到文本文件 a.txt 中（以负数成绩表示输入结束），然后再从该文件中读出并显示。

\*/

#define MAX\_NAME\_LEN 50

typedef struct Student

{

char id[10];

char name[MAX\_NAME\_LEN + 1];

double score;

struct Student \*next;

}Student;

int main() {

FILE \*file = fopen("a.txt", "w");

Student student;

printf("请输入学生数据(学号 姓名 成绩)，以负数成绩结束:\n");

while (1) {

printf("学号: ");

scanf("%s", student.id);

printf("姓名: ");

scanf("%s", student.name);

printf("成绩: ");

scanf("%lf", &student.score);

if (student.score < 0) {

break;

}

fprintf(file, "%s %s %.1lf\n", student.id, student.name, student.score);

}

fclose(file);

file = fopen("a.txt", "r");

printf("\n从文件读出的数据:\n");

while (fscanf(file, "%s%s%lf", student.id, student.name, &student.score) != EOF) {

printf("学号: %s, 姓名: %s, 成绩: %.1lf\n", student.id, student.name, student.score);

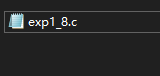
}

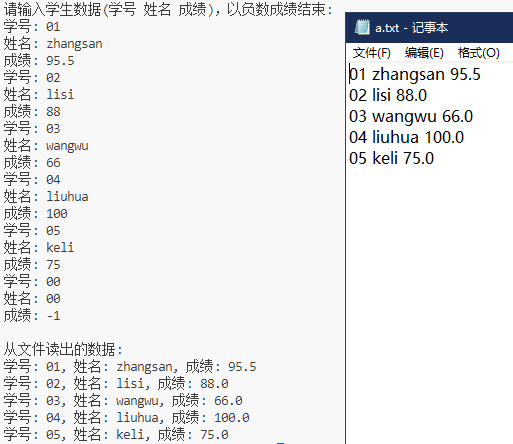
fclose(file);

return 0;

}

**【运行结果截图】：**

运行前：

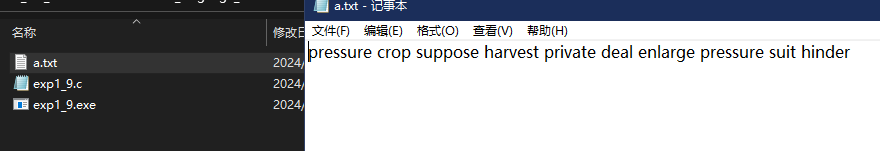
运行后：

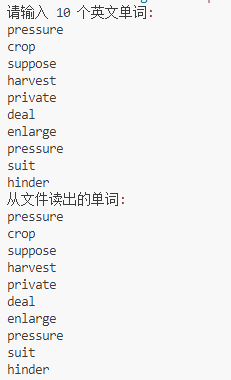
9、程序功能：从键盘输入10个英文单词到二维字符数组a并保存到文本文件a.txt中（单词间以空格分隔），然后再从该文件中读出所有单词并显示到屏幕。

（本题文件命名: exp1\_9.c或exp1\_9.cpp ）

**【代码文本】：**

**【运行结果截图】：**





10、程序功能：从键盘输入若干个学生数据（包括学号、姓名和成绩）保存到二进制文件 a.dat（以负数成绩表示输入结束），然后再从该文件中读出并显示。

（本题文件命名: exp1\_10.c或exp1\_10.cpp ）

**【代码文本】：**

#include<stdio.h>

#include<string.h>

/\*

从键盘输入若干个学生数据（包括学号、姓名和成绩）保存到二进制文件 a.dat（以负数成绩表示输入结束），然后再从该文件中读出并显示。

\*/

#define MAX\_ID\_LEN 10

#define MAX\_NAME\_LEN 50

typedef struct Student

{

char id[MAX\_ID\_LEN + 1];

char name[MAX\_NAME\_LEN + 1];

double score;

struct Student \*next;

}Student;

int main() {

FILE \*file = fopen("a.dat", "wb");

Student student;

printf("请输入学生数据(学号 姓名 成绩)，以负数成绩结束:\n");

while (1) {

printf("学号: ");

scanf("%s", student.id);

printf("姓名: ");

scanf("%s", student.name);

printf("成绩: ");

scanf("%lf", &student.score);

if (student.score < 0) {

break;

}

fwrite(&student, sizeof(Student), 1, file);

}

fclose(file);

file = fopen("a.dat", "rb");

printf("\n从文件读出的数据:\n");

while (fread(&student, sizeof(Student), 1, file) == 1) {

printf("学号: %s, 姓名: %s, 成绩: %.1lf\n", student.id, student.name, student.score);

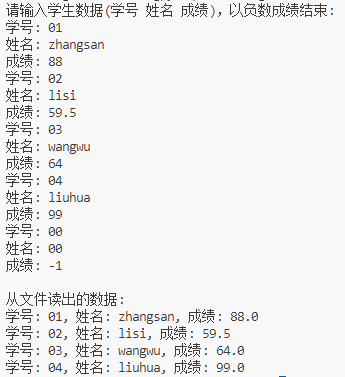
}

fclose(file);

return 0;

}

**【运行结果截图】：**



11、程序功能：两个非空二进制文件 a1.dat 和 a2.dat，分别保存若干学生数据（包括学号、姓名和成绩），且按成绩升序排列。编写程序，把 a1.dat 和 a2.dat 两个文件归并到 a3.dat 文件，使得 a3.dat 文件中的学生数据也按成绩升序排列。要求如下：

① 定义 mergeFile()函数：将两个二进制文件归并成一个按成绩升序排列的新文件。

② 定义 main()函数：先打开二进制文件 a1.dat、a2.dat、a3.dat，再调用 mergeFile()函数将 a1.dat、a2.dat 两个文件归并到按成绩升序排列的 a3.dat 文件，最后输出 a3.dat 文件内容。

（本题文件命名: exp1\_11.c或exp1\_11.cpp ）

**【代码文本】：**

**【运行结果截图】：**

**四、实验总结（ 20分）**

总结实验过程中遇到的问题，列出解决问题的方法，谈谈体会