## 3.结构体

#### 一、定义结构体

##### 1.概念

结构体(struct)是由一系列具有相同类型或不同类型的数据构成的数据集合，叫做结构。

##### 2.为什么需要结构体

结构体和其他类型基础数据类型一样，例如int类型，char类型 只不过结构体可以做成你想要的数据类型。

在实际项目中，结构体是大量存在的。研发人员常使用结构体来封装一些属性来组成新的类型。由于C语言内部程序比较简单，研发人员通常使用结构体创造新的“属性”，其目的是简化运算。

结构体在函数中的作用不是简便，其最主要的作用就是封装。封装的好处就是可以再次利用。让使用者不必关心这个是什么，只要根据定义使用就可以了。

##### 3.定义结构体

结构体的定义如下所示，struct为结构体关键字，tag为结构体的标志，member-list为结构体成员列表，其必须列出其所有成员；variable-list为此结构体声明的变量。

Struct tag{

Member-list

} variable-list;

在一般情况下，tag、member-list、variable-list这3部分至少要出现2个。以下为示例:

//此声明声明了拥有3个成员的结构体，分别为整型的a，字符型的b和双精度的c

//同时又声明了结构体变量s1

//这个结构体并没有标明其标签

struct {

    int a;

    char b;

    double c;

} s1;

//同上声明了拥有3个成员的结构体，分别为整型的a，字符型的b和双精度的c

//结构体的标签被命名为SIMPLE,没有声明变量

struct SIMPLE{

     int a;

     char b;

    double c;

};

//用SIMPLE标签的结构体，另外声明了变量t1、t2、t3

struct SIMPLE t1, t2[20], \*t3;

//也可以用typedef创建新类型

typedef struct{

    int a;

    char b;

    double c;

} Simple2;

//现在可以用Simple2作为类型声明新的结构体变量

Simple2 u1, u2[20], \*u3;

在上面的声明中，第一个和第二声明被编译器当作两个完全不同的类型，即使他们的成员列表是一样的，如果令t3=&s1，则是非法的。

结构体的成员可以包含其他结构体，也可以包含指向自己结构体类型的[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88" \t "_blank)，而通常这种[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88" \t "_blank)的应用是为了实现一些更高级的数据结构如链表和树等。

//此结构体的声明包含了其他的结构体

struct COMPLEX{

    char string[100];

    struct SIMPLE a;

};

//此结构体的声明包含了指向自己类型的指针

struct NODE{

    char string[100];

    struct NODE \*next\_node;

};

如果两个结构体互相包含，则需要对其中一个结构体进行不完整声明，如下所示:

struct B;

//对结构体B进行不完整声明

//结构体A中包含指向结构体B的指针

struct A{

    struct B \*partner;

    //other members;

};

//结构体B中包含指向结构体A的指针，在A声明完后，B也随之进行声明

struct B{

    struct A \*partner;

    //other members;};

#### 二、结构体数组

一个结构体变量中可以存放一组数据（如一个学生的学号，姓名，成绩等数据）。如果有10个学生的数据需要参加运算，显然应该用数组，这就是结构体数组。结构体数组与数据值型数组不同之处在于每个数组元素都一个结构体类型的数据，它们分别包括各个成员（分量）项。

##### 1. 定义结构体数组

和定义结构体变量的方法相仿，只需说明其为数组即可。

　　struct student

　　{

　　　　int num;

　　　　char name[20];

　　　　char sex;

　　　　int age;

　　　　float score;

　　　　char addr[30];

　　};

　　struct student stu[3];

　　以上定义了一个数组 stu，其元素为 struct student 类型数据，数组有 3 个元素。也可以直接定义一个结构体数组。如：

　　struct student

　　{

　　　　int num;

　　　　....

　　}stu[3];

或

　　struct

　　{

　　　　int num;

　　　　...

　　}stu[3];

##### 2. 结构体数组的初始化

与其它类型数组一样，对结构体数组可以初始化如：

　　struct student

　　{

　　　　int mum;

　　　　char name[20];

　　　　char sex;

　　　　int age;

　　　　float score;

　　　　char addr[30];

　　}stu[3] = {{10101,"Li Lin", 'M', 18, 87.5, "103 Beijing Road"},

　　　　　　　　{10101,"Li Lin", 'M', 18, 87.5, "103 Beijing Road"},

　　　　　　　　{10101,"Li Lin", 'M', 18, 87.5, "103 Beijing Road"}};

　　定义数组 stu 时，元素个数可以不指定，即写成以下形式：

　　stu[] = {{...},{...},{...}};

编译时，系统会根据给出初值的结构体常量的个数来确定数组元素的个数。

　　当然，数组的初始化也可以用以下形式：

　　struct student

　　{

　　　　int num;

　　　　...

　　};

　　struct student stu[] = {{...},{...},{...}};

即先声明结构体类型，然后定义数组为该结构体类型，在定义数组时初始化。

##### 3．结构体数组应用举例

　　下面例子说明结构体数组的定义和引用。

#include <stdio.h>

#include <string.h>

#include <stlib.h>

struct person

{

　　char name[20];

　　int count;

}leader[3] = {{"Li", 0},

　　　　　　　{"Zhang", 0},

　　　　　　　{"Fun", 0}};

void main()

{

　　int i, j;

　　char leader\_name[20];

　　for(i = 1; i<= 10;i++)

　　{

　　　　scanf("%s", leader\_name);

　　　　for(j=0;j<3;j++)

　　　　　　if(strcmp(leader\_name, leader[j].name) == 0)

　　　　　　　　leader[j].count ++;

　　}

　　printf("\n");

　　for(i=0;i<3;i++)

　　　　printf("%5s: %d\n", leader[i].name, leader[i].count);

　　system("pause");

}

运行结果如下：

LI

Li

Fun

Zhang

Zhang

Fun

Li

Fun

Zhang

Li

   Li: 3

Zhang: 3

  Fun: 3

#### 三、结构体指针

##### 1、指向结构体的指针变量：

C 语言中“->”是一个整体，它是用于指向结构体，假设我们在程序中定义了一个结构体，然后声明一个指针变量指向这个结构体，那么我们要用指针取出结构体中的数据，就要用到指向运算符“->”。

举例说明：

struct SunLL

{

  int a;

  int b;

  int c;

};

struct SunLL  \* p;               //定义结构体指针

struct SunLL A = {1,2,3};    //定义一个SunLL类型的变量A

int x;   //定义一个变量x

p = &A ; //让p指向A

x = p->a; //等价于x=(\*p).a  (\*p)表示p指向的结构体变量

//这句话的意思就是取出p所指向的结构体中包含的数据项a赋值给x

//由于此时p指向A，因而 p->a == A.a,也就是1

##### 2、指向结构体数组的指针变量：

指向结构体的指针变量也可以指向结构体数组及其元素。

程序如下：

struct SunLL \*p,sun[3];

p = sun;

如果假定sun[0]的地址是1000，指针变量p指向结构体数组sun的首地址，由于size of(struct SunLL)的值是6，每个结构体元素占内存空间6个字节，因此p+1指向地址1006，p+2指向地址1012.

使用指针变量指向结构体变量或结构体数组时，应注意运算符的优先级，在c语言中"()" "[ ]" "->" "."四个优先级相同，具有最高的优先级，其次是”\*“ ”++“ ”--“ ”&“四个相同优先级的运算符。如：++P->a的表达式等价于++(p->a)

(++p)->a先计算++p，p指向sun[1];

P++->a;则表达式等价于(p++)->a;

p->a++的表达式等价于(p->a)++;

#### 四、结构体参数

##### 1.结构体作为函数参数与结构体指针作为函数参数

举例:

struct st

{

int a；

char b；

}；

fun（struct st bc）

{

bc.a+=5;

bc.b='A';

printf("%d,%c\n",bc.a,bc.b);

}

main()

{

struct st bl;

bl.a=3;

bl.b="c";

fun(bl);

printf("%d,%c\n",bl.a,bl.b);

}

运行结果为：8，A        3，c

举例2：

struct st

{

int a；

char b；

}；

fun（struct st \*bp）

{

bp->a+=5;

bp->b='A';

printf("%d,%c\n",bc.a,bc.b);

}

main()

{

struct st bl;

bl.a=3;

bl.b="c";

fun(&bl);

printf("%d,%c\n",bl.a,bl.b);

}

运行结果为：8，A        8，A

总结：

将结构体变量作为函数参数，是将实参值传递给形参，属于值传递。将结构体所有参数传递给形参使用，但本身的值不会改变。将指向结构体变量的指针作为实参进行传递时，是将结构体变量的地址传递给形参，属于地址传递，会改变实际结构体变量的成员值的。

#### 五、任务

##### 学生信息

请定义一个描述学生基本信息的结构，包括姓名，学号，籍贯，身份证号，并定义一个结构体数组。编程：

1) 编写函数 input() , 输入基本信息（3～5 条记录）；

2) 编写函数 print()，输出全体记录信息；

3) 编写函数 search(), 检索一个指定的学生信息并返回, 由主函数打印到屏幕上；

4) 说明，访问结构的时候，什么时候应该用运算符“.”，什么时候应该用运算符“->”。

#include <stdio.h>

#include <string.h>

typedef struct

{

    char name[20];

    char school\_id[20];

    char location[20];

    char identify\_id[20];

} student;

student s[10]; // 结构体数组s

int n;         // 记录条数n

void input()

{

    for (int i = 0; i < n; i++)

    {

        scanf("%s %s %s %s", s[i].name, s[i].school\_id, s[i].location, s[i].identify\_id);

    }

}

void print()

{

    for (int i = 0; i < n; i++)

    {

        printf("%s %s %s %s\n", s[i].name, s[i].school\_id, s[i].location, s[i].identify\_id);

    }

}

// 通过学号查找

student search(char id[])

{

    student res;

    for (int i = 0; i < n; i++)

    {

        if (strcmp(s[i].school\_id, id) == 0)

        {

            res = s[i];

        }

    }

    return res;

}

int main()

{

    char id[10];

    printf("输入记录条数: \n");

    scanf("%d", &n);

    printf("\n输入%d条学生信息: \n", n);

    input(n);

    printf("\n输入你想找的学生学号: \n");

    scanf("%s", id);

    printf("\n你要找的学生信息如下: \n");

    student a = search(id);

    printf("%s %s %s %s\n", a.name, a.school\_id, a.location, a.identify\_id);

    printf("\n接下来显示所有人的信息: \n");

    print();

    return 0;

}

使用.运算符：当你有一个结构体变量的实例时，使用.运算符来访问该结构体变量的成员。

使用->运算符：当你有一个指向结构体的指针时，使用->运算符来访问指针所指向的结构体的成员。

##### 模拟洗牌和发牌的过程

一副扑克有52张牌，分为4种花色（suit）：黑桃（spades）、红桃（hearts） 草花(clubs)、 方块（diamonds）,每种花色又有13张牌面（face）：A,2,3,4,5,6,7,8,9,10,Jack,Queen,King，编程完成洗牌和发牌的过程。

思路：

需要一个结构体存储花色和牌面

需要一个数组存储这52张牌

需要一个随机函数生成一个随机数，知道发哪一张牌。

洗牌就是把产生的随机数对应到保存的数组里。

#include <stdio.h>

#include <time.h>

char suit[4][10] = {

    "spades", "hearts",

    "clubs", "diamonds"};

char face[13][10] = {

    "A", "2", "3", "4", "5",

    "6", "7", "8", "9", "10",

    "Jack", "Queen", "King"};

typedef struct

{

    char s[10];

    char f[10];

    int order;

} card;

void wash\_card(card c[])

{

    int i;

    srand(time(NULL)); // 使用当前时间作为随机数种子

    for (i = 51; i > 0; i--)

    {

        int j = rand() % (i + 1); // 生成一个0到i的随机数

        card temp = c[j];

        c[j] = c[i];

        c[i] = temp;

    }

}

int main()

{

    // 定义结构体数组存储52张牌

    card c[53];

    int k = 0;

    int i, j;

    for (i = 0; i < 4; i++)

    {

        for (j = 0; j < 13; j++)

        {

            strcpy(c[k].s, suit[i]);

            strcpy(c[k].f, face[j]);

            k++;

        }

    }

    // 洗牌

    wash\_card(c);

    // 洗牌后的扑克牌

    printf("\n洗牌后的扑克牌：\n");

    for (i = 0; i < 52; i++)

    {

        printf("%s of %s\n", c[i].f, c[i].s);

    }

    // 发牌   题目没看懂，下面应该是错的

    for (int i = 0; i < 5; i++)

    {

        printf("%s %s\n", c[i].f, c[i].s);

    }

    return 0;

}

##### 一元多项式加法

编写一元多项式加法器，输入两个一元稀疏多项式，然后对它们进行加法操作。

在具体实现上，要求用线性链表形式来存储一个多项式，每个链表的节点包括两个成员

变量，系数和指数（均为整数）。例如

*A*(*x*) = 75 + 30*x* + 95*x*8 + 50*x*9，

*B*(*x*) = 80*x* + 25*x*7 + 90*x*8

*A*(*x*)+ *B*(*x*)=75+110x+ 25*x*7+185*x*8+ 50*x*9

可以用下面的链表表示：

说明：

(1) 每个链表节点都是根据需要动态创建的；

(2) 程序采用多函数形式来实现，至少包括创建链表、打印链表、加法函数等。

(3) 多项式系数可正、可负；指数肯定是非负整数，且按照递增顺序排列

输入格式：第一行是一个整数M，表示第一个多项式的项数。

接下来有M行，每行有两个整数ci和ei，分别表示第i项的系数和指数。

再接下来是输入第二个多项式，方法同第一个多项式输入。

输出格式：输出两个多项式相加的结果。第一行是整数K，表示新多项式的项数。

接下来有K行，每一行为两个整数，分别代表系数和指数。

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node

{

    int ci;

    char ei[10];

    struct Node \*next;

} node;

void print(node \*phead)

{

    node \*cur = phead;

    while (cur != NULL)

    {

        printf("%d %s\n", cur->ci, cur->ei);

        cur = cur->next;

    }

    printf("\n");

}

node \*create\_node(int x1, char x2[10])

{

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

    newNode->ci = x1;

    strcpy(newNode->ei, x2);

    newNode->next = NULL;

    return newNode;

}

void push\_back(node \*\*pphead, int x1, char x2[10])

{

    node \*newNode = create\_node(x1, x2);

    if (\*pphead == NULL)

    {

        \*pphead = newNode;

    }

    else

    {

        node \*tail = \*pphead;

        while (tail->next != NULL)

        {

            tail = tail->next;

        }

        tail->next = newNode;

    }

}

node \*add(node \*phead1, node \*phead2)

{

    node \*head = create\_node(0, " ");

    node \*current = head;

    // 75 1 30 x 95 x8 50 x9

    // 80 x 25 x7 90 x8

    // 75 110x

    while (phead1 && phead2)

    {

        if (strcmp(phead1->ei, phead2->ei) == 0)

        {

            int sum = phead1->ci + phead2->ci;

            if (sum != 0)

            {

                current->next = create\_node(sum, phead1->ei);

                current = current->next;

            }

            phead1 = phead1->next;

            phead2 = phead2->next;

        }

        else if (strcmp(phead1->ei, phead2->ei) < 0)

        {

            current->next = create\_node(phead1->ci, phead1->ei);

            current = current->next;

            phead1 = phead1->next;

        }

        else

        {

            current->next = create\_node(phead2->ci, phead2->ei);

            current = current->next;

            phead2 = phead2->next;

        }

    }

    while (phead1)

    {

        current->next = create\_node(phead1->ci, phead1->ei);

        current = current->next;

        phead1 = phead1->next;

    }

    while (phead2)

    {

        current->next = create\_node(phead2->ci, phead2->ei);

        current = current->next;

        phead2 = phead2->next;

    }

    return head->next;

}

int main()

{

    int m, k;

    scanf("%d", &m);

    node \*phead1 = create\_node(0, " ");

    node \*phead2 = create\_node(0, " ");

    int i;

    for (i = 0; i < m; i++)

    {

        int ci;

        char ei[10];

        scanf("%d %s", &ci, ei);

        push\_back(&phead1, ci, ei);

    }

    scanf("%d", &k);

    for (i = 0; i < k; i++)

    {

        int ci;

        char ei[10];

        scanf("%d %s", &ci, ei);

        push\_back(&phead2, ci, ei);

    }

    node \*result = add(phead1->next, phead2->next);

    print(result);

    return 0;

}

// 4

// 75 1

// 30 x

// 95 x8

// 50 x9

// 3

// 80 x

// 25 x7

// 90 x8

##### 循环淘汰

有N 个同学，编号分别为1,2,3……,N，围成一圈，随便选定一个整数m,让大家按顺时针依次报数，报到m 的同学便会从圈子中退出，从而被淘汰，直到最后剩下一个人。

编写函数实现上述循环淘汰功能。

编写一个程序测试上述函数的功能，要求用户能够任意输入N 与m；程序输入最后剩

下人的编号。

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node

{

    int id;

    struct Node \*next;

} node;

int n, m;

node \*create(int id)

{

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

    newNode->id = id;

    newNode->next = NULL;

    return newNode;

}

void push\_back(node \*phead, int id)

{

    node \*newNode = create(id);

    if (phead->next == NULL)

    {

        phead->next = newNode;

    }

    else

    {

        node \*tail = phead;

        while (tail->next != NULL)

        {

            tail = tail->next;

        }

        tail->next = newNode;

    }

}

int cir\_out(node \*phead)

{

    node \*cur = phead;

    int i = 0;

    while (cur != cur->next)

    {

        // 遍历到第m-1个

        for (i = 0; i < m - 1; i++)

        {

            cur = cur->next;

        }

        node \*temp = cur->next;

        cur->next = temp->next;

        free(temp);

        cur = cur->next;

    }

    int last\_Id = cur->id;

    return last\_Id;

}

int main()

{

    // 头结点之前的前置节点

    node \*head = create(0);

    printf("请输入同学数量n: \n");

    scanf("%d", &n);

    printf("请输入报到数m: \n");

    scanf("%d", &m);

    int i;

    for (i = 1; i <= n; i++)

    {

        push\_back(head, i);

    }

    // 得到循环链表

    node \*cur = head->next;

    while (cur->next != NULL)

    {

        cur = cur->next;

    }

    cur->next = head->next;

    // 循环淘汰

    printf("最后剩下的是同学编号: %d\n", cir\_out(head->next));

    return 0;

}

##### 工资单处理

（1） 编写函数：有两个单向链表,头指针分别为list1、list2，链表中每一结点包含员工

号（员工号为关键字段，不重复）、姓名、工资基本信息，请编一函数，把两个链表拼组成

一个链表，并返回拼组后的新链表，按员工号升序排列。

（2）编写函数：利用指针数组，实现按照工资进行升序排列的功能。返回排序完成的指

针数组

（3）编写一程序，分别输出按员工号排序后的新链表，以及按照工资排序的结果。

假设链表list1 初始化内容为：

{002, name002,3000}, {005, name005,2500}, {003, name003,3500}

链表list2 初始化内容为：

{006, name006,2800}, {004, name004,3700}, {001, name001,3000}, {007,

name007, 3600},

**3个函数在下面一个程序中集成了**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node

{

    char id[20];

    char name[20];

    char salary[20];

    Node \*next;

} node;

// 创建

node \*create(char id[], char name[], char salary[])

{

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

    strcpy(newNode->id, id);

    strcpy(newNode->name, name);

    strcpy(newNode->salary, salary);

    newNode->next = NULL;

    return newNode;

}

// 尾插

void push\_back(node \*\*phead, char id[], char name[], char salary[])

{

    node \*newNode = create(id, name, salary);

    if (\*phead == NULL)

    {

        \*phead = newNode;

    }

    else

    {

        node \*tail = \*phead;

        while (tail->next != NULL)

        {

            tail = tail->next;

        }

        tail->next = newNode;

    }

}

// 冒泡排序

node \*bubbleSort(node \*head)

{

    if (head == NULL || head->next == NULL)

        return head;

    int swapped;

    node \*current;

    node \*tail = NULL;

    do

    {

        swapped = 0;

        current = head;

        while (current->next != tail)

        {

            if (strcmp(current->id, current->next->id) > 0)

            {

                // 交换节点的id、name和salary

                char temp[20];

                strcpy(temp, current->id);

                strcpy(current->id, current->next->id);

                strcpy(current->next->id, temp);

                strcpy(temp, current->name);

                strcpy(current->name, current->next->name);

                strcpy(current->next->name, temp);

                strcpy(temp, current->salary);

                strcpy(current->salary, current->next->salary);

                strcpy(current->next->salary, temp);

                swapped = 1;

            }

            current = current->next;

        }

        tail = current;

    } while (swapped);

    return head;

}

// 合并 成一个新的，返回新的合并后的指针

node \*merge(node \*phead1, node \*phead2)

{

    node \*list3 = create(" ", " ", " ");

    node \*cur3 = list3;

    node \*cur1 = phead1;

    node \*cur2 = phead2;

    while (cur1 != NULL)

    {

        cur3->next = cur1;

        cur1 = cur1->next;

        cur3 = cur3->next;

    }

    while (cur2 != NULL)

    {

        cur3->next = cur2;

        cur2 = cur2->next;

        cur3 = cur3->next;

    }

    cur3->next = NULL;

    return list3;

}

//(2)

// 冒泡排序

void bubbleSort2(node \*\*list, int size)

{

    int i, j;

    node \*temp;

    for (i = 0; i < size - 1; i++)

    {

        for (j = 0; j < size - i - 1; j++)

        {

            if (strcmp(list[j]->salary, list[j + 1]->salary) > 0)

            {

                temp = list[j];

                list[j] = list[j + 1];

                list[j + 1] = temp;

            }

        }

    }

}

void print(node \*list)

{

    while (list != NULL)

    {

        printf("%s %s %s\n", list->id, list->name, list->salary);

        list = list->next;

    }

}

int main()

{

    char id[20], name[20], salary[20];

    node \*list1 = create(" ", " ", " ");

    node \*list2 = create(" ", " ", " ");

    int n, m; // n是list1 m是list2 节点

    scanf("%d%d", &n, &m);

    for (int i = 0; i < n; i++)

    {

        scanf("%s%s%s", id, name, salary);

        push\_back(&list1, id, name, salary);

    }

    for (int i = 0; i < m; i++)

    {

        scanf("%s%s%s", id, name, salary);

        push\_back(&list2, id, name, salary);

    }

    // 合并

    node \*list3 = merge(list1->next, list2->next);

    // 打印链表

    //  while(list1!=NULL)

    //  {

    //      printf("%s %s %s\n",list1->id,list1->name,list1->salary);

    //      list1 = list1->next;

    //  }

    // 按照员工号排序

    node \*sortedList = bubbleSort(list3);

    print(sortedList);

    // 按照工资排序 （指针数组）

    bubbleSort2(&list3, n + m);

    print(sortedList);

    return 0;

}

##### 单向链表

设节点结构：

学号 姓名 后继结点指针

链表结构：

Head +节点

要求程序实现如下功能：

1. 链 表 生 成。

键盘输入学生信息， 建立一个节点按学号递增有序的单链表

A={a1,a2,..,an}，比如包含5～10 条记录； //假设输入的学号依次为 2010002，

2010005，2010009，2010007，2010003，2010000，姓名自己随便定义

2) 节点计数。对单链表A={a1,a2,..,an}编写节点计数函数f，求单链表中的节点个数。

主函数调用节点计数函数f，并将其返回值（整数）显示到屏幕；

3) 对单链表 A={a1,a2,..,an}编写函数fv，将它倒序为A={an,an-1,..,a1}；

4) 编写输出单链表函数 *list*。每次操作（插入一个新节点或者倒序）之后，调用函数

*list*，在屏幕上显示链表的全部记录数据。

5) 编写一个函数 *search*，输入学号，检索链表A，如果指定学号记录存在则返回指向

该节点的指针，主函数打印纪录信息。若学生纪录不存在，则返回空指针，主函数

给出检索失败的信息。

**5个函数在下面一个程序中集成了**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node

{

    int id;

    char name[20];

    struct Node \*next;

} node;

node \*Create(int id, char name[])

{

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

    newNode->id = id;

    strcpy(newNode->name, name);

    newNode->next = NULL;

    return newNode;

}

void Push\_Back(node \*phead, int id, char name[])

{

    node \*newNode = Create(id, name);

    if (phead == NULL)

    {

        phead = newNode;

    }

    else

    {

        node \*tail = phead;

        while (tail->next != NULL)

        {

            tail = tail->next;

        }

        tail->next = newNode;

    }

}

node \*bubbleSortById(node \*head)

{

    if (head == NULL || head->next == NULL)

        return head;

    int swapped;

    node \*current;

    node \*tail = NULL;

    do

    {

        swapped = 0;

        current = head;

        while (current->next != tail)

        {

            if (current->id > current->next->id)

            {

                char temp[20];

                strcpy(temp, current->id);

                strcpy(current->id, current->next->id);

                strcpy(current->next->id, temp);

                strcpy(temp, current->name);

                strcpy(current->name, current->next->name);

                strcpy(current->next->name, temp);

                swapped = 1;

            }

            current = current->next;

        }

        tail = current;

    } while (swapped);

    return head;

}

int count(node \*phead)

{

    int cnt = 0;

    node \*cur = phead;

    while (cur != NULL)

    {

        cur = cur->next;

        cnt++;

    }

    return cnt;

}

node \*reverse(node \*head)

{

    node \*prev = NULL;

    node \*cur = head;

    node \*next = NULL;

    while (cur != NULL)

    {

        next = cur->next;

        cur->next = prev;

        prev = cur;

        cur = next;

    }

    return prev;

}

void list(node \*head)

{

    node \*cur = head;

    while (cur != NULL)

    {

        printf("学号: %d 姓名: %s \n", cur->id, cur->name);

        cur = cur->next;

    }

}

node \*search(node \*head, int id)

{

    node \*cur = head;

    while (cur != NULL)

    {

        if (cur->id == id)

        {

            return cur;

        }

        cur = cur->next;

    }

    return NULL;

}

int main()

{

    int id;

    char name[20];

    int n;

    node \*phead1 = Create();

    printf("请输入数据条数n: \n");

    scanf("%d", &n);

    for (int i = 0; i < n; i++)

    {

        scanf("%d %s", &id, name);

        Push\_Back(phead1, id, name);

    }

    // 按照学号升序排序

    bubbleSortById(phead1);

    printf("节点的数量:\n");

    printf("%d\n", count(phead1->next));

    reverse(phead1);

    list(phead1->next);

    printf("输入你要查找的学号\n");

    scanf("%d", &id);

    node \*student = search(phead1->next, id);

    if (student != NULL)

    {

        printf("%d %s\n", student->id, student->name);

    }

    else

    {

        printf("不存在该学生记录\n");

    }

    return 0;

}