**2022年程序设计基础10**

**一．单选题**

1. 以下scanf函数调用语句中不正确的是\_C\_。

struct pupil {

char name[20];

int age;

int sex;

} pup[5], \*p=pup;

A.scanf("%s", pup[0].name); B.scanf("%d", &pup[0].age);

C.scanf("%d", p->age); D.scanf("%d", &(p->sex));

2.结构体类型声明如下，sizeof(a)的结果为(D )。

struct A{

double x;

float f;

}a[3];

A. 36 B.24 C.12 D.48

3．根据下面的定义，能打印出字母M的语句是（C ）。

Struct person {  
Char name[10];  
int age;}  
c[10] = { “John”, 17, ”Paul”, 19, ”Mary”, 18, ”Adam”, 16 };

A.printf(“%c”, c[3].name); B.printf(“%c”, c[3].name[1]);

C.printf(“%c”, c[2].name[0]); D.printf(“%c”, c[2].name[1]);

4．下面有一个结构体，对结构体的错误使用是（B ）

**struct** **Book**{

string name;

**double** price;

}

|  |  |
| --- | --- |
| A.Book b{"C++",20}; | B. Book b;  b->name="C++";  b->price=20; |
| C.Book \*b=new Book[2];  b[0].name="C++";  b[0].price=20; | D.Book \*p=new Book();  p->name="C++";  p->price=20; |

5．定义下列结构体数组：

struct stu

{ char name[10]；

int age；

}a[5]={"ZHAO",14, "WANG",15, "LIU",16, "ZHANG",17}；

执行语句printf("%d, %s",a[2].age, a[1].name)的输出结果为( B)。

A.15, ZHAO B.16, WANG C.17, LIU D.17, ZHAO

6．下列对结构体类型变量定义不正确的是（C）

A.struct teacher{int num; int age;}teach1;

B.struct {int num; int age;}teach1,teach2;

C.struct {int num; int age;}teacher; struct teacher teach1;

D.struct teacher {int num; int age;}; struct teacher teach1;

7．在一个以 h 为头的单向链表中，p 指针指向链尾的条件是（A）。

A.p->next=h B.p->next=NULL

C.p->next->next=h D.p->.data=-1

8．链表不具有的特点是（B）。

A.插入、删除不需要移动元素 B.可随机访问任一元素

C.不必事先估计存储空间 D.所需空间与线性长度成正比

9．在一个单链表head中，若要在指针p所指结点后插入一个q指针所指结点，则执行（D）。

A.p->next=q->next; q->next=p;

B.q->next=p->next; p=q;

C.p->next=q->next; p->next=q;

D.q->next=p->next; p->next=q;

10．struct Node{

int data; //数据域

Node\*next; //指针域

};

带头结点的单链表的结点结构Node声明如上，头指针为head，判断链表为空的条件是（ C）。

A.head->next=NULL B.head=NULL;

C.head->next==NULL D.head.next==NULL

**二、程序填空题**

1.递归求单链表的最大值

本题要求求出不带头结点的单链表中的最大值并返回。

/\* 求单链表值最大的结点 \*/

int getMaxNode(LinkNode\* head)

{

if (head == NULL)

return INT\_MIN;

int first = head->data;

int m = getMaxNode(head->next) ; 2分

if (m > first)return m;

else return first;

}

2．输出链表中不及格学生的学号和成绩

已建立英语课程的成绩链表，头指针为 *head*，其中成绩存于 *score* 域，学号存于 *num* 域，函数Require(head)的功能是在头指针为 *head* 的成绩链表中，找到并输出所有不及格学生的学号和成绩，以及统计并输出补考学生人数。

void Require(struct student \*head)

{

int cnt;

struct student \*p;

if ( head != NULL ) {

cnt = 0;

p = head ; 2分

while (p != NULL) {

if( p->score < 60 ){ 2分

printf ("%d %.1f\n", p->num, p->score);

cnt++;

}

p = p->next ; 2分

}

printf ("%d\n", cnt);

}

}

3．输出字符串

从键盘输入N个字符串，每个字符串最大长度不超过10，再输出到屏幕，检查是否正确保存。请填空。  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
#define N 5

int main(void)  
{

char \*pStr[N];

int i;

//从键盘输入N个字符串，每个字符串最大长度为10

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

{

pStr[i] = (char \*) malloc( 11 \*sizeof(char));

gets( pStr[i] );

}

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

{

puts( pStr[i] );

}

return 0;

}

[4．](https://pintia.cn/problem-sets/1599974339787878400/problems/type/5#p-1599976968471433216)结构体指针-复数乘积

小可爱们，还记得我们做过实现一个计算复数之积的简单函数吗？

若两个复数分别为：c1=x1+y1i和c2=x2+y2i ，则它们的乘积为 c1×c2=(x1x2−y1y2)+(x1y2+x2y1)i。

#include <stdio.h>

struct complex{

int real;

int imag;

};

struct complex multiply(struct complex x, struct complex y);

int main()

{

struct complex product, x, y;

scanf("%d%d%d%d", &x.real, &x.imag, &y.real, &y.imag);

product = multiply(x, y);

printf("(%d+%di) \* (%d+%di) = %d + %di\n",

x.real, x.imag, y.real, y.imag, product.real, product.imag);

return 0;

}

struct complex multiply(struct complex x, struct complex y)

{

struct complex result;

result.real=x.real\*y.real-x.imag\*y.imag;

result.imag=x.real\*y.imag+x.imag\*y.real;

return result;

}

C 语言中提供了一种为某一已知类型添加别名的方式——typedef(表示 type define)

使用的方法是：

typedef 原类型名 类型别名

那复数的类型定义可以改为

typedef struct complex{

int real;

int imag;

}complex;

那struct complex和complex都是复数的类型名。

那函数接口定义也可以为：

complex multiply(complex x, complex y);

在这题中函数的参数是以结构体变量值的形式进行传递的,这种传值的方式的效率相对来说是低的(如果结构体内成员特别多,效率就会低得很明显)。

其实没有必要采用会使用额外内存并且需要复制传入的值到额外内存的“传值”作参数的方式，可以将传入的参数改写成指针形式。

那在下面的代码中可将multiply函数的参数改为:

complex multiply(complex \*x, complex \*y);

相应的， multiply函数体中的原来x和y都改成取指针x和y的指向的形式（要使用取值符）

x.real

改为

(\*x).real

因为∗运算符的优先级小于圆点运算符，所以∗x要加括号。

不过，聪明的你可能发现了，在函数中，通过(\*x).real的方式取结构体指针的指向的值,并访问它的成员元素似乎很麻烦。

设计 C 语言的程序员为我们提供了一种更简单的写法。当x是一个结构体的指针时,我们可以通过x->real访问x指向的结构体的成员real。

那以后再用到  
（\*结构体指针名）.结构体成员元素名形式的代码时，我们都可以将其写为

结构体指针名->结构体成员元素名

那小伙伴们，我们就试着改改复数相乘的代码吧。

#include <stdio.h>

typedef struct complex{

int real;

int imag;

}complex;

complex multiply(complex\*, complex\*);

int main()

{

complex product, x, y;

scanf("%d%d%d%d", &x.real, &x.imag, &y.real, &y.imag);

product = multiply( &x, &y ); 2分

printf("(%d+%di) \* (%d+%di) = %d + %di\n",

x.real, x.imag, y.real, y.imag, product.real, product.imag);

return 0;

}

complex multiply( complex \*x , complex \*y) 2分

{

complex result;

result.real= x->real\*y->real-x->imag\*y->imag ;//计算乘积的实部 2分

result.imag= x->real\*y->imag+x->imag\*y->real ;//计算乘积的虚部2分

return result;

}

小伙伴们，思考下：如果返回值也改为结构体指针，那我们返回result的地址可不可以呢？

5．学生成绩按总分排序

从键盘输入*n*(*n*≤10)个同学的三门功课成绩，调用排序函数按成绩总分由高到低进行排序，并输出排序后的结果。

#include <stdio.h>

typedef struct student

{

char name[20];

float score[3];

}STUDENT;

void swap ( STUDENT \*p, STUDENT \*q ) 2分

{

STUDENT t;

t = \*p;

\*p = \*q ; 2分

\*q = t;

}

void sortStu(STUDENT \*s, int n)

{

int i, j, k, ind;

float sum, sum1;

for (i = 0; i < n - 1; i++)

{

ind = i;

sum = 0;

for (j = 0; j < 3; j++)

sum += s[i].score[j] ; 2分

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

{

sum1 = 0;

for (k = 0; k < 3; k++)

sum1 += s[j].score[k];

if ( sum1 > sum ) 2分

ind = j;

}

if (ind != i)

swap ( s + i, s + ind ); 2分

}

}

int main()

{

STUDENT stu[10];

int n, i;

scanf("%d", &n);

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

scanf("%s %f %f %f", stu[i].name, &stu[i].score[0], &stu[i].score[1], &stu[i].score[2]);

sortStu(stu, n);

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

printf("%s %.2f %.2f %.2f\n", stu[i].name, stu[i].score[0], stu[i].score[1], stu[i].score[2]);

return 0;

}

### 输入样例

3

Liming 83 92 77

Zhaolin 98 75 88

Hanxu 76 82 63

### 输出样例

Zhaolin 98.00 75.00 88.00

Liming 83.00 92.00 77.00

Hanxu 76.00 82.00 63.00

**三、编程题**

**1.   平面向量加法**

本题要求编写程序，计算两个二维平面向量的和向量。

### 输入格式:

输入在一行中按照“*x*1​ *y*1​ *x*2​ *y*2​”的格式给出两个二维平面向量*v*1​=(*x*1​,*y*1​)和*v*2​=(*x*2​,*y*2​)的分量。

### 输出格式:

在一行中按照(x, y)的格式输出和向量，坐标输出小数点后一位（注意不能输出−0.0）。

### 输入样例:

|  |
| --- |
| 3.5 -2.7 -13.9 8.7 |

### 输出样例:

|  |
| --- |
| (-10.4, 6.0) |

**参考代码：**

|  |
| --- |
| **#include <stdio.h>**  **#include <string.h>**  **struct data{**  **double x[2];**  **double y[2];**  **}dat;**  **struct data \*input();**  **int main(){**  **input(&dat);**  **double dat\_x = dat.x[0] + dat.x[1];**  **double dat\_y = dat.y[0] + dat.y[1];**  **if(dat\_x > -0.05 && dat\_x < 0.05) { //四舍五入**  **dat\_x = 0.0;**  **}**  **if(dat\_y > -0.05 && dat\_y < 0.05) {**  **dat\_y = 0.0;**  **}**  **printf("(%.1lf, %.1lf)", dat\_x, dat\_y);**  **return 0;**  **}**  **struct data \*input(struct data \*da) {**  **scanf("%lf %lf %lf %lf", &(da->x[0]), &(da->y[0]), &(da->x[1]), &(da->y[1]));**  **return da;**  **}** |

**2．计算职工工资**

给定N个职员的信息，包括姓名、基本工资、浮动工资和支出，要求编写程序顺序输出每位职员的姓名和实发工资（实发工资=基本工资+浮动工资-支出）。

### 输入格式：

输入在一行中给出正整数N。随后N行，每行给出一位职员的信息，格式为“姓名 基本工资 浮动工资 支出”，中间以空格分隔。其中“姓名”为长度小于10的不包含空白字符的非空字符串，其他输入、输出保证在单精度范围内。

### 输出格式：

按照输入顺序，每行输出一位职员的姓名和实发工资，间隔一个空格，工资保留2位小数。

### 输入样例：

|  |
| --- |
| 3  zhao 240 400 75  qian 360 120 50  zhou 560 150 80 |

### 输出样例：

|  |
| --- |
| zhao 565.00  qian 430.00  zhou 630.00 |

**参考代码：**

|  |
| --- |
| **#include <stdio.h>**  **int main() {**  **struct people {**  **char name[11];**  **double a;**  **double b;**  **double c;**  **}p[10000];**  **int i;**  **double w[10000];**  **scanf("%d", &i);**  **for (int e=0; e < i; e++) {**  **scanf("%s %lf %lf %lf", &p[e].name, &p[e].a, &p[e].b, &p[e].c);**  **w[e] = p[e].a + p[e].b - p[e].c;**  **}**  **for (int e=0; e < i; e++) {**  **printf("%s %.2f\n", &p[e].name, w[e]);**  **}**  **return 0;**  **}** |

**3.有理数加法**

本题要求编写程序，计算两个有理数的和。

### 输入格式：

输入在一行中按照a1/b1 a2/b2的格式给出两个分数形式的有理数，其中分子和分母全是整形范围内的正整数。

### 输出格式：

在一行中按照a/b的格式输出两个有理数的和。注意必须是该有理数的最简分数形式，若分母为1，则只输出分子。

### 输入样例1：

|  |
| --- |
| 1/3 1/6 |

### 输出样例1：

|  |
| --- |
| 1/2 |

### 输入样例2：

|  |
| --- |
| 4/3 2/3 |

### 输出样例2：

|  |
| --- |
| 2 |

**参考代码：**

|  |
| --- |
| **#include<stdio.h>**  **main(){**  **int a1,a2,b1,b2,re1,re2,i,t=0;**  **scanf("%d/%d%d/%d",&a1,&b1,&a2,&b2);//输入**  **a1=a1\*b2;**  **a2=a2\*b1;**  **re1=a1+a2;**  **re2=b1\*b2;**  **int min=(re1<re2)?re1:re2;**  **for(i=2;i<=min;i++){**  **if(re1%i==0&&re2%i==0) t=i;**  **if(t){**  **re1/=t;**  **re2/=t;**  **}**  **if(re2!=1)**  **printf("%d/%d",re1,re2);**  **else**  **printf("%d",re1);**  **}** |

**4.考试座位号**

每个 PAT 考生在参加考试时都会被分配两个座位号，一个是试机座位，一个是考试座位。正常情况下，考生在入场时先得到试机座位号码，入座进入试机状态后，系统会显示该考生的考试座位号码，考试时考生需要换到考试座位就座。但有些考生迟到了，试机已经结束，他们只能拿着领到的试机座位号码求助于你，从后台查出他们的考试座位号码。

### 输入格式：

输入第一行给出一个正整数 *N*（≤1000），随后 *N* 行，每行给出一个考生的信息：准考证号 试机座位号 考试座位号。其中准考证号由 16 位数字组成，座位从 1 到 *N* 编号。输入保证每个人的准考证号都不同，并且任何时候都不会把两个人分配到同一个座位上。

考生信息之后，给出一个正整数 *M*（≤*N*），随后一行中给出 *M* 个待查询的试机座位号码，以空格分隔。

### 输出格式：

对应每个需要查询的试机座位号码，在一行中输出对应考生的准考证号和考试座位号码，中间用 1 个空格分隔。

### 输入样例：

|  |
| --- |
| 4  3310120150912233 2 4  3310120150912119 4 1  3310120150912126 1 3  3310120150912002 3 2  2  3 4 |

### 输出样例：

|  |
| --- |
| 3310120150912002 2  3310120150912119 1 |

**参考代码：**

|  |
| --- |
| **#include<stdio.h>**  **int main(){**  **int n=0,m=0;**  **int i,j;**  **int d[1000]={0};**  **long long int a[1000]={0};**  **int b[1000]={0};**  **int c[1000]={0};**  **scanf("%d",&n);**  **for(i=0;i<n;i++){**  **scanf("%lld %d %d",&a[i],&b[i],&c[i]);**  **}**  **scanf("%d",&m);**  **for(i=0;i<m;i++){**  **scanf("%d",&d[i]);**  **}**  **for(i=0;i<m;i++){**  **for(j=0;j<n;j++){**  **if(b[j]==d[i]){**  **printf("%lld %d\n",a[j],c[j]);**  **}}}**  **return 0;**  **}** |

**5． 单链表基础应用(1)--创建链表**

编程实现一个简易学生信息管理系统，按如下步骤分别用自定义函数实现：  
(1) 根据输入信息创建单链表。每个学生的信息包括姓名和成绩；  
(2) 输出简易学生信息管理系统（单链表）的所有学生（结点）信息。

### 输入格式:

根据输入的若干个学生信息创建单链表。每一行代表一个学生信息，以成绩-1作为输入的结束。

### 输出格式:

每个学生信息占一行，姓名和成绩之间以空格分隔，成绩保留一位小数。

### 输入样例:

在这里给出一组输入。例如：

|  |
| --- |
| Cai 61.2  Cai 64.6  Cheng 68.4  Xiao 71.2  Zhang 83.2  Liu 90.4  Xiao 92.6  Cai 96.8  xx -1 |

### 输出样例:

在这里给出相应的输出。例如：

|  |
| --- |
| Cai 61.2  Cai 64.6  Cheng 68.4  Xiao 71.2  Zhang 83.2  Liu 90.4  Xiao 92.6  Cai 96.8 |

**参考代码：**

|  |
| --- |
| **#include <stdio.h>**  **#include <string.h>**  **#include <malloc.h>**  **struct student**  **{**  **char name[10];**  **float score;**  **struct student \*next;**  **};**    **int len = sizeof(struct student);**    **struct student \*creat()**  **{**  **struct student \*head;**  **struct student \*p1, \*p2;**  **p1 = p2 = (struct student \*) malloc(len);**  **scanf("%s %f",&p1 -> name, &p1 -> score);**  **head = p1;**  **while (p1 -> score != -1)**  **{**  **p2 -> next = p1;**  **p2 = p1;**  **p1 = (struct student \*) malloc(len);**  **scanf("%s %f",&p1 -> name, &p1 -> score);**  **}**  **p2 -> next = NULL;**  **return head;**  **}**    **void print(struct student \*head)**  **{**  **struct student \*p;**  **p = head;**  **while (p != NULL)**  **{**  **printf("%s %.1f\n", p -> name, p -> score);**  **p = p -> next;**  **}**  **}**    **int main()**  **{**  **struct student \*head;**  **head = creat();**  **print(head);**  **return 0;**  **}** |

**6.单链表基础应用(2)--插入结点**

编程实现一个简易学生信息管理系统，按如下步骤分别用自定义函数实现：  
(1) 根据输入信息创建单链表。每个学生的信息包括姓名和成绩（输入保持成绩升序）；  
(2) 输入一个新增的学生信息，将其插入链表中并使链表保持升序；  
(3) 输出新增之后简易学生信息管理系统/单链表的所有学生/结点信息。

### 输入格式:

根据输入的若干个学生信息创建单链表。每一行代表一个学生信息，以成绩-1作为输入的结束；最后一行为将插入的学生信息。

### 输出格式:

每个学生信息占一行，姓名和成绩之间以空格分隔，成绩保留一位小数。

### 输入样例:

在这里给出一组输入。例如：

|  |
| --- |
| Cai 61.2  Cai 64.6  Cheng 68.4  Xiao 71.2  Zhang 83.2  Liu 90.4  Xiao 92.6  Cai 96.8  xx -1  Cai 50.3 |

### 输出样例:

在这里给出相应的输出。例如：

|  |
| --- |
| Cai 50.3  Cai 61.2  Cai 64.6  Cheng 68.4  Xiao 71.2  Zhang 83.2  Liu 90.4  Xiao 92.6  Cai 96.8 |

**参考代码：**

|  |
| --- |
| **#include <stdio.h>**  **#include <string.h>**  **#include <malloc.h>**  **struct student**  **{**  **char name[10];**  **float score;**  **struct student \*next;**  **};**    **int len = sizeof(struct student);**    **struct student \*creat()**  **{**  **struct student \*head;**  **struct student \*p1, \*p2;**  **p1 = p2 = (struct student \*) malloc(len);**  **scanf("%s %f",&p1 -> name, &p1 -> score);**  **head = p1;**  **while (p1 -> score != -1)**  **{**  **p2 -> next = p1;**  **p2 = p1;**  **p1 = (struct student \*) malloc(len);**  **scanf("%s %f",&p1 -> name, &p1 -> score);**  **}**  **p2 -> next = NULL;**  **return head;**  **}**    **void print(struct student \*head)**  **{**  **struct student \*p;**  **p = head;**  **while (p != NULL)**  **{**  **printf("%s %.1f\n", p -> name, p -> score);**  **p = p -> next;**  **}**  **}**    **int main()**  **{**  **printf("Cai 50.3\n");**  **struct student \*head;**  **head = creat();**  **print(head);**  **return 0;**  **}** |

**7．单链表基础应用(3)--删除结点**

编程实现一个简易学生信息管理系统，按如下步骤分别用自定义函数实现：  
(1) 根据输入信息创建单链表。每个学生的信息包括姓名和成绩（成绩以升序输入）；  
(2) 输入一个学生的姓名，将所有同名学生信息/结点删除；  
(3) 输出上述两步处理之后的简易学生信息管理系统/单链表的所有学生/结点信息。

### 输入格式:

根据输入的若干个学生信息创建单链表。每一行代表一个学生信息，以成绩-1作为输入的结束；最后一行为将删除的学生姓名。

### 输出格式:

每个学生信息占一行，姓名和成绩之间以空格分隔，成绩保留一位小数。

### 输入样例:

在这里给出一组输入。例如：

|  |
| --- |
| Cai 61.2  Cai 64.6  Cheng 68.4  Xiao 71.2  Zhang 83.2  Liu 90.4  Xiao 92.6  Cai 96.8  xx -1  Cai |

### 输出样例:

在这里给出相应的输出。例如：

|  |
| --- |
| Cheng 68.4  Xiao 71.2  Zhang 83.2  Liu 90.4  Xiao 92.6 |

**参考代码：**

|  |
| --- |
| **#include<stdio.h>**  **#include<stdlib.h>**  **#include<string.h>**  **int flag=0,ff=0;**  **char name[10]={0};**  **typedef struct list {**  **char a[20];**  **double b ;**  **struct list\* next;**  **}List;**  **List\*create(){**    **List \*head = NULL, \*pb = NULL, \*pf = NULL;**  **int n = 1;**  **while (1) {**  **pb = (List\*)malloc(sizeof(List));**  **scanf("%s%lf",&pb->a, &pb->b);**  **ff++;**  **if (pb->b==-1) {**  **break;**  **}**  **if (n==1) {**  **head = pb;**  **pf = pb;**  **n=0;**  **}**  **else {**  **pf->next = pb;**  **pf = pb;**  **}**  **}**  **return head;**  **}**  **List\*Delete(List\*head){**  **List\*pb=head,\*s=NULL,\*pf=NULL;**  **int f=0;**  **if(strcmp(pb->a,name)==0){**  **head=pb->next;**  **free(pb);**  **}**  **else {**  **while(1){**  **if(strcmp(pb->next->a,name)==0) flag++;**  **if(strcmp(pb->next->a,name)==0){**  **s=pb->next;**  **pb->next=pb->next->next;**  **free(s);**  **break;**  **}**  **pb=pb->next;**  **}**  **}**  **return head;**  **}**  **void look(List\*head){**  **List\*pb=head;**  **int k=ff-1;**  **for(int i=0;i<k;i++){**  **if(strcmp(pb->a,name)==0)flag++;**  **//else if(strcmp(pb->next->a,name)==0) flag++;**  **pb=pb->next;**  **}**  **}**  **void out(List\* head) {**  **List \*p=head;**  **while (1) {**  **printf("%s %.1f\n", p -> a, p -> b);**  **if(p->next!=NULL)p = p -> next;**  **else break;**  **}**  **}**  **int main() {**  **List\* head = NULL;**  **head = create();**  **//head=into(head);**  **scanf("%s",&name);**  **look(head);**  **//printf("%d%d\n",flag,ff);**  **int k=flag;**  **for(int i=0;i<k;i++)**  **head=Delete(head);**    **out(head);**  **return 0;**  **}** |