程设笔试拾遗

By wln, 2020.12

1. 关于算数

   （int）d=2*4.5+9=18(先算完再舍去小数)

   （int）x=8.75+8.25=17

   （int）x=5/2+5/2=4 (每一项算完并舍去小数得出结论后，再加起来)

   （int）x=5.0/2.0+5.0/2.0=5　(5.0/2.0=2.5)

   (int)　x=5.0/2.0+5.0/2=5　(5.0/2=2.5)

   (float)　x=1.0+1/2+1/3=1.000000

2. 三目运算符 z=（x>y）？x：y　（若 x>y 则 z=x，否则 z=y）

   多重：从右向左如 a>b?a:c>d?c:d 等价于　　a>b?a:（c>d?c:d）

3. Float 型数字：小数点后保留六位 %7.2f：总宽度为 7（包括小数点、e、+），小数保留 2

   位，超额则自动补齐

```
int main()
{ int i;
double s=1.0;
for(i=2;i<=4;i=i+2) s=s+1/i; /*注意算式中并非float，1/i（1/2、1/4）的结果都是0*/
printf("%f\n",s); }
```

   输出结果：1.000000

4. (1)字符串输入输出：如果用 scanf 来读入字符串的话，那么一旦遇到空格或者回车就自动终止了，后边再输入的东西的读不进这个字符数组。如果要输入包含空格的字符串的话，应该用 gets（）

（1）

```
int main()
 { static char str1[]= "how do you do";
     char str2[10],*p1,*p2;
    p1=str1; p2=str2;
     scanf("%s",p2);
     printf("%s",p2);
     printf("%s\n",p1);
   }
```

输入：HOW DO YOU DO

输出：HOWhow do you do　（事实上只有 HOW 被读入 str2）

（2）　（scanf,gets 都会优先读入上次缓冲区遗留的东西）

```
#include<stdio.h>
int main()
{char str1[]="how do you do";
 char str2[20],str3[20],*p1,*p2,*p3;
  p1=str1;p2=str2;p3=str3;
  scanf("%s",p2);/*实际上只读入了HOW*/
  gets(p3);/*gets会读入上次输入剩下的东西直到回车，也就是（空格）DO YOU DO*/
```

```
printf("%s",p2);
printf("%s",p1);
printf("%s\n",p3);
return 0;
}
```
输入：HOW DO YOU DO

输出：.HOW how do you do DO YOU DO

（3）关于数组的指针

(3. 1)

```
int main()
{char  b[ ]="abcde";
printf("%s",&b[2]); /*既然输出的是字符串，那么&b[2]就是首地址,输出的字符串是它以及后边的字符*/
}
```
输出：cde

（3.2）

```
#include<stdio.h>
int main()
{ char s[]="9876",*p;
for (p=s;p<s+2;p++)  //第一次是p指向s[0]，然后输出p及以后字符串9876，第二次p指向s[1],然后输出此时p及以后字符串876
printf("%s",p);
}
```
输出：9876876

5. 转义字符：\a \b \f \n \t \\ ..... （算是一个字符）

Strlen:不算结束符\0，但算空格

Sizeof：比 strlen 长一位，算上\0

（1）

```
#include <stdio.h>
#include <string.h>
int main( ){ char s[ ]="\ta(空格)\\bc";
printf("%d,%d\n",sizeof(s), strlen(s));}
```
输出：7,6

（2）

```
#include<stdio.h>
#include<string.h>
int main()
{char st[21]="hahaa\0\t\'\\";//到\0的时候，字符串就终止了，字符串的内容就是之前的hahaa，strlen也只有5
  printf("%d,%d,%s\n",strlen(st),sizeof(st),st);//但是sizeof（st）一开始就规定好21了，无论如何数组大小不会变
  return 0;
}
```

输出：5 21 hahaa

6.关于外部函数里边static类型变量：会延续上次调用的时候最后的值

（1）

```
# include <stdio.h>
int a=4;
int f(int n)
{
    static int a=5;
int t = 0;
if (n%2)
 { int a=6; t += a++; } /*注意if、else内部定义的a优先在这个if里边作用，但是跟外边的a无关*/
else
 { int a=7; t += a++; }
return t + a++;
}
int main( )
{ int s=a, i=0;
for (; i<2;i++) s += f(i);
    printf("%d\n", s);}
```

实际算式：

S=4

f(0):t=0+6=6 f(0)=6+5=11

f(1):t=0+7=7 f(1)=7+6=13 （注意算f（0）的最后把static int a改成了5+1=6，这个6会保留到下一次调用函数的时候，所以算f（1）的时候a已经是6了）

输出：4+11+13=28

（若把static去掉则算f（1）的时候a还是带入5，最后输出27）

（2）

```
#include<stdio.h>
void f(){
static int a=3;
 a++;
printf("%d ",a);
}
int main() {
   f();
   f();
}
```

输出：4 5

（第二次调用f（）的时候a是以4为起点的）

7.

函数传递问题;

（1. 1）

```
#include<stdio.h>
int main()
{ int a=3,b=2,c=1,f(int,int,int);
f(a,b,c);
printf("c=%d\n",c);
}
int f(int a,int b,int c)
{
    c=a+b;
printf("c=%d\n",c);
return c;
    }
```

输出：c=5（函数里边自己输出的）

c=1（主函数里边定义的c还是没变，因为是数值结合，没有传地址）

（1.2）

```
#include<stdio.h>
int f(int x,int y)
{int c;
c=x+y;
return c;
}
int main()
{ int a=3,b=2;
printf("%d\n",f(a,b));
    }
```

输出：5

（2）

传递一个struct变量，与传递数组不同！它和传递一个普通变量名一样，不会改变主函数。

```
#include<stdio.h>
struct p
{
 int x[2];
 char c;
};
void f1(struct p b)/*这一步实际上对主函数不构成任何影响*/
  {b.x[0]=30;b.x[1]=40;b.c='y';}
void f2(int x[2])/*这一步传的是数组名x，等同于传指针，会改变主函数*/
```

```
    {x[0]=50;x[1]=60;}
int main()
{
 struct p a={10,20,'x'};
      f1(a);/*这一步实际上对主函数不构成任何影响*/
      printf("%d,%d,%c\n",a.x[0],a.x[1],a.c);
      f2(a.x);/*这一步传的是数组名x，等同于传指针，会改变主函数*/
      printf("%d,%d,%c\n",a.x[0],a.x[1],a.c);
return 0;
}
```

输出：10,20,x
50,60,x

(3)

```
#include <stdio.h>
#include <string.h>
typedef struct {
 char name[9];
char sex;
float score[2];} STU;
STU f(STU a)   /*未把结构体变量c的地址传过来，因此这个函数刚刚作用后实际上主函数里的c没变*/
{
 STU b={"Zhao", 'm', 85.0, 90.0};
int i;
strcpy(a.name, b.name);/*strcpy:把后边的复制到前边*/
a.sex = b.sex;
 for (i=0; i<2; i++)
      score[i] = b.score[i];
      return a;}
int main( )
{
 STU c={"Qian", 'f', 95.0, 92.0};
c=f(c); //这一步是关键！最终输出的c是函数中的内容，关键在于这一步赋值，也就是把函数返回值赋给c。如果仅仅摆一个f（c）再输出c，那么c就保留初始值Qian...
 printf("%s,%c,%2.0f,%2.0f\n", c.name, c.sex, c.score[0], c.score[1]);
}
```

输出：Zhao,m,85,90 （最终赋值那一步决定了c的改变）

(4)

```
 #include <stdio.h>
typedef struct { char name[9]; char sex; float score[2];} STU;
void f(STU *a){
 STU b={"Zhao", 'm', 85.0, 90.0},*p=&b;
 a=p;            //这里a=p只是改变了指针的值，这是无效的，要是真想改就得传指针的指针
```

```
    a->sex = 'm';
    a->score[0] = 66.0;} //甚至由于a=p使得a指跑了，这两步也无效。但如果去掉a=p的话，sex和score就真的改变了
int main( )
{ STU c={"Qian", 'f', 95.0, 92.0}, *d=&c; f(d);
printf("%s,%c,%2.0f,%2.0f\n", d->name, c.sex, c.score[0], c.score[1]);
   }
```

输出：Qian,f,95,92　（实际上函数没起作用）

8.指针以及指针的指针问题

（1）

```
#include<stdio.h>
int main()
{
 int**k,a[6]={3,15,17,19,11,5},z;
 int*p=a;
 k=&p;//k一直指向p当前的位置
 z=*p;//z被赋值为p现在所指的内容a[0],即给z赋值3
 p=p+1;//p指向a[1]
 z=z+**k;//k一直跟着p当前位置，即**k=*（p+1）=a[1]=15
 printf("%d\n",z);
 return 0;
   }
```

输出：18

（2）

```
# include <stdio.h>
void swap(int **a, int **b)//传过来指针的指针
{ int *t;
t=*a;
*a=*b;
*b=t;//*a、*b实际上是指针的值，由于传过来的是指针的指针，因此这里改变指针值会影响主函数
}
int main( )
{ int i=3,j=5, *p=&i, *q=&j;
swap(&p, &q);
printf("%d,%d,%d,%d\n", i, j, *p, *q);
}
```

输出：3,5,5,3　（原i，j的值没变，指针值交换）

9.关于自增自减问题

（关于逗号表达式：如（part1，part2，part3），part1和part2都已经执行了一遍，但表达式最终输出的值是part3）

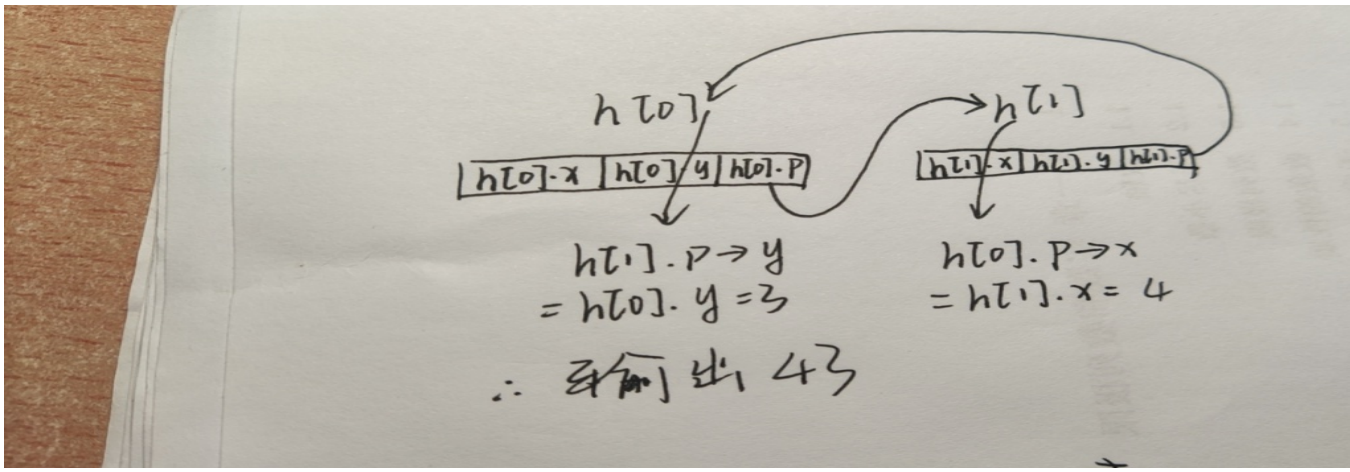
```
#include<stdio.h>
int f(int a,int b)
  {int c;c=a+b;return c;}
int main()
{int x=7,y=8,z=9;
  printf("%d\n",f((x++,y,x+y),z--));/*x++执行完（x变成8）后才会进入下一个part，这里++x和x++效果没有区别，因为这一步x没有参与任何表达式，并且最终算x+y的时候这个x已经是自增后的
                                而z--是带入函数的，所以是先带入函数（z=9）再自减（z=8）*/

  printf("%d %d %d\n",x,y,z);
    return 0;}
```

输出：25

8 8 8

10关于链表（对于怎么写链表有些启发）.

```
#include<stdio.h>
struct HAP
{int x;int y;
struct HAP*p;
}h[2];
int main()
{h[0].x=3;h[0].y=3;
 h[1].x=4;h[1].y=4;
  h[0].p=&h[1];
  h[1].p=h;      printf("%d%d\n",h[0].p->x,h[1].p->y);}
```



11. 二维数组问题：（1）（大小问题）a[3][4]则：sizeof（a）=48　（3*4*4）

sizeof（a[0]）=16　(4*4)　(第0行的大小)

sizeof（a+1）=4　　（a+1指的是第1行的首地址）

```
#include<stdio.h>
#include<malloc.h>
void f(int*p,int(*a)[3],int n)
{int i,j;
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
    {*p=a[i][j]+1;
     p++;
    }
 }
int main()
{int*p,a[3][3]={{1,3,5},{2,4,6},{7,8,9}};
   p=(int*)malloc(sizeof(int)*100);
   f(p,a+1,sizeof(a)/sizeof(a[0]));//由于这里传过去的是a+1，因此实际上指针p所指的只是第2,3行{{2,4,6}，{7,8,9}}
   printf("%d%d\n",p[2],p[5]);
return 0;
}
```

输出结果：7 10　（6+1,9+1）

（2）

（a和p本质上是一个意思）

*(*p+i) 其实就是p[0][i]，并且p[0][i]可以通过i遍历所有元素（只要i足够大）

“*p”到达的是行层面，“**p”到达的是元素层面（如*p实际上指的第0行（的首地址），*（p+i）是第i行（的首地址），*（*p+i）指的是第0行第i个元素，*（*（p+i）+j）指的是第i行第j个元素），

Eg：*(*(a+i)+i))其实就是a[i][i]

例1：

```
#include<stdio.h>
int f(int(*p)[4],int n)
{int i;
 int m;
 m=**p;
for(i=1;i<n;i++)
    if(*(*p+i)>m) m=*(*p+i);//其实是在找最大元素
return m;
}
int main()
{int a[3][4]={1,2,3,4,5,6,7,6,5,4,3,2},n=3*4;
   printf("%d\n",f(a,n));
return 0;
}
```

输出结果：7

例2

```
#include<stdio.h>
int main()
{char a[4][5]={"DCBA","EFGH","IJKL","MNOP"};
  int i;
  for(i=0;i<4;i++)
       printf("%c",*(*(a+i)+i));
  return 0;
}
```

输出：DFKP//输出的是a[i][i]

12.复合嵌套问题

（1）

```
#include<stdio.h>
int main()
{int x;
  x=10;
  {x=20;  // 注意这里并没有重新定义x，里边的x就是外边的x，加上{}也无所谓
    printf("%d\n",x);
  }
  printf("%d\n",x);
}
```

输出：20

20

（2）

```
#include<stdio.h>
int main()
{int x;
  x=10;
  {int x=20; //重新定义x后，里边的x跟外边的x不是一个，里边的赋值无法带到外边
    printf("%d\n",x);
  }
  printf("%d\n",x);
}
```

输出：20

10

13.文件

(1)

```
# include   <stdio.h>
void main( )
{ FILE *fp;
int k,  n,  a[6]={1,2,3,4,5,6};
```

```
fp = fopen("d2.dat", "w");
fprintf(fp, "%d%d%d\n", a[0],a[1],a[2]);
fprintf(fp, "%d%d%d\n", a[3],a[4],a[5]);
fclose(fp);
fp = fopen("d2.dat", "r");
fscanf(fp, "%d%d", &k, &n);
printf("%d, %d\n", k, n);
fclose(fp);
}
```

输出：123,456 （写入文件后，文件里边是123\n456\n,读出的时候读两个整数，则读123,456，把中间的\n换成空格也能达到这个效果）

(2)

```
# include   <stdio.h>
void main( )
{   FILE *fp;
int i, a[6]={1,2,3,4,5,6};
fp = fopen("d3.dat", "w+b");
fwrite(a, sizeof(int), 6, fp);//每个数据项大小为sizeof（int）,并且写入6个数据项 即a[0]~a[5]
fseek(fp, sizeof(int)*3, SEEK_SET);//从开始处（seek-set）把文件指针往后挪三位到文件中4的位置
fread(a, sizeof(int), 3, fp);//从当前文件指针位置开始读三个数据项（4、5、6）到a（也就是a[]首地址a[0] 处）即把a[0]a[1]a[2]读成4、5、6
fclose(fp);
for (i=0;i<6; i++)
printf("%d,", a[i]);
}
```

输出结果：4、5、6、4、5、6

13.指针数组（概念复习）

```
#include <stdio.h>
#include <string.h>
void fun(char *s[], int n)
{ char *t;
int i,j;
for (i=0; i<n-1; i++)
for (j=i+1; j<n; j++)
if (strcmp(s[i], s[j]) > 0)
{ t = s[i];
s[i] = s[j]; s[j] = t; }
}
int main( )
{ char *ss[]={"bcc","bbcc","xy", "aaaacc", "aabcc"};//这个指针数组的意思是数组中的每个
```

元素都是指针,每个指针指向一个字符串，如ss[0]指向“bcc”

```
fun(ss, 5);
printf("%s,%s\n", ss[0],ss[4]);
}
```

输出：aaaacc，xy　（选择排序，从小到大）

14.易错

```
#include<stdio.h>
void fun(int a[], int n)
{ int t,i,j;
for (i=0; i<n-1;i++)
for (j=i+1; j<n; j++)
{  if (a[i]<a[j])
{   t=a[i];a[i]=a[j];a[j]=t;
}
}
}
int main( )
{ int c[10]={1,2,3,4,5,6,7,8,9,0},i;
fun(c, 6);//注意是前六个从大的到小排序!
for (i=0;i<10;
i++)
printf("%d,", c[i]);
printf("\n");
}
```

输出：6,5,4,3,2,1,7,8,9,0,