C 语言笔试复习 tip

by 蔡蔡

1.转义字符

'\ddd' 为八进制的 ascii 字符。 最大'\377' （不可出现 8）

'\xdd' 为十六进制的 ascii 字符,最大'\xff'

'\+a、b、f、n、r、t'

eg."\t234\017\b"的 strlen 值为 6

"\t234\018\b" 的 strlen 值为 7

"\t234\088\b" 的 strlen 值为 4

2.复合语句

将单语句替换成用大括号括起来的一系列语句。括号内定义的变量对括号外不起作用，内层执行结果不带入外层，若重复定义则局部优先。

eg:

（1）main( ) { int x, y; x=10; y=100; { x=20; printf("y=%d\n", y); printf("x=%d\n", x); } printf("x=%d\n", x); 执行结果为: 100 20 20

（2）main( ) { int x, y; x=10; y=100; { int x; x=20; printf("y=%d\n", y); printf("x=%d\n", x); } printf("x=%d\n", x); 执行结果为: 100 20 10

3.条件运算符

条件表达式的一般形式为: 表达式 1？表达式 2：表达式 3

从右至左结合，优先级>赋值运算符；优先级<算术和关系运算符

eg: d = a>b ? (a>c ? a : c) : (b>c? b : c); 也可以写为: d = a>b ? a>c ? a : c : b>c? b : c;

4.关于 switch

在执行 switch 结构时，当执行完某 case 的语句后，如果没有遇到终止语句，将顺序继续执行后面 case 或 default 的语句，直到全部执行完后面的语句或遇到 break 语句才退出整个 switch 结构的执行。

5.关于输出

%d 十进制 %o 八进制 %x 十六进制 %hd 短整型 %ld 长整型

%md 输出数据长度为 m 少补（左补空格）多不删

%-md 左对齐右补格 %0md 或%.md 右对齐左补零

%f 小数 %e 指数 %g 系统决定用前面两种之一

%m.nf m 为整个数据宽度（包含小数点，少补空格多不删），n 为小数位数（若无则默认六位；不足补零）

%m.ne m 的长度包含"e/e+" eg: printf(" %13 . 2 e \n", f) []1.23e+002

%.f 只输出整数部分

%m.ng m 为数据宽度 n 为有效数字位数（默认六位）

%0m,nf 等与整型同理

%mc 字符前加 m-1 个空格

%ms 长度为 m 少（左）补（空格）多不删
%.ms 截取前 m 个字符

%% 输出%符号

6.长短输出互换

short int：-32768~32767　0~ 65535

eg：main( )
{ int xx, yy, zz;
xx=1; yy=-65535; zz=1;
printf("xx=%ld, yy=%ld, zz=%ld\n", xx, yy, zz);
printf("xx=%hd, yy=%hd, zz=%hd\n", xx, yy, zz); }
该程序运行的结果如下：　xx=1, yy=-65535, zz=1　xx=1, yy=1, zz=1 xx=1

7.关于四舍五入

强制类型转化不四舍五入，直接去尾
%m.nf、%g 等截取小数会四舍五入

8.printf 与自加运算符

int k=8; printf("%d,%d\n" , k, ++k); 输出结果是 9,9。
int k=8;printf("%d,%d\n" , ++k, ++k);输出结果是 10,10 。
int k=8;printf("%d,%d\n" , k++, k++); 输出结果是 9,8

9.%m.nf 中的 m 和 n 可以是变量

eg: printf("%*d", m, i); printf("%*.*f" , m, n, f);
其中 m 和 n 不输出

10.关于 scanf 输入

如果各格式说明符之间包含其他字符，则在输入数据时， 应输入与这些字符相同的字符作为间隔。

当整型或字符型格式说明符中有宽度说明时，按宽度说明截取数据。

如果输入缓冲区中已经有数据（上一个输入函数读剩下的），则依次按照“格式控制”中的格式说明符从输入缓冲区中取出数据。遇到回车或换行字符，则将输入缓冲区清空。而且只读符合格式说明符类型的数据，否则跳过（事实上会输出乱码）

eg: while(getchar( )!='\n');用于读空缓存区

数字之间不需要间隔符，若插入了间隔符，系统也将按指定的宽度来读取数据， 从而会引起输入混乱。除非数字一开始就是“粘联”在一起，否则不提倡指定输入数据所占的宽度

可以在%和格式字符之间加入“*”号，作用是跳过对应的输入数据。

eg：scanf("%d%*d%d%d", &x, &y, &z);

printf("%d %d %d\n", x, y, z);
若是输入：12 34 56 78 则输出是：12 56 78

11.关于赋值运算符

从右往左结合 eg：x=y=4+5 等价于 x=(y=4+5) a %= b+3; 等价于： a = a % (b+3)

12.负数求余

(-12)%5 的值为-2， 12%(-5)的值为 2 ，(-12)%(-5)的值为-2

13.关于运算

当前运算符两侧的数据项的类型，若其中一个为实型，而另一个不是，则把不是实型的转换为实型后进行运算。若运算符两侧的数据项的类型一致，则不做任何类型转换。形如 1,2 为整型，1.0,2.0 为浮点型。

eg： int n=5;
a=(double)(1/n);b=double(1)/n;c=1/(double)n;
则 a=0,b=c=0.200000

14.关于关系运算符

有连续几个&&（||）的表达式中，从左向右，只要有一个关系运算结果为假（真），整个结果将为假（真），不再执行后面的关系运算。

eg：int a=5, b=4, c=3, d=3; 当执行语句： if (b>a && (d=c>a)) 时，由于 b>a 为假，故不再执行(d=c>a) ， 所以此语句执行完 d 仍为 3，而不是 0

15.sizeof 的迷惑操作

sizeof('\n')为 4 ； sizeof((char)'\n')为 1
('\n'的 asci 码为整型)

16.关于逗号运算符

子表达式 1，子表达式 2，…，子表达式 n 求解过程是： 按从左到右的顺序分别计算各子表达式的值，其中最右边的子表达式 n 的值就是整个逗号表达式的值。

eg：① x=3+4, 5+7, 10*4 ② x=(3+4, 5+7, 10*4) ① x 值为 7， ② x 值为 40
t=a, a=b, b=t; 它等价于 t=a; a=b; b=t
eg： (a=2*4, a*5), a-3 最后得到整个逗号表达式的值为 5。

17.关于宏定义

(#define 待替换字符串 字符串）程序中用双引号(")括起来的字符串，即使与定义中需要替换的字符串相同，也不进行替换。

变量的字符串化（stringizing）。即 #标识符 -> "标识符"(如#define pr(x) printf("%s=%d\n", #x, x) 若 int a=1 则 pr(a)的结果为 a=1)

字符串连接（token-pasting），即 <标识符>##<宏变量> -> <标识符><宏变量>
(如#define mp(x) printf("%d", a##x) mp(1)相当于打印 a1)

18.关于#ifdef、#ifndef、#if

#ifdef 标识符 程序段 1 #else 程序段 2 #endif 如果“标识符”已经定义过（一般是指用#define 命令定义过），则程序段 1 参加编译，而程序段 2 不参加编译

#ifndef 与上面相反

#if 标识符 检测的是标识符的值是否为真（非零）

19.关于变量

局部静态变量时若不赋初值，则在编译时将自动赋初值 0 或空字符。

全局（外部）变量如果不在文件开头定义全局变量，则只限于在定义点到文件结束范围内的函数使用该变量。若想在定义点前的函数中使用，则需在函数中用 extern 声明该变量在外部定义过（可省略类型名）。

外部变量均为静态变量，extern 与 static 的区别在于对不对其他文件起作用。

20.关于数组的沙雕问题

从 0 开始编号!!!!!!!!

在被调用函数中改变了形参数组元素的值，就改变了实参数组元素的值。实参数组与形参数组的大小可以一致也可以不一致，调用时只将实参数组的首地址传给形参数组。

可以只给一维数组的前若干个元素赋初值，此时后面的元素均将自动赋初值 0（或‘\0’，后同）。如果是全局/局部静态数组则自动赋初值 0。

在分行给二维数组赋初值时，对于每一行都可以只对前几个元素赋初值，后面未赋初值的元素系统将自动赋初值 0；并且，还可以只对前几行元素赋初值，后面行自动为 0。

21.关于字符串

一定不要忘记末尾的‘\0’

（ eg.char b[ ]="china"; 则 sizeof(b)=6）

在用格式说明符%s(scanf)为字符型数组输入数据时，遇空格回车 tab 停止，字符串后自动加‘\0’，但空格回车 tab 保留在缓存区。输出则不受空格等影响，遇‘\0’停止

gets 可以读空格和制表符（tab），遇回车停止，可读取回车并将其转化成‘\0’，回车不保留在缓存区

puts 可识别转义符（char str[ ]="china\nbeijing"; puts(str);结果换行）

strncat 可以自动加‘\0’,strncpy 不行；其中 n 的大小可以包含‘\0’

(eg. char s1[20];char s2[ ]="cdef";strncpy(s1, s2, 5);printf("%s\n", s1);
则结果自带\0)

Strcpy会完全覆盖原字符串（与两个字符串大小关系无关）

Strcmp 左大于右返回1，小于返回-1，相等返回0

strlwr(字符串）将字符串中大写字母转换成小写字母 strupr(字符串）将字符串中小写字母转换成大写字母

22.关于指针大小

任何类型指针大小都为4字节

Eg. int *a[10]; short (*s)[100] sizeof(a)=40,sizeof(s)=4 (s 为行指针) *r++ 和 (*r)++ 的区别

23. *r++ 和 (*r)++等的区别

n = *r++ 是n先取r所指单元的内容，然后指针r加1（指向下一个单元），不改变指针所指

单元内容。*(r++) 与 *r++ 效果相同。
m = (*r)++ 是m先取r所指单元的内容，然后将r指针所指单元的内容加1
n = *++r 是指针r先加1，n取r所指单元a[1]的内容。*(++r) 与 *++r 效果相同。
n = ++*r 先将r指针所指单元的内容先加1，然后n取r所指单元的内容。++(*r) 与 ++*r 效果相同

24.关于指针的等价与不同

*的优先级高于算术运算符*p+1 = (*p)+1 != *(p+1); 但与自加自减是同级，按从右往左的先后顺序

a[i] 等价于 *(a+i) 等价于 *(i+a) 等价于 i[a]

&a[i] 等价于 &*(a+i) 等价于 a+i

a+i表示的是二维数组中第i行（即下标为i的行）中第一个元素的首地址（即&a[i][0]）a[0]+i表示的是二维数 组中第i个元素（以行为主排列）的首地址。

25.关于strstr()

确认字符串2是否在字符串1中出现过，是则返回第一次出现的位置，否则返回NULL(返回值是字符串1中字符串2的首地址，即一个指针)

26.main 形参

main(int argc, char *argv[ ]) { … } 其中: argc的值是命令行参数个数+1, argv是字符型指针数组，指向每一个参数字符串

eg. 输出命令行中除命令符外以空格分隔的所有字符串

```
main(int argc, char *argv[ ])
{ int k;
for (k=1; k<=argc-1; k++)
printf("%s\n", argv[k]); }
```

输入 file new good China asdfg 后 file 不输出

27.关于结构体

“.”为成员运算符，它的执行优先级是C语言中最高的

28. 结构体的大小与#pragma中pack

若不用pack声明，结构体总是默认以结构体中最长的成员类型的字节数对齐。sizeof（结构体）的值不为各成员字节数之和，而应该按对齐后的形式算

eg. struct student {char name[10]; long sno; char gender; float score[4]; };
sizeof=4*(3+1+1+4)=36

结构体成员存放时，总是以成员数据单元大小的整倍数作为起始位置。

在寻找最宽基本类型成员时，应当包括复合类型成员的子成员，而不是把复合成员看成是一个整体;在确定复合类型成员的偏移位置时则是将复合类型作为整体看待。

eg.若将float换成double则size为56（sno放在name[9]后空的一段里（四个一段））

若再将sno与gender的顺序互换则size为48（gender在name[9]的后的空字节里，sno位置与上同）

共用体与结构体类似，按最大组员取pack大小，所占内存单元的大小 就是几个数据项中长度最大的一个。

29.共用体

共用体各成员的值同时改变，保持一致

```
Eg. union EXAMPLE
 { struct { int x, y; } in;
   int a, b;
   } e;
main( )
{ e.a=1; e.b=2;
e.in.x=e.a*e.b; e.in.y=e.a+e.b;
printf("%d,%d\n",e.in.x,e.in.y); }
```

结果为4,8

30.枚举类型

常量值从0开始

enum week {sun, mon, tue, wed, thu, fri, sat}; 定义了枚举类型week，在这个类型中，共有七个常量枚举元素，分别为: sun(值为0), mon(值为1), tue(值为2)……

但C语言还允许在对枚举类型定义时 显式地给出各枚举元素的值。

31.关于拷贝

浅复制(浅拷贝) 形如a=b;结构体整体赋值或参数传递（若结构体内部有指针，则指针指向相同内存）

深拷贝 在浅拷贝基础上给a中指针单独开辟内存空间，再用strcpy等复制内容

32.关于文件

打开文件时w（b）和w+（b）都会删除原有文件

fgetc fputc 从文件读取/向文件写入字符

fgets fputs 从文件读取(末尾自动加\0)/向文件写入字符串（不自动写入\0和换行）

在执行fgets( )的过程中，如果在未读满n-1 个字符时，就已经读到一个换行符或文件结束标志EOF， 则将结束本次读操作，此时读入的字符就不够n-1个，但包括回车符也被读入到字符串string中，且回车符算入字符串总长度。

fscanf fprintf 按指定格式从文件读取/写入(eg.11.4按%f写入后变成11.400000)

fprintf要考虑输出格式，如果不在输出格式中加空格等分隔符，数据会粘连在一起，系统按小数点或空格乱分段

eg.用fprintf分别写入123，但用fscanf读入的时候把123视为一个数字

fread fwrite (二进制)读取/写入

eg.fread(读入数据的存放地址，一个数据项的字节数，读取数据项数，文件指针)

EOF为文件的结束标志，值为-1

feof(fp) 若文件结束返回值为非0；否则返回值为0。慢半拍

```
eg. while(!feof(fin)){fgets(a,80,fin);fputs(a,fout);}
```

文件最后一行会被输出两次；原因：上面循环最后一次读不成功，但随后的fputs把上次读入的字符串又写了一次。

rewind 移至开头

fseek(FILE *指针, long 移动长度, int 参考位置) SEEK_SET 或 0 表示从文件首（开头）

SEEK_CUR 或 1 表示从当前读写的位置 SEEK_END 或 2 表示从文件尾

ftell( )返回当前指针位置

若文件打开方式是"a+b",在执行读操作后（即使指针已到文件尾），如果想成功写入，必须需要先用fseek或rewind移动文件指针（位置不限，可以原地不动；由于使用a打开，一定是从文件末尾开始写，不会从指针位置开始写）！否则fwrite无法正常写入，只能使指针移动对应长度，且到文件末尾为止。

fflush的功能是: 清空文件的输入输出缓冲区，使文件的输入输出缓冲区中的内容立刻输出到实际的文件中。若成功执行，则函数返回值为0，若出错，则函数返回值为-1。

33.其他tips

长整型只占四个字节

一定要注意看循环结构/条件结构的分号在哪里，有没有大括号

注意输出格式的逗号，空格，换行符

fgets（b,3,fp）只能读到两个字符!!!!!!!

一定要注意数据类型（精确度）

指针相减=(地址1-地址2)/sizeof(类型)=相隔单元数

文件所占字节数=总字符数+行数*2（因为有回车符和换行符）