4.综合题（18道）

4.1下面代码输出是几?

```
int main()
{
    int j = 2;
    int i = 1;
    if(i = 1) j = 3;
    if(i = 2) j = 5;
    printf("%d", j);
}
```

答案：输出为5。

解读：注意if的条件语句用的是赋值符“=”而不是等号“==”，因此条件一直为真。

4.2负数和正数的反码、补码分别是什么?

（1）负数的反码：对原码除符号位外的其余各位逐位取反就是反码。
（2）负数的补码：负数的补码就是对反码加1。
（3）正数的原码、反码、补码都一样。

4.3编译和链接有什么不同？（如对外部符号的处理）

（1）编译（+汇编）生成的是目标文件(*.o)。编译过程中对于外部符号（如用extern跨文件引用的全局变量）不做任何解释和处理，外部符号对应的就是“符号”。
（2）链接生成的是可执行程序。链接将会解释和处理外部符号，外部符号对应的是地址。

4.4函数参数的传递方式有几种?

（1）两种：值传递、指针传递。
（2）严格来看，只有一种传递，值传递，指针传递也是按值传递的，复制的是地址。

4.5局部变量能否和全局变量重名?

答案：能，局部会屏蔽全局。要用全局变量，需要使用“::”。
注意：对于有些编译器而言，在同一个函数内可以定义多个同名的局部变量，比如在两个循环体内都定义一个同名的局部变量，而那个局部变量的作用域就在那个循环体内。

4.6如何引用一个已经定义过的全局变量?

答案: 可以用引用头文件的方式（不建议，可能会造成重复定义），也可以用extern关键字。

注意:

（1）如果用引用头文件方式来引用某个在头文件中声明的全局变理，假定你将那个变量写错了，那么在编译期间会报错。

（2）如果你用extern方式引用时，假定你犯了同样的错误，那么在编译期间不会报错，而在链接期间报错。

4.7全局变量可不可以定义在可被多个.C文件包含的头文件中?

答案: 可以，但要将该全局变量定义为static全局变量，否则会重复定义，但此时没有达到一般意义上的全局变量的效果，而是静态全局变量。

注意: 建议不要在头文件中定义变量，只做变量的声明。

4.8语句for(；1；)有什么问题？它是什么意思?

在C语言中没问题，和while(1)相同，因为for循环中只要中间的表达式的值为TRUE即可。

4.9下面代码输出是几?

```
for(i = 0; i < 2, i < 3, i < 4; i++)
     printf("%d \t", i);
```

答案: 输出：0　1　2　3

解读: for中间的几条表达式中只要有一条为TRUE就可以了。

4.10 ’0’的ASCII码为48，’A’的ASCII码为65，’a’的ASCII码为97。

4.11下面代码输出结果是什么?

```
void main()
{
    char *p1 = “name”;
    char *p2;
    p2 = (char*)malloc(20);
    memset (p2, ‘0’, 20);
    while(*p2++ = *p1++);
    printf(“%s\n”, p2);
}
```

答案: 输出15个’0’。

解读: 由于在语句“while(*p2++ = *p1++); ”中，*p2每次接受*p1的赋值之后，都会进行一次自增操作，因此当p1指向的字符串共5个字符（包含’\0’）都赋值给*p2后，p2已经指向第六个字符’0’。

4.12指针只有在指向数组元素时才有意义吗?

不一定，比如说指向GPIO寄存器地址/字符串时。

4.13以下代码是否合乎语法?

```
int a[5];
*(2 + a) = 1;
a[2 - 1] = 1;
```

答案: 是。

解读:

（1）对指针解引用前，在其前面加上一个数字，并不会报错，只是解引用的地址会变。

（2）访问数组时，其下标允许是常量表达式。

4.14一个32位的指针，如何按8字节的整数倍向下对齐，请写出代码。

答案: 设该指针为p，则代码为:

```
p &= ~(8 -1);
```

解读:

（1）在存储的时候，为了提高效率，一般都会让地址偏移量落在2的m次方的位置上，而且经常有向上取整和向下取整两种需求。

（2）向下取整：

```
#define  PALIGN_DOWN(x, align)  ((x) & ~((align) -1))
```

（3）向上取整：

```
#define  PALIGN_UP(x, align)  ((x) + ((align) - 1)) & ~((align) - 1)
```

4.15下面这段程序的运行结果?

```
int main()
{
    int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    memcpy(a + 3, a, 5);
    for(int i = 0; i < 10; i++)
    {
        printf("%d ", a[i]);
    }
}
```

答案：输出为0 1 2 0 1 5 6 7 8 9

解读：因为mencpy的最后一个参数是需要拷贝的字节数目，而一个int类型的数据占4个字节，本题拷贝5个字节，实际上只能拷贝两个数字（从低位开始拷贝）。

4.16 gcc优化代码执行速度的编译选项是?

-O0/没有-O	不进行任何优化，用尽可能直接的方法来编译源代码，每行代码被直接转换为可执行文件中对应的指令。当调试一个程序时，这是最佳选项。
-O1/-O	使用能减少目标文件大小、执行时间，同时又不会使编译时间明显增加的代码，在编译大型程序的时候会显著增加编译时内存的使用。
-O2	使目标文件执行速度变快，但会增加编译时间。包含-O1的优化，同时执行了不会使目标文件变大、执行速度变慢的优化，不执行循环展开以及函数内联。
-O3	执行所有-O2优化选项，同时打开更深度的优化来提升执行文件的速度，比如函数内嵌，但也可能增加其大小。
-Os	专门优化目标文件的大小，执行所有不增加目标文件大小的-O2优化选项，并执行专门优化目标文件大小的选项。

4.17 new和malloc的区别?

（1）new、delete、是C++中独有的操作符，而malloc、free是C/C++中的标准库函数。

（2）C++允许重载new/delete操作符，而malloc/free是一个函数，不能重载。

（3）new返回的是指定类型的指针，并且可以自动计算所申请内存的大小；而malloc返回的是void*类型，需要进行强制类型转换，并且需要指定要申请内存的大小。

（4）new操作符从自由存储区上为对象动态分配内存空间，而malloc函数从堆上动态分配内存。

（5）new内存分配失败时，会抛出bac_alloc异常；malloc内存分配失败时返回NULL。

（6）使用new创建对象在内存分配的时候会自动调用构造函数，同时也可以完成对对象的初始化，delete也能自动调用析构函数。

4.18指针与引用的相同和区别?

相同：都是地址的概念，指针是所指内存的地址；引用是某块内存的别名，本质是指针常量，也是占内存的，有的编译器优化后就不占内存了。

不同：

（1）引用仅仅是一个别名，而指针是一个实体。

（2）引用只能在定义时被初始化一次，之后不可改变；非const指针可以改变。

（3）引用不能为空，指针可以为空。

（4）引用没有const，指针有const。

（5）引用使用时无需解引用，指针需要解引用。

（6）引用和指针自增（++）的意义不一样。

（7）sizeof引用跟指针得到的大小不一样。

帖子还没人回复，快来抢沙发！