TCPL第1-3课作业检查

ex2-sum.c

第九行：

ln9: sum = a + b;

之后想要检查结果是否正确，可以用 assert(num == 579);
不过这个 assert 只在 debug 版本有效，实际写测试程序时还有更好的工具。

ex3-max.c

int max(int x, int y);

一般应该放到函数的外面，文件的前部，例如：

```
int max(int x, int y);
int min(int x, int y);

int main() {

}
```

如果这些函数只在本文件内部使用，还可以加 static 关键字：

```
static int max(int x, int y);
static int min(int x, int y);

int main() {

}
```

第 13 行，main 函数结尾之后应该加空行。

max 函数写得有点冗长，c 语言的写法可以更加精简，例如：

```
int max(int x, int y) {
    if (x > y) return x;
    else return y;
}
```

或者加上必要的括号和换行：

```
int max(int x, int y) {
    if (x > y) {
        return x;
    } else {
        return y;
    }
}
```

或者，最简单的写法：

```
int max(int x, int y) {
    return (x > y) ? x : y;
}
```

x5-comment.c

比较长的注释通常放到代码行的前面，例如：

```
/* This is the main function that is called right after starting the program
and
 * do the program's main work.
 */
int man() {
```

而且：
1. 注释开始的星号之后有一个空格，
2. 注释在80或120列之内换行，
3. 注释结尾的星号和斜线单独占一行，
4. 每行的星号是对齐的（一般的编辑器都会在回车时停在对齐的位置）。

比较短的注释可以放到代码行的后面。并且通常采用 // 风格的注释。例如：

```
printf("max=%d\n", c); // Print the max value
```

注意，// 后面也有一个空格。

按照上面的规则，hw1-calculate.c 可以改进一下哦。

1. 函数的声明要移到 main 函数的前面去。
2. 第 10 行计算完后，可以加个 assert 检查结果是否符合预期。
3. calculate 函数可以用 ?: 表达式减少代码，只需要一行。

ex2-factorial.c

第 10 行的乘法，可以简写为：product *= i; 。
C 语言里头很多操作都可以这么写，例如：

+=
-=
*=
/=
&=
|=

x op= y 表示 x = x op y。

注意测试一下输入不同的 n 的表现。例如 n 为 0，负数，较大的数。

ex3-odd_factorial.c 也类似。

ex5-leapyear.c

函数声明 int isleap(int year); 应该挪到 main 函数外面。

可以利用 return 语句提前返回，减少 if 语句嵌套的层数。

```
int isleap(int year) {
  if (year % 4 != 0) {
    return 0;
  }

  // 能被4整除，但是不能被100整除
  if (year % 100 != 0) {
    return 1;
  }

  // 能被4整除，也能被100整除，还能被400整除
  if (year % 400 == 0) {
    return 1;
  } else {
    return 0;
  }
}
```

对于这种返回值是 true/false 的情况，可以用 C 语言中的数据类型 bool。

```
boo is_leap(int year) {
  if (year % 4 != 0) {
     return false;
   }
...
}
```

ex6-summation.c

里头的变量 j 可以去掉。只用变量 i。不过需要把 i 改成 int 类型。

另外：
1. 浮点数的赋值最好用：float f = 1.0f，加上小数点和f后缀。
2. For 循环中可以定义循环变量：for (int I = 0; I < n; I++) {

hw1-prime.c

for 循环可以写成：

```
for (int i = 2; i < 1000; i++) {
```

isprime() 中的循环变量可以从 2 开始，去掉 j != 1 的判断。
求质数的算法还可以更快一点，例如：
1. j 其实不需要循环到 i - 1。只需要到 i 的二次方。这个二次方不好写，条件可以写成 j * j <= i。注意这个等号，考虑完全平方数。
2. 埃拉托色尼筛法。因为题目要求的是一个质数表，这个方法最快。

hw2-primesum.c

计数的循环变量 i 不要用浮点数。
求倒数和求和时确保它是浮点数运算，可以把：

```
sum = sum + (1 / i)
```

改成：

```
sum = sum + (1.0f / i)
```

或

```
sum += 1.0f / i
```

countline.c

注意，行数是以\n标记的行的个数。如果一行数据abc，后面没有\n，它算作0行。如果不按照这个定义，就比较麻烦一些。例如：

1. 文件没有任何字符，应该是0行。
2. 文件有abc，但是没有\n，算作1行。
3. 文件有abc，最后是个\n，算作2行。

实际的语义参照命令 wc 来就好了，它采用的就是第一种数\n的方法。

全文完。