5.6

(a)

对于只含有输入, 输出和循环结构的算法而言, 三个合理放置的不变式就可以证明其正确性.

(b)

无循环结构的流程图.

(c)

4个.

(d)

4个. 无论是顺序的两个循环结构还是嵌套的循环结构, 在循环开始的位置分别插入断言即可.

分为两类: 第一类是输入后和输出前的断言; 第二类是循环开始位置的断言.

5.8

Algorithm rev(X)

INPUT: a finite string X

OUTPUT: the reverse of X

Y = empty string

while (X != empty string)

Y = Y·last(X)

X = all-but-last(X)

return Y

部分正确性:

建立循环不变式

其中和为循环执行第次后的和, . 下面使用归纳法证明.

Basis n = 0或1时, 易知其成立.

I.H. n = k时成立.

I.S. n = k + 1时, 根据循环里执行的语句有

部分正确性证毕.

完全正确性:

每次循环均导致串X长度减1, 循环出口为串X为空, 由于串X为有限字符串, 故该循环一定会终止, 即该算法一定终止, 证毕.

5.10

(a)

rev(S)的正确性证明见5.8. 特别地, 题目必须保证equal(S, Y)算法是正确的, 且其语义满足 S = Y时返回true, S != Y时返回false. 否则无法证明算法Pall()的正确性.

对于算法Pall()而言, 这是一个只有顺序执行的流程. 只需保证输入合法时, 输出一定合法即可. 该程序语义是判断一个串和其逆是否相同, 和回文串的定义一致, 故这里部分正确性是显然的. 由于rev(S)和equal(S,Y)是完全正确的, 且Pall()只是顺序调用了它们, 故Pall()的完全正确性也是显然的.

(b)

若某个正确性待证的程序的流程是顺序调用多个完全正确性已被证明的子程序, 只要我们能证明该程序的语义是正确的, 则其完全正确性得证.

5.12

(a)

部分正确性:

这里循环不变式为

等价于

记循环执行的次数为n, 下面用归纳法证明.

Basis

n = 1时, 循环执行了一次, 判断了S的第一个和倒数第一个字符是否相同, 若相同则

E维持true但无法说明S是否回文, 若不同则E变为false且说明了S不是回文, 循环不变式成立.

I.H. n = k时成立

I.S.

若E在n = k 时为false, 则这里根据循环语句可以得到n = k + 1时E依然为false, 且其语义继承自以前的循环, 也就是说这里.

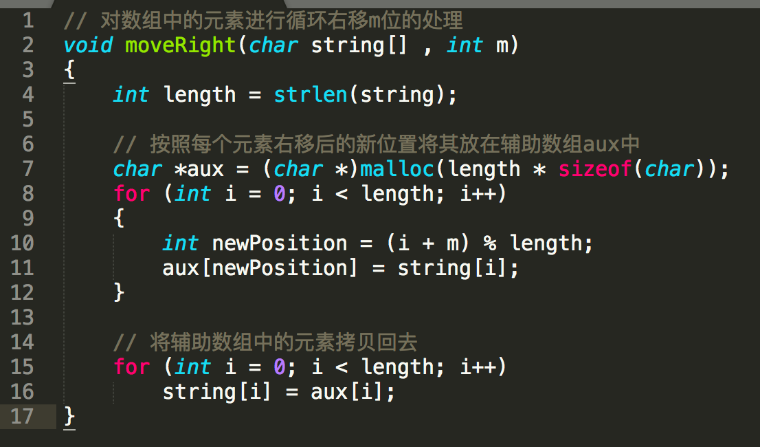
若E在n = k 时为true, 则n = k + 1时行为和第一次循环类似, 也就是说可以保持循环不变式成立.

综上, 部分正确性得证.

(b)

Pal2在S = “AB”时无法终止.

任取一个你所编写程序中的循环，写出相应的循环不变式，证明循环的正确性



循环不变式:

每次循环结束后aux字符串循环左移m位后的前i个元素与原字符串string的前i个元素相等.

部分正确性:

当i = 0时, 由于其newPosition就是从0这一位置开始循环右移m所得位置, 故对第一个字符串显然成立.

若前i次循环时循环不变式均成立, 那么执行完第i + 1次循环当中的语句后, 有:

也就是说, 此时将aux中处于newPosition上的字符循环左移m位, 其位置就会回到i + 1处. 由于前i次循环时循环不变式均成立, 由归纳法我们可以得知, 循环不变式成立.

完全正确性:

由于每次循环结束后, 循环记数变量i会增加1, 因此i是向length靠近的, 而且length和i都是整数, 因此算法总能结束.