第二讲

1．

bool isOverflow(int x, int y){

int mul = x \* y;

return ((x ^ y) > 0 && mul < 0) || (x != 0 && ((mul / x) ^ y) != 0);

}

int POW(int x, int y)

//@requires y >= 0;

{

if (y == 0) return 1;

else{

int pow = POW(x, y - 1);

if (isOverflow(x, pow)){

error("Overflow");

}

return x \* pow;

}

}

int f(int x, int y)

//@requires y >=0;

//@ensures \result == POW(x,y);

{

int r = 1;

int b = x; /\* base \*/

int e = y; /\* exponent \*/

while (e > 0)

//@loop\_invariant e >= 0;

//@loop\_invariant r \* POW(b,e) == POW(x,y);

{

if(e % 2 == 1){

if (isOverflow(b, r)){

error("Overflow");

}

r = b \* r;

}

b = b \* b;

e = e / 2;

}

//@assert e == 0;

return r;

}

2．当f的输入值x=0，y=0时，由于0的0次方没有数学意义，所以此时不满足循环不变量的条件。

第三讲

1．

int quot(int x, int y)

//@requires y != 0;

{

int sign = x ^ y;

int res = 0;

x = x < 0 ? (~x) + 1 : x;

y = y < 0 ? (~y) + 1 : y;

int i = 31;

while(i >= 0){

if((x >> i) >= y){

res += (1 << i);

x -= (y << i);

}

i -= 1;

}

if(sign < 0){

res = ~res + 1;

}

return res;

}

int rem(int x, int y){

return x - quot(x, y) \* y;

}

2．

string int2hex(int x)

//@requires x >= 0 && x <= (1 << 31) - 1;

{

char[] A = alloc\_array(char, 16);

char[] s = alloc\_array(char, 11);

for(int k = 0; k < 11; k++)

s[k] = '0';

A[0] = '0'; A[1] = '1'; A[2] = '2'; A[3] = '3'; A[4] = '4';

A[5] = '5'; A[6] = '6'; A[7] = '7'; A[8] = '8'; A[9] = '9';

A[10] = 'A'; A[11] = 'B'; A[12] = 'C'; A[13] = 'D'; A[14] = 'E';

A[15] = 'F';

s[0] = '0'; s[1] = 'X';

int t = x;

int i = 2;

while(t > 0){

s[i] = A[rem(t, 16)];

t = quot(t, 16);

i += 1;

}

int l = 2;

int r = 9;

char tmp;

while(l < r){

tmp = s[l];

s[l] = s[r];

s[r] = tmp;

l += 1;

r -= 1;

}

s[10] = '\0';

string str = string\_from\_chararray(s);

return str;

}

3．

int lsr(int x,int n){

return (x >> n) & (~(~0 << (32 - n)));

}

2的补码是1111 1000，lsr(2，1)的含义是将2的二进制补码逻辑右移一位，并在最高位补0，即此时不考虑2的二进制补码的符号位，逻辑右移的结果为0111 1000。

第四讲

1．

int[] array\_part(int[] A, int i, int j)

//@requires i >= 0 && i <= j;

//@ensures \length(\result) == j - i + 1;

{

int n = j - i + 1;

int[] B = alloc\_array(int, n);

int a;

int b = 0;

for(a = i - 1; a <= j - 1;)

//@loop\_invariant b <= n;

{

B[b] = A[a];

a++;

b++;

}

return B;

}

2．

int copy\_into(int[] source, int i, int n, int[] target, int j)

//@requires \length(source) >= i + n && j + n <= \length(target);

{

int t = 0;

for(; t < n; t++)

//@loop\_invariant t >= 0;

{

target[j + t - 1] = source[i + t - 1];

}

return j + t - 1;

}

3．

由于函数的返回值为int类型，函数的参数为int数组类型，所以函数f的功能可以是获取一个数组中元素的个数。