# 位运算专题

# 基本运用

计算机原码、反码、补码详解：<http://www.cnblogs.com/zhangziqiu/archive/2011/03/30/ComputerCode.html>

Java 位运算详解：<http://www.cnblogs.com/zhengtao/articles/1916751.html>

## 清零

解释：清空整数的指定的位数。

例题：给定整数a，清空a的后k位。

解答：

|  |
| --- |
| Program01 |
| public static int clearLastK(int a,int k){  int flash=0;  for(int i=0;i<k;i++){  flash+=(1<<i);  }  return a&(0xffffffff-flash);  } |

## 获取数据的指定位

例题：给定整数a，获取低k位或者高k位的数值。

解答：

|  |
| --- |
| Program02 |
| public static int getHighKbit(int a,int k){  return a>>>(32-k);  }    public static int getLowKbit(int a,int k){  return a<<k;  } |

# 小试牛刀

例2.1 在不使用额外变量的情况下，如何交换两个整数？（Pay attention!）

解答：

|  |
| --- |
| Program03 |
| public static void switchAB(int a,int b){  System.*out*.println(a+" "+b);  a=a^b;  b=a^b;  a=a^b;  System.*out*.println(a+" "+b);  } |

例2.2 统计给定正整数中 位值为1的数目。(Leedcode 191 Number of 1 Bits)

|  |
| --- |
| Program04 |
| public static int count1bitCount(int a){  int result=0;  while(a!=0){  result+=(a&1);  a>>>=1;  }  return result;  } |

例2.3 反转一个整数。(Leedcode 190 Reverse Bits)

|  |
| --- |
| Program05 |
| public static int reverseInt(int a){  int bit=0;int result=0;  for(int i=0;i<32;i++){  bit=a&1;  result<<=1;  result+=bit;  a>>>=1;  }  return result;  } |

例2.4 判断一个数是否是2的整数幂（Leedcode Power of two）

|  |
| --- |
| Program 06 |
| public boolean isPowerOfTwo(int n) {  if(n<=0)  return false;  int counter=0;  while(n!=0){  counter+=n & 1;  if(counter > 1)  return false;  n>>>=1;  }  return true;  } |

# 算法进阶

例3.1 给定一个整数数组，该数组中，只有一个数出现过一次，其他数均出现两次，要求在线性时间内找到该单个元素。

|  |
| --- |
| Program 07 |
| public int singleNumber(int[] nums) {  int result=0;  for(int i=0;i<nums.length;i++)  result ^=nums[i];  return result;  } |

例3.2 给定一个数组，该数组中，只有一个元素出现过一次，其他元素出现过三次，要求在线性时间内找到该单个元素。

|  |
| --- |
| Program 08 |
| public static int singleNumber(int[] nums) {  int first=0;  int second=0;  for(int i=0;i<nums.length;i++){  first=(first^nums[i])&(~second);  second=(second^nums[i])&(~first);  }  return first;  } |
| 考虑有限状态机  （0,0）-（a）->(a,0)-(a)->(0,a)-(a)->(0,0) |

例3.3 给定一个数组，该数组中，有且仅有2个元素出现过1次，其他元素均出现过2次，要求在线性时间内找到这两个单个元素。

|  |
| --- |
| Program 09 |
| public static int[] singleNumber(int[] nums) {  int[] result={0,0};  int xor=0;  //first step,we get xor result.  for(int i=0;i<nums.length;i++){  xor^=nums[i];  }  //second step, we find the first bit that x and y is not equals.  xor&=-xor;  //third step, we distinguish x and y by this bit.  for(int i=0;i<nums.length;i++){  if((nums[i]&xor)==0){  result[0]^=nums[i];  }  else  result[1]^=nums[i];  }  return result;  } |

例3.4 给定一个元素各异的整数数组，求出该数组的所有子集。

|  |
| --- |
| Program 10 |
| public static List<List<Integer>> subsets(int[] nums) {  List<List<Integer>> result=new ArrayList<List<Integer>>();  result.add(new ArrayList<Integer>());  for(int i=1;i<Math.*pow*(2, nums.length);i++){  List<Integer> list=new ArrayList<Integer>();  int t=i;int k=0;  while(t!=0){  if((t&1)==1)  list.add(nums[k]);  Collections.*sort*(list);  k++;  t>>=1;  }  result.add(list);  }    return result;  } |