**深圳大学实验报告**

**课程名称： 计算机系统(2)**

**实验项目名称： 数据表示实验**

**学院： 计算机与软件学院**

**专业： 软件工程（腾班）**

**指导教师： 罗胜**

**报告人： 赵美玲 学号： 2023155025**

**实验时间： 2025 年 4 月 8 日**

**实验报告提交时间： 2025 年 4 月 8 日**

**教务处制**

|  |
| --- |
| **实验目的与要求：**   1. 了解各种数据类型在计算机中的表示方法 2. 掌握C语言数据类型的位级表示及操作 |
| **方法、步骤：**   1. 安装gcc-multilib：     或者：    2、根据bits.c中的要求补全以下的函数：  intbitXor(int x, int y);  inttmin(void);  intisTmax(int x);  ntallOddBits(int x);  int negate(int x);  intisAsciiDigit(int x);  int conditional(int x, int y, int z);  intisLessOrEqual(int x, int y);  intlogicalNeg(int x);  inthowManyBits(int x);  unsignedfloat\_twice(unsigned uf);  unsigned float\_i2f(int x);  int float\_f2i(unsigned uf);  3、在Linux下测试以上函数是否正确，指令如下：  \*编译：./dlcbits.c  \*测试：makebtest  ./btest |
| **实验过程及内容：**   1. 安装gcc-multilib：   a743f5f2ac094771622697d96588db2   1. 根据bits.c中的要求补全以下的函数：   （1）int bitXor(int x, int y);  依离散数学知识和德摩根律如下  a3faf3417be4ccda59467d782be5b05  （2）int tmin(void);  对于32位整数，最小值即为0X 8000 0000，即将1左移31位。  e2be3ca39cf759786e1fc0cc9daeefa  （3）int isTmax(int x);  可以知道，最大值为0x7fff ffff，加一后将变为0x8000 0000，且此数加上本身后将变为0。本身加本身为0的数只有0和0x8000 0000，因此，只需将0xffffffff排除即可：  f9c76ed5230daea7f0d3e4b12f0a58c  （4）int allOddBits(int x);  在二进制下，有且仅有所有位为奇数的数与0x5555 5555进行与运算后由0xffff ffff，变为0x0000 0000。因此可以通过对其与0x5555 5555进行与运算并取反再进行取逻辑反获得结果。又因为题干中不允许使用大于256的整数，故需要通过一些操作获得0x5555 5555。  可以通过将0x0505分别进行左移4、8、16、24得到4个数，并将四个数求和获得0x5555 5555。  4bff11c9d093e5b3c6d4927a49dc86d  （5）int negate(int x);  即取反并加一返回即可。  64c58173847b2ced7b620304d7f3579  （6）int isAsciiDigit(int x);  即对于每个输入的x，需要满足x>=‘0’且x<=‘9’，因此可以将x与临界值进行作差。并通过右移31位判断对符号位进行判断是0还是1即可。  d146aa9b6ce63aec7003e19f55b071e  （7）int conditional(int x, int y, int z);  对于x的判断可以通过t=!x进行判断实现，当x为0时返回1；当x不为0时返回0。可以将表达式大致转成( \_ &y)|( \_ &z)的格式进行配凑。  对于前面的空格，当x不为0，即t=0时，需要 t转换为0xffff ffff（-1）。可以通过对1按位取反再加一1获得。  对于后面的空格，当x为0，即t=1时，也需要 t转换为0xffff ffff（-1）。此时直接对t进行取反并加一即可。  e97837c9a8f17e79144e3e648ea5fdc  （8）int isLessOrEqual(int x, int y);  直接采用y-x并判断符号位的方法进行判断，但如果作差相减，有可能会发生int类型溢出，因此需要考虑其他方法。  a. 当x，y同号：此时，即可将问题转化为转换为p=y-x>=0，并对p的符号位（通过右移获得）进行判断，获得运行结果。  b.当x，y异号：此时只要x>=0，就可以返回0，否则返回1。  c.是否同号的判断：可以通过对符号位进行求和判断是否同号。  0ba07e99b05f7dfdd1e803785cfd059  （9）int logicalNeg(int x);  可以通过取相反数进行非零判断。令y=~x+1（y=-x）并讨论x与y的符号位，有如下几种情况：  a. 当x为0时，两者符号位都为0。  b. 当x=0x8000 0000时，两者符号位都为1。  c. 当x既不为0也不为0x8000 0000时，两者符号位为01或10。  e8410914e3bda7e02efb3b45a95c7ea  （10）int howManyBits(int x);  通过二分完成算法实现，具体设计如下：  我们举一个例子进行说明：x=0000 1000 1001 0000 0000 0000 0000 0000  a.令y=x>>16=0000 1000 1001 0000，shift16=(!!y)<<4使用(!!y)判断y是否为0，如果y为0，则返回0，说明前16位都为0，shift16=0，否则!!y=1，shift16=16，使x=x>>shift16=10001001 0000;  b.同理，令y=x>>8=0000 1000，shift8=(!!y)<<3使用(!!y) 判断y是否为0，如果y为0，则返回0，说明前8位都为0，shift8=0，否则!!y=1，shift8=8，使x=x>>shift8=00001000;  c.令y=x>>4=0000，shift4=(!!y)<<2使用(!!y) 判断y是否为0，如果y为0，则返回0，说明前4位都为0，shift4=0，否则!!y=1，shift4=4，使x=x>>shift4=00001000;  d.令y=x>>2=0000 10，shift2=(!!y)<<1使用(!!y) 判断y是否为0，如果y为0，则返回0，说明前2位都为0，shift2=0，否则!!y=1，shift2=2，使x=x>>shift2=000010;  e.令y=x>>1=0000 1，shift1=(!!y),使用(!!y) 判断y是否为0，如果y为0，则返回0，说明前1位都为0，shift1=0，否则!!y=1，shift1=1，使x=x>>shift1=00001;  f.最后对各个二分进行求和即可；  g.对于负数取反码即可，对于-1和0需要进行特殊考虑。  7294bb0ce2e6693d415e4787d571fa2  （11）unsigned float\_twice(unsigned uf);  （exp表示阶码位、frac表示尾数位）  通过分析可知，共存在三种可能的情况，分别考虑如下：  a.当exp=0xff时，直接返回本身即可；  b.当exp=0时，分两种情况考虑：  当uf[22]=0时，然后将frac左移一位即可  当uf[22]=1时，将exp自增1，然后再将frac左移一位即可  c.对于其他情况，将exp自增1，然后再分两种情况：  当exp==0xff，令frac=0即可。  其余情况正常左移并返回即可  59a6defb703651ac1afc67bcd81c247  （12）unsigned float\_i2f(int x);  首先，我们可以知道int型数据的表示范围为-231~231-1。  通过分析可知，主要有三种情况：  a.用二进制下科学计数法表示int型数时，尾数位数<=23，例如0x00008001，此时将0x8001左移24-16=8位得到frac，而exp则为127+16-1；  b.当尾数位数>23时，找到位数最末一位记作x[i]，然后对尾数的舍去分3种情况考虑，并初始化c=0：  当x[i-1]=1且x[i-2]、x[i-3]…x[0]都为0，且x[i]=1,令c=1；  当x[i-1]=1且x[i-2]、x[i-3]…x[0]不都为0，令c=1;  其余情况 令c=0；  c.特殊情况。  对于x为0或为0x8000 0000的情况，在处理前进行特判即可。  ffb81b2e5d6d82c8164adfd040042ff  （13）int float\_f2i(unsigned uf);  为了方便进行计算，可以将uf（unsigned float）切割成符号位s，阶码exp和位数frac，分情况讨论：  a.当exp=0或exp-127<0时，返回0；  b.当exp-127>=31时候，超出表示范围，于是返回0x80000000u；  c.当exp-127<=23，根据符号位返回值num>>(23-(exp-127))  4ea0402556e474599de3ec594a713b5  3、在Linux下测试以上函数是否正确，指令如下：  \*编译：./dlcbits.c  \*测试：makebtest  ./btest 02dd4428b7a3821cea630e729c5251e |
| **实验结论：**  位操作运算比高级语言的运算要更为抽象，难理解，需要注意很多细节以及底层运算逻辑。在写程序的过程中着重要注意特解，往往会影响程序的正确性。 |
| **心得体会：**  在运行过程中系统显示我btest有问题，如下  8b05fb6a9157758c59a979f056c0b47  显示使用了未初始化的变量，但是当我初始化后又有别的错，修改很久最后发现使用最简单的循环就可以解决，很奇怪，因为我觉得他们的实际运行是一样的，不知道为什么原代码就会报错。以下是修改后的代码片段。  cdc140ca60546f97e6200c7c69c06d1 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2018年 月 日 |
| 备注： |