不同类型数据的二进制编码方法

姓名：杨硕        学号：202411070224

1. 整型数据的编码

原码：

原码是一种简单的机器数表示法，最高位为符号位，正数的符号位为 0，负数的符号位为 1，其余各位表示数值的绝对值。

优点：直观易懂

缺点：加减法运算复杂，存在+0和-0两种表示形式

反码：

正数的反码与原码相同，负数的反码是在原码的基础上，符号位不变，其余各位按位取反（即 0 变 1，1 变 0）。

优点：与原码转换方便

缺点：存在+0和-0两种表示形式，加减法运算不够简便

补码：

正数的补码与原码、反码相同，负数的补码是在其反码的基础上再加 1。补码的主要作用是能方便地实现计算机中的减法运算，通过将减法转化为加法来进行计算。

在计算机中，整型数据通常以补码的形式存储和运算，因为补码可以将加法和减法统一起来，简化了计算机的运算电路设计，提高了运算效率。而且能有效处理数据的正负性以及溢出等问题。

优点：加减法运算统一且简便，只有一种表示形式，方便处理溢出问题

缺点：理解相对困难

选取你的学号的个位、十位，以及你生日的日子，组成一个四位数：

4228 -4228

求它和它的相反数的原码、反码、补码，要给出计算过程：

原码: 0001000010000100 1110111101111100

反码: 0001000010000100 1001000010000011

补码: 0001000010000100 1001000010000100

计算过程：

原码：

4228=4096+128+4

4228=（-1）\*（4096+128+4）

反码：

4228同原码

-4228 （-1）\*（！0！0！1！0！0！0！0！1！0！0！0！0！1！0！0）

二、实数的编码

浮点数编码方法

IEEE 754 是电气和电子工程师协会（IEEE）制定的关于浮点数运算的工业标准，目前广泛应用于计算机系统中。该标准定义了两种基本的浮点数格式：单精度（32 位）和双精度（64 位）

符号位（Sign）：占 1 位，0 表示正数，1 表示负数。

指数位（Exponent）：占 8 位，用于表示指数部分，但实际存储的指数值需要进行一定的偏移处理，这个偏移量对于单精度浮点数是 127。即实际指数值 = 存储的指数值 - 127。

尾数位（Mantissa）：占 23 位，它表示的是小数部分，但隐含了一个最高位的 1（除了表示 0 等特殊情况），也就是实际表示的小数是 1.xxxx（xxxx 为尾数位表示的部分）。

双精度浮点数（64 位）的编码方式类似，只是各部分的位数不同：符号位占 1 位，指数位占 11 位（偏移量为 1023），尾数位占 52 位。

实数进行算术运算的步骤

1. 对阶

目的：使参与运算的两个浮点数的指数部分相同。因为只有指数相同，才能直接对小数部分（尾数）进行相加运算。

操作：比较两个浮点数的指数值，将指数较小的浮点数的尾数进行右移操作，同时其指数值相应增加，直到两个浮点数的指数值相等为止。

2. 尾数相加

操作：在对阶完成后，直接将两个浮点数的尾数部分进行相加运算，就如同普通的二进制数相加一样。

3. 规格化

目的：保证结果的浮点数符合标准的浮点数表示形式，即尾数部分的最高位为 1（除了表示 0 等特殊情况）。

操作：如果尾数相加后的结果不符合规格化要求（比如最高位不是 1），则需要对尾数进行左移或右移操作，同时相应地调整指数值。

4. 舍入

目的：在进行尾数的移位操作（如对阶、规格化等过程中）可能会导致尾数的位数超出规定的位数（单精度 23 位，双精度 52 位），舍入就是为了处理这种情况，以保证结果的准确性和符合浮点数的标准表示。

操作：常见的舍入方法有四种，分别是就近舍入、朝零舍入、向上舍入和向下舍入。例如，就近舍入就是将尾数按照最接近的可表示的值进行舍入，通常采用的是四舍五入的原则（但不完全相同）。

假设在前面的运算过程中，尾数相加后需要舍入，按照就近舍入原则进行处理。

5. 处理结果的符号

目的：确定运算结果的符号，根据参与运算的两个浮点数的符号以及运算类型（加法、减法等）来判断。

操作：如果是加法运算，两个正数相加结果为正数，两个负数相加结果为负数，正数与负数相加则根据绝对值大小来确定结果的符号。

减法、乘法、除法等算术运算的基本步骤与加法类似，只是在具体的操作细节上有所不同，比如乘法运算需要分别相乘尾数和相加指数等，但整体的思路都是先处理指数部分（对阶等），然后处理尾数部分（相乘、相加等），最后进行规格化、舍入等操作以得到符合标准的浮点数结果。

三、汉字的编码

外码

定义：外码也叫输入码，是指用户从键盘上输入汉字时所使用的编码。其主要目的是为了方便用户将汉字输入到计算机系统中，不同的输入法对应着不同的外码体系。

常见类型及示例：

拼音码：这是最为常见的一种外码，它按照汉字的读音来进行编码。

五笔字型码：它依据汉字的字形结构和笔画特征进行编码。

区位码：将汉字或其他字符按照一定的区域和位置进行编号形成的编码。

交换码

定义：交换码是用于汉字信息处理系统之间或者汉字信息处理系统与通信系统之间进行信息交换时所使用的编码。它使得不同的系统在交换汉字信息时能够统一标准，保证信息准确无误地传输和理解。

机内码

定义：机内码是计算机内部存储、处理和传输汉字时所使用的编码。它是将交换码进行一定的转换后得到的，以便计算机能够更好地识别、处理和存储汉字信息。

字形码

定义：字形码是用于表示汉字字形的编码，它主要是为了将汉字以可视化的形式呈现出来，比如在显示器上显示汉字或者在打印机上打印汉字时，就需要用到字形码。

常见类型及示例：

点阵字形码：它通过将汉字的字形用点阵的方式来表示。

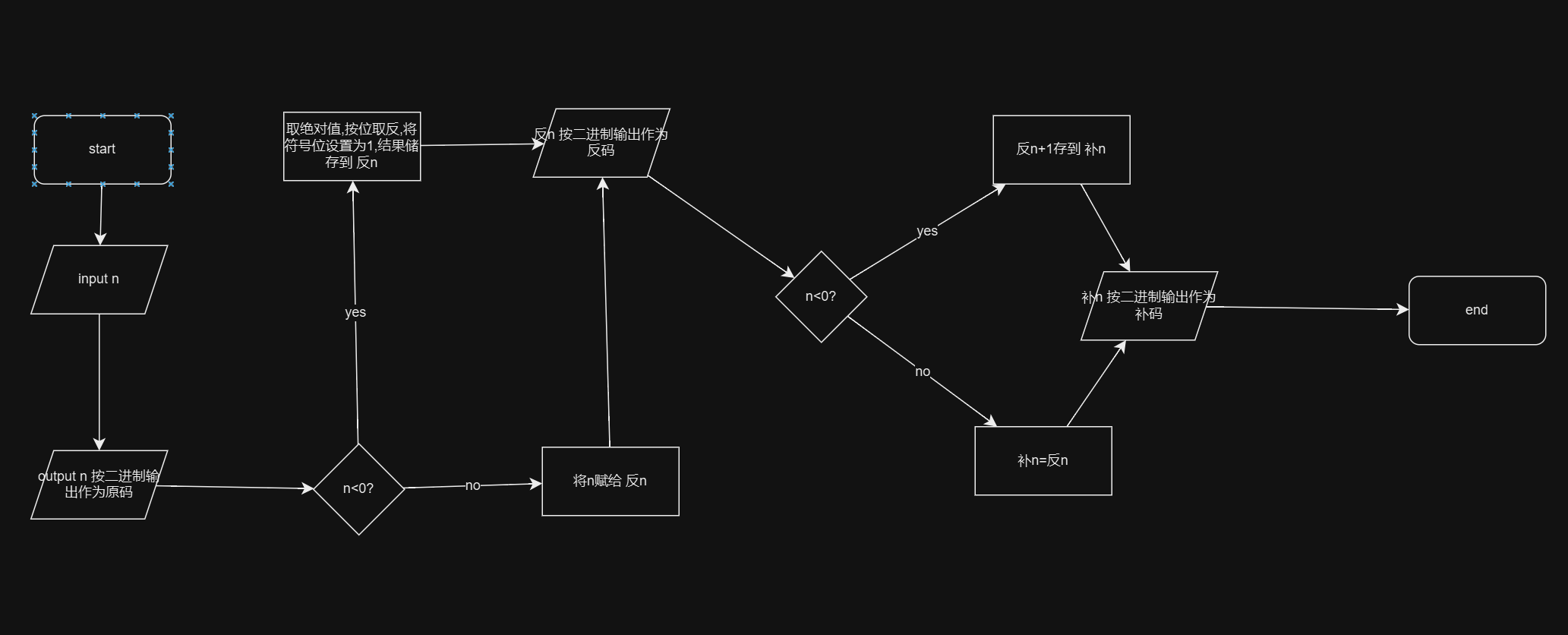
矢量字形码：它不是用点阵来表示汉字字形，而是用数学公式和矢量来描述汉字的笔画和外形。这种方式的优点是在放大或缩小汉字时不会出现失真现象，因为它是基于数学描述的。比如，在一些专业的图形设计软件或高质量的打印设备中，可能会采用矢量字形码来呈现汉字字形。

综上所述，汉字的外码方便用户输入，交换码用于系统间信息交换，机内码供计算机内部处理，字形码则用于汉字的可视化呈现，它们在汉字信息处理的不同环节发挥着重要作用

四、编程求整数的编码

编写程序，输入一个整数，输出它的原码、反码和补码。

首先给出流程图，再给出代码。



#include <stdio.h>

void prBit(int num, char \*chs) {

  printf("%s: ", chs);

  for (int i = 15; i >= 0; i--) {

    int bit = (num >> i) & 1;

    printf("%d", bit);

  }

  printf("\n");

}

int main() {

  int num;

  scanf("%d", &num);

  prBit(num, "原码");

  // 反码

  int fannum = num;

  if (num < 0) {

    // 负数，先取绝对值，然后按位取反，再设置符号位为1

    fannum = -num;

    for (int i = 0; i < 16; i++) {

      fannum = fannum ^ (1 << i);

    }

    fannum = fannum | (1 << 15);

  }

  prBit(fannum, "反码");

  // 补码

  int bunum = fannum;

  if (num < 0) {

    // 负数，反码加1得到补码

    bunum++;

  }

  // 补码

  prBit(bunum, "补码");

  return 0;

}