1

Lab

**BÁO CÁO BÀI THỰC HÀNH SỐ 1**

**Tính toán bit**

**Môn học: Lập trình hệ thống**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | ThS. Đỗ Thị Hương Lan |
| **Sinh viên thực hiện** | Nguyễn Phan Hữu Khánh (22520645)  Trần Anh Khôi |
| **Mức độ hoàn thành** | Hoàn thành |
| **Thời gian thực hiện** | 06/03/2024-11/03/2024 |
| **Tự chấm điểm** | 9/10 |

1. **Phần trên lớp:**

(file cpp có bổ sung giải thích)

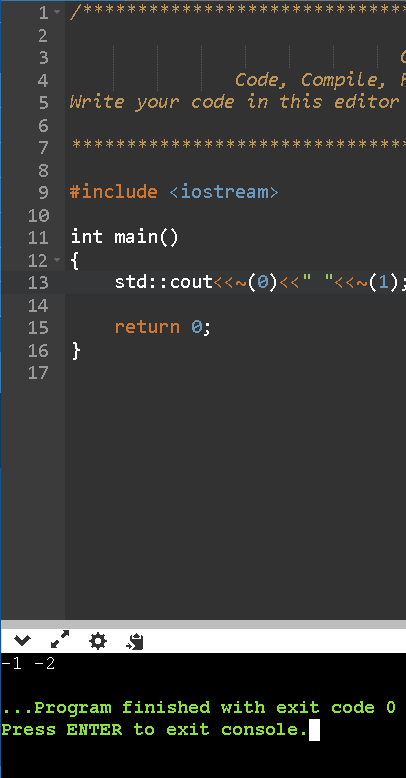
1. **Phần về nhà:**

2.1

Để so sánh dấu của 2 số, ta so sánh bit dấu của chúng(x\_sign và y\_sign)

Vì cùng dấu phải trả về 1, khác dấu trả về 0 -> Bảng chân trị phép XNOR

Nếu dùng XNOR thì:“return ~(x\_sign^y\_sign) +2”, vì trong C++ ~(0)=-1, ~(1)=-2



Để không cần +2 ta có thể XOR, sau đó dùng “!” để đúng ý đồ của phép XNOR mà ta muốn.

2.2

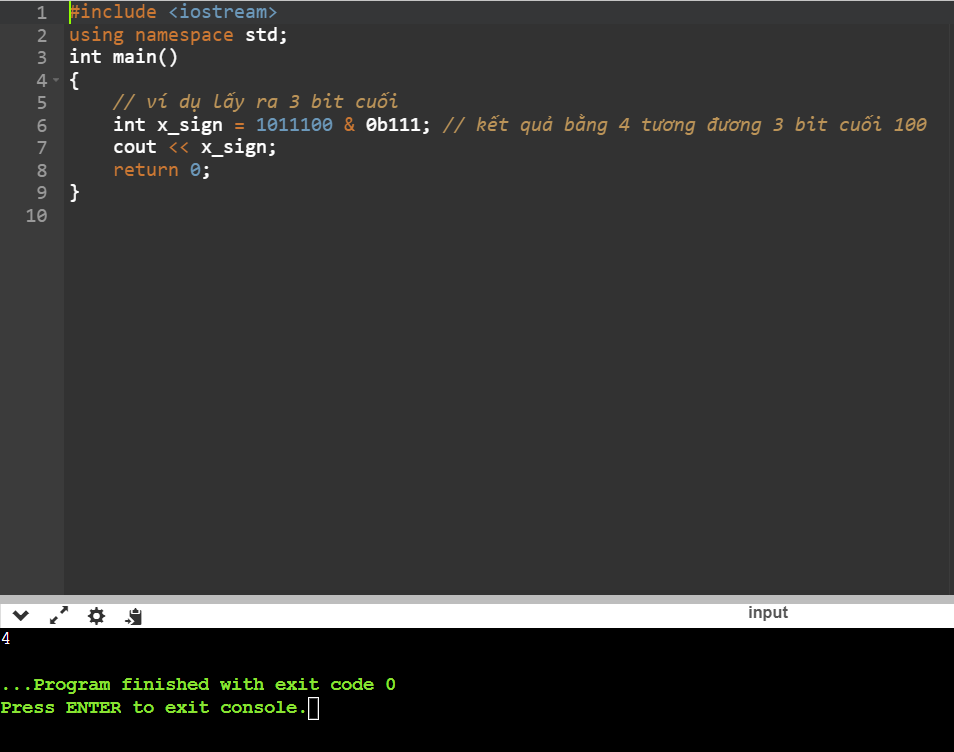
Để chia hết cho 8 thì 3 bit cuối phải là 0 (vì chia 8 đồng nghĩa với dịch phải 3 bit)

VD: 8/8=1 dư 0, 1000>>3=1

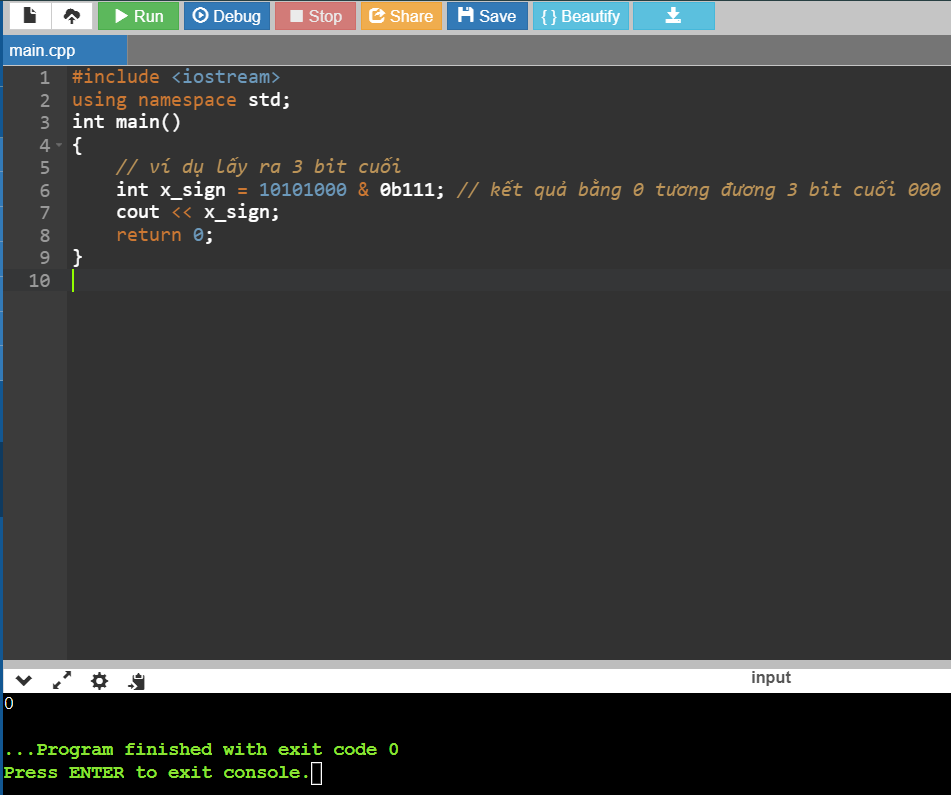
9/8=1 dư 1, 100**1**>>3=1

10/8=1 dư 2, 10**10**>>3=1

Để lấy 3 bit cuối thì cần AND với mask 0b111



Nếu 3 bit cuối đều là 0 thì ta cần in ra 1, còn lại tất cả đều in 0->dùng dấu “!” vì !(a) với a khác 0 thì đều cho kết quả 0

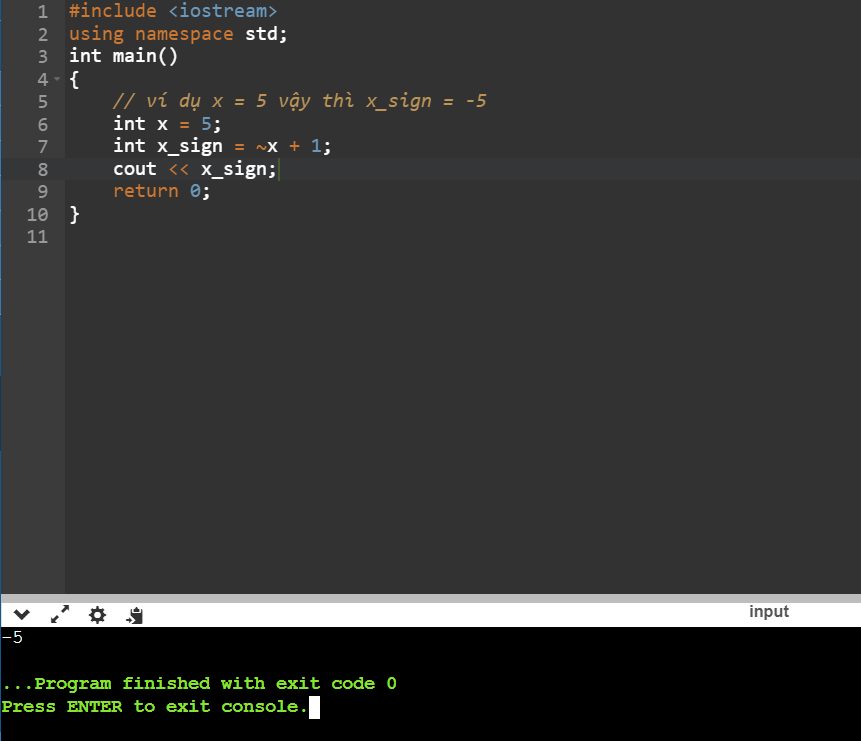


2.3

X dương thì in 1, ngược lại in 0

X âm bit dấu là 1, x dương bit dấu là 0->NOT bit dấu, nhưng vậy thì không thỏa khi x=0

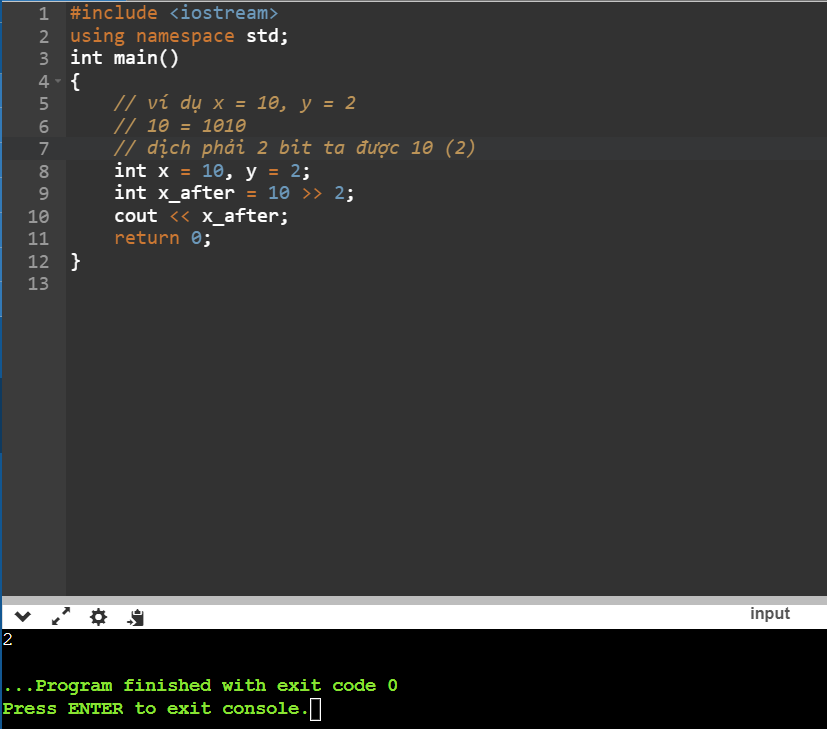
Ta thay bằng lấy bit dấu của số bù 2 của nó (vẫn cho kết quả như phép NOT bit dấu), điều này sẽ làm thỏa cả trường hợp x=0 vì số bù 2 của 0 là 0, bit dấu là 0 (thỏa)



2.4

Ta viết lại biểu thức là : x\*2^(-n)<1 ( vì n dương nên 2^n luôn dương)

Ta tính được x\*2(-n) dựa vào dịch bit

Sau khi làm một vài ví dụ:  
  
VD: x=10, n=2 ->trả về 0

x>>2=0b10

x=7, n=2 ->trả về 0

x>>2=0b01

x=2, n=2 ->trả về 1

x>>2=0x0

x=1, n=2 -> trả về 1

x>>2=0x0

Ta thấy quy luật ở đây là nếu trả về 1 thì x sau khi dịch bit đều có giá trị là 0x0 ngược lại thì sẽ có các giá trị khác

Từ đây, ta chỉ cần xem sau khi x dịch bit thì x có bằng 0x0 không là được.