

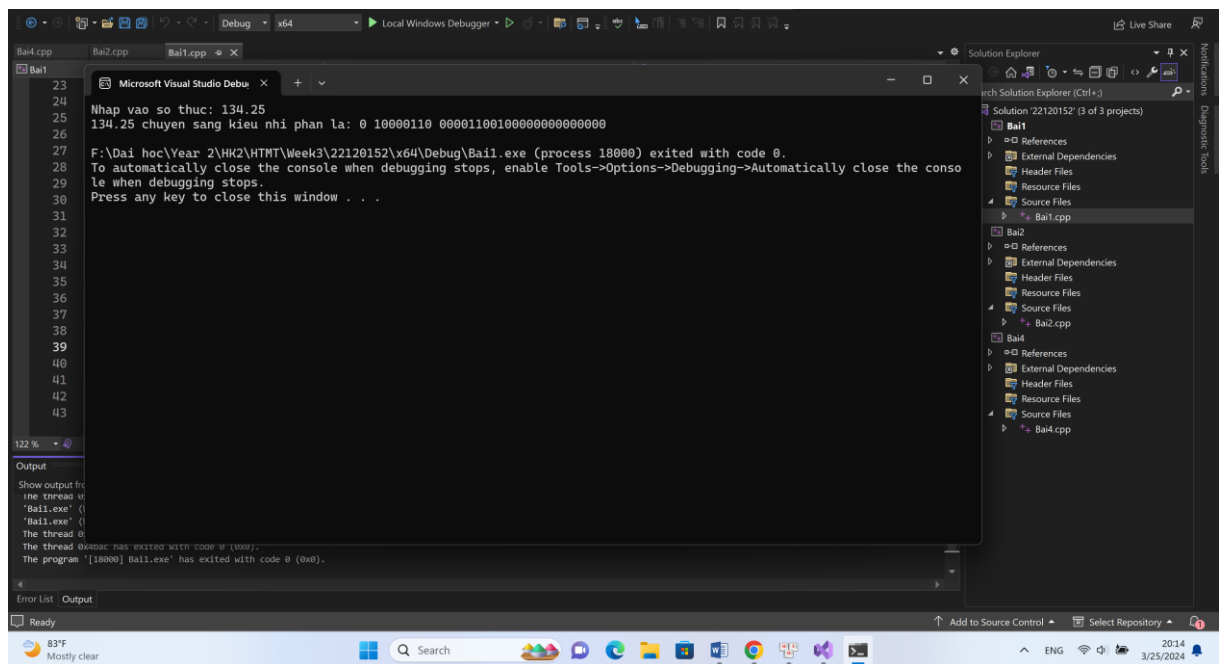
# BÀI TẬP 3: KHẢO SÁT SỐ CHẤM ĐỘNG

## I. Đánh giá

Bài	Mức độ hoàn thành
Bài 1	Đã hoàn thành
Bài 2	Đã hoàn thành
Bài 3	Đã hoàn thành
Bài 4	Đã hoàn thành

## II. Kết quả bài làm

### 1. Bài 1



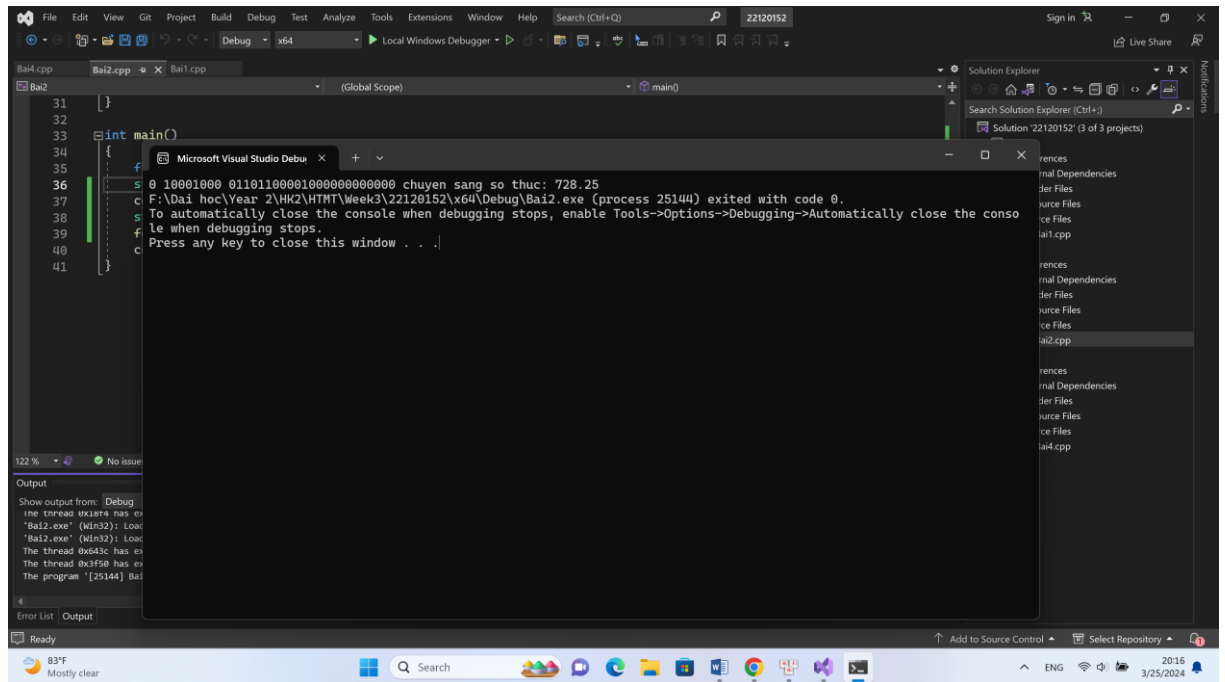
```
23
24
25 Nhập vào số thực: 134.25
26 134.25 chuyển sang kiểu nhị phân là: 0 10000110 0000110010000000000000
27 F:\Dai hoc\Year 2\HK2\HTMT\Week3\22120152\x64\Debug\Bai1.exe (process 18000) exited with code 0.
28 To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
29 Press any key to close this window . . .
30
31
32
33
34
35
36
37
38
39
40
41
42
43
```

Output

```
Show output from the threads:
'Bai1.exe' (0)
'Bai1.exe' (0)
The thread 0:
The thread 0x00000000 has exited with code 0 (user).
The program '[18000] Bai1.exe' has exited with code 0 (0x0).
```

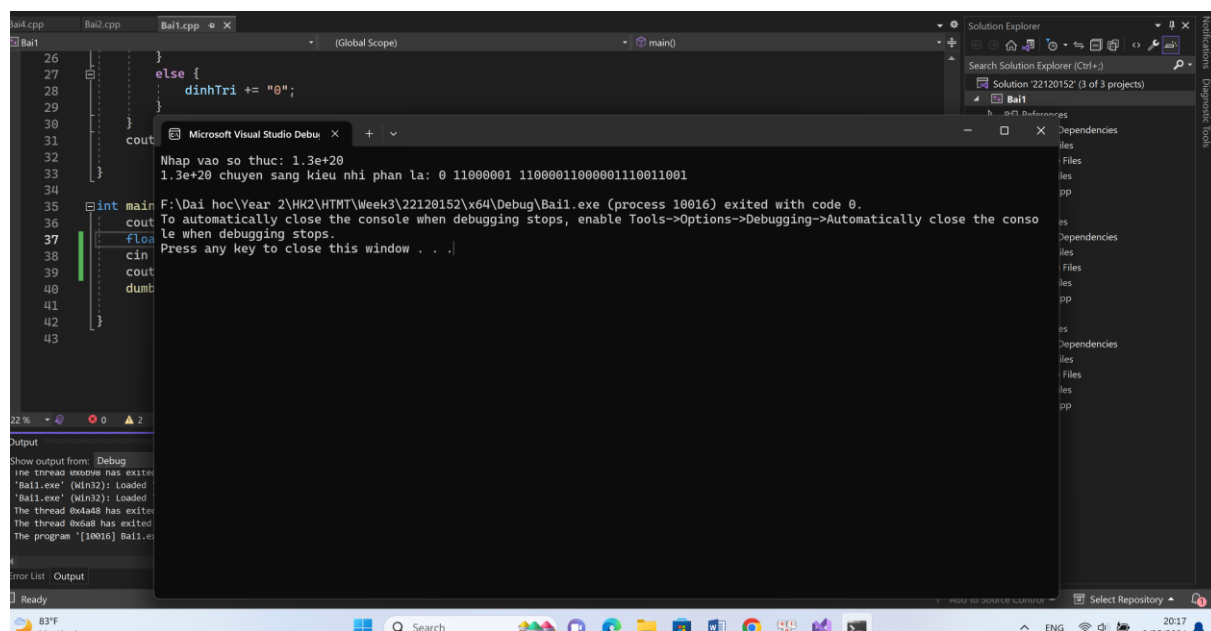
Ready

### 2. Bài 2



### 3. Bài 3

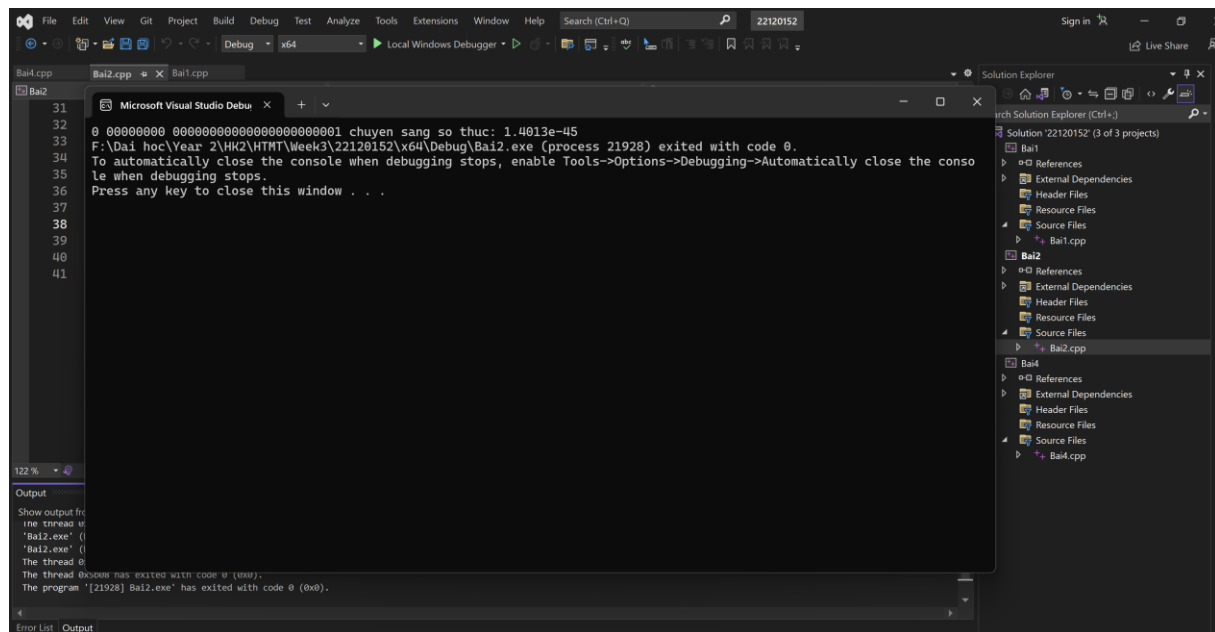
- **Câu 1:  $1.3E+20$  có biểu diễn nhị phân ra sao?**



- **Câu 2: Số float nhỏ nhất lớn hơn 0 là số nào? Biểu diễn nhị phân của nó?**

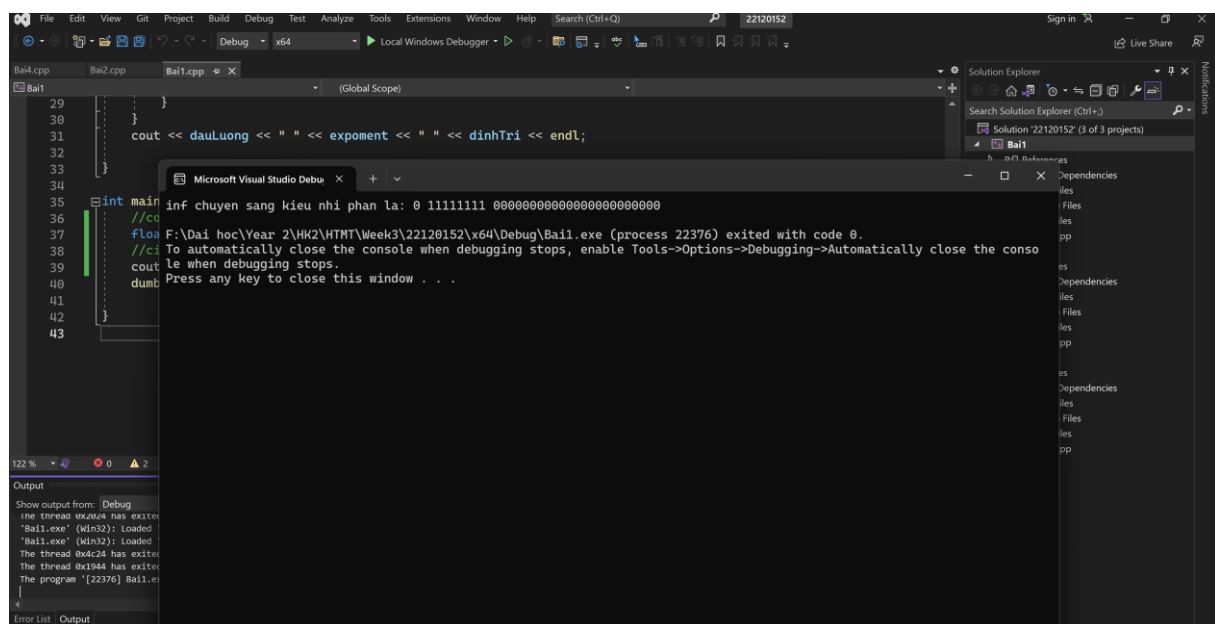
Với 32 bit lưu trữ, số float nhỏ nhất là :

0 00000000 000000000000000000000001

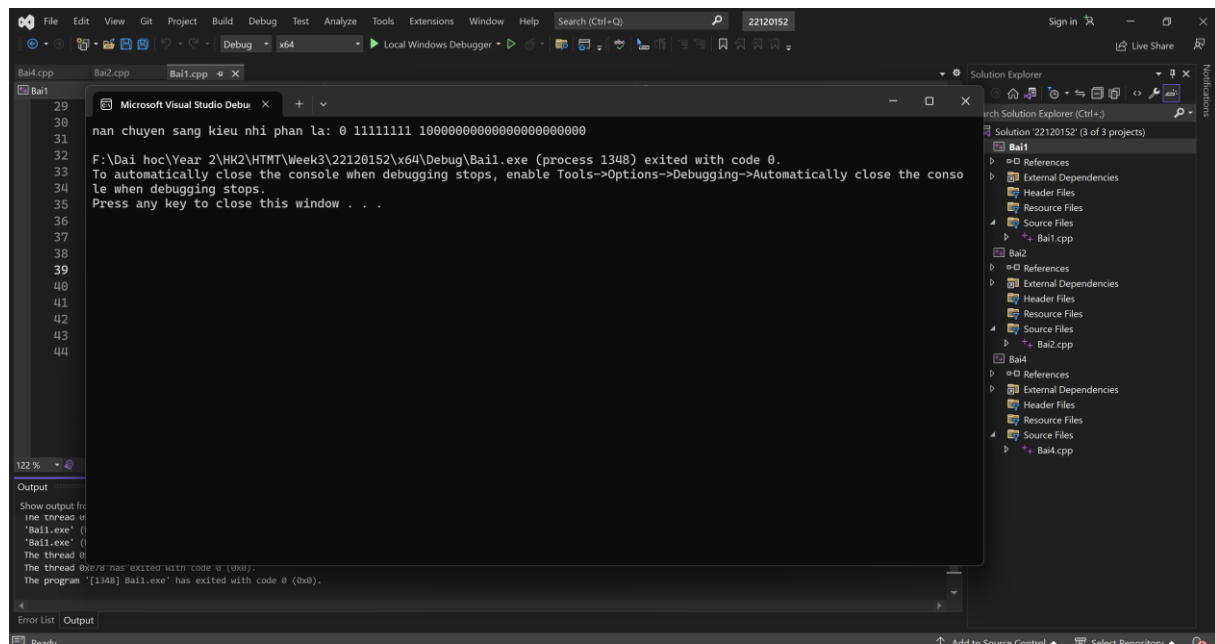


- **Câu 3: Những trường hợp nào tạo ra các số đặc biệt (kiểu float) (viết chương trình thử nghiệm và giải thích kết quả):**

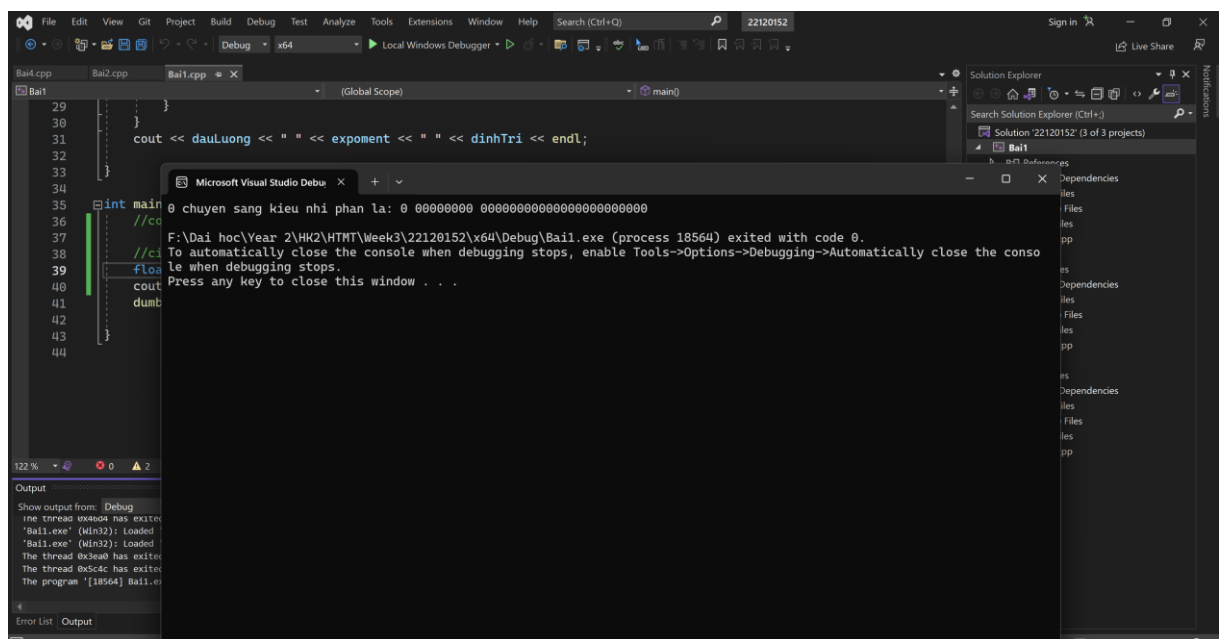
+ Số vô cùng: gồm Exponent = 111...1 (toàn bit 1), Significand = 0



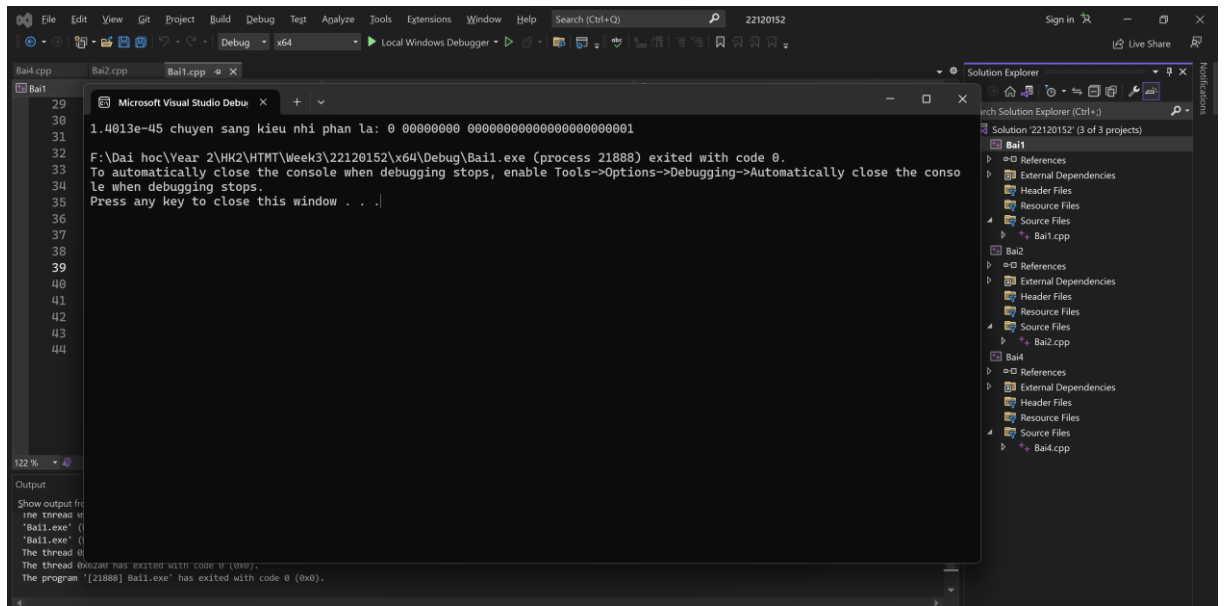
+ Số báo lỗi NaN: gồm Exponent = 111...1 (toàn bit 1), Significand != 0



+ Số 0: gồm Exponent = 0, Significand = 0

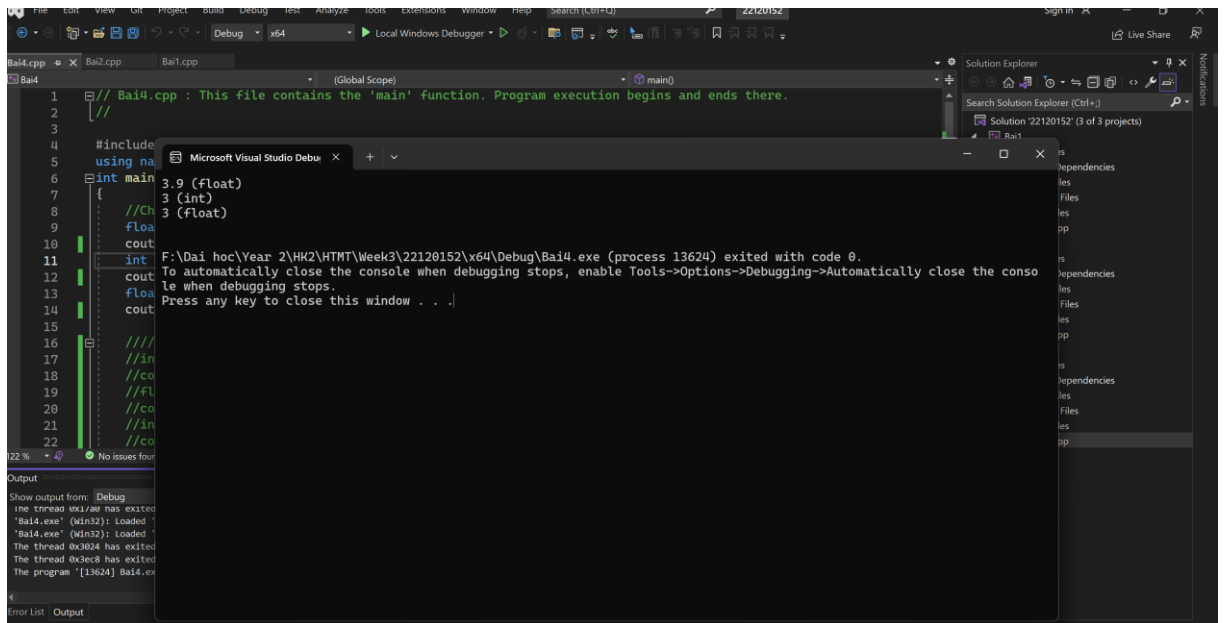


+ Số không thể chuẩn hóa: gồm Exponent = 0, Significand != 0



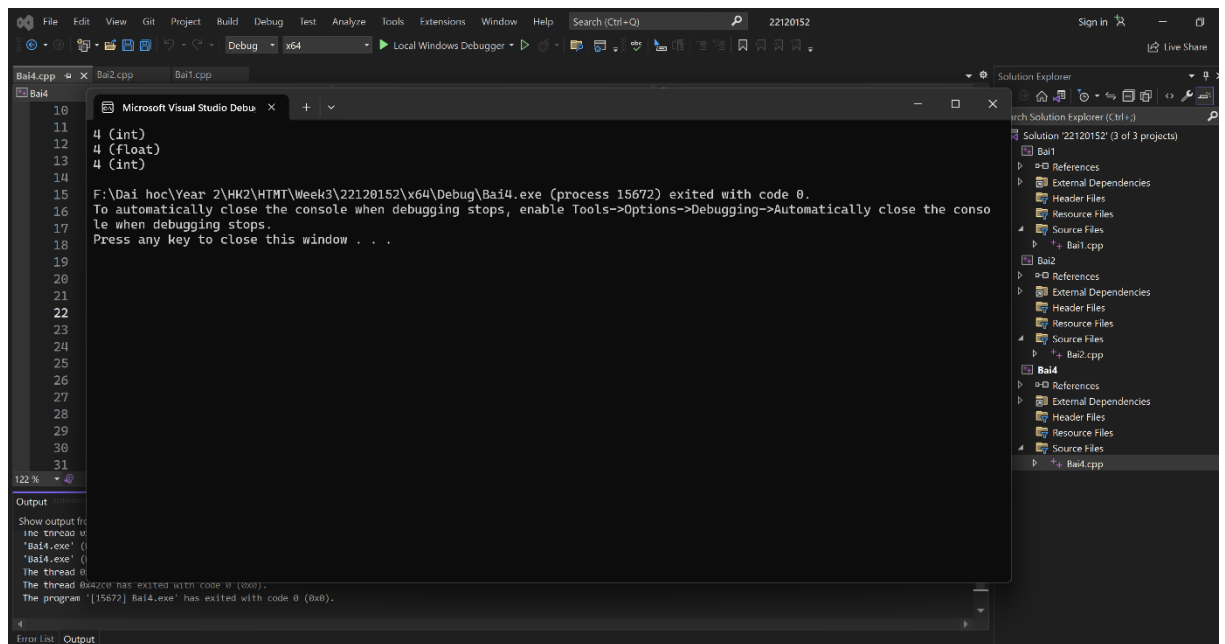
#### 4. Bài 4

### Câu 1: Chuyển đổi float -> int -> float. Kết quả như ban đầu ?



Kết quả không như ban đầu vì biến y chỉ lưu phần nguyên của biến x (3), biến z lưu kết quả của biến y nên biến z chỉ chứa phần nguyên của biến x ban đầu dẫn đến kết quả khác.

### Câu 2: Chuyển đổi int -> float -> int. Kết quả như ban đầu ?



Kết quả như nhau vì kiểu int không có phần thập phân nên khi ép kiểu về float thì biến float sẽ chứa cả phần nguyên của kiểu int và phần thập phân bằng 0

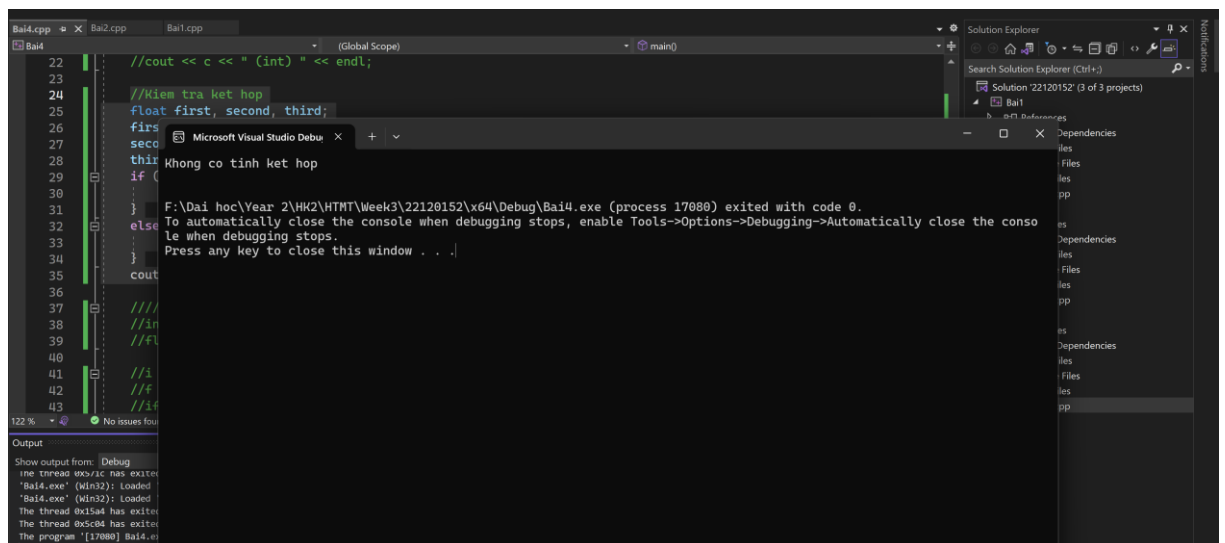
Khi ép lại kiểu int thì phần thập phân bằng 0 nên sẽ không ảnh hưởng kết quả

### Câu 3:

```

//Kiểm tra kết hợp
float first, second, third;
first = 1.2;
second = 1.4;
third = 1.5;
if ((first + second) + third == first + (second + third)) {
    cout << "Có tính kết hợp" << endl;
}
else {
    cout << "Không có tính kết hợp" << endl;
}
cout << endl;

```



Vì đối với kiểu float phân định trị có 23 bit do đó khi phân thập phân của các số được biểu diễn dưới dạng nhị phân mà xuất hiện các bit được lặp đi lặp lại thì sẽ không thể chứa hết các bit đó, dẫn đến sai số.

#### - Câu 4:

Với i là biến kiểu int, f là biến kiểu float

4. `i = (int) (3.14159 * f);`

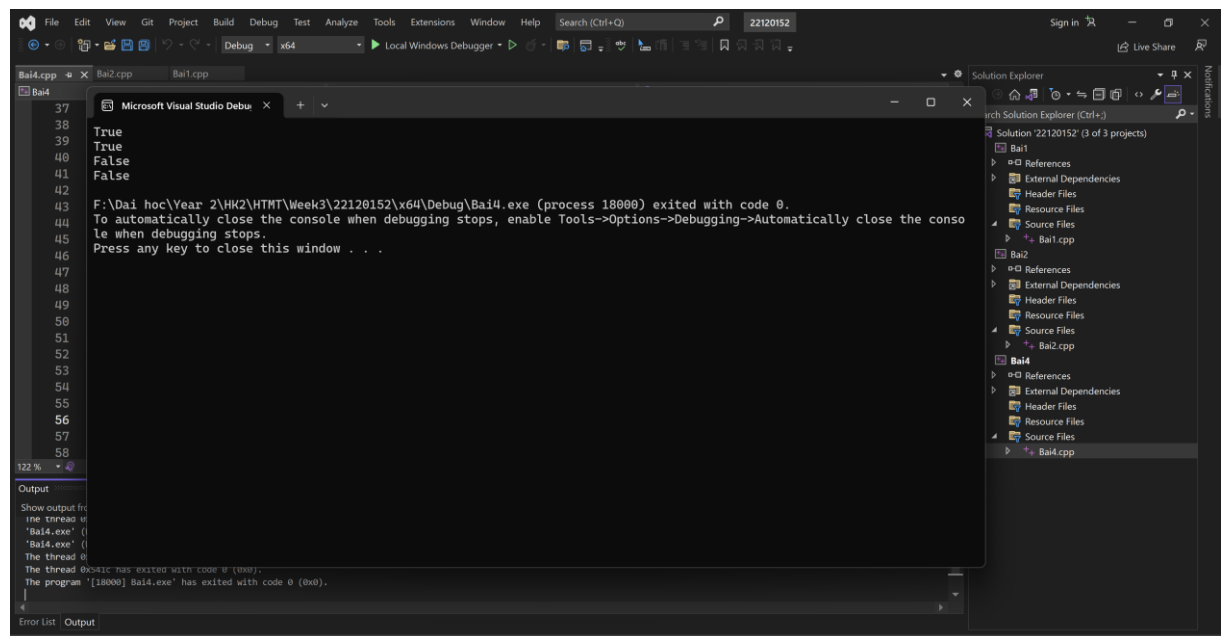
5. `f = f + (float) i;`

6. `if (i == (int)((float) i)) { printf("true");`

7. `if (i == (int)((double) i)) { printf("true"); }`

8. `if (f == (float)((int) f)) { printf("true"); }`

9. `if (f == (double)((int) f)) { printf("true"); }`



Với i là biến kiểu int, f là biến kiểu float

- Câu lệnh `i = (int) (3.14159 * f);` sẽ cho một số nguyên là phần nguyên của  $3.14159 * f$
- Câu lệnh `f = f + (float) i;` sẽ cho 1 số float là  $f + (float)i$ , nhưng vì i đã là số nguyên nên  $(float)i$  sẽ được cộng vào phần nguyên của f
- `if (i == (int)((float) i)) { printf("true");` vì i là số nguyên nên khi ép kiểu về float xong ép lại về int (Câu 2) thì kết quả không thay đổi, trả ra true
- `if (i == (int)((double) i)) { printf("true"); }` tương tự với kiểu ép về double
- `if (f == (float)((int) f)) { printf("true"); }` Lúc này f được ép về int xong ép về float (Câu 1) thì chỉ trả về phần nguyên của f, trả về false
- `if (f == (double)((int) f)) { printf("true"); }` tương tự ý trên với kiểu ép là double

