

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



CSC10009 - Hệ thống máy tính

BÁO CÁO BÀI TẬP

Khảo sát số chấm động của các ngôn ngữ lập trình
trên nền kiến trúc x86

Họ tên
Nguyễn Lê Hồ Anh Khoa

MSSV
23127211

Giảng viên hướng dẫn
Lê Viết Long

Ngày 28 tháng 10 năm 2024

Mục lục

1	Thông tin sinh viên	2
2	Đánh giá	2
2.1	Bảng tự đánh giá các yêu cầu đã hoàn thành	2
2.2	Đánh giá tổng thể mức độ hoàn thành của bài nộp	3
3	Kết quả và giải thích	4
3.1	Bài 1	4
3.2	Bài 2	4
3.3	Bài 3	4
3.3.1	Zero (Số 0)	5
3.3.2	Denormalized Number (Số không chuẩn hóa)	5
3.3.3	Infinity (Số vô cùng)	5
3.3.4	NaN (Not a Number - Số không phải là số)	5
3.4	Bài 4	6
3.4.1	Chuyển đổi float -> int -> float. Kết quả như ban đầu?	7
3.4.2	Chuyển đổi int -> float -> int. Kết quả như ban đầu?	7
3.4.3	Phép cộng số chấm động có tính kết hợp?	7
3.4.4	i = (int) (3.14159 * f);	7
3.4.5	f = f + (float) i;	7
3.4.6	if (i == (int)((float) i)) printf("true");	7
3.4.7	if (i == (int)((double) i)) printf("true");	7
3.4.8	if (f == (float)((int) f)) printf("true");	8
3.4.9	if (f == (double)((int) f)) printf("true");	8

1 Thông tin sinh viên

Họ và tên: Nguyễn Lê Hồ Anh Khoa. MSSV: 23127211. Lớp: 23CLC09

2 Đánh giá

2.1 Bảng tự đánh giá các yêu cầu đã hoàn thành

Bảng 1: Bảng tự đánh giá bài 1

STT	Yêu cầu	Mức độ hoàn thành
1	Viết chương trình nhập vào số chấm động. Hãy xuất ra biểu diễn nhị phân từng thành phần (dấu, phần mũ, phần trị) của số chấm động vừa nhập	100%
	Tổng cộng	100%

Bảng 2: Bảng tự đánh giá bài 2

STT	Yêu cầu	Mức độ hoàn thành
1	Viết chương trình nhập vào số chấm động. Hãy xuất ra biểu diễn nhị phân từng thành phần (dấu, phần mũ, phần trị) của số chấm động vừa nhập	100%
	Tổng cộng	100%

Bảng 3: Bảng tự đánh giá bài 3

STT	Yêu cầu	Mức độ hoàn thành
1	Biểu diễn nhị phân số $1.3E+20$	100%
2	Biểu diễn nhị phân số float nhỏ nhất lớn hơn 0	100%
3	Những trường hợp tạo ra các số đặc biệt	100%
	Tổng cộng	100%

Bảng 4: Bảng tự đánh giá bài 4

STT	Yêu cầu	Mức độ hoàn thành
1	Khảo sát chuyển đổi float -> int -> float	100%
2	Khảo sát chuyển đổi int -> float -> int	100%
3	Khảo sát tính kết hợp của phép cộng	100%
4	Khảo sát <code>i = (int) (3.14159 * f)</code>	100%
5	Khảo sát <code>f = f + (float) i;</code>	100%
6	Khảo sát <code>if (i == (int)((float) i)) printf("true");</code>	100%
7	Khảo sát <code>if (i == (int)((double) i)) printf("true");</code>	100%
8	Khảo sát <code>if (f == (float)((int) f)) printf("true");</code>	100%
9	Khảo sát <code>if (f == (double)((int) f)) printf("true");</code>	100%
	Tổng cộng	100%

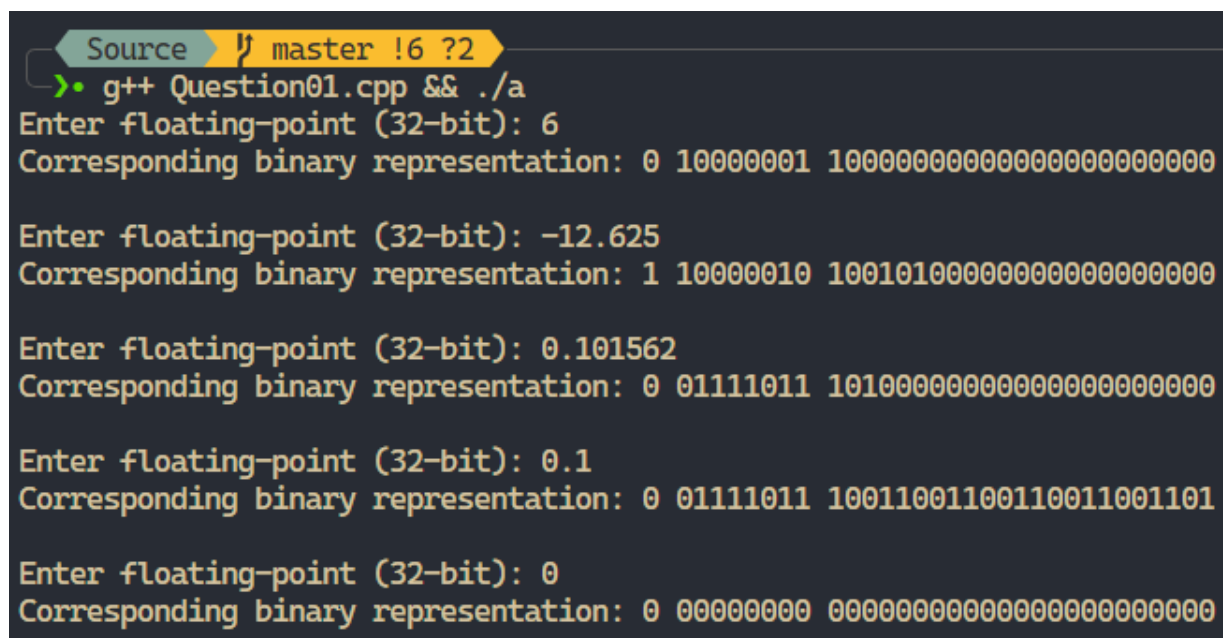
2.2 Đánh giá tổng thể mức độ hoàn thành của bài nộp

Bài nộp đã hoàn thành đầy đủ các yêu cầu đề ra trong bài tập. Tất cả các yêu cầu đều đã được cài đặt và kiểm thử thành công. Tổng thể, bài nộp đã hoàn thành 100% các yêu cầu đề ra.

3 Kết quả và giải thích

3.1 Bài 1

Viết chương trình nhập vào số chấm động. Hãy xuất ra biểu diễn nhị phân từng thành phần (dấu, phần mũ, phần trị) của số chấm động vừa nhập.



```
Source master !6 ?2
> g++ Question01.cpp && ./a
Enter floating-point (32-bit): 6
Corresponding binary representation: 0 10000001 100000000000000000000000

Enter floating-point (32-bit): -12.625
Corresponding binary representation: 1 10000010 100101000000000000000000

Enter floating-point (32-bit): 0.101562
Corresponding binary representation: 0 01111011 101000000000000000000000

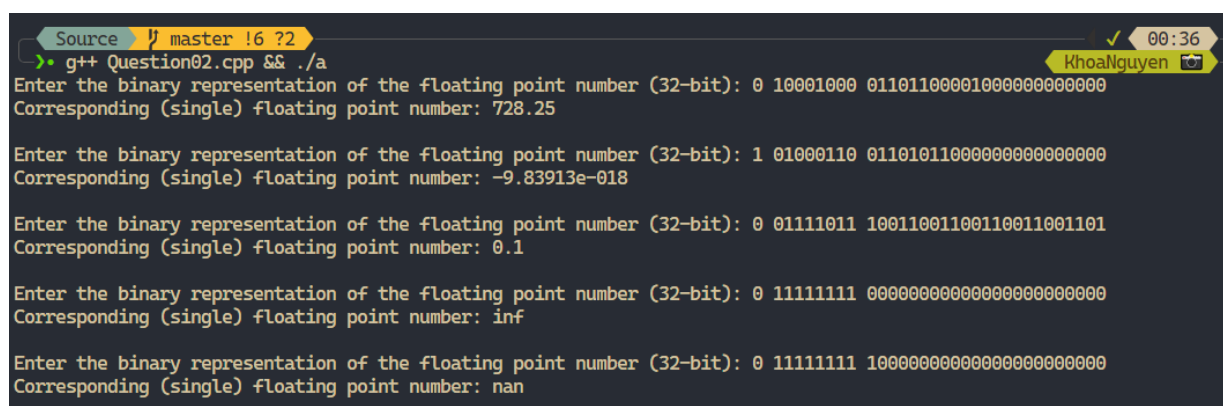
Enter floating-point (32-bit): 0.1
Corresponding binary representation: 0 01111011 10011001100110011001101

Enter floating-point (32-bit): 0
Corresponding binary representation: 0 00000000 000000000000000000000000
```

Hình 1: Chụp màn hình kết quả bài 1.

3.2 Bài 2

Viết chương trình nhập vào biểu diễn nhị phân của số chấm động. Hãy xuất ra biểu diễn thập phân tương ứng



```
Source master !6 ?2
> g++ Question02.cpp && ./a
Enter the binary representation of the floating point number (32-bit): 0 10001000 011011000010000000000000
Corresponding (single) floating point number: 728.25

Enter the binary representation of the floating point number (32-bit): 1 01000110 011010110000000000000000
Corresponding (single) floating point number: -9.83913e-018

Enter the binary representation of the floating point number (32-bit): 0 01111011 10011001100110011001101
Corresponding (single) floating point number: 0.1

Enter the binary representation of the floating point number (32-bit): 0 11111111 000000000000000000000000
Corresponding (single) floating point number: inf

Enter the binary representation of the floating point number (32-bit): 0 11111111 100000000000000000000000
Corresponding (single) floating point number: nan
```

Hình 2: Chụp màn hình kết quả bài 2.

3.3 Bài 3

Dùng 2 hàm đã viết để khảo sát các câu hỏi:

- $1.3E+20$ có biểu diễn nhị phân ra sao.
- Số float nhỏ nhất lớn hơn 0 là số nào? Biểu diễn nhị phân của nó?
- Những trường hợp nào tạo ra các số đặc biệt (kiểu float) (viết chương trình thử nghiệm và giải thích kết quả)

3.3.1 Zero (Số 0)

Số 0 trong biểu diễn số chấm động có hai dạng: số 0 dương và số 0 âm. Cả hai đều có phần mũ và phần trị bằng 0, chỉ khác nhau ở bit dấu.

- Số 0 dương: bit dấu = 0, phần mũ = 0, phần trị = 0.
- Số 0 âm: bit dấu = 1, phần mũ = 0, phần trị = 0.

Ví dụ, biểu diễn nhị phân của số 0 dương trong chuẩn IEEE 754 (32-bit) là:

0 00000000 000000000000000000000000

3.3.2 Denormalized Number (Số không chuẩn hóa)

Số không chuẩn hóa là các số rất nhỏ gần bằng 0, được sử dụng để biểu diễn các giá trị nhỏ hơn giá trị nhỏ nhất có thể biểu diễn được bằng số chuẩn hóa. Trong biểu diễn số không chuẩn hóa, phần mũ bằng 0 nhưng phần trị khác 0.

- Bit dấu: 0 hoặc 1.
- Phần mũ: 0.
- Phần trị: khác 0.

Ví dụ, biểu diễn nhị phân của một số không chuẩn hóa trong chuẩn IEEE 754 (32-bit) là:

0 00000000 000000000000000000000001

3.3.3 Infinity (Số vô cùng)

Số vô cùng được sử dụng để biểu diễn các giá trị vượt quá phạm vi của số chấm động. Có hai dạng: vô cùng dương và vô cùng âm.

- Vô cùng dương: bit dấu = 0, phần mũ = 255 (toàn bit 1), phần trị = 0.
- Vô cùng âm: bit dấu = 1, phần mũ = 255 (toàn bit 1), phần trị = 0.

Ví dụ, biểu diễn nhị phân của số vô cùng dương trong chuẩn IEEE 754 (32-bit) là:

0 11111111 000000000000000000000000

3.3.4 NaN (Not a Number - Số không phải là số)

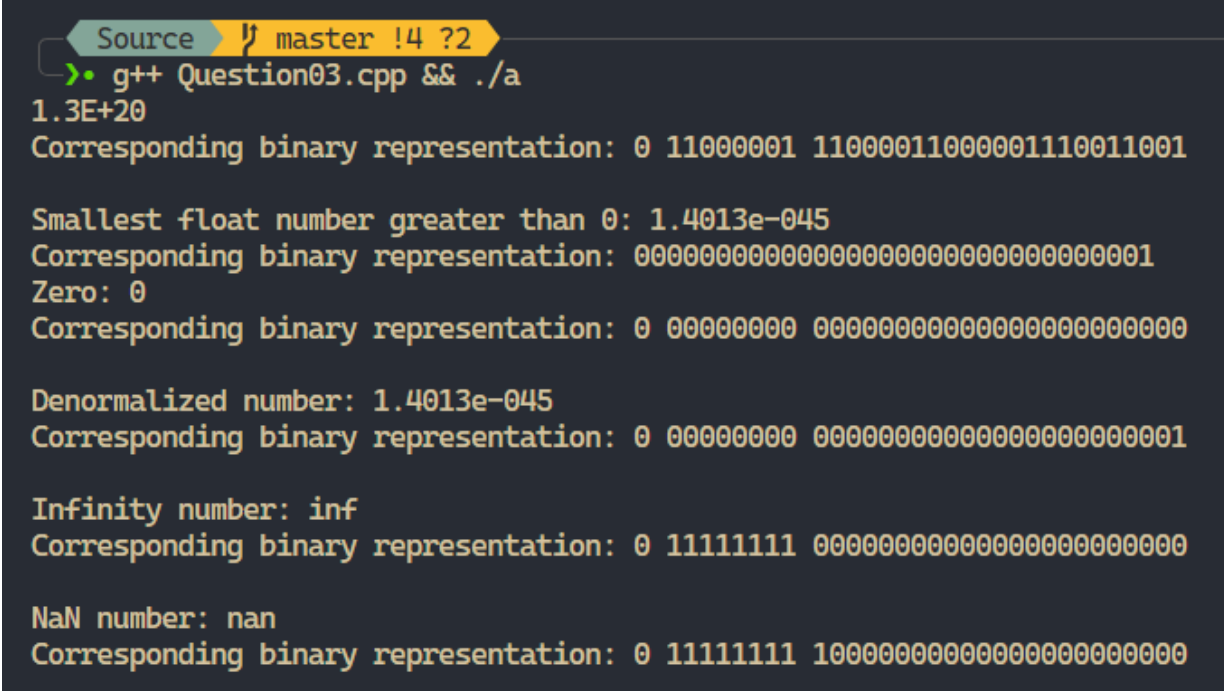
NaN được sử dụng để biểu diễn các giá trị không hợp lệ hoặc không xác định, chẳng hạn như kết quả của phép chia 0 cho 0 hoặc căn bậc hai của số âm. Có hai loại NaN: NaN

yên lặng (quiet NaN) và NaN báo lỗi (signaling NaN).

- Bit dấu: 0 hoặc 1.
- Phần mũ: 255 (toàn bit 1).
- Phần trị: khác 0.

Ví dụ, biểu diễn nhị phân của một NaN trong chuẩn IEEE 754 (32-bit) là:

0 11111111 100000000000000000000000



```
Source master !4 ?2
> g++ Question03.cpp && ./a
1.3E+20
Corresponding binary representation: 0 11000001 11000011000001110011001

Smallest float number greater than 0: 1.4013e-045
Corresponding binary representation: 00000000000000000000000000000001
Zero: 0
Corresponding binary representation: 0 00000000 000000000000000000000000

Denormalized number: 1.4013e-045
Corresponding binary representation: 0 00000000 000000000000000000000001

Infinity number: inf
Corresponding binary representation: 0 11111111 000000000000000000000000

NaN number: nan
Corresponding binary representation: 0 11111111 100000000000000000000000
```

Hình 3: Chụp màn hình kết quả bài 3.

3.4 Bài 4

Khảo sát các trường hợp sau đây (viết chương trình thử nghiệm và giải thích kết quả):

- Chuyển đổi float -> int -> float. Kết quả như ban đầu ?
- Chuyển đổi int -> float -> int. Kết quả như ban đầu ?
- Phép cộng số chấm động có tính kết hợp ?
- `i = (int) (3.14159 * f);`
- `f = f + (float) i;`
- `if (i == (int)((float) i)) printf("true");` ‘
- `if (i == (int)((double) i)) printf("true");`
- `if (f == (float)((int) f)) printf("true");`

- `if (f == (double)((int) f)) printf("true");`

3.4.1 Chuyển đổi float -> int -> float. Kết quả như ban đầu?

Khi chuyển đổi từ 'float' sang 'int', phần thập phân của số 'float' sẽ bị loại bỏ, chỉ giữ lại phần nguyên. Do đó, khi chuyển đổi ngược lại từ 'int' sang 'float', kết quả sẽ không giống như ban đầu nếu số 'float' ban đầu có phần thập phân khác 0. Ví dụ, nếu giá trị ban đầu là '3.14159', sau khi chuyển đổi sang 'int' sẽ thành '3', và khi chuyển đổi ngược lại sang 'float' sẽ thành '3.0'. Kết quả này không giống như giá trị ban đầu '3.14159'.

3.4.2 Chuyển đổi int -> float -> int. Kết quả như ban đầu?

Khi chuyển đổi từ 'int' sang 'float', giá trị của 'int' sẽ được giữ nguyên và thêm phần thập phân '.0'. Khi chuyển đổi ngược lại từ 'float' sang 'int', phần thập phân sẽ bị loại bỏ, giữ lại phần nguyên. Do đó, kết quả sẽ giống như ban đầu nếu số 'float' không có phần thập phân khác 0. Ví dụ, nếu giá trị ban đầu là '42', sau khi chuyển đổi sang 'float' sẽ thành '42.0', và khi chuyển đổi ngược lại sang 'int' sẽ thành '42'. Kết quả này giống như giá trị ban đầu '42'.

3.4.3 Phép cộng số chấm động có tính kết hợp?

Phép cộng số chấm động không có tính kết hợp do các vấn đề về độ chính xác và làm tròn trong biểu diễn số chấm động. Ví dụ, với ba số 'a = 1.0f', 'b = 1e10f', và 'c = -1e10f', kết quả của '(a + b) + c' và 'a + (b + c)' có thể khác nhau do thứ tự thực hiện phép toán ảnh hưởng đến độ chính xác của kết quả. Trong trường hợp này, '(a + b) + c' có thể cho ra kết quả khác với 'a + (b + c)'.

3.4.4 `i = (int) (3.14159 * f);`

Khi thực hiện phép toán '3.14159 * f', kết quả sẽ là một số 'float'. Sau đó, khi chuyển đổi kết quả này sang 'int', phần thập phân sẽ bị loại bỏ, chỉ giữ lại phần nguyên. Ví dụ, nếu 'f = 2.0f', thì '3.14159 * 2.0' sẽ là '6.28318', và khi chuyển đổi sang 'int' sẽ thành '6'.

3.4.5 `f = f + (float) i;`

Khi thực hiện phép toán 'f = f + (float) i', giá trị của 'i' sẽ được chuyển đổi sang 'float' trước khi thực hiện phép cộng. Ví dụ, nếu 'f = 2.0f' và 'i = 3', thì '2.0 + 3.0' sẽ là '5.0'.

3.4.6 `if (i == (int)((float) i)) printf("true");`

Khi thực hiện phép toán 'i == (int)((float) i)', giá trị của 'i' sẽ được chuyển đổi sang 'float' và sau đó chuyển đổi ngược lại sang 'int'. Kết quả sẽ giống như ban đầu nếu giá trị của 'i' không có phần thập phân khi chuyển đổi sang 'float'. Ví dụ, nếu 'i = 3', thì '3' sẽ được chuyển đổi sang '3.0' và sau đó chuyển đổi ngược lại thành '3', do đó điều kiện sẽ đúng và in ra 'true'.

3.4.7 `if (i == (int)((double) i)) printf("true");`

Khi thực hiện phép toán 'i == (int)((double) i)', giá trị của 'i' sẽ được chuyển đổi sang 'double' và sau đó chuyển đổi ngược lại sang 'int'. Kết quả sẽ giống như ban đầu nếu giá

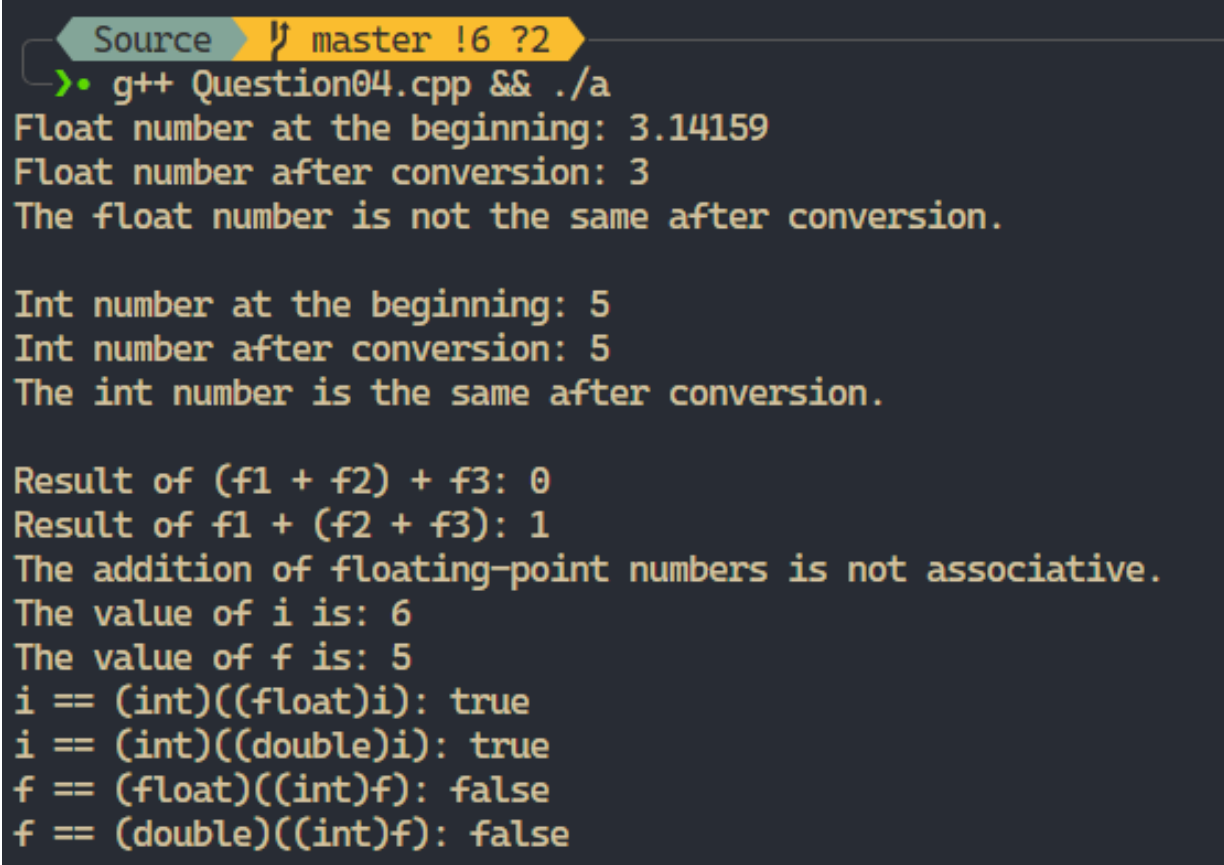
trị của 'i' không có phần thập phân khi chuyển đổi sang 'double'. Ví dụ, nếu 'i = 3', thì '3' sẽ được chuyển đổi sang '3.0' và sau đó chuyển đổi ngược lại thành '3', do đó điều kiện sẽ đúng và in ra 'true'.

3.4.8 if (f == (float)((int) f)) printf("true");

Khi thực hiện phép toán 'f == (float)((int) f)', giá trị của 'f' sẽ được chuyển đổi sang 'int', loại bỏ phần thập phân, và sau đó chuyển đổi ngược lại sang 'float'. Kết quả sẽ giống như ban đầu nếu giá trị của 'f' không có phần thập phân khác 0. Ví dụ, nếu 'f = 3.5f', thì '3.5' sẽ được chuyển đổi sang '3' và sau đó chuyển đổi ngược lại thành '3.0', do đó điều kiện sẽ sai và in ra 'false'.

3.4.9 if (f == (double)((int) f)) printf("true");

Khi thực hiện phép toán 'f == (double)((int) f)', giá trị của 'f' sẽ được chuyển đổi sang 'int', loại bỏ phần thập phân, và sau đó chuyển đổi ngược lại sang 'double'. Kết quả sẽ giống như ban đầu nếu giá trị của 'f' không có phần thập phân khác 0. Ví dụ, nếu 'f = 3.5f', thì '3.5' sẽ được chuyển đổi sang '3' và sau đó chuyển đổi ngược lại thành '3.0', do đó điều kiện sẽ sai và in ra 'false'.



```
Source master !6 ?2
> g++ Question04.cpp && ./a
Float number at the beginning: 3.14159
Float number after conversion: 3
The float number is not the same after conversion.

Int number at the beginning: 5
Int number after conversion: 5
The int number is the same after conversion.

Result of (f1 + f2) + f3: 0
Result of f1 + (f2 + f3): 1
The addition of floating-point numbers is not associative.
The value of i is: 6
The value of f is: 5
i == (int)((float)i): true
i == (int)((double)i): true
f == (float)((int)f): false
f == (double)((int)f): false
```

Hình 4: Chụp màn hình kết quả bài 4.