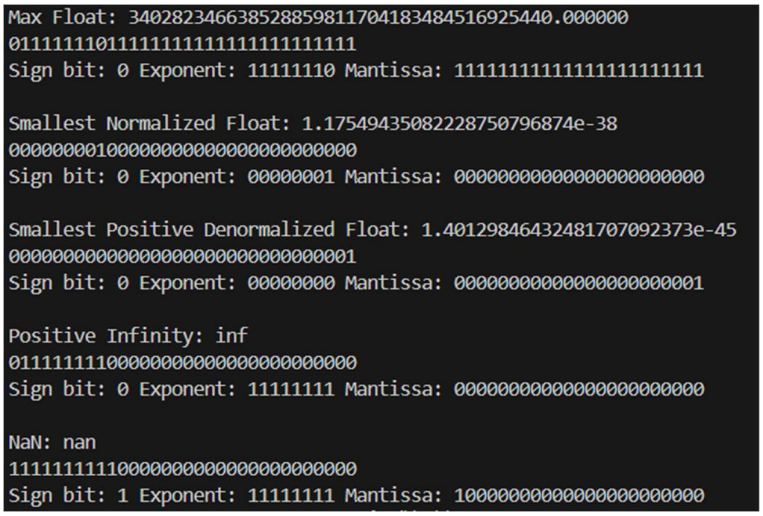
Alvin Tan Yee Jia E94115011 資訊 115

2024 NCKU Program Design I Homework 4 Question 3
(3) We first run the code to see what the output show:

```c
#include <stdio.h>
#include <string.h>

void printBinary(unsigned int num) {
    for (int i = sizeof(num) * 8 - 1; i >= 0; i--) {
        printf("%u", (num >> i) & 1);
    }
    printf("\n");
}

void printBinarySeperated(unsigned int num) {
    unsigned int sign = (num >> 31) & 1;
    unsigned int exponent = (num >> 23) & 0xFF;
    unsigned int mantissa = num & 0x7FFFFF;

    printf("Sign bit: %u ", sign);
    printf("Exponent: ");
    for (int i = 7; i >= 0; i--) {
        printf("%u", (exponent >> i) & 1);
    }

    printf(" Mantissa: ");
    for (int i = 22; i >= 0; i--) {
        printf("%u", (mantissa >> i) & 1);
    }
    printf("\n");
}

int main() {
    float maxFloat = 3.4028235e+38f;
    printf("Max Float: %f\n", maxFloat);
    printBinary(*(unsigned int*)&maxFloat);
    printBinarySeperated(*(unsigned int*)&maxFloat);
    printf("\n");

    float minNormFloat = 1.17549435e-38f;
    printf("Smallest Normalized Float: %.23e\n", minNormFloat);
    printBinary(*(unsigned int*)&minNormFloat);
    printBinarySeperated(*(unsigned int*)&minNormFloat);
    printf("\n");

    float minDenormFloat = 1.401298464324817e-45f;
    printf("Smallest Positive Denormalized Float: %.23e\n", minDenormFloat);
    printBinary(*(unsigned int*)&minDenormFloat);
    printBinarySeperated(*(unsigned int*)&minDenormFloat);
    printf("\n");

    float infFloat = 1.0f / 0.0f;
    printf("Positive Infinity: %f\n", infFloat);
    printBinary(*(unsigned int*)&infFloat);
    printBinarySeperated(*(unsigned int*)&infFloat);
    printf("\n");

    float nanFloat = 0.0f / 0.0f;
    printf("NaN: %f\n", nanFloat);
    printBinary(*(unsigned int*)&nanFloat);
    printBinarySeperated(*(unsigned int*)&nanFloat);

    return 0;
}
```

Output:



So the code above verifies that:

a) Maximum Float Value
The maximum representable float value in C is approximately **3.4028235 × 10^38**.

binary representation (IEEE 754):
Sign bit: 0 (positive)
Exponent: 11111110 (which is 254 in decimal, bias is 127)
Mantissa: 11111111111111111111

There are two types of minimum float value, later I will explain what are the differences.
b) **Minimum Float Value** (smallest positive normalized float)
The minimum positive normalized float value in C is approximately **1.17549435 × 10^-38**.

binary representation (IEEE 754):
Sign bit: 0 (positive)
Exponent: 00000001
Mantissa: 00000000000000000000

c) **Minimum Float Value** (smallest positive denormalized float)
The smallest positive denormalized float value is approximately **1.401298464324817e-45**.

binary representation (IEEE 754):
Sign bit: 0 (positive)
Exponent: 00000000 (which indicates a denormalized number)
Mantissa: 00000000000000000001

d) **Infinity (Inf)**
An example of infinity that I have given is infFloat = 1.0f / 0.0f.
Infinity is represented by setting the exponent to all 1s and the mantissa to all 0s.

binary representation (IEEE 754):
Sign bit: 0 (positive)
Exponent: 11111111
Mantissa: 00000000000000000000000

e) **Not a Number (NaN)**
An example of NaN that I have given is nanFloat = 0.0f / 0.0f.
NaN is represented by setting the exponent to all 1s and the mantissa to any non-zero value.

binary representation (IEEE 754):
Sign bit: 1 (both signs are valid and indicate that the operation produced an undefined result)
Exponent: 11111111
Mantissa: 10000000000000000000000

QnA:

1)What are the differences between denormalized float and normalized float?

Ans:
Normalized Floats

A normalized float is a floating-point number where the leading bit of the mantissa is implicitly 1, mantissa is assumed to be in the form 1.xxxxxx. This means that the representation takes full advantage of the available bits to provide maximum precision for larger values.

Bit Structure:

Sign Bit: 1 bit (0 for positive, 1 for negative)

Exponent: Biased exponent, which is offset by a bias (127 for 32-bit floats) to allow for both positive and negative exponents.

Mantissa (Significand): A fraction that has an implicit leading 1 (not stored explicitly).

Denormalized Floats

A denormalized float (or subnormal float) allows representation of values that are very close to zero, effectively filling the gap around zero that normalized floats cannot represent.

Bit Structure:

Sign Bit: 1 bit

Exponent: All bits are set to zero, indicating a denormalized number.

Mantissa: The significand has no implicit leading 1 and can start with zero, which means they can only represent numbers as 0.xxxxx. This means that denormalized floats can only represent very small numbers with much less precision.

Example :

-A normalized float might represent the mantissa as 1.00000000000000000000000.

-A denormalized float might represent the mantissa as 0.00000000000000000000001

∴ In essence, normalized floats are designed for most standard operations where values are larger and more precise, while denormalized floats provide a way to represent very small values, ensuring that calculations can continue smoothly without abrupt underflows to zero. This distinction is critical in floating-point arithmetic, especially in numerical applications.

2) For the **Maximum Float Value above** 0 11111110 11111111111111111111111 (**3.4028235 × 10^38**), the exponent seems to not at its fullest, why can't the largest be 0 11111111 11111111111111111111111?

Ans: If we look at the formula below

$$\text{value} = (-1)^{\text{sign}} \times (1 + \text{mantissa}) \times 2^{\text{exponent}}$$

Convert the exponent from binary to decimal: 11111110 = 254
Subtract the bias (127) from the exponent: 254 – 127 = 127
so we know the maximum bias can be 128 for exponent 11111111

But as the Inf and NaN shown above, the exponent 11111111 is used to handle these two different cases, so even if we input a binary representation with exponent 11111111, despite the calculation is larger than **3.4028235 × 10^38, but the computer will convert this to either Inf or NaN as output depending on the mantissa.**