Numerical Method

Dept. Material Science & Engineering
Undergraduate Course (4th year)
Chang Jae Yu

Homework #1 - 1

- 0.00001을 백만 번 더하면서 매 십만 번째마다 결과를 출력하시오.

```c
#include <stdio.h>
int main(){
    double sum = 0;
    int count = 1;
    printf(" <Homework 1 - 1 : 0.00001을 백만번 더했을 때>₩n₩n");

    while(count <= 1000000){
        sum = sum + 0.00001;
        if (count % 100000==0){
            printf(" %d 번째 :  %.16f₩n", count, sum);
        }
        count = count + 1;
    }
    return 0;
}
```

Algorithm

- Sum에 0.00001을 한 번 더할 때마다 count가 1씩 올라간다.
- Count가 100000의 배수일 때, printf로 결과가 출력되도록 한다.
- 64-bit 내에서의 소수를 표현하기 위해 sum을 double로 선언했다.

Homework #1 - 1

- 0.00001을 백만 번 더하면서 매 십만 번째마다 결과를 출력하시오.

```
<Homework 1 - 1 : 0.00001을 백만번 더했을 때>

100000번째      : 0.99999999999980838
200000번째      : 2.00000000000046350
300000번째      : 3.00000000000111862
400000번째      : 4.00000000000177369
500000번째      : 4.99999999998798792
600000번째      : 5.99999999999420215
700000번째      : 6.99999999999041638
800000번째      : 7.99999999998663061
900000번째      : 8.99999999998284483
1000000번째     : 9.99999999997905906
계속하려면 아무 키나 누르십시오 . . .
```

Results

- 일반적으로 0.00001을 100000의 배수만큼 더했을 때, 1~10의 자연수가 나온다.
- 하지만, 컴퓨터에서는 이를 정확하게 표현하지 못한다.

Homework #1 - 2

- 1을 백만 번 더하면서 매 십만 번째마다 결과/100000을 출력하시오.

```c
#include <stdio.h>
int main() {
    double sum = 0;
    int count = 1;

    printf("<Homework 1 - 2 : 1을 백만번 더했을 때>₩n₩n");
    while (count <= 1000000) {
        sum = sum + 1;
        if (count % 100000 == 0) {
            printf("%d  번째 :  %f₩n", count, sum/100000);
        }
        count = count + 1;
    }
    return 0;
}
```

Algorithm

- Sum에 1을 한 번 더할 때마다 count가 1씩 올라간다.
- Count가 100000의 배수일 때, sum을 100000으로 나눈 값을 printf로 출력되도록 한다.
- sum을 십만으로 나눈 값을 계산하기 위해 double로 선언했다.

Homework #1 - 2

- 1을 백만 번 더하면서 매 십만 번째마다 결과/100000을 출력하시오.

```
<Homework 1 - 2 : 1을 백만번 더했을 때>

100000   번째 :   1.000000
200000   번째 :   2.000000
300000   번째 :   3.000000
400000   번째 :   4.000000
500000   번째 :   5.000000
600000   번째 :   6.000000
700000   번째 :   7.000000
800000   번째 :   8.000000
900000   번째 :   9.000000
1000000  번째 :   10.000000
계속하려면 아무 키나 누르십시오 . . .
```

Results

- 일반적으로 1을 100000의 배수만큼 더하고 이를 100000으로 나눴을 때, 1~10의 자연수가 나온다.
- 컴퓨터에서 계산하였더니 같은 결과가 출력된다.

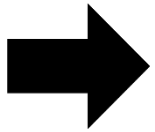
Homework #1 - 3

- 이 system이 몇 개의 bit로 가수(mantissa)를 표현하는지 판정하시오.

Computer에서의 수의 체계

$$(-1)^{s} \times 2^{c-1023} \times (1+f)$$

s: sign indicator
c: exponent
f: mantissa



Algorithm

- 1에 각각 1/2, 1/4, … $1/2^{f}$ 를 더하는 (sum) 행동을 반복한다.
- $1/2^{f}$ 의 값이 매우 작아져 mantissa로 인식을 못하게 되면, 결국 sum이 1로 수렴하게 된다.
- Sum이 1로 수렴할 때, 반복문의 loop에서 break하여 반복을 종료한다.

Homework #1 - 3

- 이 system이 몇 개의 bit로 가수(mantissa)를 표현하는지 판정하시오.(32-bit, float)

```c
#include <stdio.h>
int main() {
    float sum = 1;
    float sub = 1;
    int exponent=0;

    printf(" <Homework 1 - 3 : mantissa를 표현하는 bit의 갯수>₩n₩n");
    while (1) {
        exponent = exponent + 1;
        sub = sub / 2;
        sum = 1 + sub;
        printf(" %d %.18f₩n", exponent, sum);
        if (sum == 1) {
            printf("₩n Result : 32-bit (float) 에서는 %d-bit로 mantissa를 표현한다.₩n₩n", exponent - 1);
            break;
        }
    }
}
```

Homework #1 - 3

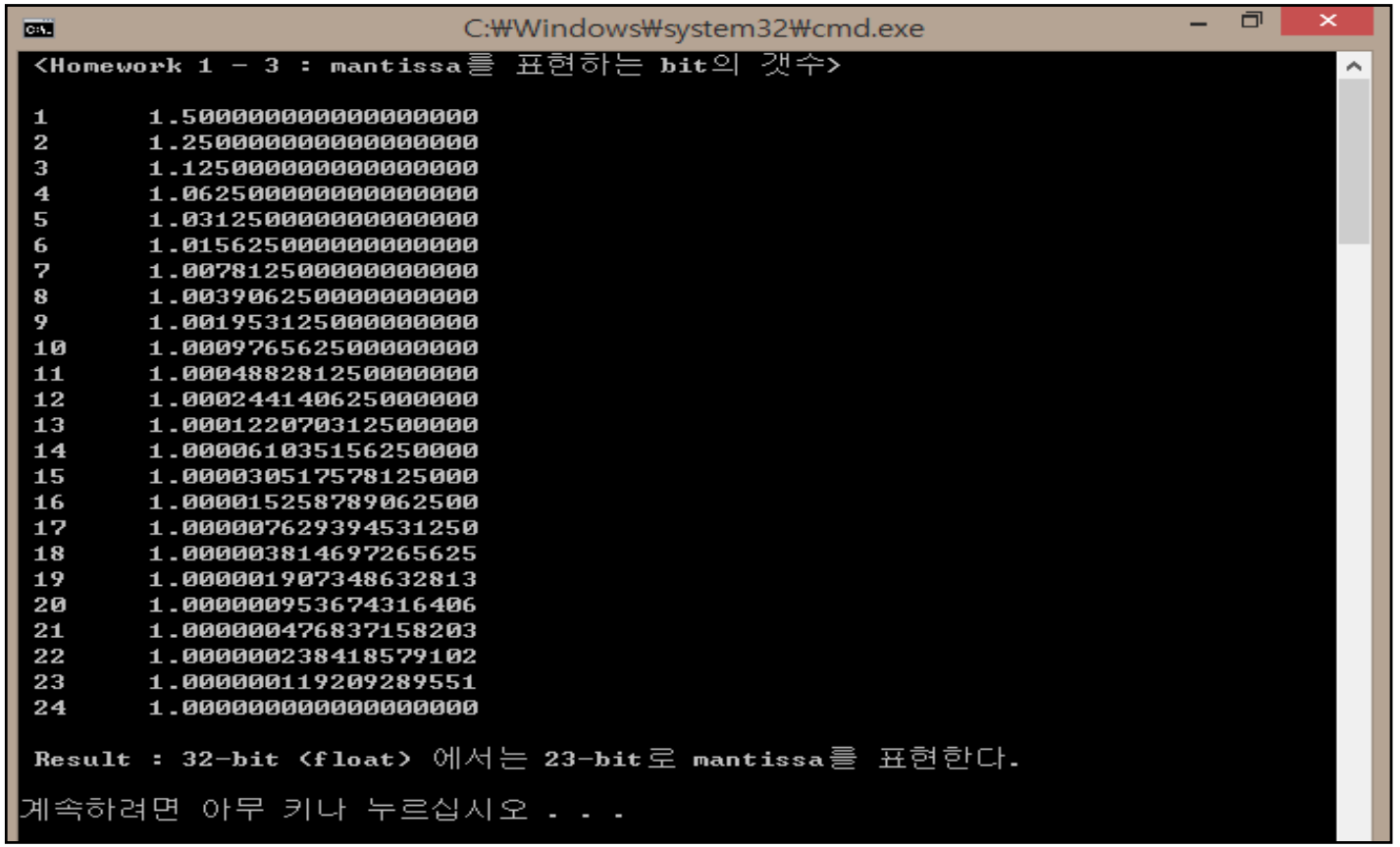
- 이 system이 몇 개의 bit로 가수(mantissa)를 표현하는지 판정하시오.(64-bit, double)

```c
#include <stdio.h>
int main() {
    double sum = 1;
    double sub = 1;
    int exponent=0;

    printf(" <Homework 1 - 3 : mantissa를 표현하는 bit의 갯수>₩n₩n");
    while (1) {
        exponent = exponent + 1;
        sub = sub / 2;
        sum = 1 + sub;
        printf(" %d %.18f    ", exponent, sum);
        if (exponent % 2 == 0) {
            printf("₩n");
        }
        if (sum == 1) {
            printf("₩n₩n Result : 64-bit (double) 에서는 %d-bit로 mantissa를 표현한다.₩n₩n", exponent - 1);
            break;
        }
    }
}
```

Homework #1 - 3

- 이 system이 몇 개의 bit로 가수(mantissa)를 표현하는지 판정하시오.

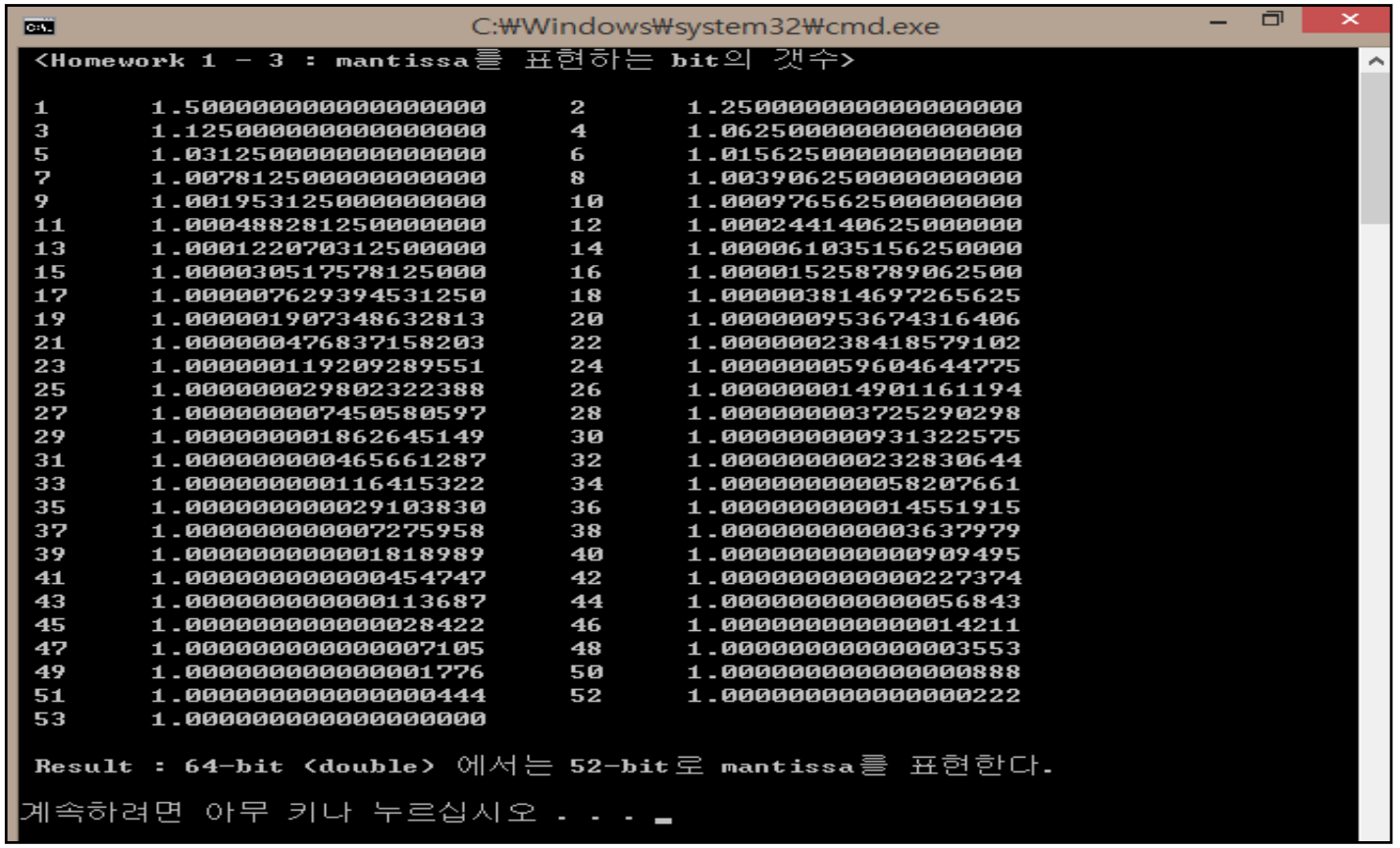


Results

- 32-bit에 해당하는 float로 선언했을 시 $1/2^{24}$ 이하의 값을 인식할 수 없었다.
- 이는 mantissa를 23개의 bit로 표현한다는 것을 알 수 있다.

Homework #1 - 3

- 이 system이 몇 개의 bit로 가수(mantissa)를 표현하는지 판정하시오.



Results

- 64-bit에 해당하는 double로 선언했을 시 $1/2^{53}$ 이하의 값을 인식할 수 없었다.
- 이는 mantissa를 52개의 bit로 표현한다는 것을 알 수 있다.

Compiler & Language

- Microsoft Visual Studio 2015 with C programming language.

Question & Answer