

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.

```
#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       : 0.9999999999980838
200000번째       : 2.00000000000046350
300000번째       : 3.000000000000111862
400000번째       : 4.000000000000177369
500000번째       : 4.99999999999798792
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.

```
#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.

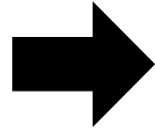
Computer에서의 수의 체계

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

s: sign indicator

c: exponent

f: mantissa



Algorithm

- 1에 각각 $1/2$, $1/4$, \dots $1/2^f$ 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)

```
#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)

```
#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.

```
C:\Windows\system32\cmd.exe
<Homework 1 - 3 : mantissa를 표현하는 bit의 갯수>
1      1.50000000000000000000
2      1.25000000000000000000
3      1.12500000000000000000
4      1.06250000000000000000
5      1.03125000000000000000
6      1.01562500000000000000
7      1.00781250000000000000
8      1.00390625000000000000
9      1.00195312500000000000
10     1.00097656250000000000
11     1.00048828125000000000
12     1.00024414062500000000
13     1.00012207031250000000
14     1.00006103515625000000
15     1.00003051757812500000
16     1.00001525878906250000
17     1.00000762939453125000
18     1.00000381469726562500
19     1.00000190734863281300
20     1.00000095367431640600
21     1.00000047683715820300
22     1.00000023841857910200
23     1.00000011920928955100
24     1.00000000000000000000

Result : 32-bit <float> 에서는 23-bit로 mantissa를 표현한다.
계속하려면 아무 키나 누르십시오 . . .
```

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.

```
C:\Windows\system32\cmd.exe
<Homework 1 - 3 : mantissa를 표현하는 bit의 갯수>
1      1.50000000000000000000000000000000    2      1.25000000000000000000000000000000
3      1.12500000000000000000000000000000    4      1.06250000000000000000000000000000
5      1.03125000000000000000000000000000    6      1.01562500000000000000000000000000
7      1.00781250000000000000000000000000    8      1.00390625000000000000000000000000
9      1.00195312500000000000000000000000   10     1.00097656250000000000000000000000
11     1.00048828125000000000000000000000   12     1.00024414062500000000000000000000
13     1.00012207031250000000000000000000   14     1.00006103515625000000000000000000
15     1.00003051757812500000000000000000   16     1.00001525878906250000000000000000
17     1.00000762939453125000000000000000   18     1.00000381469726562500000000000000
19     1.00000190734863281300000000000000   20     1.00000095367431640600000000000000
21     1.00000047683715820300000000000000   22     1.00000023841857910200000000000000
23     1.00000011920928955100000000000000   24     1.00000005960464477500000000000000
25     1.00000002980232238800000000000000   26     1.00000001490116119400000000000000
27     1.00000000745058059700000000000000   28     1.00000000372529029800000000000000
29     1.00000000186264514900000000000000   30     1.00000000093132257500000000000000
31     1.00000000046566128700000000000000   32     1.00000000023283064400000000000000
33     1.00000000011641532200000000000000   34     1.00000000005820766100000000000000
35     1.00000000002910383000000000000000   36     1.00000000001455191500000000000000
37     1.00000000000727595800000000000000   38     1.00000000000363797900000000000000
39     1.00000000000181898900000000000000   40     1.00000000000090949500000000000000
41     1.00000000000045474700000000000000   42     1.00000000000022737400000000000000
43     1.00000000000011368700000000000000   44     1.00000000000005684300000000000000
45     1.00000000000002842200000000000000   46     1.00000000000001421100000000000000
47     1.00000000000000710500000000000000   48     1.00000000000000355300000000000000
49     1.00000000000000177600000000000000   50     1.00000000000000088800000000000000
51     1.00000000000000044400000000000000   52     1.00000000000000022200000000000000
53     1.00000000000000000000000000000000

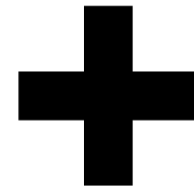
Result : 64-bit <double> 에서는 52-bit로 mantissa를 표현한다.
계속하려면 아무 키나 누르십시오 . . .
```

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
