| 부동 소수점(floating-point)

부동 소수점(floating-point)
1. 단정도(single precision)
   1) 32 bit
   2) 부호(1 bit) + 지수(8 bit) + 가수(23 bit)
2. 배정도(double precision)
   1) 64 bit
   2) 부호(1 bit) + 지수(11 bit) + 가수(52 bit)

Copyright FASTCAMPUS Corp. All Rights Reserved

| 부동 소수점(floating-point)

float	1 bit	8 bit	23 bit
double	1 bit	11 bit	52 bit

$$\pm 1.man \times 2^{Exp - bias}$$

Copyright FASTCAMPUS Corp. All Rights Reserved

10진수 에서 2진수로 : 실수의 경우

1 0 -1 -2 -3

10.625

$$10.625 = 8 + 2 + 0.5 + 0.125$$
$$= 2^3 + 2^1 + 2^{-1} + 2^{-3}$$
$$= 1010.101_2$$

$$\mathbf{1010.101_2}$$

Copyright FASTCAMPUS Corp. All Rights Reserved

정규화(Normalization)

정수 부분을 0이 아닌 자연수로 만드는 것

$10.625 \rightarrow 1.0625 \times 10^{1}$

$1062.5 \rightarrow 1.0625 \times 10^{3}$

Copyright FASTCAMPUS Corp. All Rights Reserved

정규화(Normalization)

2진수의 경우 0이 아닌 자연수는 1 밖에 없으므로
정규화를 하면 정수 부분은 반드시 1이다

$$1010.101_2 \rightarrow 1.010101_2 \times 2^3$$

Copyright FASTCAMPUS Corp. All Rights Reserved

| 부동 소수점(floating-point)

$$\pm 1.man \times 2^{Exp\ -bias}$$

$$+1.010101_2 \times 2^3$$

1. 부호 : 0
2. Exp – bias = 3
3. man = 010101

Copyright FASTCAMPUS Corp. All Rights Reserved

bias

$$\text{bias} = 2^{n-1} - 1$$

n : 지수부의 비트 수

float의 경우 $2^{8-1} - 1 = 127$

Copyright FASTCAMPUS Corp. All Rights Reserved

bias

$$E_{re} = E_{mem} - \text{bias}$$

Copyright FASTCAMPUS Corp. All Rights Reserved

$$+1.010101_2 \times \mathbf{2^3}$$

$$\mathbf{3 = E_{mem} - 127}$$

$$\mathbf{E_{mem} = 130}$$

Copyright FASTCAMPUS Corp. All Rights Reserved

| 부동 소수점(floating-point)

$$\pm 1.man \times 2^{Exp\ -bias}$$

$$+1.010101_2 \times 2^3$$

1. sign : 0
2. Exp : 10000010
3. man : 010101...(나머지 비트는 0)

Copyright FASTCAMPUS Corp. All Rights Reserved

| 부동 소수점(floating-point)

float : 4 byte(32 bit)

$$+1.010101_2 \times 2^3$$

sign	exponent	mantissa
0	10000010	010101(나머지 비트는 0)
1 bit	8 bit	23 bit

Copyright FASTCAMPUS Corp. All Rights Reserved

| 부동 소수점(floating-point)

0.01은 가수부 23 bit로 온전히 표현할 수 없다.
→ 근사치로 표현하므로
0.01을 100번 더한 값은 정확히 1.0이 아니다.

Copyright FASTCAMPUS Corp. All Rights Reserved

| 엡실론(epsilon)

실수 1.0 $+1.0_2 \times 2^0$

sign	exponent	mantissa
0	01111111	0000...(0이 23개)
1 bit	8 bit	23 bit

1.0 다음으로 표현할 수 있는 수는?

Copyright FASTCAMPUS Corp. All Rights Reserved

엡실론(epsilon)

$$+1.00 \ldots \ldots 001_2 \times 2^0$$

sign	exponent	mantissa
0	01111111	0000… … 1
1 bit	8 bit	23 bit

22 bit까지 모두 0 마지막 bit만 1

$$+1.0_2 \times 2^0 + \mathbf{2^{-23}}$$

Copyright FASTCAMPUS Corp. All Rights Reserved

엡실론(epsilon)

엡실론(Epsilon)
- 1.0과 그 다음 표현할 수 있는 수 사이의 차이

$$2^{-23} = 1.192092896e\text{-}07$$

Copyright FASTCAMPUS Corp. All Rights Reserved

엡실론(epsilon)의 쓰임

실수 10.5와 그 다음 표현 가능한 수 사이의 차이는 얼마일까?

$10.5 = 8 + 2 + 0.5 = 2^3 + 2^1 + 2^{-1} = 1.0101_2 \times 2^3$

$$\boldsymbol{diff = 2^E \times epsilon}$$
$$= 2^E \times 2^{-23}$$
$$= \mathbf{2^{-20}}$$
$$\mathbf{= 9.53674e\text{-}07}$$

Copyright FASTCAMPUS Corp. All Rights Reserved

| 엡실론(epsilon)의 쓰임

$$1.0 \leq \frac{|num|}{2^E} < 2.0$$

$$2^E \leq |num| < 2^{E+1}$$

$|num|$이 2^E로 근사하면

$$\boldsymbol{diff = 2^E \times epsilon}$$
$$\boldsymbol{\approx |num| \times epsilon}$$

Ex) num = $1.0101_2 \times 2^3$ 이

$$1.0 \leq \frac{1.0101_2 \times 2^3}{2^3} < 2.0$$

$$2^3 \leq 1.0101_2 \times 2^3 < 2^4$$

$$\mathbf{8 \leq 10.5 < 16}$$

$$\boldsymbol{diff \approx |num| \times epsilon}$$
$$\boldsymbol{\approx 10.5 \times 2^{-23}}$$
$$\boldsymbol{\approx 1.25169e\text{-}06}$$

Copyright FASTCAMPUS Corp. All Rights Reserved

| 엡실론(epsilon)의 쓰임

실수 비교
- 실수는 if(a==b)처럼 직접 비교하면 안된다.
- 그러므로 a, b를 비교하는 함수를 작성해 비교한다.

1. Absolute comparison
2. Relative comparison

Copyright FASTCAMPUS Corp. All Rights Reserved

| 엡실론(epsilon)의 쓰임

1. Absolute comparison

```
def Is_equal1( a, b):
    return |a - b| <= 1e-10
```

두 수의 차이가 1e-10보다 작다면 같다고 생각해도 무방
여기서 1e-10은 임의로 정한 충분히 작은 수

Copyright FASTCAMPUS Corp. All Rights Reserved

| 엡실론(epsilon)의 쓰임

2. Relative comparison

```
def Is_equal2( a, b, allowed=0):
    ep ← 2^-23
    diff = |a – b|
    return diff <= max(|a|, |b|) * ep * 2^allowed
```

$$diff \approx |num| \times \text{epsilon}$$

→ 두 수 중 큰 수에 epsilon을 곱하면
→ 그 수와 그 이웃하는 수 차이의 근사값

Copyright FASTCAMPUS Corp. All Rights Reserved