

## Zadanie 1.

### 1.1 Opis problemu:

1.1.1 Napisać program języku `julia` wyznaczający iteracyjnie epsilony maszynowe dla wszystkich dostępnych typów zmiennopozycyjnych `Float16`, `Float32`, `Float64` i porównać z wartościami zwracanymi przez funkcję `eps` oraz nagłówkami `float.h` języka c. Jaki związek ma liczba `macheps` z precyzją arytmetyki?

1.1.2 Napisać program w języku `Julia` wyznaczający iteracyjnie liczbę `eta` taką, że `eta > 0.0` dla wszystkich typów zmiennopozycyjnych `Float16`, `Float32`, `Float64`, zgodnych ze standardem IEEE 754 ( `half`, `single`, `double` ), i porównać z wartościami zwracanymi przez funkcje: `nextfloat(Float16(0.0))`, `nextfloat(Float32(0.0))`, `nextfloat(Float64(0.0))`. Jaki związek ma liczba `eta` z liczbą `MINsub`?

1.1.3 Napisać program w języku `Julia` wyznaczający iteracyjnie liczbę `MAX` dla wszystkich typów zmiennopozycyjnych i porównać z wartościami zwracanymi przez funkcję `realmax()` oraz wartościami w pliku nagłówkowym `float.h`.

### 1.2 Rozwiązanie:

1.2.1 Weźmy więc reprezentację jedynek w danym typie i dodawajmy do niej coraz mniejszą liczbę również w tym samym typie. Tą drugą będzie szukany `macheps`. Zaczniemy więc od jedynek i dzielimy ją przez 2, sprawdzając cały czas czy nasza suma jest większa od jedynek. Jeśli nasz warunek nie będzie spełniony, znaczy to że liczba dzielona cały czas przez 2 jest już za mała. Przerwywamy pętlę a liczbę mnożymy razy 2 aby otrzymać najmniejszą liczbę spełniającą warunek `fl(1 + macheps) > 0`.

1.2.2 Weźmy reprezentację jedynek w danym typie. Będzie to szukana liczba `eta`. Dzielimy teraz ją przez 2 tak długo dopóki liczba ta jest większa od zera. W ten sposób otrzymamy najmniejszą liczbę `eta > 0`.

1.2.3 Weźmy liczbę 1 w danym typie. Wiemy jak wygląda jej reprezentacja (możemy to sprawdzić funkcją `bits()`). Zwykle mnożenie razy 2 nie pozwoli nam na osiągnięcie wartości maksymalnej, ponieważ część ułamkowa mantysy – `f` będzie wynosiła 0. Zaczniemy więc od uzupełnienia części ułamkowej reprezentacji liczby 1 jedynekami. Teraz mnożąc razy dwa i kontrolując naszą liczbę funkcją `isinf()` będziemy pewni że otrzymamy maksymalną wartość w danym typie.

### 1.3 Wyniki:

1.3.1 Program załączony zwraca 2 wyniki dla każdego typu. Jeden obliczony przy użyciu funkcji `eps()`, drugi przy użyciu metody iteracyjnej opisanej powyżej:

```
Float64: [ eps: 2.220446049250313e-16]
```

```
Float32: [ eps: 1.1920929e-7]
```

```
Float16: [ eps: 0.000977]
```

```
Float64: eps 2.220446049250313e-16
```

Float32: eps 1.1920929e-7

Float16: eps 0.000977

Poniżej wypisane są wyniki z nagłówka `float.h`:

DBL\_EPSILON = 2.22045e-16

```
FLT_EPSILON = 1.19209e-07
```

LDBL\_EPSILON = 1E-9 or smaller

1.3.2 Poniżej wypisane są wyniki zwracane przez funkcję `nextloat()`:

```
Float64: [ nextfloat: 5.0e-324]
```

```
Float32: [ nextfloat: 1.0e-45]
```

```
Float16: [ nextfloat: 6.0e-8]
```

Tutaj wyniki zwracane w obliczeniach, dodatkowo wyświetlana binarna reprezentacja:

[illegible]

```
Float32: eta [1.0e-45, 0000000000000000000000000000000000000000000000000000000]
```

```
Float16: eta [6.0e-8, 000000000000000001]
```

1.3.3 Poniżej prezentowane są wyniki zwracane przez funkcję `realmax`:

```
Float64: [ MAX: 1.7976931348623157e308]
```

Float32: [ MAX: 3.4028235e38]

Float16: [ MAX: 6.55e4]

Tutaj wyniki mojej funkcji liczącej:

Float64: max 1.7976931348623157e308

Float32: max 3.4028235e38

Float16: max 6.55e4

Wartości z pliku nagłówkowego `float.h`:

```

FLT MAX 3.40282 e+38

```

```
DBL MAX 1.79769 e+308
```

## 1.4 Wnioski:

1.4.1 Obliczona wartość `macheps` jest precyzją arytmetyki. Przykład dla arytmetyki single (Float64):

$$\varepsilon = \frac{1}{2} \cdot \beta^{1-\tau} = 2^{-52} = 2.220446049250313080847263336181640625 \times 10^{-16} = \text{macheps}$$

Pnadtó obliczone wartości są równe z wartościami zwracanymi przez funkcję `eps()` oraz przez nagłówek `float.h`.

1.4.2

$$MIN_{\text{sub}} = 2^{-(t-1)} \cdot 2^{(c_{\text{min}})}$$

Wynik tego działania dla typu double (float64) to:

$$= 4.9 \cdot 10^{-324}$$

Co jest równe naszemu wynikowi, oraz wartości zwróconej przez funkcję `nextfloat()`.

Zgodnie z przewidywaniami wartość `eta` jest wartością „minimal subnormal”.

1.4.3 Wartości zwracane przez funkcję `realmax()` są takie same jak wartości otrzymane w rozwiązaniu. Jest to maksymalna wartość jaką można otrzymać w danej arytmetyce.