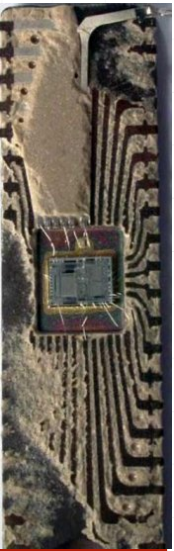


Arytmetyka zmiennoprzecinkowa

wer. 11 z drobnymi modyfikacjami!

Wojciech Myszka

2023-10-23 15:23:44 +0200

Politechnika Wrocławska

Część I

Wprowadzenie

Podsumowanie

Jako podsumowanie poprzedniego wykładu chciałbym przedstawić trzy podstawowe rzeczy, które umożliwiły na burzliwy rozwój komputerów:

1. Dwójkowy system liczenia zamiast powszechnie używanego dziesiętnego.

Podsumowanie

Jako podsumowanie poprzedniego wykładu chciałbym przedstawić trzy podstawowe rzeczy, które umożliwiły na burzliwy rozwój komputerów:

1. Dwójkowy system liczenia zamiast powszechnie używanego dziesiętnego.
2. Zastąpienie operacji arytmetycznych przez proste operacje logiczne.

Podsumowanie

Jako podsumowanie poprzedniego wykładu chciałbym przedstawić trzy podstawowe rzeczy, które umożliwiły na burzliwy rozwój komputerów:

1. Dwójkowy system liczenia zamiast powszechnie używanego dziesiętnego.
2. Zastąpienie operacji arytmetycznych przez proste operacje logiczne.
3. Użycie systemu *uzupełnieniowego do dwu* jako podstawy wszystkich operacji.

Podsumowanie

Jako podsumowanie poprzedniego wykładu chciałbym przedstawić trzy podstawowe rzeczy, które umożliwiły na burzliwy rozwój komputerów:

1. Dwójkowy system liczenia zamiast powszechnie używanego dziesiętnego.
2. Zastąpienie operacji arytmetycznych przez proste operacje logiczne.
3. Użycie systemu *uzupełnieniowego do dwu* jako podstawy wszystkich operacji.

Podsumowanie

Jako podsumowanie poprzedniego wykładu chciałbym przedstawić trzy podstawowe rzeczy, które umożliwiły na burzliwy rozwój komputerów:

1. Dwójkowy system liczenia zamiast powszechnie używanego dziesiętnego.
2. Zastąpienie operacji arytmetycznych przez proste operacje logiczne.
3. Użycie systemu *uzupełnieniowego do dwu* jako podstawy wszystkich operacji.

Co ciekawe, to ostatnie było już postulowane przez von Neumanna w 1945 roku, ale upowszechniło się dopiero w latach sześćdziesiątych.

Podsumowanie

Jako podsumowanie poprzedniego wykładu chciałbym przedstawić trzy podstawowe rzeczy, które umożliwiły na burzliwy rozwój komputerów:

1. Dwójkowy system liczenia zamiast powszechnie używanego dziesiętnego.
2. Zastąpienie operacji arytmetycznych przez proste operacje logiczne.
3. Użycie systemu *uzupełnieniowego do dwu* jako podstawy wszystkich operacji.

Co ciekawe, to ostatnie było już postulowane przez von Neumanna w 1945 roku, ale upowszechniło się dopiero w latach sześćdziesiątych.
Ale cay czas pamiętać trzeba, że **ograniczona** liczba bitów wprowadza istotne problemy — zakres liczb jest ograniczny; jeżeli wynik operacji nie „mieści" się — powstaje przepełnienie (ang overflow).

Jak przedstawiać wartości niecałkowite?

W arkuszu kalkulacyjnym możemy wybierać między następującymi formatami:

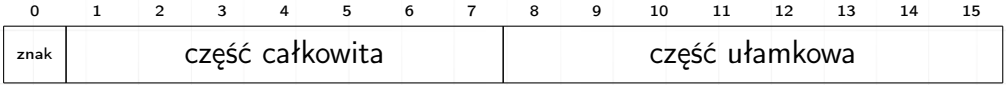
- ogólny: 1,5782
- liczbowy 1,58
- procenty 157,82%
- waluta 1,58 zł
- naukowy 1,58E+00
- separator tysięcy

Ale to tylko zewnętrzna reprezentacja!
Jaka jest wewnętrzna reprezentacja?

Stały przecinek

Załóżmy, że przeznaczamy szesnaście bitów na przechowywanie wartości niecałkowitych:

1. Pierwszych osiem bitów dla części całkowitej
2. Kolejnych osiem bitów dla części ułamkowej

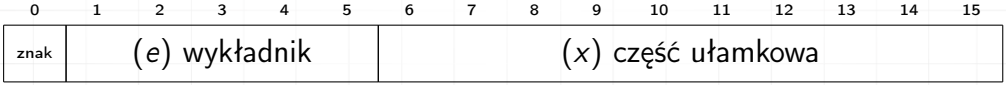


Zmienny przecinek

Szesnaście bitów (ale, ogólnie, 32 lub 64) używane są do przechowywania:

1. *znak*
2. część ułamkowa x
3. wykładnik e

liczby zapamiętywane są jako[1] $znak \cdot x \cdot 2^e$



[1] Tak, na prawdę, jest to nieco bardziej skomplikowane!

Ułamki

Powinniśmy wiedzieć już wszystko na temat arytmetyki liczb całkowitych. Teraz zajmiemy się „liczbami zmiennoprzecinkowymi".
Przez analogie do
$345{,}5 = 3 * 10^2 + 4 * 10^1 + 5 * 10^0 + 5 * 10^{-1}$
mamy:
$101011001{,}1_{(2)} = 2^8 + 2^6 + 2^4 + 2^3 + 2^0 + 2^{-1}$
ale nie jest to zapis zbyt wygodny... Trzy cyfry dwójkowe po przecinku $0{,}001_{(2)} = 0{,}125_{(10)}$ to (w przybliżeniu) jedna cyfra dziesiętna.

Duże liczby dziesiętne...

...zapisujemy korzystając z notacji wykładniczej (naukowej). Zamiast pisać

$$c = 299792458\ \mathrm{m/s}$$

piszemy

$$c = 2{,}99792458 * 10^8\ \mathrm{m/s}$$

albo

$$c = 2{,}99792458\ \mathrm{E}\,8\ \mathrm{m/s}$$

albo (w przybliżeniu)

$$c \sim 3\ \mathrm{E}\,8\ \mathrm{m/s}$$

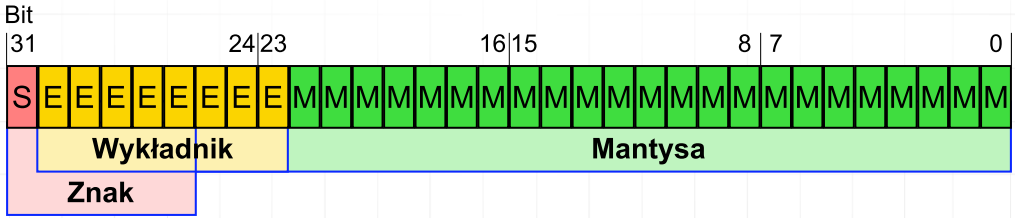
2,99792458 to mantysa; 8 to wykładnik (albo cecha).

Duże liczby binarne...

...można zapisywać tak samo:

$$1{,}11_2 * 2^7_{10}$$

Jest nawet specjalna norma na ten temat: IEEE-754. Można również przeczytać ciekawy artykuł zatytułowany Co student PWR o IEEE 754 wiedzieć powinien.



Dokładność

Liczby 32-bitowe

- Dokładność „dwójkowa”: 24 bity
- Dokładność „dziesiętna”: $\sim$7,2 cyfry dziesiętne

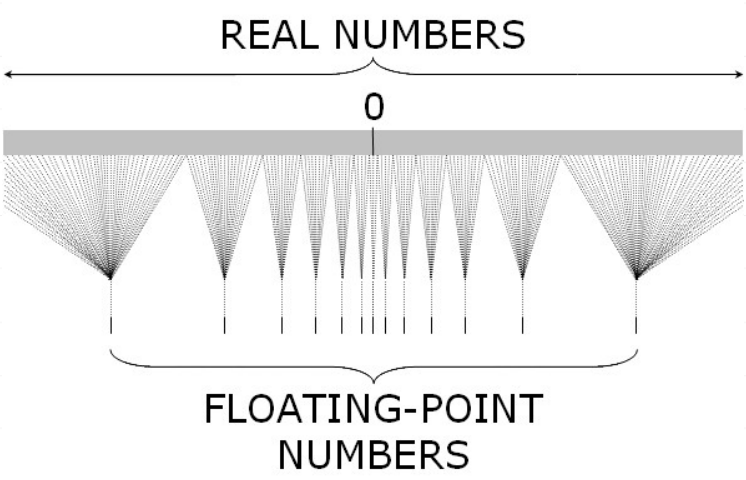
Liczby 64-bitowe

- Dokładność „dwójkowa”: 53 bity
- Dokładność „dziesiętna”: $\sim$15,9 cyfry dziesiętne

Zakres

1. 32-bit: 1,17549 $E-038$ do 3,40282 $E+038$
2. 64-bit: 2,22507 $E-308$ do 1,79769 $E+308$

Niejednoznaczność



Rysunek: Niejednoznaczność przedstawienia liczb zmiennoprzecinkowych

Operacje na liczbach zmiennoprzecinkowych I

1. Mnożenie.
   Jest proste: mnożymy mantysy i dodajemy wykładniki.
   $1{,}33\ e{+}3 * 1{,}55\ e{+}7 = 2{,}0615\ e{+}10$
   Następnie trzeba wynik „obciąć" (zaokrąglając) do odpowiedniej liczby miejsc znaczących (w naszym przypadku niech to będą trzy cyfry) — $2{,}06\ e{+}10$
   W przypadku liczb binarnych będzie podobnie.
   Uwaga: czasami może zdarzyć się problem: w wyniku mnożenia liczba może ulec „denormalizacji" — wówczas trzeba ją znormalizować, zaokrąglić i skorygować wykładnik: $5{,}55\ e{+}0 * 6{,}33e{+}0 = 35{,}13\ e{+}0 = 3{,}51\ e{+}1$
2. Dzielenie.
   Postępujemy analogicznie jak w przypadku mnożenie (dzielimy mantysy, odejmujemy wykładniki). W przypadku „de-normalizacji" postępujemy jak wyżej
   $1{,}33\ e{+}0 / 9{,}88\ e{+}0 = 0{,}134615385\ e{+}0 = 1{,}35\ e{-}1$

Operacje na liczbach zmiennoprzecinkowych II

3. Dodawanie.
   Sprawa nieco bardziej skomplikowana. Aby dodawać liczby zmiennoprzecinkowe trzeba je najpierw „zdenormalizować" i doprowadzić do równości wykładników:
   $1{,}22\,e{+}0 + 3{,}35\,e - 4 = 1{,}22\,e{+}0 + 0{,}000335\,e{+}0 = 1{,}220335\,e{+}0 = 1{,}22\,e{+}0$
   a następnie zaokrąglić i znormalizować...
4. Odejmowanie.
   Identycznie jak dodawanie.

Parę problemów

1. Zawsze(?) ograniczona liczba bitów przeznaczona na zapamiętanie liczby (ale znane są specjalne programy, które starają się te ograniczenie przezwyciężać).
2. Wynik działań arytmetycznych często prowadzi do powstania nadmiaru (czyli przekroczenia maksymalnej dopuszczalnej wartości liczb).
3. Większość liczb które (z przyzwyczajenia) traktujemy jako dokładne, nie ma dokładnej reprezentacji dwójkowej (0,5 jest OK ale 0,1 już nie).

Część II

Błędy bezwzględne

Podstawowe definicje

W teorii błędów maksymalnych jest pięć podstawowych pojęć:

1. Wielkość.
2. Dokładna wartość wielkości.
3. Przybliżona wartość wielkości.
4. Błąd bezwzględny wartości przybliżonej.
5. Liczba przybliżona.

Zostaną one zdefiniowane w dalszej części.

Podstawowe definicje

Definicja (Wielkość)

Dowolna stała matematyczna, wynik jakiejś operacji matematycznej (działania), pierwiastek rozwiązywanego równania. π jest określony jako stosunek obwodu okręgu do jego średnicy; $\sqrt{2}$ jest pierwiastkiem równania kwadratowego $X^2 - 2 = 0$.

Definicja (Wartość dokładna wielkości)

Wartość wynikająca wprost z definicji wielkości, nie obarczona żadnymi błędami.

Definicja (Wartość przybliżona wielkości)

Wartość liczbowa uzyskana w wyniku obliczeń. Zazwyczaj w wyniku obliczeń nie uzyskujemy dokładnej wartości.

Wielkości fizyczne

Ciśnienie, temperatura, długość, stężenie — to przykłady wielkości fizycznych, które bardzo często „mierzymy".
Każdy pomiar obarczony jest błędem wynikającym z dokładności użytego narzędzia pomiarowego.
Wartość dokładna to temperatura w jakimś miejscu sali; wartość przybliżona — to wartość zmierzona jakimś termometrem.

Obliczenia

- Używamy komputera do dokonania jakichś obliczeń.

Obliczenia

- Używamy komputera do dokonania jakichś obliczeń.
- Komputer podaje wynik ($a = 5{,}34273343$) mający 8 cyfr po przecinku.

Obliczenia

- Używamy komputera do dokonania jakichś obliczeń.
- Komputer podaje wynik ($a = 5{,}34273343$) mający 8 cyfr po przecinku.
- Czy możemy powiedzieć, że wyznaczona liczba ma wszystkie cyfry **poprawne**?, Czy różni się od wartości dokładnej o mniej niż $0{,}5 \cdot 10^{-8}$?

Obliczenia

- Używamy komputera do dokonania jakichś obliczeń.
- Komputer podaje wynik ($a = 5{,}34273343$) mający 8 cyfr po przecinku.
- Czy możemy powiedzieć, że wyznaczona liczba ma wszystkie cyfry **poprawne**?, Czy różni się od wartości dokładnej o mniej niż $0{,}5 \cdot 10^{-8}$?
- A co z sytuacją, że zastosowana metoda obliczeń jest mało dokładna?

Błąd bezwzględny wartości przybliżonej I

Definicja (Błąd bezwzględny wartości przybliżonej)

Niech A będzie wartością dokładną, a a wartością przybliżoną pewnej wielkości. ***Błędem bezwzględnym wartości przybliżonej*** *nazywamy każdą liczbę Δa spełniającą warunek:*

$$|A - a| \leq \Delta a,$$

to znaczy taką liczbę, że

$$a - \Delta a \leq A \leq a + \Delta a.$$

Wartość przybliżona a i jej błąd bezwzględny Δa wyznaczają przedział:

$$< a - \Delta a;\ a + \Delta a >,$$

do którego należy dokładna wartość A

Błąd bezwzględny nie jest określony jednoznacznie!

Liczba przybliżona

Definicja (Liczba przybliżona)

Jeżeli a jest wartością przybliżoną dla wartości dokładnej A, obciążoną błędem Δa, to parę liczb Δa, a zapisaną w postaci

$$\begin{matrix} \Delta a \\ a \end{matrix}$$

będziemy nazywali ***liczbą przybliżoną*** *dla A.*

Liczba π

Wiemy, że $\pi = 3{,}14159265\ldots$. Wartością przybliżoną π często używaną w rachunkach, jest liczba 3,14.
Jej błędem bezwzględnym jest, na przykład, liczba $\Delta a = 0{,}0016$.
Dokładna wartość π jest zawarta między liczbami:

$$3{,}14 - 0{,}0016 \leq \pi \leq 3{,}14 + 0{,}0016$$

to znaczy π znajduje się w przedziale

$$< 3{,}1384;\ 3{,}1416 >$$

Zatem możemy zapisać $\pi = 3{,}14^{0{,}0016}$

„Równość w przybliżeniu”

Jeżeli liczby przybliżone $\overset{\alpha}{a}$ i $\overset{\beta}{b}$ są takie, że przedział $< a - \alpha;\ a + \alpha >$ jest zawarty w przedziale $< b - \beta;\ b + \beta >$ to mówimy, że **liczba $\overset{\alpha}{a}$ jest w przybliżeniu równa liczbie $\overset{\beta}{b}$**. Zapisujemy to $\overset{\alpha}{a} \Rightarrow \overset{\beta}{b}$.

Z tego że „$\overset{\alpha}{a}$ jest w przybliżeniu równe $\overset{\beta}{b}$ NIE WYNIKA, że $\overset{\beta}{b}$ jest w przybliżeniu równe $\overset{\alpha}{a}$!

Zaokrąglanie liczb przybliżonych

Dla dowolnej liczby przybliżonej $\overset{\alpha}{a}$ i dowolnej liczby rzeczywistej b zachodzi związek:

$$\overset{\alpha}{a} \Rightarrow \overset{\alpha+|a-b|}{b}$$

czyli $\overset{\alpha}{a}$ jest w przybliżeniu równe $\overset{\alpha+|a-b|}{b}$

Zaokrąglanie stosujemy wtedy, gdy wynik jakichś działań ma zbyt wiele cyfr.

Zastępując liczbę $\overset{\alpha}{a}=3{,}14159$ liczbą 3,14 możemy oszacować błąd. Wynosi on $0{,}0000027 + |3{,}14159 - 3{,}14|$. Czyli: 0,0015927. Zatem:

$$\overset{0{,}0000027}{3{,}14159} \Rightarrow \overset{0{,}0015927}{3{,}14}$$

Zaokrąglanie liczb przybliżonych

Jeżeli $\beta \geq \alpha$, to:

$$\overset{\alpha}{a} \Rightarrow \overset{\beta}{b}$$

Zatem $\overset{0,0015927}{3,14} \Rightarrow \overset{0,0016}{3,14}$

Reguły zaokrąglania I

- Gdy wynik działania arytmetycznego ma (za) dużo cyfr — możemy odrzucić „ostatnie, zbędne cyfry" (pamiętając o zwiększeniu błędu zaokrąglenia).
- Gdy pierwszą odrzuconą cyfrą jest 0, 1, 2, 3, 4 — cyfr pozostawionych w wartości przybliżonej nie zmieniamy.
- Jeżeli pierwszą odrzuconą cyfrą jest 5, 6, 7, 8, 9 — do pozostawionej części wartości przybliżonej dodajemy 1 na ostatnim zostawianym miejscu dziesiętnym.

Taka zmiana liczby przybliżonej nazywa się **poprawnym zaokrągleniem**.
Zadanie domowe: poczytać o innych sposobach zaokrąglania (zaokrąglanie „w dół", zaokrąglanie „do góry", zaokrąglanie „bankierskie"... Która z tych reguł to zaokrąglanie **poprawne**?

Działania na liczbach przybliżonych

suma

$$\overset{\alpha}{a} + \overset{\beta}{b} = \overset{\alpha+\beta}{a+b}$$

Działania na liczbach przybliżonych

suma

$$\overset{\alpha}{a} + \overset{\beta}{b} = \overset{\alpha+\beta}{a+b}$$

różnica

$$\overset{\alpha}{a} - \overset{\beta}{b} = \overset{\alpha+\beta}{a-b}$$

Działania na liczbach przybliżonych

iloczyn

$$\overset{\alpha}{a} \cdot \overset{\beta}{b} \Rightarrow \overset{|a|\beta+|b|\alpha+\alpha\beta}{ab}$$

Działania na liczbach przybliżonych

iloczyn

$$\overset{\alpha}{a} \cdot \overset{\beta}{b} \Rightarrow \overset{|a|\beta+|b|\alpha+\alpha\beta}{ab}$$

dzielenie

$$\overset{\alpha}{a} : \overset{\beta}{b} \Rightarrow \frac{\overset{\gamma}{a}}{b}$$

gdzie

$$\gamma = \frac{\alpha + |\frac{a}{b}|\beta}{|b| - \beta}.$$

Działania na liczbach przybliżonych

Zadanie domowe

Jak będzie w przypadku innych operacji:

- podnoszenie do potęgi?
- pierwiastkowanie (dowolnego stopnia)
- jakieś inne operacje. . . ?

Działania na liczbach przybliżonych

suma

1. Pierwszy najmniej korzystny przypadek:

$$a - \alpha + b - \beta = (a + b) - (\alpha + \beta)$$

2. Drugi najmniej korzystny przypadek:

$$a + \alpha + b + \beta = (a + b) + (\alpha + \beta)$$

Działania na liczbach przybliżonych

suma

1. Pierwszy najmniej korzystny przypadek:

$$a - \alpha + b - \beta = (a + b) - (\alpha + \beta)$$

2. Drugi najmniej korzystny przypadek:

$$a + \alpha + b + \beta = (a + b) + (\alpha + \beta)$$

(Zadanie domowe: jak będzie w przypadku różnicy? A w przypadku iloczynu?)

Przykład

Obliczyć wartość wielomianu

$$w(x) = a_0x^4 + a_1x^3 + a_2x^2 + a_3x + a_4$$

dla $x = 2{,}1$.
Przyjmijmy, że współczynniki wielomianu są liczbami dokładnymi i równają się:

$$a_0 = 2{,}3,\ a_1 = 3,\ a_2 = -4{,}5,\ a_3 = 7{,}2,\ a_4 = -0,1$$

Najpierw obliczenia wykonamy z dokładnością do dwóch miejsc po przecinku, a później z dokładnością do czterech.

Przykład cd I

dwie cyfry

$$x^2 = \overset{0,0}{2,1} \times \overset{0,0}{2,1} = \overset{0,00}{4,41}$$

$$x^3 = \overset{0,00}{4,41} \times \overset{0,0}{2,1} = \overset{0,000}{9,261} \Rightarrow \overset{0,001}{9,26}$$

$$x^4 = \overset{0,001}{9,26} \times \overset{0,0}{2,1} \Rightarrow \overset{0,0021}{19,446} \Rightarrow \overset{0,0061}{19,45}$$

$$2,3 \times x^4 = \overset{0,0}{2,3} \times \overset{0,0061}{19,45} \Rightarrow \overset{0,01403}{44,735} \Rightarrow \overset{0,01903}{44,74} \Rightarrow \overset{0,02}{44,74}$$

$$3x^3 = \overset{0}{3} \times \overset{0,001}{9,26} \Rightarrow \overset{0,003}{27,78}$$

$$-4,5x^2 = \overset{0,0}{-4,5} \times \overset{0,00}{4,41} \Rightarrow \overset{0,000}{-19,845} \Rightarrow \overset{0,005}{-19,85}$$

$$7,2x = \overset{0,0}{7,2} \times \overset{0,0}{2,1} \Rightarrow \overset{0,00}{15,12}$$

Przykład cd II

dwie cyfry

suma:

$$w(2,1) = \overset{0,02}{44,74} + \overset{0,003}{27,78} - \overset{0,005}{19,85} + \overset{0,00}{15,12} - \overset{0,0}{0,1} = \overset{0,028}{67,69}$$

Przykład cd I

cztery cyfry

$$x^2 = \overset{0,0}{2,1} \times \overset{0,0}{2,1} = \overset{0,00}{4,41}$$

$$x^3 = \overset{0,00}{4,41} \times \overset{0,0}{2,1} = \overset{0,000}{9,261}$$

$$x^4 = \overset{0,000}{9,261} \times \overset{0,0}{2,1} = \overset{0,0000}{19,4481}$$

$$2,3 \times x^4 = \overset{0,0}{2,3} \times \overset{0,0000}{19,4481} = \overset{0,0000}{44,73063} \Rightarrow \overset{0,00003}{44,7306}$$

$$3x^3 = \overset{0}{3} \times \overset{0,000}{9,261} = \overset{0,000}{27,783}$$

$$-4,5x^2 = \overset{0,0}{-4,5} \times \overset{0,00}{4,41} = \overset{0,000}{-19,845}$$

$$7,2x = \overset{0,0}{7,2} \times \overset{0,0}{2,1} = \overset{0,00}{15,12}$$

Przykład cd II

cztery cyfry

suma

$$w(2,1) = \overset{0,00003}{44,7306} + \overset{0,000}{27,783} - \overset{0,000}{19,845} + \overset{0,00}{15,12} - \overset{0,0}{0,1} = \overset{0,00003}{67,6886}$$

Przykład cd I

Załóżmy teraz, że współczynniki obarczone są błędami i równają się:

$$a_0 \overset{0,01}{=} 2,3,\ a_1 \overset{0}{=} 3,\ a_2 \overset{0,02}{=} -4,5,\ a_3 \overset{0,02}{=} 7,2,\ a_4 \overset{0,01}{=} -0,1$$

dwie cyfry

$$w(2,1) \overset{0,42}{\Rightarrow} 67,69$$

cztery cyfry

$$w(2,1) \overset{0,3678}{\Rightarrow} 67,6886 \overset{0,3692}{\Rightarrow} 67,69 \overset{0,37}{\Rightarrow} 67,69$$

Prowadzenie obliczeń z dokładnością do czterech cyfr po przecinku praktycznie nie zwiększyło dokładności!

Przykład cd II

Wynika to stąd, że dane obarczone są dużym błędem (już druga cyfra po przecinku nie jest dokładna).

Część III

Dygresja

Przykład

$$x_0 \in [0; 1] \tag{1}$$

$$x_{k+1} = \mu x_k (1 - x_k) \tag{2}$$

Niech $\mu = 3{,}7$

Wyniki obliczeń I

iteracja	single	double	extended
2	0.2566875	0.2566875	0.2566875
4	0.7680534	0.7680533	0.7680533
6	0.8312892	0.8312889	0.8312889
8	0.9236761	0.9236760	0.9236760
10	0.7133774	0.7133778	0.7133778
12	0.6814939	0.6814953	0.6814953
14	0.5850334	0.5850375	0.5850375
16	0.3381789	0.3381866	0.3381866
18	0.5266685	0.5266460	0.5266460
20	0.2649377	0.2649240	0.2649240
24	0.7726893	0.7727947	0.7727947
28	0.9247919	0.9247575	0.9247575

Wyniki obliczeń II

32	0.6627011	0.6632869	0.6632869
36	0.2665924	0.2677791	0.2677791
40	0.7597211	0.7502111	0.7502111
45	0.3075923	0.4160662	0.4160662
50	0.8943822	0.9210730	0.9210730
55	0.2570739	0.7404139	0.7404139
60	0.5249998	0.5649204	0.5649208
70	0.9157254	0.6021104	0.6020892
80	0.9222577	0.4007202	0.4019857
90	0.2573895	0.5755109	0.6455021
100	0.7139580	0.3158045	0.8947899
110	0.2567323	0.7575933	0.5199780

Wyniki dla 1000 cyfr dziesiętnych i $\mu = 3{,}7$

```
 10: 0.7134705
 20: 0.2625198
 30: 0.8721161
 40: 0.6717834
 50: 0.267952
 60: 0.9100494
 70: 0.8216705
 80: 0.9156799
 90: 0.9235535
100: 0.4925537
```

Program komputerowy

```
#include <stdio.h>
int main()
{
        float s;
        double d;
        long double e;
        int i;
        s = d = e = 0.5;
        for(i=1;i<=110;i++)
        {
                s = 3.7F * s * (1.F - s);
                d = 3.7* d * (1. - d);
                e = 3.7L * e * (1.L - e);
                if (i%10==0)
                    printf("%10i %.7f %.7lf %.7Lf\n", i, s, d, e);
        }
        return 0;
}
```

Literatura dodatkowa I

- Zuber R., *Metody numeryczne i programowanie*, WSziP 1975, fragmenty: https://kmim.wm.pwr.edu.pl/myszka/wp-content/uploads/sites/2/2020/10/zuber.pdf.
- Myszka W., *Przykładowe programiki pokazujące problemy numeryczne* 2008, dostępne pod adresem https://kmim.wm.pwr.edu.pl/myszka/wp-content/uploads/sites/2/2020/10/kod.pdf.
- Goldberg D., *What every computer scientist should know about Floating-Point arithmetic*, [w:] *Numerical Computation Guide*, Sun Microsystems, Palo Alto 2000, URL http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.
- Ward A., *Some issues on floating-point precision under linux*, Linux Gazette, , 53, 2000, URL https://linuxgazette.net/issue53/ward.html.

Kolofon

Prezentacja złożona w systemie LaTeX 2_ε z wykorzystaniem klasy beamer. Użyto fontu Carlito. Ilustracja na stronie tytułowej jest fragmentem zdjęcia, przedstawiającego 8-bitowy mikrokontroler Intel 8742, zawierający w jednym układzie CPU o szybkości 12 MHz, 128 bajtów RAM, 2048 bajtów EPROM oraz układy wejścia/wyjścia.
Sameli, Ioan. 2006. Old processor. Maj 25. Flickr.
http://www.flickr.com/photos/biwook/153062645/.