Politechnika Wrocławska

# Arytmetyka komputerów

**Wersja: 5 z drobnymi modyfikacjami!**

Wojciech Myszka

2012-11-09 09:23:41 +0100

# Część I

# Liczby binarne i arytmetyka komputerów

# Arytmetyka komputerów

- Zapis liczb — dwójkowy.
  - Każda z liczb zapisywana jest za pomocą cyfr **0** i **1**.
  - Układ jest pozycyjny — waga cyfry zależy od miejsca, w którym została ustawiona.
  - Najmniej znaczące miejsca są po stronie prawej. . .
  - 1010 to $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$ czyli $8 + 0 + 2 + 0 = 10$
  - NB liczby parzyste mają zero na końcu, nieparzyste — 1.
- Arytmetyka dwójkowa — bardzo prosta.
  - $0 + 0 = 0$
  - $1 + 0 = 0 + 1 = 1$
  - $1 + 1 = 10$
  - $1 \times 1 = 1$
  - $1 \times 0 = 0 \times 1 = 0$
  - $0 \times 0 = 0$

# Operacje logiczne

(Podstawowe) operacje logiczne to suma logiczna (**OR**), iloczyn logiczny (**AND**), negacja (**NOT**), różnica symetryczna (**XOR**)

| **OR** | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| **AND** | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| **XOR** | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# Arytmetyka komputera

## Arytmetyka „klasyczna”

Jesteśmy przyzwyczajeni do następujących „rzeczy”:

1. Jeżeli $x \neq 0$ to $\forall a \quad a + x \neq a$

# Arytmetyka komputera

## Arytmetyka „klasyczna”

Jesteśmy przyzwyczajeni do następujących „rzeczy”:

1. $$\text{Jeżeli } x \neq 0 \text{ to } \forall a \quad a + x \neq a$$

2. $$a + b + \cdots + z = z + y + \cdots + b + a$$

# Arytmetyka komputera

## Arytmetyka „klasyczna”

Jesteśmy przyzwyczajeni do następujących „rzeczy”:

1. $$\text{Jeżeli } x \neq 0 \text{ to } \forall a \quad a + x \neq a$$

2. $$a + b + \cdots + z = z + y + \cdots + b + a$$

3. $$\forall a, b \in \Re \; a < b \quad \exists c : \; a < c < b$$

# Arytmetyka komputera

## Arytmetyka „klasyczna”

Jesteśmy przyzwyczajeni do następujących „rzeczy”:

1. $$\text{Jeżeli } x \neq 0 \text{ to } \forall a \quad a + x \neq a$$

2. $$a + b + \cdots + z = z + y + \cdots + b + a$$

3. $$\forall a, b \in \Re \; a < b \quad \exists c : \; a < c < b$$

W arytmetyce komputerowej powyższe zasady **nie zawsze** obowiązują!

# Liczby „zmiennoprzecinkowe”

# Liczby całkowite I

- Sytuacja dosyć klarowna.
- Na $n$ bitach możemy zapisać liczby całkowite dodatnie z zakresu od zera do $2^n - 1$
- Jest pewien problem z liczbami ujemnymi: trzeba zarezerwować miejsce na znak
- Trzeba to tak zrobić, żeby podstawowe operacje (dodawanie, odejmowanie i mnożenie,. . . ) były wykonywane tak samo gdy argumenty są dodatnie jak i wtedy gdy są ujemne.
- Układ „uzupełnieniowy” to załatwił.

# Liczby całkowite II

- Czasami korzysta się z kodu BCD (Binary Coded Decimal — (cyfry) dziesiętne kodowane binarnie: liczba zapisywana jest w układzie dziesiętnym (za pomocą cyfr dziesiętnych), ale poszczególne cyfry kodowane są binarnie
  $321_{(10)}$ zapisywane jest jako $0011\ 0010\ 0001_2$

# Liczby ujemne

1. Trzeba zarezerwować jeden bit na zapamiętanie znaku!
2. Wariant najprostszy $3 - 0000011$
3. Wariant najprostszy $-3 - 1000011$
   Jest to zapis „znak-moduł”
4. Ale jak dodawać takie liczby?

# Liczby ujemne

Tablica odejmowania:

| $-$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

(Zakładamy, że operujemy na liczbach czterobitowych!)

$$0011 - 1 = 0010$$

$$0010 - 1 = 0001$$

$$0001 - 1 = 0000$$

$$0000 - 1 = 1111$$

Zatem $-1$ to 1111 (czterobitowo!)

# Liczby ujemne

Dokonajmy prostego sprawdzenia:

$$5 + (-1)$$

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \hline \end{array}$$

# Liczby ujemne

Dokonajmy prostego sprawdzenia:

$$5 + (-1)$$

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \end{array}$$

# Dygresja

Liczby dziesiętne, dwucyfrowe:

$$\begin{array}{r} 3\ \ 3 \\ 9\ \ 9 \\ \hline \end{array}$$

# Dygresja

Liczby dziesiętne, dwucyfrowe:

$$\begin{array}{cc} 3 & 3 \\ 9 & 9 \\ \hline 1 & 3 \quad 2 \end{array}$$

# Negacja liczby

Mnemotechniczny algorytm negacji jest bardzo prosty:
„negujemy” wszystkie bity i powstałą liczbę zwiększamy o 1:
1 to 0001
negacje: 1110
zwiększenie o 1: 1111
2 to 0010
negacja: 1101
zwiększenie o 1: 1110
sprawdzenie $5 + (-2)$

| | | | | |
|---|---|---|---|---|
| | 0 | 1 | 0 | 1 |
| | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

# Liczby „stałoprzecinkowe”

1. Liczby w których na zapamiętanie części całkowitej przeznacza się kilka(naście/dziesiąt) bitów
2. Na zapamiętanie części ułamkowej również używa się kilku(nastu?) bitów:

   | 1 | 0 | 1 | 0 | , | 1 | 0 | 1 | 0 |
   |---|---|---|---|---|---|---|---|---|

   co odczytujemy jako:
   $1*2^3+0*2^2+1*2^1+0*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}+0*2^{-4}$
   lub $8+2+\frac{1}{2}+\frac{1}{8}$ czyli 10,625
3. Używany bardzo rzadko (finanse??)
4. Z matematycznego punktu widzenia są to liczby wymierne
5. Jak w tej postaci zapisać liczbę 1,1

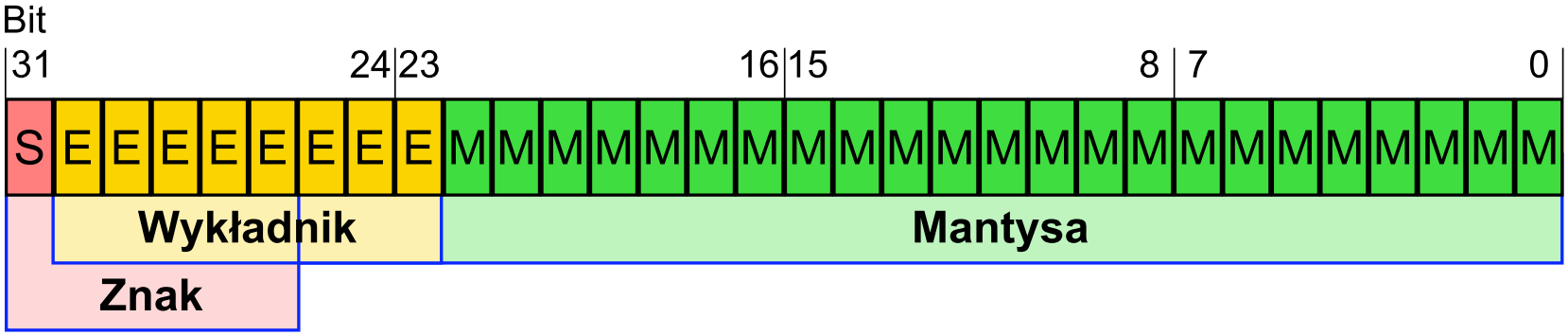
# Liczby „zmiennoprzecinkowe” I

1. Są to liczby zapisywane (kodowane) w sposób podobny do znanego nam: $c = 299792458 \sim 3 * 10^8$ m/s
2. Czyli w postaci mantysa (2,99792458) plus wykładnik 8, zatem 2,99792458*$10^8$ albo inaczej 2,99792458 e8
3. W przypadku komputerów podstawa kodowania (tak mantysy jak i wykładnika) to 2!
4. Dodatkowo liczby zapisywane są zawsze w postaci „znormalizowanej” czyli takiej, że cyfra przed przecinkiem (kropką) dziesiętnym jest zawsze z zakresu między 1 a 9. (a w układzie dwójkowym zawsze jest równa 1!)
5. Na zapamiętanie mantysy i wykładnika przeznaczana jest zawsze skończona liczba bitów.

# Liczby „zmiennoprzecinkowe” II

6. Z matematycznego punktu widzenia są to liczby wymierne.
7. Sposób zapisu liczb zmiennoprzecinkowych reguluje standard IEEE-754.

Zgodnie ze standardem liczba zmiennoprzecinkowa zapisywana jest tak:




- Zakres liczb: $\pm 1{,}18 \cdot 10^{-38}$ do około $\pm 3{,}4 \cdot 10^{38}$.
- Dokładność: 7–8 cyfr dziesiętnych.

# Liczby „zmiennoprzecinkowe” III

- Przypadki szczególne:
  - $+0$ — wszystkie bity są zerami
  - $-0$ — bit znaku ustawiony, wszystkie bity są zerami
  - liczby małe (*denormalized numbers*) — nie zakłada się niezerowego pierwszego bitu mantysy
  - $\pm\infty$ — ustawione wszystkie bity wykładnika, mantysa równa zero (jako wynik dzielenia przez zero, na przykład)
  - NaN (*Not a Number*) ustawione wszystkie bity wykłądnika, mantysa różna od zera.

# 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).

# 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” 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: $5,55\,e{+}0 * 6,33e{+}0 = 35,13\,e{+}0 = 3,51\,e{+}1$

# Operacje na liczbach zmiennoprzecinkowych II

2. Dzielenie.
   Postępujemy analogicznie jak w przypadku mnożenie (dzielimy mantysy, odejmujemy wykładniki). W przypadku „denormalizacji„ postępujemy jak wyżej
   $1,33\,e{+}0 / 9,88\,e{+}0 = 0,134615385\,e{+}0 = 1,35\,e{-}1$
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. . .

# Operacje na liczbach zmiennoprzecinkowych III

4. Odejmowanie.
   Identycznie jak dodawanie.

# Konwersje

## Dziesiętny do dwójkowego

Liczby całkowite:

Liczbę dzielimy przez dwa zapisując reszty z dzielenia:

10 |

# Konwersje

## Dziesiętny do dwójkowego

Liczby całkowite:

Liczbę dzielimy przez dwa zapisując reszty z dzielenia:

| | |
|---|---|
| 10 | |
| 5 | 0 |

# Konwersje

## Dziesiętny do dwójkowego

Liczby całkowite:

Liczbę dzielimy przez dwa zapisując reszty z dzielenia:

| | |
|---|---|
| 10 | |
| 5 | 0 |
| 2 | 1 |

# Konwersje

## Dziesiętny do dwójkowego

Liczby całkowite:

Liczbę dzielimy przez dwa zapisując reszty z dzielenia:

| | |
|---|---|
| 10 | |
| 5 | 0 |
| 2 | 1 |
| 1 | 0 |

# Konwersje

## Dziesiętny do dwójkowego

Liczby całkowite:

Liczbę dzielimy przez dwa zapisując reszty z dzielenia:

| | |
|---|---|
| 10 | |
| 5 | 0 |
| 2 | 1 |
| 1 | 0 |
| 0 | 1 |

# Konwersje

## Dziesiętny do dwójkowego

Liczby całkowite:

Liczbę dzielimy przez dwa zapisując reszty z dzielenia:

| | |
|---|---|
| 10 | |
| 5 | 0 |
| 2 | 1 |
| 1 | 0 |
| 0 | 1 |

Reszty z dzielenia zapisujemy „od końca” otrzymując 1010

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| ,33

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |
| 0 | ,24 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |
| 0 | ,24 |
| 0 | ,48 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |
| 0 | ,24 |
| 0 | ,48 |
| 0 | ,96 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |
| 0 | ,24 |
| 0 | ,48 |
| 0 | ,96 |
| 1 | ,92 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |
| 0 | ,24 |
| 0 | ,48 |
| 0 | ,96 |
| 1 | ,92 |
| 1 | ,84 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |
| 0 | ,24 |
| 0 | ,48 |
| 0 | ,96 |
| 1 | ,92 |
| 1 | ,84 |
| 1 | ,68 |

# Konwersje

## Dziesiętny do dwójkowego

Część ułamkowa:
Liczbę mnożymy przez dwa i zapisujemy część całkowitą:

| | |
|---|---|
| | ,33 |
| 0 | ,66 |
| 1 | ,32 |
| 0 | ,64 |
| 1 | ,28 |
| 0 | ,56 |
| 1 | ,12 |
| 0 | ,24 |
| 0 | ,48 |
| 0 | ,96 |
| 1 | ,92 |
| 1 | ,84 |
| 1 | ,68 |

$0{,}0101010001111_2 = 0{,}329833984375_{10}$

# Konwersje

## Dwójkowy do dziesiętnego

Do zrobienia w domu!

# Wnioski

1. Konwersja liczb dziesiętnych do dwójkowych często wprowadza błędy.
2. Ograniczona liczba bitów powoduje, że każde działanie wykonywane jest z błędem.
3. W przypadku wielokrotnego powtarzania jakiegoś obliczenia sprawa zaczyna mieć znaczenie. . .

# Część II

## Błędy numeryczne

# Błędy

Bardzo prosty przykład. Mamy dwa wyrażenia:

## Wyrażenie 1

$$x = ((b + a) - a)/b$$

# Błędy

Bardzo prosty przykład. Mamy dwa wyrażenia:

## Wyrażenie 1

$$x = ((b + a) - a)/b$$

## Wyrażenie 2

$$y = (b + (a - a))/b$$

# Błędy

Bardzo prosty przykład. Mamy dwa wyrażenia:

## Wyrażenie 1

$$x = ((b + a) - a)/b$$

## Wyrażenie 2

$$y = (b + (a - a))/b$$

Jaki powinien być wynik działań dla:

1. $a = 5$ i $b = 100$

# Błędy

Bardzo prosty przykład. Mamy dwa wyrażenia:

## Wyrażenie 1

$$x = ((b + a) - a)/b$$

## Wyrażenie 2

$$y = (b + (a - a))/b$$

Jaki powinien być wynik działań dla:

1. $a = 5$ i $b = 100$
2. $a = 1.0^4$ oraz $b = 1.0^{-4}$

# Błędy

Bardzo prosty przykład. Mamy dwa wyrażenia:

## Wyrażenie 1

$$x = ((b + a) - a)/b$$

## Wyrażenie 2

$$y = (b + (a - a))/b$$

Jaki powinien być wynik działań dla:

1. $a = 5$ i $b = 100$
2. $a = 1.0^4$ oraz $b = 1.0^{-4}$
3. $a = 1.0^{10}$ oraz $b = 1.0^{-10}$

# Błędy

Bardzo prosty przykład. Mamy dwa wyrażenia:

## Wyrażenie 1

$$x = ((b + a) - a)/b$$

## Wyrażenie 2

$$y = (b + (a - a))/b$$

Jaki powinien być wynik działań dla:

1. $a = 5$ i $b = 100$
2. $a = 1.0^4$ oraz $b = 1.0^{-4}$
3. $a = 1.0^{10}$ oraz $b = 1.0^{-10}$

Sprawdźmy w praktyce. . .

# Błędy I

Najprostsze dodawanie (źródło programu strona MSDN zatytułowana „Why Floating-Point Numbers May Lose Precision”)

```
// Floating-point_number_precision.c
// Compile options needed: none. Value of c is printed with a
// point precision of 10 and 6 (printf rounded value by defau
// show the difference
#include <stdio.h>

#define EPSILON 0.0001    // Define your own tolerance
#define FLOAT_EQ(x,v) (((v - EPSILON) < x)
                          && (x <( v + EPSILON)))

int main() {
   float a, b, c;

   a = 1.345f;
```

# Błędy II

```
    b = 1.123f;
    c = a + b;

    // if (FLOAT_EQ(c, 2.468)) // Remove comment for correct r
    if (c == 2.468)            // Comment this line for correc
       printf("Liczby sa rowne.\n");
    else
       printf("Liczby nie sa rowne!  Wartosc c wynosi %13.10f
                  "or %f\n",c,c);
}
```

**Komentarz:** *Program dodaje dwie liczby, a wynik zapisuje w zmiennej o nazwie* `c`*, a następnie sprawdza czy wynik jest taki jak oczekiwano. Niestety nie jest.*

Wyniki:

# Błędy III

```
Liczby nie sa rowne!  Wartosc c wynosi  2.4679999352
```

# Długotrwałe obliczenia I

Program w Pascalu dokonujący wielokrotnie prostej dosyć operacji. Na podstawie Some issues on floating-point precision under Linux

```
program caos;

{$n+}          { you need to activate hardware floatin
                 in order to use the extended type }

uses
   crt;

var
   s : single;     { 32-bit real }
```

# Długotrwałe obliczenia II

```
    r : real;        { 48-bit real }
    d : double;      { 64-bit real }
    e : extended;    { 80-bit real }

    i : integer;

begin
    clrscr;

    s := 0.5;
    r := 0.5;
    d := 0.5;
    e := 0.5;
```

# Długotrwałe obliczenia III

```
for i := 1 to 100 do begin
    s := 3.8 * s * (1 - s);
    r := 3.8 * r * (1 - r);
    d := 3.8 * d * (1 - d);
    e := 3.8 * e * (1 - e);

    if (i/10 = int(i/10)) then begin
        writeln (i:10, s:16:5, r:16:5,
                                d:16:5, e:16:5);
    end;
end;
```

# Długotrwałe obliczenia IV

```
    readln;
end.
```

Program wykonuje dość proste obliczenia: sto razy wylicza wyrażenie:

$$x = 3.8 * x * (1 - x)$$

Drukowana jest co dziesiąta wartość:

# Długotrwałe obliczenia V

| | | | | |
|---|---|---|---|---|
| 10 | 0.18510 | 0.18510 | 0.18510 | 0.18510 |
| 20 | 0.23951 | 0.23963 | 0.23963 | 0.23963 |
| 30 | 0.88423 | 0.90200 | 0.90200 | 0.90200 |
| 40 | 0.23013 | 0.82493 | 0.82493 | 0.82493 |
| 50 | 0.76654 | 0.53714 | 0.53714 | 0.53714 |
| 60 | 0.42039 | 0.66878 | 0.66878 | 0.66879 |
| 70 | 0.93075 | 0.53190 | 0.53190 | 0.53203 |
| 80 | 0.28754 | 0.93557 | 0.93557 | 0.93275 |
| 90 | 0.82584 | 0.69203 | 0.69203 | 0.79884 |
| 100 | 0.38775 | 0.41983 | 0.41983 | 0.23138 |

Widać wyraźnie, że precyzja liczb ma wpływ na wyniki obliczeń. . .

# Podstawowe definicje

**Wielkość** — dowolna stała matematyczna, wynik jakiejś operacji matematycznej (działania), pierwiastek rozwiązywanego równania. $\pi$ jest określony jako stosunek obwodu okręgu do jego średnicy; $\sqrt{2}$ jest pierwiastkiem równania kwadratowego $X^2 - 2 = 0$.
**Wartość dokładna wielkości** — wartość wynikająca wprost z definicji wielkości, nie obarczona żadnymi błędami.
**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

Niech $A$ będzie wartością dokładną, a $a$ wartością przybliżoną pewnej wielkości. **Błędem bezwzględnym wartości przybliżonej** nazywamy każdą liczbę $\Delta a$ 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 $\Delta a$ wyznaczają przedział:

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

do którego należy dokładna wartość $A$
Błąd bezwzględny nie jest określony jednoznacznie!

# Liczba przybliżona

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

$$\frac{\Delta a}{a}$$

będziemy nazywali **liczbą przybliżoną** dla A.

# $\pi$

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

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

to znaczy $\pi$ 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 $\overset{\alpha}{a}=\overset{0,0000027}{3,14159}$ 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

- 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**.

# Błąd względny

Błędem względnym wartości przybliżonej $a$ obarczonej błędem bezwzględnym $\Delta a$ nazywamy liczbę:

$$\varepsilon a = \frac{\Delta a}{|a|}$$

# 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

## 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

## dwie cyfry

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

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

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

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

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

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

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

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

## cztery cyfry

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

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

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

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

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

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

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

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

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

# Literatura dodatkowa I


John Bayko.
Great microprocessors of the past and present.
http://jbayko.sasktelwebsite.net/cpu.html, 2003.


Jeremy G. Butler.
Information technology history - outline.
http://www.tcf.ua.edu/AZ/ITHistoryOutline.htm, 1998.


David Goldberg.
What every computer scientist should know about Floating-Point arithmetic.
*Numerical Computation Guide*. Sun Microsystems, Palo Alto, 2000.

# Literatura dodatkowa II

Dostępne jako: http://docs.sun.com/source/806-3568/ncg_goldberg.html#674.


Witold Komorowski.
*Architektura Minikomputerow*.
Wydawnictwa Naukowo-Techniczne, Warszawa, 1980.


Witold Komorowski.
*Architektura Komputerów*.
Wydawnictwo Politechniki Wrocławskiej, Warszawa, 1986.


Witold Komorowski.
*Architektura Komputerow: Skrypt*.
Politechnika Wrocławska, Wrocław, 1987.

# Literatura dodatkowa III


Witold Komorowski.
*Instrumenta Computatoria: [wybrane Architektury Komputerów].*
"Helion", Gliwice, 2000.

Witold Komorowski.
*Krótki Kurs Architektury I Organizacji Komputerów.*
Mikom, Warszawa, 2004.

Wojciech Myszka.
Przykładowe programiki pokazujące problemy numeryczne.
Dostępne pod adresem
`http://www.immt.pwr.wroc.pl/~myszka/TI/kod.html`,
październik 2008.

# Literatura dodatkowa IV

Roman Zuber.
*Metody numeryczne i programowanie*.
WSziP, 1975.
fragmenty:
http://www.immt.pwr.wroc.pl/~myszka/TI/zuber.pdf
oraz
http://www.immt.pwr.wroc.pl/~myszka/TI/zuber1.pdf.

# Kolofon

Prezentacja złożona w systemie LaTeX $2_\varepsilon$ z wykorzystaniem klasy beamer. Użyto fontu MS Trebuchet. Ilustracja na stronie tytułowej jest zdjęciem przedstawiającym fragment procesora. Pavert, rob van de. CPU, Styczeń 1, 2005. http://www.flickr.com/photos/rob20/101300834/.