Wstęp do informatyki

Wykład 3

Uniwersytet Wrocławski
Instytut Informatyki

Dane przetwarzane przez „komputer”

Tydzień temu:

– Wejście, wyjście, wartości przetwarzane: liczby całkowite (dowolnie duże)

Dziś:

– Podstawowa jednostka informacji, binarna reprezentacja informacji;

– Cyfrowa/binarna reprezentacja liczb: naturalnych, całkowitych, rzeczywistych;

– Cyfrowa/binarna reprezentacja: znaków, obrazów, wideo, dźwięku, …

– TABLICE jako struktura danych w algorytmie

Podstawowa jednostka informacji

Bit:

– dwa stany (0 i 1)

Dlaczego tak „mało”:

– urządzenia cyfrowe („komputery”) powinny „automatycznie” i niezawodnie odróżniać wartości

Jak uzyskać więcej wartości?

– ciąg $\boldsymbol{k}$ bitów może przyjmować $\mathbf{2}^{\boldsymbol{k}}$ różnych wartości

Bity a komórki pamięci komputera

Maszyna RAM:

– komórka pamięci przechowuje dowolnie dużą liczbę całkowitą

Świat rzeczywisty:

– komórka pamięci to ciąg bitów o ustalonej długości – „bajt” lub „słowo maszynowe”;

– 1 bajt = 8 bitów (zazwyczaj…);

– formalnie (czasem…): bajt to najmniejsza „adresowalna” jednostka pamięci komputera

Binarna reprezentacja liczb naturalnych

Binarna reprezentacja liczb naturalnych

Ciąg binarny = ciąg, którego elementami mogą być tylko zera i jedynki.

Binarna reprezentacja liczby naturalnej x>0 to ciąg binarny $x_k \ldots x_1 x_0$ taki, że:

$$x = \sum_{i=0}^{k} x_i \cdot 2^i$$

oraz $x_k = 1$.

Binarna reprezentacja zera = 0.

Binarna reprezentacja liczb naturalnych

Fakt. Każda liczba naturalna posiada dokładnie jedną reprezentację binarną (bez zbędnych zer „wiodących”).

Dowód: Indukcyjny:

- Krok bazowy: n = 0
- Krok indukcyjny:
  - Zał.: Fakt zachodzi dla każdej liczby $m<n$
  - Teza: Fakt zachodzi dla n
  - Idea dowodu: skorzystaj z zał. indukcyjnego oraz reprezentacji $n = 2^k+m$, gdzie $0 \leq m < 2^k$

Binarna reprezentacja liczb naturalnych

Problem algorytmiczny

Wejście: a – liczba naturalna

Wyjście: $x_k \ldots x_1 x_0$ – ciąg tworzący binarną reprezentację liczby a

Binarna reprezentacja liczb naturalnych

Algorytm

1. $i \leftarrow 0$
2. $x[0] \leftarrow a \bmod 2$
3. $a \leftarrow a \operatorname{div} 2$
4. dopóki $a > 0$:
   a) $i \leftarrow i + 1$
   b) $x[i] \leftarrow a \bmod 2$
   c) $a \leftarrow a \operatorname{div} 2$
5. dopóki $i \geq 0$:
   a) wypisz $x[i]$
   b) $i \leftarrow i - 1$

Oznaczenia:

- a mod b = reszta z dzielenia a przez b
- a div b = wynik dzielenia całkowitego a przez b (zaokrąglenie wyniku w dół do liczby całkowitej)
- x[0], x[i]… co to? tablica

Binarna reprezentacja liczb naturalnych

Poprawność algorytmu - idea:

– ostatnia cyfra reprezentacji binarnej liczby a jest równa a mod 2

– a = (a div 2) · 2 + (a mod 2)

– reprezentację binarną liczby (a div 2) uzyskujemy poprzez usunięcie ostatniej cyfry reprezentacji binarnej liczby a:

$$a = \sum_{i=0}^{k} a_i \cdot 2^i$$

$$a \operatorname{div} 2 = \left(\sum_{i=0}^{k} a_i \cdot 2^i \right) \operatorname{div} 2 = \sum_{i=1}^{k} a_i \cdot 2^{i-1}$$

TABLICE

Tablica

Tablica:

struktura danych przechowująca ustaloną liczbę elementów n tego samego typu,

- indeksowanych kolejnymi liczbami naturalnymi 0, 1, 2,…, n – 1 (Ansi C).

Tablica

Tablica C:

```
#define N 200
int n = 100
int tabA[ 100 ], tabB[ N ], tabC[ n ];

tabA[ 55 ] = 7;
tabC[ n – 1 ] = tabA[0] + tabB[ 1 ];
```

Tablica

Tablica Python (array_demo.py):

```
from wdi import *
print "Tablica jednowymiarowa\n"
N = 10
A = Array(N)              # Tworzenie tablicy
B = Array(N)
for i in range(N):
    A[i] = i * i
    print A[i]
B[5] = A[0]+B[3]
```

Tablica

Tablica Python (array_demo.py):

Na WdI nie używamy list pythonowych!

```
A = Array(10)
```

Tablica

Tablica Python (array_demo.py):

Na WdI nie używamy list pythonowych!

```
A = []
```

```
A = Array(10)
```

```
append, extend, insert, remove, count,
pop, …
```

Binarna reprezentacja liczb rzeczywistych

Binarna reprezentacja liczb rzeczywistych

Binarna reprezentacja liczby rzeczywistej a>0 to $x_k \ldots x_1 x_0 \, , \, x_{-1} x_{-2} x_{-3} x_{-4} \ldots$ taka, że:

$$a = \sum_{i=0}^{k} x_i \cdot 2^i + \sum_{i=1}^{\infty} x_{-i} \cdot 2^{-i}$$

oraz $x_k=1$ lub $k=0$, $x_i \in \{0,1\}$ dla każdego *i*.

x_4	x_3	x_2	x_1	x_0	,	x_{-1}	x_{-2}	x_{-3}	x_{-4}	x_{-5}	…
2^4	2^3	2^2	2^1	2^0	,	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	…

Binarna reprezentacja liczb rzeczywistych

Binarna reprezentacja liczby rzeczywistej a>0 to $x_k \ldots x_1 x_0 , x_{-1} x_{-2} x_{-3} x_{-4} \ldots$ taka, że:

$$a = \sum_{i=0}^{k} x_i \cdot 2^i + \sum_{i=1}^{\infty} x_{-i} \cdot 2^{-i}$$

oraz $x_k=1$ lub k=0, $x_i \in \{0,1\}$ dla każdego *i*.

Reprezentacja nieskończona?

- Jeśli $x_{-p}=0$ dla każdego p>j: wartości x_{-j-1}, x_{-j-2} , x_{-j-3} ,… <u>pomijamy!</u>
- Analogia: dziesiętna reprezentacja ½, 1/3,…

Binarna reprezentacja liczb rzeczywistych

Fakt

Jeśli liczba *a* posiada skończoną reprezentację binarną, to reprezentacja ta jest jednoznaczna.

(inaczej: każda liczba posiada co najwyżej jedną <u>skończoną</u> reprezentację binarną)

Dowód – idea:

- część całkowita – p. dowód wcześniej
- część ułamkowa – indukcja ze względu na liczbę cyfr najkrótszej reprezentacji, wykorzystanie nierówności (*j*>*k*): $2^{-k} > \sum_{i=k+1}^{j} 2^{-i}$

Binarna reprezentacja liczb rzeczywistych

Problem algorytmiczny

Wejście:

a – liczba rzeczywista z przedziału $(0;1)$;

n – dodatnia liczba naturalna

Wyjście:

$x_{-1}x_{-2}\ldots x_{-n+1}x_{-n}$ – *n* pierwszych bitów ułamkowej części w binarnej reprezentacji liczby *a*

(czyli $a = 0{,}x_{-1}x_{-2}\ldots x_{-n+1}x_{-n}\ldots$)

Binarna reprezentacja liczb rzeczywistych

Algorytm

1. $i \leftarrow 1$
2. dopóki ($a \neq 0$) oraz ($i \leq n$):
   1. $a \leftarrow 2 \cdot a$
   2. jeśli ($a \geq 1$):
      a) wypisz 1 (czyli $x_{-i} \leftarrow 1$)
      b) $a \leftarrow a - 1$
   3. w przeciwnym przypadku:
      a) wypisz 0 (czyli $x_{-i} \leftarrow 0$)
   4. $i \leftarrow i + 1$

Poprawność algorytmu: p. efekt mnożenia przez 2 liczby z przedziału $(0;1)$ …

Binarna/cyfrowa reprezentacja liczb całkowitych

Cyfrowa reprezentacja liczb całkowitych

Reprezentacje liczb całkowitych :

- znak-moduł
- kod uzupełnień do 1
- kod uzupełnień do 2

Wspólna cecha powyższych reprezentacji liczb całkowitych:

- k – ustalona długość reprezentacji (liczba bitów)
- najstarszy bit (skrajnie lewy) pozwala określić znak liczby

Reprezentacja znak-moduł

Wartość opisywana ciągiem $x_{k-1}\, x_{k-2} \ldots x_1\, x_0$

– wartość bezwzględna odpowiada binarnej reprezentacji $x_{k-2} \ldots x_1\, x_0$ czyli

$$\sum_{i=0}^{k-2} x_i \cdot 2^i$$

– znak liczby:
  – dodatnia gdy $x_{k-1} = 0$,
  – ujemna gdy $x_{k-1} = 1$

Kod uzupełnień do 1 (U1)

Wartość opisywana ciągiem $\boldsymbol{x_{k-1}\ x_{k-2} \ldots x_1\ x_0}$

– Jeśli $\boldsymbol{x_{k-1}}$=0: „standardowa" reprezentacja liczby dodatniej

$$\sum_{i=0}^{k-2} x_i \cdot 2^i$$

– Jeśli $\boldsymbol{x_{k-1}}$=1: liczba ujemna, bity wartości bezwzględnej zanegowane

$$-\sum_{i=0}^{k-2} (1 - x_i) \cdot 2^i$$

Kod uzupełnień do 2 (U2)

Wartość opisywana ciągiem $x_{k-1}\ x_{k-2} \ldots x_1\ x_0$ jest równa

$$-x_{k-1} \cdot 2^{k-1} + \sum_{i=0}^{k-2} x_i \cdot 2^i$$

Zakres reprezentowanych liczb:

Kodowanie	**Zakres**	**Najmniejsza**	**Największa**	**zero**
Znak-moduł	[–(2$^{k–1}$ –1), 2$^{k–1}$ –1]	1 11…11	011…11	00…00 10…00
U1	[–(2$^{k–1}$ –1), 2$^{k–1}$ –1]	1 00…00	011…11	00…00 11…11
U2	[–2$^{k–1}$, 2$^{k–1}$ –1]	1 00…00	011…11	00..00

Reprezentacje liczb całkowitych

Przykład. k=8

Liczba	Znak-moduł	U1	U2
0	00000000 10000000	00000000 11111111	00000000
109	01101101	01101101	01101101
-109	11101101	10010010	10010011

Kod uzupełnień do 2 (U2) – liczby przeciwne

Wejście:

k – liczba naturalna,

x=$x_{k-1}, x_{k-2}, \ldots x_1, x_0$ – reprezentacja liczby a w U2 na k bitach

Wyjście: y – reprezentacja liczby –a w U2 na k bitach

Algorytm – idea:

1. zaneguj wszystkie bity w ciągu x (negowanie: 1$\rightarrow$0, 0 $\rightarrow$1);
2. do zanegowanego ciągu (traktowanego jako liczba naturalna w zapisie binarnym) dodaj 1.

Kod uzupełnień do 2 (U2) – liczby przeciwne

Wejście:

k – liczba naturalna

x=$x_{k-1},x_{k-2},\ldots x_1,x_0$ – reprezentacja liczby a w U2 na k bitach

Wyjście: y – reprezentacja liczby –a w U2 na k bitach

Algorytm:

1. Jeśli x=100…0:

   zwróć „Brak reprezentacji"
2. x' $\leftarrow$ x z zanegowanym każdym bitem
3. x'' $\leftarrow$ binarna reprezentacja bez znaku x'+1
4. y $\leftarrow$ x''
4. Zwróć y

Kod uzupełnień do 2 (U2) – liczby przeciwne

Poprawność algorytmu:

- zanegowanie bitów $0, 1, \ldots, k-2$: zamiana wartości liczby naturalnej bez znaku w na $2^{k-1} - 1 - w$
- jeśli $a \geq 0$:
  - $x_{k-1}=0$, $a=w$, zanegowanie x_{k-1} daje wartość $2^{k-1} - 1 - w - 2^{k-1} = -w - 1 = -a - 1$
- jeśli a<0:
  - $x_{k-1}=1$, $a=w-2^{k-1}$, zanegowanie $x_{k-1}=1$ „usuwa" ujemny składnik -2^{k-1} i daje wartość $2^{k-1} - 1 - w = -(w - 2^{k-1}) - 1 = -a - 1$

Kod uzupełnień do 2 (U2)

Problem

Wejście:

x - liczba całkowita, k – dodatnia liczba naturalna

Wyjście:

zapis liczby x w kodzie U2 na k bitach

Algorytm – idea:

1. Jeśli x poza przedziałem $[-2^{k-1}, 2^{k-1} -1]$: liczba poza zakresem; zakończ
2. Jeśli $x \geq 0$: zwróć binarną (k-bitową) reprezentację x
3. Jeśli x<0:
   - Zaneguj wszystkie bity binarnej (k-bitowej) reprezentacji -x
   - Dodaj 1
   - Zwróć wynik dodawania

Kod uzupełnień do 2 (U2)

Problem

Wejście:

x - liczba całkowita,

k – dodatnia liczba naturalna

Wyjście:

zapis liczby x w kodzie U2 na k bitach

Uwaga:

binarna(x) zwraca zapis binarny liczby x.

neg(x) oznacza ciąg utworzony przez zanegowanie każdego bitu w x

Algorytm

1. $x' \leftarrow |\, x \,|$
2. Jeśli $x = -2^{k-1}$: zwróć 10…0
3. $bx \leftarrow$ binarna (x')
4. $p \leftarrow$ długość(bx)
5. Jeśli $p>k-1$: liczba przekracza zakres; stop
6. Uzupełnij bx z lewej strony $k - p$ zerami;
7. Jeśli $x \geq 0$: zwróć bx,
   w przeciwnym przypadku:
   $bx \leftarrow$ neg$(bx) + 1$
8. Zwróć bx

Kod U2 – dodawanie

Problem

Wejście:

k – liczba naturalna

x_1, x_2 – k-bitowe zapisy liczb całkowitych w U2

Wyjście:

k-bitowy zapis w U2 liczby x_1+x_2

Algorytm (idea)

1. $x \leftarrow x_1+x_2$ gdzie x_1 i x_2 traktowane jako liczby zapisane binarnie bez znaku,
2. Obetnij x do k skrajnie prawych bitów!
3. Potraktuj x jako liczbę w U2 oraz:
   a) jeśli znak(x_1)=znak(x_2) oraz znak(x_1)≠znak(x): zwróć „przekroczenie zakresu”
   b) zwróć x

Kod U2 – liczby rzeczywiste

Reprezentacja liczb rzeczywistych w U2:

- k – liczba bitów na część całkowitą
- p – liczba bitów na część ułamkową

Zapis

$$x_{k-1}\, x_{k-2} \ldots x_2\, x_1\, x_0\, ,\, x_{-1}\, x_{-2} \ldots x_{-(p-1)}\, x_{-p}$$

reprezentuje liczbę

$$-x_{k-1} \cdot 2^{k-1} + \sum_{i=0}^{k-2} x_i \cdot 2^i + \sum_{i=1}^{p} x_{-i} \cdot 2^{-i}$$

Zmiennopozycyjna reprezentacja liczb rzeczywistych

Reprezentacja zmiennopozycyjna

Fakt Każdą liczbę rzeczywistą $x \neq 0$ można jednoznacznie zapisać

$$x = (-1)^s \cdot m \cdot 2^c$$

gdzie

- $s \in \{0, 1\}$
- m to liczba rzeczywista, $m \in [1,2)$
- c to liczba całkowita

Reprezentacja zmiennopozycyjna

Reprezentacja cyfrowa liczby

$$x = (-1)^s \cdot m \cdot 2^c$$

gdzie $s \in \{0, 1\}$, $m \in [1,2)$, c to liczba całkowita:

- s – jeden bit
- binarny (zaokrąglony) zapis $1,m_1m_2m_3\ldots_{(2)}$ liczby *m* reprezentujemy przy pomocy ciągu $m_1m_2m_3\ldots m_M$, gdzie M - ustalone
- c – liczba w zapisie U2 na C bitach.

Razem:

Zapis na 1 + M + C bitach.

Reprezentacja zmiennopozycyjna

Zakres wartości

$$x = (-1)^s \cdot m \cdot 2^c$$

dla ustalonych M i C:

$$m \in \left[1, 1 + \sum_{i=1}^{M} \frac{1}{2^i}\right) = \left[1, 2 - \frac{1}{2^M}\right) \qquad c \in \left[-2^{C-1}, 2^{C-1} - 1\right]$$

Liczba o największym module (wartości bezwzgl.):

$$\pm\left(2 - \frac{1}{2^M}\right) \cdot 2^{2^{C-1}-1} \approx \pm 2^{2^{C-1}}$$

Liczba o najmniejszym module:

$$\pm 2^{-2^{(C-1)}}$$

Reprezentacja zmiennopozycyjna

Przykład

$$x = (-1)^s \cdot m \cdot 2^c$$

dla M=4 i C=3:

$$m \in \left[1{,}1 + \sum_{i=1}^{4} \frac{1}{2^i}\right) = \left[1{,}2 - \frac{1}{2^4}\right) = \left[1{,}1\frac{15}{16}\right) \quad c \in \left[-2^2, 2^2 - 1\right] = \left[-4{,}3\right]$$

Liczba o największym module:

$$\pm\left(2 - \frac{1}{2^M}\right) \cdot 2^{2^{C-1}-1} = \pm 1\frac{15}{16} \cdot 2^3 \approx \pm 2^4$$

Liczba o najmniejszym module:

$$\pm 2^{-2^{(C-1)}} = \pm 2^{-4}$$

Reprezentacja zmiennopozycyjna - normalizacja

Normalizacja liczby

$$x = (-1)^s \cdot m \cdot 2^c$$

dla s$\in${0,1}, $m \in \Re_+$ oraz liczby całkowitej c polega na znalezieniu *m'* oraz *c'* takich, że

$$x = (-1)^s \cdot m \cdot 2^c = (-1)^s \cdot m' \cdot 2^{c'}$$

oraz $m' \in [1,2)$, *c*' całkowite.

Przykład

$$x = 47 = 101111_{(2)} = 1 \cdot 1\mathbf{01111}_{(2)} \cdot 2^0 = 1 \cdot 1{,}\mathbf{01111}_{(2)} \cdot 2^5$$

Dla M=6 i C=4 liczba x ma następującą reprezentację:

s	m	c
0	011110	0101

Reprezentacja zmiennopozycyjna - mnożenie

Wejście:
zmiennopozycyjne reprezentacje liczb x_1 i x_2:

$$x_1 = (-1)^{s_1} \cdot m_1 \cdot 2^{c_1},\ \ x_2 = (-1)^{s_2} \cdot m_2 \cdot 2^{c_2}$$

Wyjście:
zmiennopozycyjna reprezentacja

$$x = x_1 \cdot x_2 = (-1)^{s} \cdot m \cdot 2^{c}$$

Reprezentacja zmiennopozycyjna - mnożenie

Wejście: s_1, m_1, c_1, s_2, m_2, c_2

$$x_1 = (-1)^{s_1} \cdot m_1 \cdot 2^{c_1}, \;\; x_2 = (-1)^{s_2} \cdot m_2 \cdot 2^{c_2}$$

Wyjście: *s*, *m*, *c*, takie, że $s \in \{0, 1\}$, $m \in [1,2)$, c to liczba całkowita

$$x = x_1 \cdot x_2 = (-1)^{s} \cdot m \cdot 2^{c}$$

Algorytm:

- dodaj znaki modulo 2: $s = (s_1 + s_2) \bmod 2$
- pomnóż mantysy: $m = m_1 \cdot m_2$
- dodaj cechy: $c = c_1 + c_2$
- znormalizuj wynik!!

Reprezentacja zmiennopozycyjna - mnożenie

Przykład (założenie M=3**)**:

$(-1)^1 \cdot 1{,}100_{(2)} \cdot 2^8 * (-1)^1 \cdot 1{,}110_{(2)} \cdot 2^6 =$

$(-1)^0 \cdot 1{,}100_{(2)} \cdot 1{,}\ 110_{(2)} \cdot 2^{8+6} =$

$(-1)^0 \cdot 11{,}101_{(2)} \cdot 2^{14} =$

normalizacja

$(-1)^0 \cdot 1{,}1101_{(2)} \cdot 2^{15} \approx$

zaokrąglenie

$(-1)^0 \cdot 1{,}110_{(2)} \cdot 2^{15}$

Reprezentacja zmiennopozycyjna - dodawanie

Wejście: s_1, m_1, c_1, s_2, m_2, c_2

$$x_1 = (-1)^{s_1} \cdot m_1 \cdot 2^{c_1}, \; x_2 = (-1)^{s_2} \cdot m_2 \cdot 2^{c_2}$$

Wyjście: *s, m, c takie, że:* $x = x_1 + x_2 = (-1)^s \cdot m \cdot 2^c$

Algorytm:

1. wyrównaj cechy do większej
2. dodaj mantysy ze znakiem:

   $m \leftarrow (-1)^{s1} m_1 + (-1)^{s2} m_2$
3. Ustal znak na podstawie sumy mantys **ze znakiem** (i zamień mantysę na jej wartość bezwzględną):
   - Jeśli (m>0): $s \leftarrow 0$,

     wpp $s \leftarrow 1$
   - $m \leftarrow | m |$
4. Normalizuj wynik!

Reprezentacja zmiennopozycyjna - dodawanie

Przykład:

$(-1)^0 \cdot 1{,}101_{(2)} \cdot 2^8 + (-1)^1 \cdot 1{,}\ 001_{(2)} \cdot 2^6 =$ Wyrównanie cech

$1{,}101_{(2)} \cdot 2^8 - 0{,}01001_{(2)} \cdot 2^8 =$

$(1{,}101_{(2)} - 0{,}01001_{(2)}) \cdot 2^8 =$ dodawanie mantys

$1{,}01011_{(2)} \cdot 2^8 =$ normalizacja

$(-1)^0 \cdot 1{,}01011_{(2)} \cdot 2^8$

Uwaga: w realizacji komputerowej wystąpić może zaokrąglenie lub błąd wynikające z długości cechy i mantysy (M i C).

Reprezentacja zmiennopozycyjna a reprezentacja stałopozycyjna

Stałopozycyjna:

- Wyniki obliczeń bez błędów
- Mały zakres **LUB** duży rozmiar pamięci

Zmiennopozycyjna

- Błędy zaokrągleń reprezentacji
- Błędy wyników operacji arytmetycznych
- Duży zakres wartości
- Oszczędność pamięci

Reprezentacja stałopozycyjna w C i Pythonie

Ansi C:

- Typ zmiennej określa zakres wartości i rozmiar pamięci zajmowanej przez zmienną (liczbę bajtów)
- Przekroczenie zakresu wartości oznacza błędne wyniki/wartości

Python

- W niektórych typach interpreter dopasowuje zakres (a zarazem rozmiar pamięci) do aktualnej wartości zmiennej

Reprezentacja tekstu, grafiki, wideo, dźwięku,…

Reprezentacja tekstu

Rozwiązanie „bazowe”

1. Ustal zestaw znaków, jego rozmiar
2. Przyporządkuj znaki kolejnym liczbom naturalnym

Modyfikacje i uszczegółowienia

- Ustal liczbę bajtów kodujących jeden znak
- Możliwa różna liczba bajtów dla różnych znaków (np. Unicode)

Przykłady

ASCII

Unicode (UTF)

Reprezentacja obrazu

Sposoby reprezentacji grafiki:

- grafika **rastrowa**: prostokątna siatka pikseli
- grafika **wektorowa**: matematyczny opis punktów, odcinków, figur płaskich, brył, krzywych itp. w układzie współrzędnych

Uwaga:

- Prezentacja grafiki na ekranie komputera wymaga (przetłumaczenia do) grafiki rastrowej.

Opis zawartości piksela:

- obraz monochromatyczny: intensywność jako liczba z określonego zakresu
- obraz kolorowy: trzy obrazy „monochromatyczne” – np. RGB (red, green, blue)

Dźwięk i wideo

Reprezentacja dźwięku:

- dźwięk jako funkcja w dziedzinie czasu (wartości funkcji to „poziom dźwięku”, „ciśnienie”,…)
- reprezentacja cyfrowa – próbkowanie (pomiar wartości funkcji w ustalonych odstępach czasowych)

Sekwencja wideo:

- ciąg obrazów/klatek (np. 50 na sekundę)
- ścieżka dźwiękowa

Kompresja: dźwięk i wideo

Reprezentacja obrazów, dźwięku, wideo – ogromny rozmiar danych:

- zdjęcie dobrej jakości zawiera około 10^6 pikseli
- dźwięk – kilkadziesiąt tysięcy próbek na sekundę
- wideo – kilkadziesiąt „klatek” (obrazów) na sekundę

Kompresja – oszczędny sposób reprezentacji, (czasem) kosztem utraty jakości, np.:

- JPG – obraz
- MP3 – dźwięk
- MPEG - wideo

Podsumowanie

1. Bit, bajt – podstawowe jednostki informacji
2. Binarny zapis liczb naturalnych, rzeczywistych
3. Cyfrowy zapis liczb całkowitych (U2)
4. Zmiennopozycyjny zapis liczb rzeczywistych
5. Operacje arytmetyczne na zapisie zmiennopozycyjnym
6. Cyfrowa reprezentacja
   - tekstu
   - grafiki
   - dźwięku
   - wideo