Teoretyczne podstawy informatyki

Wykład 1:
Informacja i zasady jej zapisu

http://th-www.if.uj.edu.pl/~erichter/dydaktyka/Dydaktyka2010/TPI-2010

http://kiwi.if.uj.edu.pl/~erichter/Dydaktyka2010/TPI-2010

http://hibiscus.if.uj.edu.pl/~erichter/Dydaktyka2010/TPI-2010

Zakres tematyczny

1. Co to jest informacja?
2. Algorytmy i struktury danych, poprawność algorytmu
3. Złożoność obliczeniowa cz. I
4. Rekursja, indukcja, iteracja, teoria prawdopodobieństwa
5. Modele danych: drzewa, listy, zbiory, relacje, grafy
6. Złożoność obliczeniowa, cz. II
7. Wzorce, automaty, wyrażenia regularne i gramatyki
8. Języki formalne, problemy NP-zupełne

Literatura

1. H. Abelson, et al., *Struktura i intepretacja programów komputerowych*
2. A. V. Acho, J. D. Ullman, *Wykłady z informatyki z przykładami w języku C*
3. T. H. Cormen, Ch. F. Leiserson, R. L. Rivest, *Wprowadzenie do algorytmów*
4. A. Drozdek, D. L. Simon, *Struktury danych w języku C*
5. D. Harel, *Rzecz o istocie informatyki*
6. J.E. Hopcroft, J. Ullman, *Wprowadzenie do teorii automatów, jezyków i obliczeń*
7. S. Kowalski, A. W. Mostowski, *Teoria automatów i lingwistyka matematyczna*
8. Ch. H. Papadimitriou, *Złożoność obliczeniowa*
9. W. Sikorski, *Wykłady z podstaw informatyki*
10. W. M. Turski, *Propedeutyka Informatyki*
11. N. Wirth, *Algorytmy i struktury danych = programy*

Zaliczenie przedmiotu

- zaliczenie ćwiczeń z zadań rachunkowych
- egzamin pisemny: podany obowiazujący zestaw zagadnień z wykładu

Informatyka

Mimo że informatyka jest stosunkowo nową dziedziną nauki, już wpływa na niemal wszystkie aspekty działalności człowieka.

- Jej wpływ na funkcjonowanie społeczeństw jest widoczny w rozpowszechnianiu się komputerów, systemów informatycznych, edytorów tekstu, arkuszy kalkulacyjnych itd...
- Ważną cecha informatyki jest ułatwianie samego programowania i czynienie programowania bardziej niezawodnym

Informatyka: mechanizacja abstrakcji

- Zasadniczo jednak **informatyka jest**
  - nauką o *abstrakcji* , czyli nauką o tworzeniu właściwego modelu reprezentującego problem i wynajdowaniu odpowiedniej techniki mechanicznego jego rozwiązywania
  - Informatycy tworzą abstrakcje rzeczywistych problemów w formach które mogą być rozumiane i przetwarzane w pamięci komputera

Abstrakcja

oznaczać będzie pewne uproszczenie, zastąpienie skomplikowanych i szczegółowych okoliczności występujących w świecie rzeczywistym zrozumiałym modelem umożliwiającym rozwiązanie naszego problemu. Oznacza to że abstrahujemy od szczegółów które nie maja wpływu lub mają minimalny wpływ na rozwiązanie problemu. Opracowanie odpowiedniego modelu ułatwia zajęcie się istotą problemu.

Informatyka: mechanizacja abstrakcji

W ramach tego wykładu omówimy

- **modele danych**: abstrakcje wykorzystywane do opisywania problemów
- **struktury danych:** konstrukcje języka programowania wykorzystywane do reprezentowania modelów danych. Przykładowo język C udostępnia wbudowane abstrakcje takie jak struktury czy wskaźniki, które umożliwiają reprezentowanie skomplikowanych abstrakcji takich jak grafy
- **algorytmy:** techniki wykorzystywane do otrzymywania rozwiązań na podstawie operacji wykonywanych na danych reprezentowanych przez abstrakcje modelu danych, struktury danych lub na inne sposoby

Informacja i zasady jej zapisu

1. Czym jest informacja?
2. Systemy zapisu liczb
   - System dwójkowy
   - System szesnastkowy
   - Bajty
3. Znaki i teksty
4. Wielkości liczbowe
5. Obrazy i dźwięki
6. Kompresja i szyfrowanie

Czym jest informacja?

Istnieje kilka różnych definicji pojęcia informacja (encyklopedia PWN)

1. Konstatacja stanu rzeczy, świadomość.
2. **Obiekt abstrakcyjny, który w sposób zakodowany może być przesyłany, przetwarzany i używany do sterowania**
3. Powiadamianie społeczeństwa lub określonych zbiorowości w sposób zobiektyzowany, systematyczny i konkretny za pomocą środków masowego przekazu.

Interesuje nas ta druga definicja, ponadto:

- Informacją zajmuje się nauka zwana Teorią Informacji. Dotyczy ona przekazywania wiadomości ze źródła wiadomości do ich przeznaczenia – odbiorcy.
- Informację możemy mierzyć ilościowo i jakościowo.

Czym jest informacja?

Informację przekazuje możliwość porównania dwóch stanów.

- Dzwonek dzwonka informuje nas, że ktoś nacisnął przycisk. Kiedy przycisk się zatnie i dzwonek dzwoni daje, już nie informuje nas o niczym. Gdy przestanie dzwonić a my porównamy dwie sytuacje, uzyskamy informację, że usterka została usunięta.

- Brak zmian to brak informacji: niezmienny sygnał nosi nazwę **szumu**. Nie można go jednak ignorować, gdyż często zakłóca przekaz właściwej informacji.

Jednostka informacji: bit

Podstawową jednostką informacji jest bit, oznaczany też poprzez „b”
(w ang. *kawałek*, skrót od **bi**nary digi**t**, czyli cyfra dwójkowa).

- Bit jest to podstawowa elementarna jednostka informacji: wystarczająca do zakomunikowania jednego z co najwyżej dwóch jednakowo prawdopodobnych zdarzeń.
- Bit stanowi podstawę zapisu informacji w różnych typach pamięci komputera. Wszystkie inne jednostki składają się z jego wielokrotności.
- Bit przyjmuje jedną z dwóch wartości, które zwykle oznacza się jako „0” lub „1”. Jest to oznaczenie stosowane w matematyce (wartość logiczna: „0” – fałsz, „1” - prawda) oraz przy opisie informacji przechowywanej w pamięci komputera i opisie sposobów kodowanie informacji.

Systemy zapisu liczb

System liczbowy – to inaczej zbiór reguł zapisu i nazewnictwa liczb.

Do zapisu liczb zawsze używa się pewnego skończonego zbioru znaków, zwanych cyframi (np.. arabskimi lub rzymskimi), które jednak można zestawiać ze sobą na różne sposoby otrzymując nieskończoną liczbę kombinacji.

System jedynkowy

Najbardziej prymitywnym systemem liczbowym jest jedynkowy system liczbowy, w którym występuje tylko jeden znak (np. 1). W systemie tym kolejne liczby są tworzone przez proste powtarzanie tego znaku.

Przykład:

3 w tym systemie zapisujemy jako 111, a 5 jako 11111.

Systemy addytywne

W tych systemach liczby tworzy się przez dodawanie kolejnych symboli.

Przykładem addytywnego systemu jest dobrze znany i wciąż stosowany rzymski system liczbowy z podstawowymi wielokrotnościami 10 i 5.

Jego cyfry to: I - 1, V - 5, X - 10, L - 50, C - 100, D - 500, M - 1000, (jednak w tym systemie w niektórych przypadkach występuje odejmowanie, a nie tylko dodawanie).

Przykład:

jeśli "X"=10,"V"=5,"I"=1 to XVI = 10+5+1 = 16

Systemy pozycyjne

Są to systemy które posiadają symbole (cyfry) tylko dla kilku najmniejszych liczb naturalnych: 0, 1, 2, ..., $g - 1$, gdzie g to tzw. podstawa systemu, która może być dowolną liczbą naturalną większą niż 1.

Cyfry te są kolejno umieszczane w ściśle określonych pozycjach i są mnożone przez odpowiednią potęgę g. W sytuacji, gdy dana potęga nie jest potrzebna do zapisu danej liczby, zostawia się w zapisie puste miejsce, lub częściej specjalny symbol. Współcześnie jest to cyfra 0.

Na przykład liczbę 5004,3 w dziesiętnym systemie liczbowym (czyli systemie, którego podstawą jest 10) odczytuje się jako:

$$5 \cdot 10^3 + 0 \cdot 10^2 + 0 \cdot 10^1 + 4 \cdot 10^0 + 3 \cdot 10^{-1} = 5 \cdot 1000 + 4 \cdot 1 + 3 \cdot 0{,}1 = 5004{,}3$$

Systemy liczbowe

Sześćdziesiątkowy system liczbowy, stosowany w Mezopotamii, w którym podstawowymi wielkościami były 10 i 60, był częściowo addytywny, częściowo pozycyjny. Jest on najstarszym znanym systemem każdego z tych dwóch rodzajów. W życiu codziennym spotykamy ślady babilońskiego systemu w podziale godziny na 60 minut, a minuty na 60 sekund, oraz w podziale kąta na minuty i sekundy kątowe.

Zaletą systemów addytywnych jest możliwość zapisu nawet dużych liczb (pod warunkiem, że są okrągłe) za pomocą jednego znaku, a wadą złożoność, kłopoty interpretacyjne i zbyt wielka liczba cyfr przy mało okrągłych liczbach, oraz bardzo skomplikowany sposób dokonywania za ich pomocą prostych operacji arytmetycznych, wymagający zapamiętywania długich tabel.

Zaletą systemów pozycyjnych jest ich klarowność, łatwość dokonywania nawet złożonych operacji arytmetycznych oraz możliwość zapisu dowolnie dużej liczby, jednak do zapisu bardzo dużych liczb (nawet okrągłych) jest potrzebna duża liczba cyfr.

Współcześnie powszechnie używany jest system dziesiątkowy. W informatyce często stosowany jest system dwójkowy (binarny), ósemkowy i szesnastkowy (heksadecymalny).

Systemy liczbowe w informatyce

Z racji reprezentacji liczb w pamięci komputerów za pomocą bitów, najbardziej naturalnym systemem w informatyce jest system dwójkowy.

Ze względu na specyfikę architektury komputerów, gdzie często najszybszy dostęp jest do adresów parzystych, albo podzielnych przez 4, 8 czy 16, często używany jest szesnastkowy system liczbowy. Sprawdza się on szczególnie przy zapisie dużych liczb takich jak adresy pamięci, zakresy parametrów itp.

Na przykład:

- $2^{16} = 65536_{10} = 10000_{16}$
- $2^{32} = 4294967296_{10} = 100000000_{16}$

10000_{16} i 100000000_{16} są znacznie łatwiejsze do zapamiętania.

System szesnastkowy często spotykany jest też na strona WWW (HTML), gdzie stosowany jest do zapisu kolorów.

Bajt

Jest to najmniejsza adresowalna jednostka informacji pamięci komputerowej, składająca się z bitów, w praktyce przyjmuje się że jeden bajt to **8 bitów** (zostało to uznane za standard w 1964 r.).

Jeden bajt może reprezentować zatem $2^8 = 256$ różnych wartości, które mogą oznaczać zapisywane informacje.

Bajt oznaczany jest poprzez „**B**".

Stosowanie przedrostków *kilo, mega, giga* itp. jako do określania odpowiednich potęg liczby dwa jest niezgodne z wytycznymi układu SI (np. *kilo* oznacza 1000, a nie 1024).

W celu odróżnienia przedrostków o mnożniku 1000 od przedrostków o mnożniku 1024, w styczniu 1997 r. pojawiła się propozycja ujednoznacznienia opracowana przez **IEC** (ang. *International Electrotechnical Commission)* polegająca na dodawaniu litery "i" po symbolu przedrostka dwójkowego, oraz "bi" po jego nazwie.

Bajt

Nowe przedrostki nazywane zostały **przedrostkami dwójkowymi (binarnymi)**. Jednak ta propozycja rozwiązania problemu niejednoznaczności przedrostków nie została przyjęta przez wszystkie środowiska.

Przykładowo producenci nośników pamięci i urządzeń sieciowych, z powodów marketingowych, wolą korzystać z przedrostków układu SI, co bywa źródłem nieporozumień co do faktycznej pojemności pierwszych i prędkości drugich (która podawane jest w bitach na sekundę).

Producent określa, że jego urządzenie cechuje się pojemnością 1 GB, co każdy odczytuje jako 1073741824 bajtów, a w rzeczywistości produkt ma 1000000000 bajtów co daje różnicę 70 MiB.

Wielokrotności bajtów

Przedrostki dziesiętne (SI)			Przedrostki binarne (IEC)		
Nazwa	Symbol	Mnożnik	Nazwa	Symbol	Mnożnik
Kilobajt	kB / KB	$10^3 = 1000^1$	Kibibajt	KiB	$2^{10} = 1024^1$
Megabajt	MB	$10^6 = 1000^2$	Mebibajt	MiB	$2^{20} = 1024^2$
Gigabajt	GB	$10^9 = 1000^3$	Gibibajt	GiB	$2^{30} = 1024^3$
Terabajt	TB	$10^{12} = 1000^4$	Tebibajt	TiB	$2^{40} = 1024^4$

Kodowanie informacji

Jak to się dzieje że w pamięci komputera można przechowywać teksty, obrazy, dźwięki i liczby znacznie różniące się od zestawu 0 – 255?

- Dzięki kodowaniu informacji

Bez kodowania nie ma zapisu różnorodnych informacji w pamięci komputera.
Kodowanie występuje w każdym programie i na każdym poziomie.

Znaki i teksty

- Teksty składają się ze znaków.
- Podstawą zapisu jest jeden bajt.
- 1 bajt przyjmuje 256 różnych wartości.
- Ważną cechą kodowania jest jednoznaczność:

  przyjęcie pewnego sposobu kodowania powinno być powszechne:

  ASCII: 0 – 127 standardowe, 128 – 255 zależne od kraju

Np.
litera W: 01010111, czyli 87.

Znaki specjalne	0-31, 127
Spacja	32
Cyfry	48-57
Duże litery	65-90
Małe litery	97-122
Pozostałe znaki: Kropka, nawiasy, itd….	33-47, 58-64, 91-96, 123-127

Znaki i teksty

- W rozszerzonym kodzie ASCII znajdują się niektóre znaki matematyczne oraz znaki symulujące elementy grafiki na komputerach.
- Przetwarzanie informacji nie oznacza samego zapisywania tekstów. Dodatkowe informacje (wytłuszczenie, różne czcionki, akapity…) też trzeba zakodować.
  - Przykład: W kodzie ASCII znaki 0-31 i 127 nie są wykorzystane. Jeżeli umówimy się że po **jednym** z tych znaków następny zmienia znaczenie, to mamy **255** dodatkowych kodów.
    Np. kod 65 występujący po tym wybranym znaku nie będzie oznaczać litery A tylko jedną z funkcji sterujących pracą edytora.
- Dodatkowe kody pozwalają zapisać znacznie więcej informacji, ale wymagają dekodowania wg. tych samych reguł z jakimi były kodowane.

ASCII

- **ASCII** (ang. *American Standard Code for Information Interchange*) - 7-bitowy kod przyporządkowujący liczby z zakresu 0-127 literom (alfabetu angielskiego), cyfrom, znakom przystankowym i innym symbolom oraz poleceniom sterującym.
- Litery, cyfry oraz inne znaki drukowane tworzą zbiór znaków ASCII. Jest to 95 znaków o kodach 32-126. Pozostałe 33 kody (0-31 i 127) to tzw. kody sterujące służące do sterowania urządzeniem odbierającym komunikat, np. drukarką czy terminalem.
- Ponieważ kod ASCII jest 7-bitowy, a większość komputerów operuje na 8-bitowych bajtach, dodatkowy bit można wykorzystać na powiększenie zbioru kodowanych znaków. Powstało wiele różnych rozszerzeń ASCII wykorzystujących ósmy bit (np. norma ISO 8859), nazywanych stronami kodowymi. Również kodowanie UTF-8 można uważać za rozszerzenie ASCII, tutaj jednak dodatkowe znaki są kodowane na 2 i więcej bajtach.

Liczby naturalne

- Podobnie jak w dziesiętnym systemie pozycyjnym, w systemie dwójkowym liczby naturalne przedstawiamy jako sumę potęg bazy (2) z odpowiednimi wagami: 0 i 1.
- Każda liczba naturalna ma zatem reprezentację postaci:

$$\sum_{k=0}^{m} a_k 2^k \text{ , gdzie } a_k \in \{0,1\}$$

- Reprezentacja ta jest jednoznaczna, jeśli przyjmiemy, że nie stosujemy wiodących zer.

Liczby naturalne

Pierwsze 16 wartości			
$(k)_{10}$	$(k)_2$	$(k)_{10}$	$(k)_2$
0	0	8	1000
1	1	9	1001
2	10	10	1010
3	11	11	1011
4	100	12	1100
5	101	13	1101
6	110	14	1110
7	111	15	1111

Liczby naturalne

- Jeśli ustalimy z góry pewną liczbę **n** cyfr, za pomocą których będziemy reprezentowali liczby naturalne, to uzyskamy **n** - cyfrowe reprezentacje, uzupełniając je do pełnych **n** cyfr zerami z lewej strony.
- Dla **n** = 8 (1 bajt), mielibyśmy kolejno :
  00000000, 00000001, 00000010, … , 11111111.
  Czyli liczby od 0 do 255.
- Widać, że różnych wartości **n** – cyfrowych jest 2^n : od 0 do $2^n - 1$

Zapis dziesiętny do binarnego

- Jak w prosty sposób znajdować reprezentacje dwójkowe liczb naturalnych? Są dwie proste metody.

- Oznaczmy poprzez **m** liczbę w zapisie dziesiętnym, i załóżmy **m > 0**
- Znajdujemy największą liczbę **d** = 2^k nie większą niż **m**. Piszemy jedynkę, odejmujemy od **m** wartość **d**, a następnie kolejno dla wszystkich mniejszych potęg dwójki sprawdzamy, czy mieszczą się one w tym co zostało z **m**. Jeśli dana potęga dwójki się nie mieści to dopisujemy zero, wpp. dopisujemy jedynkę i odejmujemy tę wartość od tego, co zostało z **m**.
- Przykładowo dla **m = 13.**

m	Potęga dwójki	Wypisano
13	8	1
5	4	11
1	2	110
1	1	1101
0		

Zapis dziesiętny do binarnego

- Drugi sposób:
- Zaczynamy od liczby **m**, a następnie dopóki **m** jest większe od zera, dzielimy **m** przez **2**, zapisując kolejno otrzymywane reszty. Ciąg reszt odczytany od końca da nam poszukiwaną reprezentację.
- Przykładowo dla **m = 13**.

Zostało z **m** po dzieleniu przez **2**	Reszta z dzielenia **m** przez **2**
13	1
6	0
3	1
1	1
0	

Liczby naturalne

- Warto pamiętać że nie możemy reprezentować każdej liczby naturalnej. Wymagałoby to od nas nieskończonej pamięci.
- Zazwyczaj do reprezentacji liczb całkowitych używa się standardowo określonej z góry liczby bitów (typowo 8, 16, 32 lub 64).
- Stąd też pojawia się problem, że nie wszystkie działania będą wykonalne. Np. dodawanie… skoro zbiór liczb reprezentowanych w komputerze jest skończony, to istnieją w nim dwie największe liczby i dodanie ich do siebie spowoduje, że wynik będzie niereprezentowalny.

Liczby całkowite, system znak - moduł

- Umówmy się zatem, że przeznaczymy określoną liczbę **n** bitów, aby reprezentować liczby całkowite. Powstaje pytanie: jak zapisywać liczby ujemne? Istnieją co najmniej 3 sposoby.
- Kodowanie w systemie **znak-moduł:**

Umawiamy się że jeden bit, np. pierwszy z lewej, rezerwujemy na określenie znaku liczby. Pozostałe **n-1** bitów reprezentuje moduł liczby w tradycyjny sposób. Jeśli pierwszy bit znaku jest równy 0, to liczba jest nieujemna, a jeśli 1, to jest niedodatnia.

Bity	Wartość	Bity	Wartość
0000	0	1000	-0
0001	1	1001	-1
0010	2	1010	-2
0011	3	1011	-3
0100	4	1100	-4
0101	5	1101	-5
0110	6	1110	-6
0111	7	1111	-7

System znak moduł

- Problem: zero jest kodowane na 2 sposoby
  - Nieujemne zero (0000)
  - Niedodatnie zero (1000)

- Jest to duża wada systemu, gdyż trzeba uważać, by porównując wartości dwóch liczb nie stwierdzić, że **$0 \neq -0$**

- W kodzie tym dla liczb **n** – bitowych mamy zakres od $-2^{n-1} + 1$ do $2^{n-1} - 1$, różnych liczby reprezentowanych jest zatem **$2^n - 1$**.

System znak – moduł odwrotny

- System ten jest podobny do poprzedniego, z tą różnicą, iż jeśli pierwszy bit jest 1, to pozostałe **n**-1 bitów reprezentuje **negatyw** modułu liczby.
- W tym systemie znów występuje podwójne kodowanie zera.
- Różnych reprezentowanych liczb jest $2^n - 1$

Bity	Wartość	Bity	Wartość
0000	0	1000	-7
0001	1	1001	-6
0010	2	1010	-5
0011	3	1011	-4
0100	4	1100	-3
0101	5	1101	-2
0110	6	1110	-1
0111	7	1111	-0

System znak – moduł odwrotny, zalety

- Jaki jest sens kodowania liczb w ten sposób?
  - Otóż dodawanie jest dużo prostsze.
- W systemie znak – moduł, aby dodać dwie liczby przeciwnych znaków, trzeba by najpierw ustalić znak wyniku i zdecydować, od którego modułu odjąć który.
- W systemie znak – moduł odwrotny wystarczy, nie przejmując się znakami, dodać bitowo reprezentacja. Jeśli ostatecznie pojawi się w przeniesieniu całego wyniku jedynka, dodać ją jeszcze raz do otrzymanego wyniku.

Przykładowo: -4 + 6

1011	-4
+ 0110	+6
= (1) 0001	= (1) 1
= 0010	2

Kod uzupełnieniowy

- Najstarszy bit (pierwszy od lewej) **n** – bitowej reprezentacji traktujemy jako – 2^{n-1}.
- Pozostałe natomiast jako kolejno 2^{n-2}, …, 2^0.
- Dla **n** = 4, kolejno bity od lewej mają wartości: -8, 4, 2 ,1.
- Wartości są z zakresu od -2^{n-1} do $2^{n-1}-1$
- Każda wartość jest reprezentowana jednoznacznie
- Liczb ujemnych jest o 1 więcej niż dodatnich.

Bity	Wartość	Bity	Wartość
0000	0	1000	-8
0001	1	1001	-7
0010	2	1010	-6
0011	3	1011	-5
0100	4	1100	-4
0101	5	1101	-3
0110	6	1110	-2
0111	7	1111	-1

System uzupełnieniowy

- Dodawanie w systemie uzupełnieniowym, jest jeszcze bardziej wygodne. Po prostu dodaje się bitowo reprezentacje i jeśli pojawia się bit przepełnienia, to się go ignoruje.
- Istnieje górne i dolne ograniczenie zakresu wartości liczb.
- Ograniczenia te zależą od tego ile bajtów przeznaczymy na liczbę oraz od systemu kodowania znaku.
- Wszystkie otrzymane wartości są dokładne.
- Przy takim zapisie umawiamy się, że przecinek leży za prawym skrajnym znakiem (zatem reprezentujemy tylko liczby całkowite)
- Ten system kodowania nazywamy też **systemem stałoprzecinkowym**.
- Uwaga: we wszystkich stosowanych systemach, liczby dodatnie reprezentuje się identycznie.

Przykładowo: -4 + 6

1100	- 4
+ 0110	+ 6
= 0010	= 2

Ułamki

- Podobnie jak w systemie dziesiętnym korzystamy z ujemnych potęg bazy (dziesiątki) po przecinku, tak tu będziemy rozważali binarne rozwinięcia ułamków za pomocą ujemnych potęg dwójki.
- Po przecinku, oddzielającym część całkowitą od ułamkowej, kolejne bity będą odpowiadały wartościom kolejno: ½, ¼, …

System	Ułamki		
Dziesiętny	¼	½	¾
Binarny	0.01	0.1	0.11

- Zapisanie ułamka dziesiętnego binarnie jest proste, jeśli tylko mianownik ułamka jest potęgą dwójki. Wystarczy zapisać licznik binarnie, a następnie przesunąć przecinek o tyle pozycji w lewo, ile wynosi wykładnik potęgi dwójki.
- Np.: 5/16 zapisane binarnie to 0.0101, jest to po prostu 5 czyli 101, przesunięte o 4 pozycje w prawo.

Ułamki

- Co zrobić gdy mianownik nie jest potęgą dwójki?
- Przez **u** oznaczmy ułamek, który chcemy zapisać binarnie. Niech **0 < u < 1** .
- Dopóki 0 **< u**, bądź otrzymujemy nie napotkane wcześniej wartości **u**, będziemy wykonywać:
  - Pomnóż u przez 2
  - Jeżeli u jest mniejsze od 1, dopisz cyfrę 0
  - Wpp. Dopisz cyfrę 1 i odejmij od u cyfrę 1.
- Gdy powtórzy się wartość **u**, otrzymamy szukany wynik, przy czym wypisany ciąg jest okresowy, a jego okresem jest ciąg bitów między powtórzeniami **u**.

Ułamki - przykład

Bity	Wartość **u**
0.	2/7
0	4/7
1	1/7
0	2/7
0	4/7

Binarne rozwinięcie: 2/7 = 0.(010)

Bity	Wartość **u**
0.	1/10
0	2/10
0	4/10
0	8/10
1	6/10
1	2/10

Binarne rozwinięcie: 1/10 = 0.0(0011)

Zauważmy iż nawet tak prosta liczba jak 1/10 ma nieskończone binarne rozwinięcie okresowe. Gdy chcemy reprezentować ją w komputerze, jesteśmy zmuszeni do zaokrąglenia tej wartości i w rzeczywistości otrzymujemy tylko coś koło 1/10.

Ułamki - zaokrąglenia

- Skoro nie da się dokładnie reprezentować wartości wymiernych w komputerze, trzeba je zaokrąglać. Reguły są bardzo proste.
- Jeśli chcemy zaokrąglić na **k** - tej pozycji, to patrzymy na cyfrę na następnej pozycji (**k+1**)
  - Jeśli jest ona równa **0**, to zaokrąglamy w dół, 'obcinamy ogon' rozwinięcia binarnego
  - Wpp, również 'odrzucamy ogon', jednakże na **k** - tym miejscu przybliżenia dodajemy **1,** czyli zaokrąglamy w górę.

Ułamki – zaokrąglenia, przykład

Przybliżenie ułamka:

$$(1/10)_{10} = (\ 0.0(0011) \ )_2$$

Uzyskujemy najlepsze przybliżenia dla ułamków o mianownikach będących kolejnymi potęgami dwójki.

Numer bitu zaokrąglenia	Zaokrąglenie	Wartość zaokrąglenia
1	0.0	0/2
2	0.00	0/4
3	0.001	1/8
4	0.0010	2/16
5	0.00011	3/32
6	0.000110	6/64
7	0.0001101	13/128
8	0.00011010	26/256
9	0.000110011	51/512
…		

System stałopozycyjny

- Liczby rzeczywiste mają część całkowitą i ułamkową, układ stałopozycyjny charakteryzuje się tym że przeznaczamy w nim **stała**, z góry określoną liczbę (**k**) bitów na część całkowitą, tak jak i na ułamkową (**u**).
- Tym sposobem możemy przedstawić liczby z zakresu od -2^{k-1} do $2^{k-1} - 2^{-u}$, i wartości w nim reprezentowane są rozłożone równomiernie co 2^{-u}.
- Jest to sposób nieekonomiczny, gdy operujemy na dużych liczbach (np. w astronomii), nie potrzebujemy przeznaczać pamięci na część ułamkową. Gdy za to operujemy bardzo małymi liczbami (np. fizyka cząstek elementarnych), nie potrzebujemy przeznaczać dużo pamięci na część całkowitą…

System zmiennopozycyjny

- System zmiennopozycyjny jest intuicyjny i powszechnie stosowany. Gdy chcemy zapisać np. stała Plancka nikt nie będzie pisał:
  - $\mathbf{h} = 0{,}00000000000000000000000000000000663\ \mathrm{J \cdot s}$,

gdyż byłoby to nieczytelne. Zostanie użyty natomiast zapis:

  - $\mathbf{h} = 6{,}63 \cdot 10^{-34}\ \mathrm{J \cdot s}$

- W takim zapisie podaje kilka cyfr znaczących (mantysę) oraz określamy rząd wielkości (cechę) poprzez podanie potęgi podstawy systemu.

System zmiennopozycyjny

- Pojawia się problem jednoznaczności, np. liczbę 3/8 można przedstawić jako:
  - $3/8 \cdot 2^{0}$
  - $3/4 \cdot 2^{-1}$
  - $3/2 \cdot 2^{-2}$
  - $3 \cdot 2^{-3}$ ….
  - $3/16 \cdot 2^{1}$
  - $3/32 \cdot 2^{2}$ …
- Rozwiązanie problemu jednoznaczności jest przyjęcie pewnego standardu:
  - Mantysa musi mieścić się w przedziale (½, 1] dla wartości dodatnich.
  - Oraz [-1, -½) dla wartości ujemnych.
  - Wyjątkiem jest reprezentacja zera.

System zmiennopozycyjny

- Każdą niezerową liczbę rzeczywistą reprezentujemy za pomocą przybliżenia wymiernego w postaci pary (**m, c**), takich że:
  - $\mathbf{m} \in [-1, -½) \cup (½, 1]$

m jest mantysą, zaś c jest cechą. Interpretacja takiej reprezentacji wyraża się wzorem:

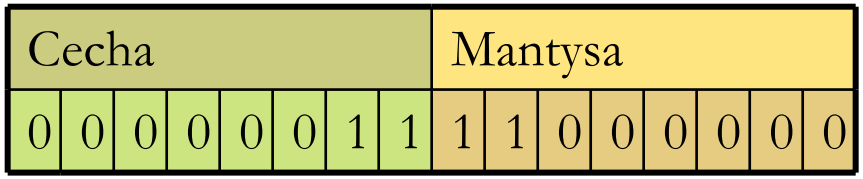
- $\mathbf{x = m\ P^{c}}$

Oczywiście w naszym przypadku $P = 2$ (podstawa systemu)

- System ten, umożliwia zapis liczb rzeczywistych z ustalonym błędem względnym.

System zmiennopozycyjny

- Liczba binarna zapisana w postaci cecha – mantysa na dwóch bajtach:



Tutaj: c = 3, m = - 1.

- W praktyce zwykle na cechę przeznaczamy jeden bajt, na mantysę minimum trzy bajty.
  - Ilość bajtów przeznaczonych na cechę decyduje o zakresie.
  - Ilość bajtów przeznaczona na mantysę decyduje o błędzie.
- Liczby w mantysie są kodowane w systemie znak – moduł.
- Zaś dla cechy w systemie uzupełnieniowym.

Standard zapisu liczb zmiennopozycyjnych

- Standard IEEE 754 (ang. Institute of Electrical and Electronics Engineers) dla liczby rzeczywistej: (4 bajty)

Kolejne bity (od lewej)	Znaczenie
1 (jeden)	Znak mantysy
2-9 (osiem)	Cecha
10-33 (dwadzieścia trzy)	Mantysa

- Standard IEEE dla liczby podwójnej precyzji: (8 bajtów)

Kolejne bity (od lewej)	Znaczenie
1 (jeden)	Znak mantysy
2-12 (jedenaście)	Cecha
13-64 (pięćdziesiąt dwa)	Mantysa

Standard IEEE – wartości specjalne

Wartość	Zapis
+0	Wszystkie bity są zerami.
-0	Bit znaku jest ustawiony, reszta jest zerami.
liczby małe (*ang. denormalized numbers*)	Wykładnik równy zero, mantysa różna od 0, nie zakłada się wiodącego niezerowego bitu; są to liczby zbyt małe aby mogły być reprezentowane z taką samą precyzją jak "zwykłe" liczby
$\pm\infty$	Ustawione są wszystkie bity cechy, mantysa jest równa 0, może się pojawić np. jako wynik dzielenia przez 0.
NaN (*ang. Not a Number*)	Ustawione są wszystkie bity cechy, mantysa różna od 0, może się pojawić np. jako wynik pierwiastkowania liczby ujemnej.

System zmiennopozycyjny

- W tym przykładzie przeznaczymy 3 bity na cechę, i 5 na mantysę.
- Jedyne legalne dodatnie mantysy to: 00101, 00110, 00111, 01000, gdyż $\mathbf{m} \in [-1, -½) \cup (½, 1]$
- Maksymalną możliwą do zapisania liczbą jest 8.
- Tworząc analogiczną tabelkę dla cech ujemnych, zauważymy iż minimalną dodatnią liczbą, możliwą do zapisania jest 5/128.
- Reprezentowane wartości nie są rozłożone równomiernie, im dalej od zera tym rzadziej.
- W obrębie jednej cechy wartości są rozłożone równomiernie

Cecha	Mantysa	Wartość
000	00101	5/8
000	00110	6/8
000	00111	7/8
000	01000	8/8
001	00101	10/8
001	00110	12/8
001	00111	14/8
001	01000	16/8
010	00101	20/8
010	00110	24/8
010	00111	28/8
010	01000	32/8

System zmiennopozycyjny a zero

- Nie da się zera przedstawić w podanej postaci, gdyż żadna z liczb $\mathbf{m\ 2^{c}}$ zerem być nie może dla mantys co do modułu większych od ½ .

- Najczęściej stosowane rozwiązanie polega na tym, że wyłącza się jedną z cech (najmniejszą) i ustala, że jeśli liczba ma tę cechę, to jest równa zero niezależnie od mantysy.

Dodawanie w systemie zmiennopozycyjnym

- Przyjrzyjmy się, jak wygląda dodawanie w naszym systemie.
- Dostając dwa argumenty, musimy pamiętać, żeby dodawane bity odpowiadały sobie wartościami.
- Trzeba zatem przed rozpoczęciem dodawania ujednolicić cechy, przesuwając o odpowiednią liczbę bitów jedną z mantys.
- Obowiązuje zasada, że dostosowujemy mniejszą cechę do większej.
- Następnie wykonujemy dodawanie i normalizujemy wynik, na końcu go zaokrąglając.

Dodawanie w systemie zmiennopozycyjnym - przykład

- Rozważmy dodawanie 3/32 + 3/32.
- Każda z tych liczb ma zapis 101 00110.
- Cechy są równe, więc nie potrzeba niczego denormalizować.
- Wykonujemy dodawanie mantys otrzymując 01100.
- Przesuwamy wynik o jeden w prawo – czyli dzielimy go przez 2 i dodajemy do cechy 1.
- Wynik, to po prostu 110 00110, czyli 3/16.
- Zauważmy przy okazji, że mnożenie przez 2 i dzielenie przez 2 można wykonywać bezpośrednio dodając lub odpowiednio odejmując jedynkę od cechy!

Dodawanie w systemie zmiennopozycyjnym - przykład

- Rozważmy teraz dodawanie 3/8 + 5/2.
- Reprezentacje bitowe tych liczb to odpowiednio:
  - 111 00110 czyli $2^{-1} \cdot 3/4$
  - 010 00101 czyli $2^{2} \cdot 5/8$
- Różnica w cechach to 3, więc o tyle bitów w prawo należy przesunąć mantysę 3/8.
- Załóżmy, że na ten czas mamy dodatkowe bity na mantysę.
- Mamy zatem do dodania dwie mantysy:
  - 00000110
  - 00101
- Wynikiem czego jest mantysa: 00101110, którą musimy zaokrąglić.
- Otrzymamy wynik: 010 00110. (Cecha bez zmian, a mantysa w skutek zaokrąglenia powiększona.)
- Ostatecznie otrzymaliśmy liczbę 3, która to najlepiej przybliża 23/8.

Dodawanie w systemie zmiennopozycyjnym - przykład

- Rozważmy teraz dodawanie 3/16 + 5/2.
- Reprezentacje bitowe tych liczb to odpowiednio:
  - 110 00110 czyli $2^{-2} \cdot 3/4$
  - 010 00101 czyli $2^{2} \cdot 5/8$
- Różnica w cechach to 4, więc o tyle bitów w prawo należy przesunąć mantysę 3/16.
- Mamy zatem do dodania dwie mantysy:
  - 000000110
  - 00101
- Wynikiem czego jest mantysa: 001010110, którą musimy zaokrąglić.
- Otrzymamy wynik: 010 00101. (Cecha bez zmian, mantysa bez zmian.)
- Zatem liczba 3/16 jest zbyt mała, by dodanie jej do liczby 5/2 zmieniło wartość tej drugiej.

Dodawanie w systemie zmiennopozycyjnym – wnioski

- Nie tylko zero nie zmienia wartości drugiego argumentu przy dodawaniu.
- Dodawanie zmiennopozycyjne nie jest łączne.
- Co widać na przykładzie 5/2 + 3/16 + 3/16.
- W przypadku gdy działania wykonamy w kolejności:
  - (5/2 + 3/16) + 3/16 otrzymamy wynik 5/2,
  - 5/2 + (3/16 + 3/16) otrzymamy wynik 3.

Obrazy, dźwięki, …

- Ciągi bajtów muszą przechowywać teksty, liczby, muzykę, animacje: wszystkie informacje zapisywane w wyniku wykonywanych działań.
- Potrzebne jest zakodowanie informacji, inne niż w przypadku liczb czy też tekstów.
- Kodowanie koloru: model RGB, model YUV.
- Kodowanie obrazu: formaty: BMP (bitmapa), GIF, JPEG
- Kodowanie muzyki: formaty: MP1, MP2, MP3, MP4, WAV, OGG, …
- Warto zauważyć pewną zależność:
  - Im większa precyzja, tym większy rozmiar pliku.

Kodowanie koloru – RGB

- Jeden z modeli przestrzeni barw, opisywanej współrzędnymi RGB.
- Jego nazwa powstała ze złożenia pierwszych liter angielskich nazw barw: R – red (czerwonej), G – green (zielonej) i B – blue (niebieskiej), z których model ten się składa.
- Jest to model wynikający z właściwości odbiorczych ludzkiego oka, w którym wrażenie widzenia dowolnej barwy można wywołać przez zmieszanie w ustalonych proporcjach trzech wiązek światła o barwie czerwonej, zielonej i niebieskie.
- Model RGB jest jednak modelem teoretycznym a jego odwzorowanie zależy od urządzenia (ang. device dependent), co oznacza, że w każdym urządzeniu każda ze składowych RGB może posiadać nieco inną charakterystykę widmową, a co za tym idzie, każde z urządzeń może posiadać własny zakres barw możliwych do uzyskania.

Kodowanie koloru – RGB

- Model RGB miał pierwotnie zastosowanie do techniki analogowej, obecnie ma również do cyfrowej. Jest szeroko wykorzystywany w urządzeniach analizujących obraz (np. aparaty cyfrowe, skanery) oraz w urządzeniach wyświetlających obraz (np. telewizory, monitory komputerowe).
- Zapis koloru jako RGB często stosuje się w informatyce (np. palety barw w plikach graficznych, w plikach html). Najczęściej stosowany jest 24-bitowy zapis kolorów (**po 8 bitów na każdą z barw składowych**), w którym każda z barw jest zapisana przy pomocy składowych, które przyjmują wartość z zakresu **0-255**. W modelu RGB **0** (dla każdej ze składowych) oznacza kolor czarny, natomiast **255** (analogicznie) kolor biały.
- Kolor RGB można obliczyć tak:
  - numer koloru = R * 256^2 + G * 256 + B

Kodowanie koloru – YUV

- Model barw, w którym Y odpowiada za jasność obrazu (luminancję), a pod UV zaszyta jest barwa - dwie chrominancje.
- Model YUV był wykorzystywany w czasie przechodzenia od telewizorów czarno-białych na kolorowe. Czarno-białe odbiorniki wyświetlały jedynie jasność obrazu, a kolorowe dodawały kolor, co pozwoliło posiadaczom czarno-białych nie pozbywać się odbiorników od razu. Y - luminacja (dla obrazu czarno-białego) U - przeskalowana składowa B V - przeskalowana składowa R
- $Y = 0.299 *R + 0.587*G +0.114 *B$
- $U = B – Y$
- $V = R - Y$

Kompresja

- Jest to działanie mające na celu zmniejszanie objętości pliku. Przy kompresji wykorzystuje się podobieństwa i regularności występujące w plikach. Program przeprowadza analizę i wybiera fragmenty, które można zapisać w sposób zajmujący mniejszą liczbę bajtów. Wyróżniamy dwa typy:
  - Kompresja bezstratna: odtworzona informacja jest identyczna z oryginałem, dekompresja jest w pełni odwracalna (np. GIF).
  - Kompresja stratna: polega ona na eliminowaniu pewnych elementów oryginału, w celu lepszej efektywności kompresji (np. JPEG).
- Możemy powiązać jakość ze stopniem kompresji.

Szyfrowanie

- Szyfr – rodzaj kodu, system umownych znaków stosowany w celu utajnienia wiadomości, żeby była ona niemożliwa (lub bardzo trudna) do odczytania przez każdego, kto nie posiada odpowiedniego klucza.
- Szyfrowanie natomiast jest procedurą przekształcania wiadomości nie zaszyfrowanej w zaszyfrowaną.
- Wiadomość przed zaszyfrowaniem nazywa się tekstem jawnym (plaintext), a wiadomość zaszyfrowaną – szyfrogramem (ciphertext).
- Kryptologia – nauka o przekazywaniu informacji w sposób zabezpieczony przed niepowołanym dostępem.
- Kryptologię dzieli się na:
  - Kryptografię, czyli naukę o układaniu systemów kryptograficznych,
  - Kryptoanalizę, czyli naukę o ich łamaniu.

Szyfrowanie

- Dwa najpopularniejsze algorytmy kryptografii asymetrycznej (czyli takiej, w której się używa zestawów dwu lub więcej powiązanych ze sobą kluczy, umożliwiających wykonywanie różnych czynności kryptograficznych) to RSA i ElGamal.
- RSA Został stworzony w 1978 przez zespół: Ronald Rivest, Adi Shamir, Leonard Adleman (nazwa RSA jest akronimem utworzonym z pierwszych liter nazwisk jego twórców). RSA opiera się na trudności faktoryzacji dużych liczb. Znalezienie szybkiej metody faktoryzacji doprowadziłoby do złamania RSA, aczkolwiek nie ma dowodu, że nie da się go złamać w inny sposób.
- ElGamal natomiast jest oparty na trudności problemu logarytmu dyskretnego w ciele liczb całkowitych modulo duża liczba pierwsza. Algorytm w połowie lat 80. XX wieku przedstawił Egipcjanin Taher Elgamal.
- Algorytm ElGamala umożliwia szyfrowanie oraz obsługę podpisów cyfrowych. Setki modyfikacji algorytmu ElGamala (podobnie jak modyfikacje algorytmu RSA) mają różne inne zastosowania.