Binárny kód

Cieľom kódovania je prispôsobenie vyjadrenia informácie **možnostiam technického zariadenia** (napr. Morseova abeceda pre telegraf) alebo **možnostiam ľudí** (Braillovo písmo pre nevidiacich). **Kód** je ľubovoľná, vopred dohodnutá a všeobecne známa množina pravidiel, ktorá dovoľuje vyjadriť informáciu tak, aby sa dala požadovane spracovať.
Šifrovanie sa používa na utajenie obsahu informácie, predovšetkým pri komunikácii. Je to postup, ktorý prevedie (zašifruje) zmysluplný text do nečitateľnej podoby. **Kľúč** je tajná informácia, bez ktorej nie je možné zašifrovaný text prečítať (dešifrovať). Symetrická šifra používa pre šifrovanie aj dešifrovanie ten istý kľúč. Asymetrická šifra používa verejný kľúč pre šifrovanie a súkromný kľúč pre dešifrovanie (okrem prijímateľa správy ho nik nemusí vedieť).
Pre počítače, z konštrukčného hľadiska, je najvhodnejší zápis informácií v binárnom kóde, pre svoju jednoduchosť a spoľahlivosť. **Binárny kód** používa len dva znaky, znak 0 a znak 1. Ktorúkoľvek z dvoch číslic, 0 alebo 1, nazývame **bit** (binary digit – dvojková číslica) a označujeme **b**. Bit je jednotkou elementárnej informácie – informácie o tom, ktorá z dvoch možností nastala, v ktorom z dvoch možných stavov sa systém nachádza (vypnutý/zapnutý, zavretý/otvorený, stojí/v pohybe,...). Informácia zapísaná v binárnom kóde je zapísané ako postupnosť núl a jednotiek. Skupinu ôsmich bitov nazývame **bajt** (byte, **B**). Informácie zapísané v binárnom kóde nazývame **digitálne informácie**.
Binárny kód a počítanie v dvojkovej sústave tvoria most medzi informáciou a elektronickými obvodmi počítača, ktoré tiež môžu byť len v jednom z dvoch stavov (tečie/netečie elektrický prúd daným miestom, je tam/nie je tam elektrické napätie, je tam/nie je tam elektrický náboj,...).

Pozičné číselné sústavy:

- dekadická (desiatková), so základom 10
  - používa cifry (znaky) 0, 1, ... , 9
  - $1024_{10} = 1.10^3 + 0.10^2 + 2.10^1 + 4.10^0 = 4 + 20 + 0 + 1000 = 1024$
- binárna (dvojková), so základom 2
  - používa len dva znaky 0 a 1
  - $1000001_2 = 1.2^6 + 0.2^5 + 0.2^4 + 0.\ 2^3 + 0.2^2 + 0.\ 2^1 + 1.2^0 = 64 + 1 = 65_{10}$
  - $1111010_2 = 1.2^6 + 1.2^5 + 1.2^4 + 1.\ 2^3 + 0.2^2 + 1.\ 2^1 + 0.2^0 = 64 + 32 + 16 + 8 + 0 + 2 + 0 = 122_{10}$
- hexadecimálna (šestnástková), so základom 16
  - používa 16 znakov: 0, 1, 2,..., 9, A (10_{10}), B (11_{10}), C (12_{10}), D (13_{10}), E (14_{10}) a F (15_{10})
  - $61_{16} = 6.16^1 + 1.16^0 = 96 + 1 = 97$
  - $7A_{16} = 7.16^1 + 10.16^0 = 112 + 10 = 122$
  - $ABBA_{16} = 10.16^3 + 11.16^2 + 11.16^1 + 10.16^0 = 43\ 962$

Algoritmus (návod) na prevod desiatkového čísla na dvojkové číslo:
1. Desiatkové číslo vydelíme 2.
2. Zapíšeme zvyšok (0 alebo 1).
3. Výsledok delenia opäť vydelíme 2.
4. Zvyšok zapíšeme pred predchádzajúci zvyšok.
5. Opakujeme 3. a 4. krok tak dlho, kým výsledok delenia nie je 0.

Napr. $3926_{10} = 111101010110_2$

Použitie algoritmu na prevod dvojkového čísla na desiatkové:
Napr. $111101010110_2 = 1.2^{11} + 1.2^{10} + 1.2^9 + 1.2^8 + 0.2^7 + 1.2^6 + 0.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + + 0.2^0 = 2048 + 1024 + 512 + 256 + 0 + 64 + 0 + 16 + 0 + 4 + 2 + 0 = 3926_{10}$

Počítanie s dvojkovými číslami:

Sčítanie:		Napr.:		
	0 + 0 = 0		**45**	**101101**
	0 + 1 = 1		**+ 93**	**+ 1011101**
	1 + 0 = 1	prenos:	1--	11111-1-
	1 + 1 = 0 prenos 1	súčet:	**138**	**10001010**

Algoritmus na prevod čísla z dvojkovej do šestnástkovej sústavy:

1. binárne číslo rozdelíme sprava na štvorčíslia
2. každé štvorčíslie prepočítame na hexadecimálne číslo, presnejšie na jednu cifru hexadecimálneho čísla
3. vypočítané cifry tvoria hľadané hexadecimálne číslo

Napr.

1. číslo 1000001_2 rozdelíme na „štvorice“ sprava: 100 0001 alebo 0100 0001
2. $100_2 = 1.2^2 = 4 = \mathbf{4}_{16}$; $0001_2 = 1.2^0 = 1 = \mathbf{1}_{16}$
3. $1000001_2 = \mathbf{41}_{16}$

alebo

1. 1111010_2 = 0111 1010
2. $111_2 = 1.2^2 + 1.2^1 + 1.2^0 = \mathbf{7}$; $1010 = 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 = 10$ a v šestnástkovej sústave sa zapisuje znakom **A**
3. $1100001_2 = \mathbf{7A}_{16}$

alebo

1. 11111111_2 = 1111 1111
2. $1111_2 = 1.2^3 + 1.2^2 + 1.2^1 + 1.2^0 = 15$ teda **F**; $1111_2 = 1.2^3 + 1.2^2 + 1.2^1 + 1.2^0 = \mathbf{F}$
3. $11111111_2 = \mathbf{FF}_{16}$

Poznámka:

V kódovacích tabuľkách ASCII aj Unicode sú pod číslami:

$65_{10} = 1000001_2 = 41_{16}$ veľké písmeno anglickej abecedy A

$97_{10} = 1100001_2 = 61_{16}$ malé písmeno anglickej abecedy a

$122_{10} = 1111010_2 = 7A_{16}$ malé písmeno anglickej abecedy z (122-97 = 25, t.j. anglická abeceda má 26 písmen)

Úloha:

Určte minimálne základy číselných sústav, v ktorých sú vyjadrené čísla 10, 99, 32, AA, 2C, F7, 51, 104, G2, 707 a povedzte o jedna väčšie číslo ako zapísané.

Napr. 707 - číslo z číselnej sústavy s minimálnym základom 8, $707_8 + 1 = 710_8$ (viete to aj dokázať?)

Digitalizácia informácií

Človek svojimi zmyslami prijíma **analógové** (spojité) informácie. Počítače spracúvajú **digitálne** informácie. Informácie z reálneho sveta (blízke človeku) treba pred spracovaním v počítači **digitalizovať** – podľa dohodnutých pravidiel zapísať v binárnom kóde.

Podstata digitalizácie – fázy digitalizácie:

1. rozdelenie informácie na elementy (text na znaky, obrázok na pixely, ...)
2. očíslovanie všetkých rôznych možností (elementov) s využitím tvrdenia:

n bitov umožňuje zakódovať 2^n rôznych hodnôt

Poznámka: Pri číslovaní rôznych možností (elementov) používame n-tice bitov, napr. pre 3 bity sú to trojice [0,0,0], [0,0,1], [0,1,0], [0,1,1], [1,0,0], [1,0,1], [1,1,0] a [1,1,1].

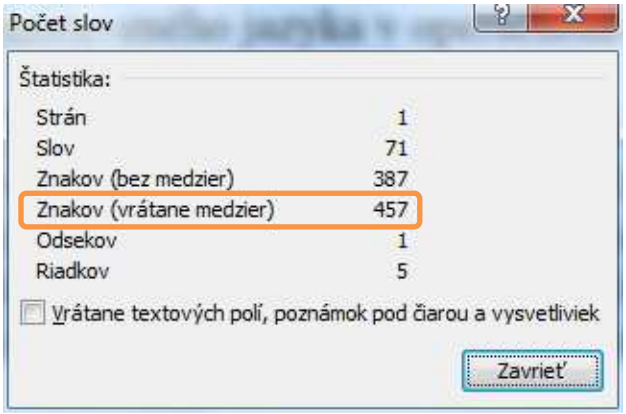
Rozlišujeme nasledovné typy informácií: textová, grafická (rastrová alebo vektorová) a multimediálna (zvuk, melódia, video, animácia). Skôr, ako si načrtneme princípy digitalizácie niektorých z nich, najprv si povieme niečo o súboroch, v ktorých sú zdigitalizované informácie (texty, obrázky, videá, zvuky,...) uložené.
Budeme v nich rozlišovať tzv. hlavičku a dátovú časť. **Hlavička** obsahuje informácie o použitom spôsobe kódovania informácie, napr. o formáte, v akom bol súbor uložený, a **dátová časť** obsahuje element po elemente informácie o ich vlastnostiach, napr. čísla znakov z textu, čísla farieb pixelov obrázka, ... Pokúsime sa odhadnúť, čo obsahuje hlavička a vypočítať veľkosť dátovej časti súboru, v ktorom je informácia uložená.
Poznámka:
Ide o princípy digitalizácie, „odvodené“ odhady o veľkosti súboru nie sú všeobecne platné! Napríklad v rôznych operačných systémoch sú používané rôzne štandardy.

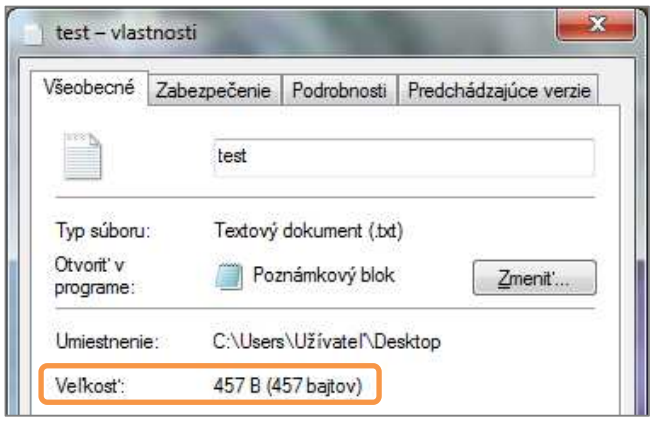
Digitalizácia textovej informácie

1. fáza – rozdelenia textu na znaky
2. fáza - znaky sú pomocou medzinárodne dohodnutej kódovacej tabuľky (v ktorej sú očíslované všetky použité znaky) prekódované na bajty.

ASCII kód (uvádzaný aj ako kódovanie ANSI)**:**
je 8 bitový, t.j. 1 znak je zakódovaný vo ôsmich bitoch resp. v jednom bajte (1B/znak); skupina 8-mich bitov umožňuje zakódovať 2^8 = 256 rôznych znakov; prvých 128 znakov je pevne daných (z písmen len veľké a malé písmená anglickej abecedy), zvyšných 128 znakov sa mení podľa nastaveného jazyka v operačnom systéme počítača (národné abecedy).

Úloha: V programe Word otvorte textový dokument a označte do bloku jeden odsek. Po kliknutí v ľavom dolnom paneli ľavým tlačidlom myši na Slová:... sa zobrazí Štatistika, v ktorej môžete zistiť počet znakov v označenom texte (obrázok nižšie vľavo je ukážka pre odsek, ktorý práve čítate). Text prekopírujte do aplikácie Poznámkový blok a uložte ako textový súbor vo formáte ANSI. Presvedčte sa, že veľkosť súboru zodpovedá súčinu: počet znakov krát 1B/znak.

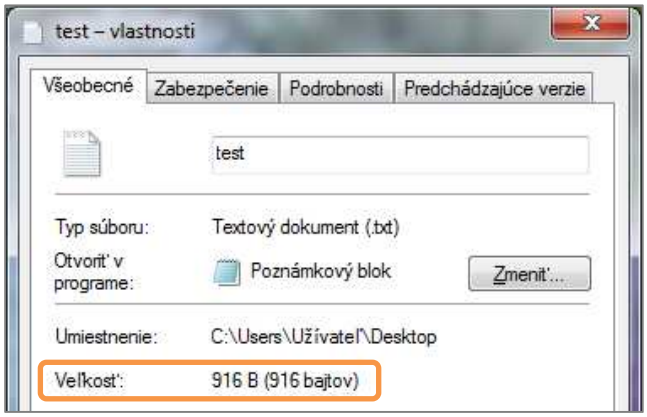




Unicode:
16 bitový, t.j. 1 znak je zakódovaný v 16-tich bitoch resp. v 2B (2B/znak); 16 bitov umožňuje zakódovať 2^{16} = 65 536 rôznych znakov; pozri napríklad vo Worde *Vložiť – Symbol... – Symboly* - Písmo: (normálny text). Každý z bežne používaných znakov (okrem čínskych, japonských, kórejských a pod.) má jednoznačne priradené číslo (šestnásticu bitov) a znak sa na „celom svete“ zobrazí rovnako. Prvých 128 znakov je zhodných so znakmi z ASCII kódu.

Ak by sme text z predchádzajúcej úlohy z Poznámkového bloku uložili do súboru v kódovaní Unicode (predpokladáme použitie operačného systému Windows; postup: Súbor - Uložiť ako... -

Kódovanie: Unicode), veľkosť súboru sa zdvojnásobí plus pribudne dvojbajtová informácia o použitom kódovaní (obrázok): 457 B x 2 + 2 B = 916 B.



Pokúsme sa o zhrnutie:

1. text sme „vytvorili“ v jednoduchom textovom editore, v ktorom je k dispozícii len základný formát písma a preto by mal súbor obsahoval len nevyhnutné informácie o vlastnostiach znakov textu
2. keď sme uložili text vo formáte ANSI (ASCII), veľkosť súboru zodpovedala veľkosti dátovej časti vypočítanej podľa vzorca: počet znakov krát 1B/znak
3. keď sme uložili text vo formáte Unicode, veľkosť dátovej časti sa zdvojnásobila, čo je pochopiteľné, lebo v Unicode (v OS Windows) je každý znak zakódovaný v 2B. Na veľkosť hlavičky súboru, obsahujúcej informáciu o použitom kódovaní, zostala veľkosť 2B.

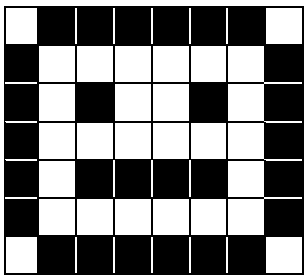
Záver:
Pre veľkosť dátovej časti súboru (dátčs), v ktorom je uložený text vo formáte ASCII alebo Unicode bez formátovacích znakov a len „s bežne používanými“ znakmi, platí:

veľkosť dátčs [B] = počet znakov [znak] x **počet bajtov použitého kódu na znak [B/znak]**

Ak by boli v texte vytvorené odseky, každé stlačenie klávesu Enter je v textovom súbore zakódované dvoma znakmi – CR (Carriage Return/na začiatok riadka), riadiaci kód 13_{10} (D_{16}) a LF (Line Feed/nový riadok), riadiaci kód 10_{10} (A_{16}).
Skutočná veľkosť súboru bude prakticky vždy väčšia, minimálne o veľkosť hlavičky, a pri použitom zložitejšom formátovaní textu, aj o ďalšie informácie o vlastnostiach textu.
Podrobnejšie pozrite napríklad na Wikipédii. O kódovaní znakov, a s ním spojenými problémami, sa môžete dočítať aj v študijnom texte Kódovanie znakov.

Digitalizácia grafickej rastrovej informácie

1. fáza – rozdelenie obrázka na pixely preložením mriežky (rastra) cez obrázok



Bitová mapa (bitmapa, bmp):

0 1 1 1 1 1 1 0
1 0 0 0 0 0 0 1
1 0 1 0 0 1 0 1
1 0 0 0 0 0 0 1
1 0 1 1 1 1 0 1
1 0 0 0 0 0 0 1
0 1 1 1 1 1 1 0

0 znamená biely pixel
1 znamená čierny pixel

Obrázok sa skladá z 8 x 7 pixelov (bitová mapa teoreticky z 56 bitov).

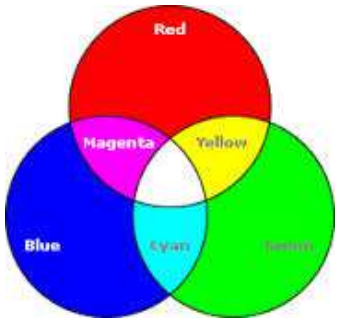
2. fáza – očíslovanie všetkých možností, u obrázka to najčastejšie znamená očíslovanie všetkých použitých farieb. Opäť platí: n bitov umožňuje zakódovať 2^n farieb.

Napr.:	2 farby (= 2^1)	potrebujeme 1 bit, použijeme 0 alebo 1
	3 alebo 4 farby (= 2^2)	potrebujeme 2 bity, použij. dvojice bitov
	5 až 8 farieb (= 2^3)	potrebujeme 3 bity, použij. trojice bitov
	16 farieb (= 2^4)	potrebujeme 4 bity, ...
	256 farieb (= 2^8)	potrebujeme 8 bitov

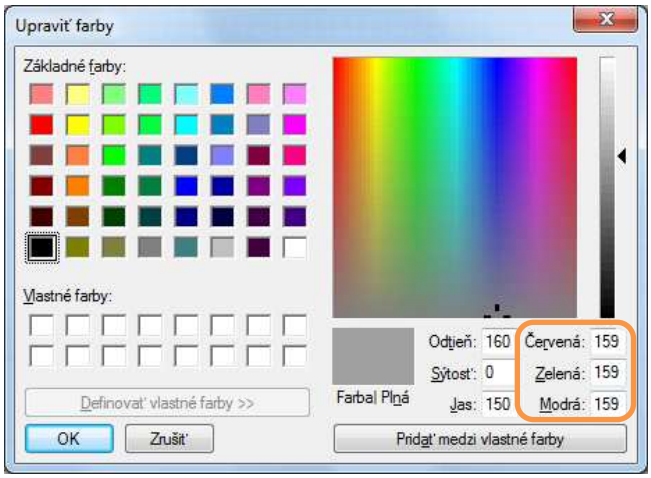
High Colour, 65 536 farieb (= 2^{16})	potrebujeme 16 bitov
True Colour, 16,7 mil. farieb (= 2^{24})	potrebujeme 24 bitov

Pri **čierno-bielom obrázku** by sa dátová časť súboru (dátčs), v ktorom je takýto obrázok uložený, mohla rovnať počtu pixelov obrázka x 1bit/px (znak „x“ budeme používať ako symbol násobenia). Hlavička súboru, v ktorom je uložený obrázok, by mala obsahovať informácie o použitom formáte pri ukladaní súboru, o rozmeroch obrázka, t.j. koľko pixelov je v riadku a koľko riadkov tvorí obrázok, o použitých farbách a pod.
Záver: Odvodiť vzorec pre výpočet veľkosti súboru, ktorý obsahuje rastrový obrázok, nie je až také jednoduché ☺.
V prvom rade si musíme niečo povedať o kódovaní farieb.

Farebný model RGB



Farebný model RGB je najčastejšie využívané kódovanie farby pixelu obrázka. Empiricky (skúsenosťou, praxou) sa zistilo, že takmer všetky farby sa dajú vytvoriť zmiešaním ľubovoľných troch nezávislých farieb. Najvýhodnejšie bolo použitie farieb červenej (Red), zelenej (Green) a modrej (Blue). Pri použití 256-tich rôznych odtieňov každej z farieb červená, zelená a modrá možno vytvoriť až 16 777 216 rôznych zložených farieb (16 777 216 = 256 x 256 x 256). Pri takomto kódovaní je odtieň každej základnej farby RGB zakódovaný 8 bitmi (8 bitov dáva 2^8 možností, t.j. 256 rôznych odtieňov) a teda výsledná farba pixelu je zakódovaná 3 x 8 bitov = 24 bitmi resp. 3B/px (výhodné pre zaznamenanie v počítači). Kód [255, 0, 0] je sýta červená farba, [0, 255, 0] je sýta zelená farba, [0, 0, 255] je sýta modrá farba, [0, 0, 0] je čierna farba a [255, 255, 255] je biela farba atď.
„Nenápadne“ používame pojem **pixel** (px) namiesto „bod obrázka“. Je to odôvodnené tým, že teraz už vieme, že každý „bod obrázka“ je vlastne zložený z troch bodov - odtieňov červenej, zelenej a modrej. Tieto tri body (subpixely) vytvárajú jeden obrazový bod alebo pixel.
Ďalším pojmom je **paleta farieb**, ktorá vypovedá o kódovaní farieb použitých v obrázku a býva súčasťou hlavičky súboru. Nech je obrázok čo i len **dvojfarebný**, súbor by mal obsahovať tabuľku, v ktorej prvá farba bude mať číslo 0 a následne bude uvedená trojica odtieňov modelu RGB - čísel od 0-255, zložením ktorých dostávame farbu číslo 0. Podobne pre farbu číslo 1. Takže súbor by mal, okrem dátovej časti veľkosti 1b pre každý pixel obrázka (použitá farba č.0 alebo farba č.1), obsahovať aj tabuľku - paletu farieb veľkosti 50b (2 farby x 25b na jednu farbu; 25b = 3 x 8b RGB + 1b číslo farby).
Každý obrázok uložený vo formáte **256 farebnej bitovej mapy**, môže obsahovať iných 256 farieb, preto súbor obsahuje tabuľku, v ktorej je v každom z 256 riadkov informácia: číslo použitej farby (od 0 po 255) a tri bajty modelu RGB, zložením ktorých dostaneme danú farbu. Celá paleta farieb má preto veľkosť 256 riadkov x 4B/riadok = 1024B.



Pri použití síce čierno-bieleho obrázka, ale **s 256 odtieňmi sivej**, by mala mať paleta farieb rovnakú veľkosť (1024B).
Pri použití formátu **24-bitovej mapy** je farba každého pixelu zakódovaná priamo v dátovej časti (3B/px) a preto hlavička takéhoto súboru neobsahuje paletu farieb. S kódovaním farieb modelu RGB sa dá pekne pohrať v Skicári - Upraviť farby (pozri obrázok).

Záver:
Súbor obsahujúci farebný rastrový obrázok obsahuje v hlavičke informácie o použitom formáte, rozmeroch obrázka (počet px v riadku a počet riadkov), paletu farieb (tabuľku kódovania požitých

farieb) a v dátovej časti podľa palety farieb zakódované informácie o farbách jednotlivých pixelov tvoriacich daný obrázok. Ak je obrázok uložený v 24-bitovej mape, súbor (presnejšie hlavička súboru) neobsahuje paletu farieb.
Dátová časť súboru, obsahujúceho nekomprimovaný rastrový obrázok (pre nás formát bmp), má veľkosť:

veľkosť dátčs bmp [B] = počet pixelov obrázka x **počet bajtov použitej palety farieb na px**

Malo by nám byť zrejmé, že skutočný súbor bude väčší o hlavičku súboru, ktorej veľkosť nevieme tak jednoducho vypočítať (veľkosť palety farieb v nej asi áno).
V niektorých príkladoch je zhoda so skutočnosťou dosť dobrá. Napríklad pre rastrový obrázok 1024 x 1024 px uložený v 24-bitovej mape mal súbor veľkosť 3 145 782 B a výpočet ukázal 3 145 728 B. Pre ten istý obrázok (1024 x 1024 px) uložený vo formáte 256 farebnej bitovej mapy mal súbor veľkosť 1 049 654 B a výpočtom sme dostali 1 049 600 B (paleta v hlavičke: 256 farieb krát 4B/farbu plus dátová časť: 1024 x 1024 px krát 1B/px). V oboch príkladoch je rozdiel medzi skutočnosťou a našim výpočtom, v ktorom sme nezarátali napríklad v hlavičke zakódované informácie o rozmeroch obrázka, rozdiel 54 B.
V každom prípade súbory, v ktorých sú uložené rastrové obrázky, najčastejšie z digitálnych fotoaparátov, mávajú veľkú veľkosť (milióny bajtov).
Na zmenšenie veľkosti súboru s rastrovým obrázkom sa používajú rôzne algoritmy, ktoré sa často ešte kombinujú:

- už spomenutá **paleta farieb** (kódovacia tabuľka - „slovník“ pre farby použité v obrázku)
  Ak by sme v každom obrázku chceli mať k dispozícii všetkých 16 777 216 farebných odtieňov modelu RGB, museli by sme neustále používať 24 bitové kódovanie farieb. Ak je v obrázku použitých napríklad len 256 farieb, zníženie pamäťových nárokov spočíva v tom, že očíslujeme všetky použité farby v obrázku číslami od 0 do 255, a potom kódujeme každý pixel tak, že uvedieme poradové číslo farby v palete. Tým miesto troch pamäťových miest (zaznamenaný odtieň červenej, zelenej a modrej) farbu každého pixelu zakódujeme len pomocou jedného pamäťového miesta (súbor však musí obsahovať v hlavičke tabuľku - paletu farieb pre v obrázku použité farby).
- ak údaje určené na kompresiu (komprimáciu) obsahujú dlhé postupnosti rovnakých hodnôt, možno tieto postupnosti nahradiť kratším zápisom: hodnota a počet opakovaní (kódovanie podľa dĺžky, úsekové kódovanie, RLE) alebo slovníkové kódovanie, kde určité postupnosti hodnôt sú zaznamenané v slovníku a nahradené kratším zápisom (pozri koniec tohto študijného textu) a pod.
- **formát gif** využíva slovník (3B RGB/px nahrádza pomocou slovníka 1B/px) aj RLE kódovanie, pri ktorom sa v najviac 256 farebnom obrázku rozpoznajú skupiny rovnakých obrazových bodov a do súboru sa o tom zapíše len krátka informácia (zjednodušene napr. modrá modrá modrá modrá modrá modrá modrá modrá sa zapíše modrá 9 krát). Preto obrázky, ktoré neobsahujú rozličné komplikované farebné prechody, zaberú na disku málo miesta. Kompresia je bezstratová. Stratovou sa stáva, ak obrázok obsahuje viac ako 256 rôznych farebných odtieňov, potom sa najprv musí zredukovať počet farieb na 256, a tak sa pri ukladaní do súboru GIF stratí jeho pôvodná kvalita. Formát GIF umožňuje vďaka priradenia „priehľadnej“ hodnoty jednej z farieb v palete (zobrazí sa pozadie obrázka) jednoduché animácie.
- **formát png** umožňuje bezstratovú kompresiu; bol vyvinutý ako zdokonalenie a náhrada formátu GIF. PNG ponúka podporu 24-bitovej farebnej hĺbky a lepšiu kompresiu. Navyše obsahuje osembitovú priehľadnosť, čo znamená, že obrázok môže byť v rôznych častiach rôzne priehľadný, neumožňuje však jednoduché animácie, ktoré naopak umožňuje formát GIF.
- **formát jpg** znamená väčšinou použitie stratovej kompresie na rastrový obrázok, pri ktorej sú nahradené body s malými zmenami farieb bodmi s príbuznou farbou (stratia sa niektoré farby, ostrosť prechodov, ľudské oko to nemusí zaregistrovať). JPEG je vhodný pre fotografické snímky alebo maľby realistických scenérií s hladkými prechodmi v tóne a farbe. V tomto

prípade funguje omnoho lepšie ako čisté bezstratové metódy (napr. GIF), pričom poskytuje stále veľmi dobrú kvalitu obrazu.

Pomer medzi veľkosťou súboru po kompresii a pred kompresiou je bezrozmerné reálne číslo k (úspešnosť kompresie), pre ktoré platí: $0 < \mathbf{k} \leq 1$. Čím je číslo **k** menšie, tým bola kompresia súboru úspešnejšia, t.j. skomprimovaný súbor má menšiu veľkosť ako pôvodný súbor. Napr. pre k = 0,5 = ½ má skomprimovaný súbor polovičnú veľkosť oproti neskomprimovanému súboru. Pre k = ⅓ je skomprimovaný súbor zmenšený na tretinu pôvodného súboru, teda je trikrát menší ako pôvodný súbor. Úspešnosť kompresie býva často číslo okolo 0,1 t.j. skomprimovaný súbor má veľkosť len jednej desatiny pôvodného súboru.
Zrejme platí:

veľkosť komprimovaného súboru [B] = k x veľkosť nekomprimovaného súboru [B]

Teoretická veľkosť súboru, v ktorom je uložené neozvučené video

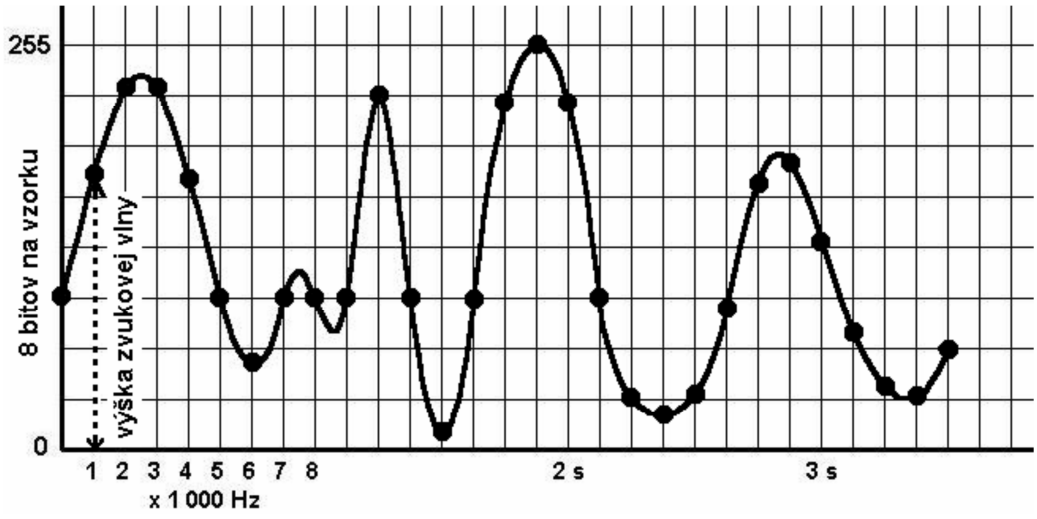
S obrázkami úzko súvisí aj neozvučené video, keďže je tvorené sekvenciou (postupnosťou) pravidelne sa vymieňajúcich obrázkov (snímok) počas trvania videa. Snímky sa vymieňajú minimálne 25 až 30 krát za sekundu, čo pri zotrvačnosti oka asi 0,1 sekundy (snímka už nie je zobrazená a my ju „ešte vidíme“) stačí na to, aby sme vnímali dej ako plynulý.

veľkosť dátčs, v ktorom je uložené video [B] =
= veľkosť snímky [B] x počet snímok za sekundu [s^{-1}] x doba trvania videa [s]

Ako vidieť z rovnice, súbory, ktoré obsahujú videá, budú mať zrejme veľkú veľkosť, ktorá je priamoúmerná kvalite a dĺžke videa. Pre videosúbory je použitie účinnej kompresie nevyhnutnosťou.

Digitalizácia zvukovej informácie

V obrázku je zakreslená analógová zvuková informácia, ktorá sa s hustotou vzorkovacej frekvencie digitalizuje odčítaním výšky zvukovej vlny, napr. posledný bod má kód $63_{10} = 00111111_2$ (v obrázku je zakreslené len každé tisíce odčítanie výšky zvukovej vlny z 8000 odčítaní za sekundu)



vz. frekvencia: 8 000 Hz

počet bitov na vzorku: 8

kvalita: mono

dĺžka v sek.: 3,5

veľkosť súboru: 27,4 KiB

veľkosť dátčs wav [B] =
= vzorkovacia frekvencia [s^{-1}] x počet bajtov na vzorku [B] x kvalita [] x dĺžka v sekundách [s]

kde
vzorkovacia frekvencia je číslo udávajúce, koľkokrát za sekundu sa zosníma výška analógovej vlny (bežne od 8 000 Hz t.j. 8 kHz a viac),
počet bitov na vzorku je číslo vyjadrujúce bohatosť zvuku; 8 bitov znamená 256 hodnotovú stupnicu v smere osi y, 16 bitov znamená 2^{16} = 65 536 hodnotovú stupnicu
kvalita znamená počet kanálov, pri mono jeden (1), pri stereo dva (2)
dĺžka v sekundách je čas trvania skladby (zvuku).

Výpočet veľkosti dátovej časti zvukového súboru uloženého vo formáte wav sa dobre zhoduje s veľkosťou súboru, ktorý vznikne v počítači napríklad po vytvorení zvukového záznamu pomocou mikrofónu. Zrejme skutočná veľkosť súboru bude väčšia o hlavičku nesúcu informácie potrebné na dekódovanie dát uložených v súbore.

Tak ako súbory typu bmp, možno aj súbory typu wav komprimovať. Najznámejší je formát **mp3**, čo je stratová kompresia aplikovaná na formát wav. Využíva nedokonalosť ľudského sluchu, keď sú zo záznamu vynechané niektoré tóny, ktoré by bežné ľudské ucho pravdepodobne nezaznamenalo. Úspešnosť kompresie býva približne jedna jedenástina (0,09).

Poznámka
Pri prezeraní vlastností uloženého zvukového súboru operačný systém častejšie ako vyššie pomenované parametre zobrazí **bitovú frekvenciu**, čo je rýchlosť dátového toku v bps. Čím je bitová frekvencia väčšia, tým je nahrávka kvalitnejšia (viac dát za sekundu a teda aj viac zvukov a tónov). Pre súbory typu wav zrejme bitová frekvencia zodpovedá súčinu vzorkovacia frekvencia x počet bajtov na vzorku x kvalita. Bitová frekvencia sa používa aj pri formáte mp3.

Analógia (podobnosť) medzi typmi grafickej a zvukovej informácie:

grafická informácia	zvuková informácia	zdôvodnenie
formát bmp	formát wav	súbor obsahuje čísla: farby pixelov obrázka resp. výšky zvukovej vlny
vektorový obrázok	formát midi	súbory obsahujú inštrukcie (návody)
formát jpg	formát mp3	stratová kompresia bmp resp. wav

Prenosová rýchlosť v počítačovej sieti

$$\text{prenosová rýchlosť [b/s]} = \frac{\text{množstvo prenesených dát v bitoch}}{\text{čas prenosu v sekundách}}$$

Používané jednotky: b/s = bs^{-1} = bps (číta se „bit per sekunda“)
Násobky: kbps, k = 10^3, t.j. 1000; Mbps, kBps, MBps a pod.

Úlohy na precvičenie
Predpokladajte, že v príkladoch sú uvedené parametre súborov zodpovedajúce vyššie „odvodeným“ vzorcom (napríklad neberte do úvahy existenciu hlavičiek súborov)!

Príklad:
Čísla 71 a 29 preveďte do dvojkovej pozičnej sústavy a sčítajte. Výsledok skontrolujte prevodom do desiatkovej pozičnej sústavy. [71 1000111; 29 11101; 100 1100100]

Príklad:
Preveď dvojkové číslo 101111001010 do šestnástkovej pozičnej číselnej sústavy.
Príklad:
Nájdite všetky riešenia rovnice 10aa01 + bbbbb = 1cccccc
kde a, b, c sú len binárna 0 alebo 1. Spravte skúšku správnosti!
Príklad:
Kód 01001101010000010101010001010101010010 (40 bitov)
obsahuje prvých päť písmen slova MATURANT. O aký kód ide, zdôvodnite a dopíšte chýbajúce bity.
Príklad:
Aká bola rýchlosť prenosu v počítačovej sieti v kbps v okamihu prenosu súboru s veľkosťou 76,8 KiB; ak prenos trval 1,7 sekundy (pozor: k znamená 10^3)? [370kbps]
Príklad:
Textový dokument bez formátovacích značiek má po kompresii s úspešnosťou 0,1 (1:10) veľkosť 15 KiB. Urči a zdôvodni použitý kód, v ktorom je text uložený, ak zaberá 24 strán, priemerne 40 riadkov na strane a 80 znakov v riadku (zanedbajte aj všetky znaky pre zlom riadka a strany).
Príklad:
Rastrový obrázok formátu bmp rozmerov 1600 x 1200 pixelov je uložený v súbore s veľkosťou 5 625 KiB. V akej farebnej palete bol uložený? Navrhni farebnú paletu, pre ktorú sa veľkosť súboru zmenší a bez výpočtu uveď koľkokrát. Ako sa zmení hlavička súboru?
Príklad:
Zvuková nahrávka bola uložená v najnižšej kvalite (8 kHz, 1B, mono) vo formáte wav a následne skomprimovaná. Jej prehranie trvá 5,5 minúty. Aká je veľkosť súborov wav a mp3, ak úspešnosť kompresie bola 1:11?
Príklad:
Rastrový obrázok rozmerov 1600 x 1200 pixelov je uložený v komprimovanom súbore (úspešnosť kompresie 1:10) s veľkosťou 562,5 KiB. V akej farebnej palete bol uložený? Navrhni farebnú paletu, pre ktorú sa veľkosť súboru zmenší a bez výpočtu uveď koľkokrát.
Príklad:
Prenos textového dokumentu bez formátovacích značiek v počítačovej sieti s rýchlosťou 409,6 kbps trval 0,1 sekundy. Urči a zdôvodni použitý kód, v ktorom je text uložený, ak má text 32 riadkov a v riadku priemerne 80 znakov.
Príklad:
Zvuková nahrávka bola uložená v najnižšej kvalite (8 000 Hz) vo formáte wav a má veľkosť 1 875 KiB. Koľko minút trvá jej prehranie? Aká by bola veľkosť nahrávky vo formáte mp3?
Príklad:
Prenos textového dokumentu bez formátovacích značiek trval v počítačovej sieti s rýchlosťou 409,6 kbps 0,1sekundy. Urči a zdôvodni použitý kód, v ktorom je text uložený, ak má text 25 riadkov a v riadku priemerne 100 znakov.
Príklad:
Rastrový obrázok rozmerov 1024 x 768 pixelov je uložený v súbore s veľkosťou 2,25MiB. V akej farebnej palete bol uložený? Navrhnite farebnú paletu, pre ktorú sa veľkosť súboru zmenší a uveďte novú veľkosť súboru bez výpočtu.
Príklad:
Koľko minút trvá prehranie zvukovej nahrávky typu mp3 (kompresný pomer 1:11), ak bola uskutočnená s parametrami 16 000 Hz, 2 B, stereo a veľkosť súboru mp3 je 852,3 KiB?
Príklad:
Prenos textového dokumentu s 18,53 KiB obrázkom trval v počítačovej sieti s rýchlosťou 400 kbps 0,4 sekundy. Urči a zdôvodni použitý kód, v ktorom je text uložený, ak má text 16 riadkov a v riadku priemerne 64 znakov.

Príklad:
Aká bola približne rýchlosť pripojenia v okamihu odosielania prílohy mailovej správy - súboru mp3, ak zvuková nahrávka bola uskutočnená s parametrami 24 000Hz, 1B, stereo a jej prehranie trvá 50 sekúnd; úspešnosť kompresie do formátu mp3 bola 1 : 11 a prenos súboru trval 15 sekúnd?
Príklad:
Fotografia formátu bmp má 3600 x 2000 pixelov. Približne koľko GiB zaberá na disku, ak bola uložená vo formáte 24-bitovej mapy?
Príklad:
Určte približne veľkosť súboru, v ktorom je uložený 15 sekundový videoklip zložený z rastrových 256 farebných obrázkov 400 x 400 px, pričom za sekundu sa zobrazí 30 snímok. Minimálne ako dlho by trval prenos v počítačovej sieti daného videoklipu s priemernou rýchlosťou prenosu 1 Mbps?
Príklad:
Textový dokument bez formátovacích značiek s obrázkom má veľkosť približne 196,875 KiB. Určte a zdôvodnite použitý kód, v ktorom je text uložený, ak zaberá 24 strán (strana má 40 riadkov, v riadku je 80 znakov) a obrázok vo formáte jpg (kompresný pomer 1 : 10) má rozmery 800 x 600 pixelov a bol uložený vo formáte 256 farebnej bitovej mapy.
Príklad*:
Akú veľkosť má paleta farieb v hlavičke súboru obsahujúceho rastrový obrázok uložený vo formáte 16 farebnej bitovej mapy?
Príklad*:
Akú veľkosť má súbor, v ktorom je uložená 2,5 minútová zvuková nahrávka s bitovou frekvenciou 96 kbps.

Prílohy

V roku 1999 zverejnila Medzinárodná elektrotechnická komisia (IEC) druhý dodatok k "IEC 60027-2: Písmenové symboly pre použitie v elektrotechnickej technológii – časť 2: Telekomunikácie a elektronika". Tento štandard, schválený v roku 1998, zaviedol prefixy kibi-, mebi-, gibi-, tebi-, pebi-, exbi- na špecifikáciu binárnych násobkov množstva. Názvy pochádzajú z prvých dvoch písmen originálnych SI prefixov, nasledovaných skratkou bi, ktorá znamená binárny.

Názov	Symbol	Význam	Hodnota násobku	
kibi-	Ki	binárne kilo	2^{10}	= 1 024
mebi-	Mi	binárne mega	2^{20}	= 1 048 576
gibi-	Gi	binárne giga	2^{30}	= 1 073 741 824
tebi-	Ti	binárne tera	2^{40}	= 1 099 511 627 776
pebi-	Pi	binárne peta	2^{50}	= 1 125 899 906 842 624
exbi-	Ei	binárne exa	2^{60}	= 1 152 921 504 606 846 976

Prečo mi pri DVD médiu s kapacitou 4,7 GB vypaľovací program hlási nedostatok miesta, keď chcem vypáliť asi len 4,6 GB dát?
Problém spočíva v tom, že jednotky udávané v počítači a jednotky udávané ako kapacity médií nie sú čo do veľkosti zhodné. Kapacita na DVD sa uvádza v GB (gigabajtoch), avšak operačné systémy Windows zobrazujú kapacitu v GiB (**gibi**bajtoch) a ešte k tomu ju označujú GB! Rozdiel spočíva v tom, že 1 GiB má v skutočnosti 1 073 741 824 B (bajtov), zatiaľ čo 1 GB má rovných 1 000 000 000 B. Na médium s udávanou kapacitou 4,7 GB sa preto zmestí len asi 4,38 GiB.
(Preložené z internetu: http://ppk.chip.cz/cs/poradna/kapacita-dvd-je-mensi-nez-se-udava.html.)

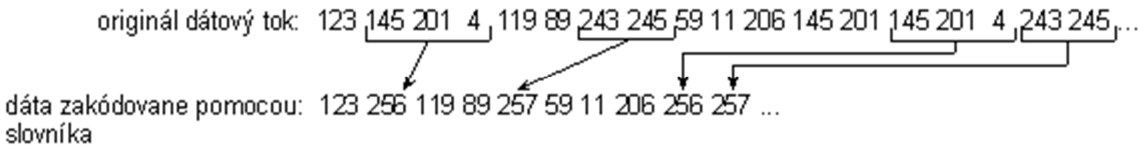
Kompresný algoritmus LZW

(prevzaté z internetu, viac napríklad na http://www.pakuj.host.sk/lzw/lzw.html)

Komprimačný algoritmus označovaný ako LZW je pomenovaný podľa svojich objaviteľov pánov A.Lempela , J.Ziva a Terryho A.Welcha, ktorý modifikoval pôvodný algoritmus LZ77. Jedná sa o **bezstratovú** komprimačnú metódu vhodnú na kompresiu ako grafických tak aj textových súborov. Rôzne modifikácie metódy LZW sa používajú v bežných komprimačných programov PKARC, PAK, PKZIP, v UNIX-e príkaz "compress" ako aj pri formátoch počítačovej grafiky GIF, TIFF, PostScript... Základným princípom kompresného algoritmu LZW je vyhľadávanie rovnakých, opakujúcich sa reťazcov v spracovávanom vstupnom súbore a priraďovanie im kódov.

Príklad slovníka	
kód	reťazce
0000	0
0001	1
:	:
0254	254
0255	255
0256	145 201 4
0257	243 245
:	:
4096	XXX XXX XXX

Nasledujúci príklad ukazuje veľmi zjednodušene (v samotnom algoritme LZW je to zložitejšie) kompresiu pomocou **slovníka** (dictionary). Predpokladáme, že slovník má kapacitu 4096 položiek. Z toho vyplýva, že jednotlivé kódy budú reprezentované pomocou 12 bitov. Kódy 0-255 v slovníku sú vyhradené a predstavujú jednotlivé byty (napr. ASCII znaky) vstupného súboru. Kompresia sa dosahuje použitím kódov 256 až 4095, ktoré reprezentujú reťazce. Napr. kód 256 reprezentuje sekvenciu 3 bytov 145 201 4. Vždy, keď kompresný algoritmus narazí na tento reťazec vo vstupnom súbore, umiestni kód 256 do výstupného kódovaného súboru. Pri dekompresii je kód 256 preložený pomocou slovníka do pôvodnej sekvencie bytov 145 201 4. Dlhé reťazce pripadajúce jednotlivým kódom a ich častý výskyt vo vstupnom súbore vedú k vysokej kompresii.



Tento prístup je značne zjednodušený a treba nájsť odpoveď na niektoré otázky:

- ako určiť, aké reťazce majú byť v slovníku
- ako odovzdať dekompresoru rovnaký slovník aký použil kompresor

Algoritmus LZW pracuje so slovníkom, ktorý sa adaptívne prispôsobuje kódovaným dátam. Táto adaptívna metóda vytvára dynamický substitučný slovník. Slovník nie je potrebné prenášať pre potreby dekompresie, pretože dekompresor si vie z prijímaných dát vytvoriť vlastný duplicitný slovník.