Úvod

- Pro toho, kdo chce proniknout do světa IT
- Ten svět má podobu 1 a 0, což **působí** neproniknutelně, jako něco tajemného a jen pro určité matematicky uvažující lidi
    - Je pravda:
        - je to svět logický a technický a jako takový není pro každého
        - Že není pro každého neznamená nutně, že by to ten člověk nezvládl, kdyby chtěl, ale třeba taky, že ho tohle prostě nebere a raději dělá jiné věci...
    - Není pravda:
        - Že je to svět zas tak tajemný a neproniknutelný, neuvěřitelně komplikovaný
        - Spousta, téměř vše, věcí v IT je naopak docela jednoduchá (i když striktně logická) a komplikované se to stává až kombinováním (proti tomu ale zase vytvořené mechanismy, které zjednodušují)
        - Příklady lidskosti: téměř všechny programovací jazyky jsou například podobné skutečnému, anglickému, jazyku:
        - Např.
            - ***Select apple***
              From appleBasket
              Where color = 'red' and taste = 'juicy'
              Order by size
              (valid SQL)
        - Nebo:

```
if (me_hungry == true) then
{
    do
    {
        bite();
        swallow();
    }
    until (me_hungry == false);
}
else
{
    exclaim("yayyy not hungry anymore!");
}
```

(pseudokód, Pascal/C#)

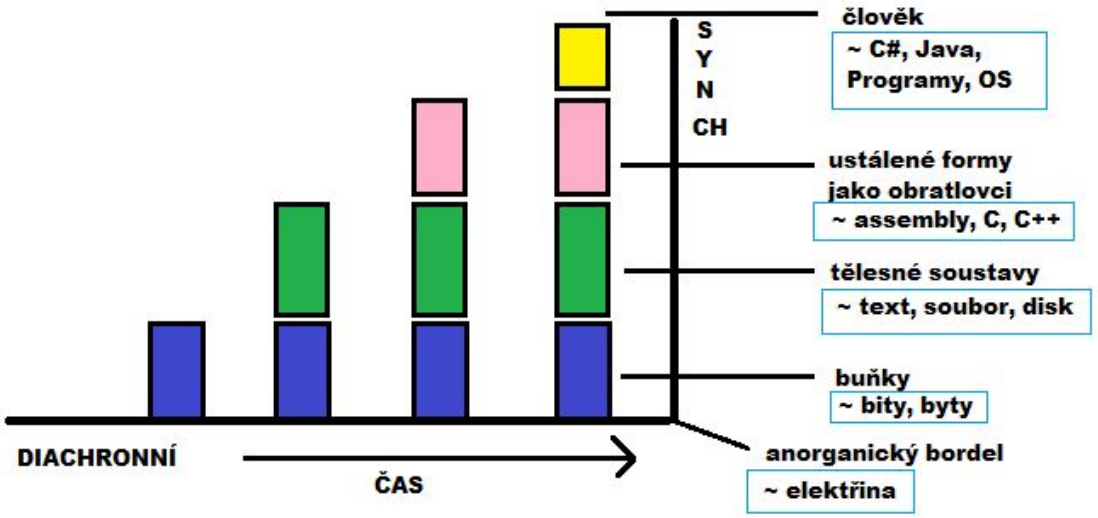
- Svět IT jsou teda 0 a 1 - v základu - jako jsou molekuly v přírodě, v základu => ITák (i programátor) ale nepracuje většinou s 0 a 1, ale s již interpretovanými celky nul a jedniček, které mají lidskou podobu (viz výše) -

mechanik taky pracuje de dacto s molekulami, ale nikdy ne přímo s nimi, ale už s hotovými celky jako šroubováky a šroubky...

- Cíl tohohle:
    - Ukázat, že různé děsivé názvy a znaky nejsou zas tak tajemné a vysvětlit jejich význam => proniknutí do toho světa, protože chápeš pojmy, které se používají
    - Ukázat, že lze pochopit, jak souvisí 0 a 1 s tím, s čím se pracuje běžně v IT světě (občas ani iťák úplně nezná)
    - Teorie (koncepty), ne praxe

Stupně evoluce výpočetní techniky (diachronně a synchronně)

- Pojmy diachronní, synchronní <> postupně, současně
- V nejobecnější rovině se IT rozvíjí podobným způsobem jako se vyvíjí živé organismy
    - Pamatuje si předchozí stádia vývoje
    - Co to má znamenat? Jak to souvisí s organismy? [glad u asked]



    - To má taky podobné důsledky jako v přírodě. Čím základnější úroveň jsem schopný pochopit a pracovat přímo s ní, tím pracnější je s tím cokoli udělat ale zároveň je to hodně stabilní v čase. Logika bitů a bytů bude nejspíše aplikovatelná i za stovky let a je tak základní, že by bylo v lidských silách ji znovu zavést. Nejvyšší úroveň nemusí být aplikovatelná za pár let třeba. Kdo zná pouze nejvyšší úroveň, tak když ta zmizí, tak už nezná nic. Kdo zná nižší úrovně bude znát základ i úrovní, které teprve budou v budoucnosti.
    - Nejvyšší úrovně mají nejvíc lidskou podobu. “00111010” vs “ovládání hlasem”

bits and bytes (2^8)

- Bit je proste moznost ano a ne, 0 a 1, pruchod elektriny ci nepruchod, dirka nebo plnost, informace o 2 moznostech (= datovy typ)
- Veskera vypocetni technika je postavena na bitech
- Byte je 8 techto bitů => 2^8 variaci…
    - Prevest bity na byty neni nic tezkeho, je to proste pocet bitů děleno 8 = počet bytů (viz inzerovani prenosovych rychlosti internetu)
- Bit se znaci “b” a byte “B”
- = jsou to pojmy, ktere toto znamenaji a nic vic
- -> představa o mnohosti kombinaci: Kolik ruznych variaci mi da jeden MB..

Číselné systémy

- Každé číslo lze zapsat v různých číselných systémech a desitkovy je pouze jedním z nich - my jsme na něj ale tak zvyklí, že mame pocit, že čísla jsou pouze to, co je takto zapsané

Jak funguje decimalní, “náš”, systém (k číselným soustavám obecně)

- Ukázat řadu 10 a jejich exponentu - 10^0 (jednotky) , 10^1 (desitky) , 10^2 (stovky) , 10^3 (tisice) atd
- Ukázat, jak ta řada souvisí s běžným číslem, treba 1273: psat pod tu radu odzadu: 3xjednotky, 7xdesitky, 2xstovky… + ukázat, ze tam je nekonečně dalších volnych míst
- Upozornit, ze tech cislic na danem stupni je vzdy právě 10 a to od 0 do 9 => kdyby jich bylo mene, nevyjadrim vsechna cisla, vice jich byt nemůže, protože to uz spadá do dalšího stupně

Další systémy (vizuální odlišení)

- Binární (2), hexadecimalni (16), sedesatkovy (stupně, minuty, sekundy), base64
- ukázat jejich reprezentace - jsme schopni rychle rozpoznat, i když ne třeba přečíst hned (jako japonske znaky a jina pisma)

Binární, dvojkový, systém

- Zapsat stejne cislo jako vyse v binárkách (0,1 => pravidlo jako vyse u decimalni soustavy)
- Ukazat, jak trojmistnym zapisem v decimalni soustavě vyjadrim az cislo 999, zatímco v binarni pouze cislo 7
- Film interstellar - binary scene (https://youtu.be/_fOwJQpBI5c)

Hexadecimalni systém

- Ukazat, jak se pouzivaji pismena, ukazat radu
- ukazat, jak se zapisuje 0 až 15 (F)
- Zapsat stejne cislo v hexu
- Trojmistny hex dava cislo 4095
- Odkaz na veci dale: vetsina cisel v PC se zapisuje hexadecimalne - vetsinou obsah souboru (hexeditor, viz dale) nebo treba barvy (dale)
- Film Marťan - hexadecimal scene

znakové sady, ASCII, Unicode

- Jedna se proste o dohodu

Vyvozovani 1

- lidský hlas prevadime do znaků, písma
- proto jsme si vytvořili abecedu jako kombinaci toho, čím můžeme ty hlasy vyjádřit
- kdyby abeceda mela pouze 2 znaky, pak bychom tim nevyjádřili vsechny mozne zvuky potrebne pro vyjadreni slov
- abecedu jiz mame a chceme ji prevest dale do digitalniho sveta 0 a 1

Vyvozovani 2

- 0 a 1, takže 1b, nam na to nestaci, tim bychom vyjádřili pouze dva znaky. Vzdy pracujeme s bity, ale muzeme je kombinovat. 2 bity uz nam davaji 4 možnosti, 3b 8 atd.
- Kombinace bitu lze brat jako cisla, cim větší je ta kombinace, tim vetsi cislo mohu vyjadrit…
    - Kódování funguje uplne jednoduse tak, ze je to domluva typu: pro cislo 1 je znak “A” a pro cislo 75 znak “K”
- Kolik moznosti potřebujeme? V historii jsou dve dulezite odpovedi: 7b (128) = ASCII; 1 až 4 byty (256 - 4 mld) = UTF
- ASCII
    - 128 moznosti = balíček 7 0/1 ve vsech kombinacich
    - Ukazat tabulku
    - Ukazat souvislost s Alt+cislo a mozna i souvislost s binarnim obsahem souboru (hex editor)
    - Kde jsou české znaky, kde čínské… = nestačí to

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

- ANSI (Windows-1250)
  - To samé, co ASCII, obsahující ASCII, ale přidán ještě jeden bit, takže 8 bitů, takže 1 byte
  - Tím jedním bitem navíc se získává dalších 128 možných znaků a tímto se měla řešit lokalizace, takže národní odlišnosti jazyků (čeština třeba plná háčků a čátek a takové znaky v ASCII nejsou)
  - Jenže každý národ potřebuje odlišné tyhle znaky, tak to funguje tak, že k těm datům ještě řekneme, v rámci jaké lokalizace je máme chápat. Čeština je například tzv “Windows-1250) => písmeno “š” je například 0x9A, takže 154, ale v jiném regionu pod tímto číslem nebude “š”, ale něco jiného
  - https://www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/WINDOWS/CP1250.TXT
- UTF [*8/16*]
  - *UTF 8 ma balicky po 8 bitech, kterych muze byt 1 až 4*
  - *UTF 16 ma balicky po 16b, takze může byt 1 (2B) nebo 2 (4B)*
  - Je zpětně kompatibilní s ASCII, což znamená, že ta dohoda porad plati => cislo 75 (0x4b) bude porad “K” i v nove rozsirene tabulce
  - Dále je to ale řešeno jednotně tak, že jsou obsaženy všechny regionální znaky a každý má svoje unikátní číslo. Takže pak už není třeba nikomu říkat, pod jakým regionem to má chápat. Proto by se už mělo používat jen UTF, protože to nevytváří problém.
  - VLC/MS Word,Poznámkový blok, různě...

Barvy RGB, CMYK a hex zapis barev

- Barvy většinou fungujou ve dvou odlisnych způsobech:
    - A) barvy jako vlnění, svetelne zareni, např záření monitoru - fungují tak, že se navzajem jaksi sčítají = kdyz vezmu vsechny barvy a pustím je přes sebe, dostanu bilou = dennimu světlu sr rika bile světlo a skrývá v sobe všechny barvy (sklenice s vodou) [filosofická poznámka: objekty kolem nás nemají samotné žádnou barvu, jen jejich povrchy jinak odrážejí a pohlcuji světlo]
    - B) barva jako barevna látka, inkoust - funguje naopak tak, ze mícháním barev dohromady postupně dostávám tmavší odstíny až dojdu k černé
- S prvnim způsobem je spjata zkratka RGB (red, green, blue), s druhým nejčastěji CMYK (cyan, magenta, yellow, black)
- Hodnoty definice libovolné barvy RGB jsou hodnoty od 0 do 255 (0xff)
- Lze je zapsat decimalne (malovani) a casteji v hexu
- protoze 255 je zaroven hodnota maximalniho dvoumistneho hexu (ff), tak se definice barvy RGB většinou zapisuje jako #FFAABB…
    - Jaka barva je: #00FF00?
    - Ukázat fakultni barvy (https://fsv.cuni.cz/pro-zamestnance/prezentace-fakulty/logo-design-manual) [příklad IMS decimal do hexu]
- Alpha kanál - míra průhlednosti
    - Relevantní jen pro některé formáty a opět se uvádí v hodnotách od 0 do 255, takže 00 až FF v hexu

Vektor x bitmapa, rozliseni, dpi, rozlišení vs informace

- Bitmapa nebo taky rastr [např .bmp <<] je jenom výčet barevných políček (pixelů) pomocí RGB hodnot + info o rozměru (jestli jdou políčka v řadě nebo třeba 800x600) => nejzákladnější obrázkový formát
- Vektor taky definuje obrázek, ale úplně jiným způsobem. Vůbec nic neví o nějakých barevných políčcích, pixelech. Definuje geometrické tvary a jak se k sobě mají (2 x větší než ten druhý, vzdálený o určitou délku, natočený tak a tak) => třeba trojúhelník: mám bod A a říkám, že bod B je od něj vzdálen jednu “délku” a boc C je kolmo od B vzdálen 2 “délky”. Tento útvar se pak má vyplnit nějakou barvou.
    - Tohle je důležité, protože zde nejsme nijak limitováni rozměry. “Délka” může být v milimetrech nebo kilometrech, je to fuk.
    - Teprve když chceme tisknout nebo zobrazit třeba na monitoru, tak musíme z těchto definic dopočítat, jakými konkrétními pixely mají být zobrazeny. To už ale lze z té definice odvodit. Naopak z bitmapy nemůžu odvodit vektor, protože nic neříká, že ta řada černých pixelů spolu nějak souvisí jakožto “úsečka”atp.
- ⇒ Uplatnění zejména při jasně geometricky definovaných obrazcích jako jsou loga. Proto by logo od grafika mělo vždy přijít zejména v nějakém vektorovém formátu. Mohu jej pak vytisknout přes celou budovu bez jakékoli ztráty kvality (kostičkování)
- Časté bitmapové formáty/přípony: bmp, jpg. gif, png, tiff…
- Časté vektorové formáty/přípony: ai, svg

- Rozlišení = odlišnost informací, jednotlivých pixelů => kolik tam takových bodů je a kolik je jich na šířku a na výšku
    - Vysoké rozlišení neznamená automaticky kvalitu obrazu! Proč? Protože můžu mít sice 1920×1080 pixelů (= full hd = 1080p) ale to neznamená, že tam nemohou být shluky stejných barev vytvářející kostičkovaný obraz. Naopak si můžu být jistý, že když mám obrázek s rozlišením třeba 640*480, tak že tam nebude těch informací více, obraz nemůže být více kvalitní, nelze tam rozlišit více pixelů
- Dpi = dots per inch. Říká kolik bodů má přijít na velikost jednoho palce. Nejde ve skutečnosti o žádnou vlastnost toho obrázku, jde o pokyn monitoru nebo tiskárně, co s těmi pixely má udělat. Když mám 800 pixelů na šířku a chci tisknout na tiskárně s kvalitou 300dpi pak 800/600 = 2,66 palce, takže asi 6,5cm v plné kvalitě. Čím menší bude dpi, tím méně bude bodů na jednom palci, obrázek se bude tisknout větší, ale méně kvalitní (extrém: třeba 2 pixely na palec…)

- V přeneseném slova smyslu lze o “rozlišení” mluvit i třeba u zvuku. Zvuková vlna v sobě zachycuje určitém množství odlišností. A analogicky: zvuk o vysokém rozlišení nemusí být nutně kvalitní, ale zvuk o nízkém rozlišení (bitrate), určitě nemůže naopak kvalitní být.

Grafické editory: pojmy související spíše s editací, nikoli výsledkem

- Pojmy jako: vrstvy, masky, filtry/shadery
- Vrstvy
    - formát .tiff prý umí zachovat vrstvy, ale jinak je výsledek většinou prostě zploštělý = to znamená, že třeba není možné později nějaký podklad posunout = už je to jednolitá plocha barev…

Datové typy

- ⇒ co mohu očekávat z hlediska typu + jak to bude velké
- ⇒ ve smyslu dat na discích to jsou všechno pořád jen čísla, bity/byty
    - Zde jde už o jejich interpretaci nějakými programy, kde v tomto panuje ale veliká shoda napříč těmi programy => dostanu nějaké 0 a 1, ale bude jich určitý počet a já vím, za co je mám považovat (je to číslo? Budu s tím počítat? Je to text? Budu s tím dělat textové operace? Je to logická hodnota? Budu s tím vyhodnocovat logické podmínky?)

- Čísla (celá, desetinná): **int** (integer), **float**
- **Booly** (ano, ne nebo taky 1 a 0)
- **Byty** (zapisují se většinou hexadecimálně, viz obrázek proč) = opět čísla
- Znaky (**char**)
- Řetězce (**string**) = řetězce znaků...