

Maturitní otázky z informatiky

Adam Krška

Obsah

1 Historie počítačů	3
1.1 Počítač	3
1.2 Historie	3
1.2.1 Abacus	3
1.2.2 Mechanický kalkulátor	3
1.2.3 Děrné štítky	3
1.2.4 Analytický stroj	3
1.3 Generace	3
1.3.1 Nultá generace	3
1.3.2 První generace (1945 – 1950)	4
1.3.3 Druhá generace (1951 – 1964)	4
1.3.4 Třetí generace (1965 – 1980)	4
1.3.5 Čtvrtá generace (od 1980)	4
1.4 Data v počítači	5
1.4.1 Ukládání dat	5
2 Informace v počítači	5
2.1 Data v počítači	5
2.2 Číselné soustavy	6
2.2.1 Dvojková (binární) soustava	6
2.2.2 Šestnáctková (hexadecimální) soustava	7
2.2.3 Osmičková (oktalová) soustava	7
2.3 Jednotky informace	8
2.4 Záznamová média	8
3 Booleova algebra	9
3.1 Zákony a pravidla	9
3.2 Základní logické operace	10
3.2.1 Konjunkce / AND	10
3.2.2 Disjunkce / OR	10
3.2.3 Implikace	10
3.2.4 Ekvivalence	11
3.2.5 Negace / NOT	11
3.3 Základní logické členy	11
3.3.1 NOT	11
3.3.2 AND	11
3.3.3 OR	12
3.3.4 NAND	12
3.3.5 NOR	12
3.3.6 XOR	12
3.3.7 XNOR	12
3.4 Logická funkce	12
3.4.1 Minimalizace logické funkce	12
3.5 Kombinační obvod	13
4 Sekvenční logický systém	13
4.1 Sekvenční logický systém	13
4.1.1 Stavba	15
4.1.2 Návrh sekvenčního systému	15
4.2 SR Latch	15
4.3 SR latch with enable (gated SR latch)	15

4.4	D latch	16
4.4.1	Rising edge D flip-flop	16
4.5	JK latch	16
4.6	T latch	16
4.7	Flip Flop	18
4.8	Konečný automat	18
5	Základní části počítače, operační systémy	18
5.1	Základní části počítače	18
5.2	Záznamová média	20
5.3	Programové vybavení počítače	20
5.3.1	Možnosti programování / programovací jazyky	20
5.3.2	Typy softwaru	20
5.4	Operační systémy	20
5.4.1	Příklady	20
5.4.2	Komponenty	21
6	Periferní zařízení počítačů	22
6.1	Vstupní zařízení	22
6.1.1	Klávesnice	22
6.1.2	Myš	22
6.1.3	Grafický tablet	23
6.1.4	Skener	23
6.1.5	Mikrofon	23
6.1.6	Fotoaparát / Kamery	24
6.2	Výstupní zařízení	24
6.2.1	Monitor	24
6.2.2	Tiskárna	25
6.2.3	Zvukový výstup	26
7	Počítačové sítě	26
7.1	Rozdelení	26
7.1.1	Podle typu připojení	26
7.1.2	Podle velikosti	26
7.2	Síťové modely	27
7.2.1	RM OSI	27
7.2.2	TCP/IP	27
7.3	Síťové prvky	30
7.3.1	Router	30
7.3.2	Modem	30
7.3.3	Switch	31
7.3.4	Wireless Access Point (WAP, AP)	31
7.3.5	NAT – Network address translation	31
7.4	Protokoly	32
7.5	Tok dat v síti	32
8	Internet	33
8.1	Připojení k internetu	33
8.2	Možnosti internetu	33
8.2.1	WWW	33
8.2.2	Email	33
8.2.3	VoIP	33
8.2.4	Přenos souborů	33
8.3	IP adresa	33

8.4 Mac adresa	34
8.5 Podsítě	34
8.6 ARP	34
8.7 Směrování (routing)	35
8.7.1 Směrovací tabulky (routing table)	35
8.7.2 Směrovací algoritmy	35
9 Textové editory	37
9.1 Druhy	37
9.2 Formáty dokumentů	37
9.3 Vzhled dokumentu	37
9.3.1 Obsah dokumentu	37
9.3.2 Formát textu	38
9.3.3 Font (písmo)	38
9.3.4 Odstavce	38
9.3.5 Seznamy	38
9.3.6 Nadpisy	38
9.3.7 Sloupce	39
9.4 Hromadná korespondence	39
9.5 Záhlaví, zápatí (header a footer)	39
9.5.1 Číslování stránek	39
9.6 Oddíly	39
9.7 Automatické číslování	39
9.8 Generování seznamů	39
9.8.1 Obsah	39
9.8.2 Rejstřík	39
9.8.3 Seznam obrázků a tabulek	40
13 Databáze MS Access, SQL	40
13.1 Databáze	40
13.1.1 MS Access	40
13.2 Struktura	40
13.2.1 Formáty dat/datatypy	41
13.3 SQL	41
13.3.1 Příkazy	41
13.3.2 SELECT	41
13.4 Práce s MS Access	42
13.4.1 Návrh databáze	42
13.4.2 Formuláře	42
13.4.3 Třídění a vyhledávání dat	42
14 Obrazová informace	42
14.1 Digitální obraz	42
14.1.1 Pixely	42
14.1.2 Barevné modely	42
14.1.3 Barevná hloubka (Color depth)	44
14.1.4 Rozlišení	44
14.1.5 DPI	44
14.2 Rastrová/bitmapová grafika	45
14.2.1 Komprese	45
14.3 Vektorová grafika	46

22 Vstup/výstup, podmínky	61
22.1 Vstup/výstup do/z programu	61
22.1.1 Uživatelský vstup	63
22.1.2 Výstup programu	63
22.1.3 Formát printf/scanf	63
22.2 Podmínky	64
22.2.1 Booleovské výrazy	64
22.2.2 Kombinace podmínek	65
23 Cykly	65
23.1 For loop	65
23.2 While cyklus	66
23.3 Do while cyklus	66
23.4 Vnořený cyklus	66
24 Práce se soubory, struktury, pole	66
24.1 Práce se soubory	66
24.1.1 Otevřání souboru	68
24.1.2 Zavření souboru	68
24.1.3 Čtení ze souboru	68
24.1.4 Zápis do souboru	68
24.2 Struktury	68
24.3 Pole	70
24.3.1 Hodnota položky	70
24.3.2 Pole struktur	71
25 Práce s polem, řadící algoritmy	71
25.1 Práce s polem	71
25.1.1 Inicializace pole	71
25.1.2 Získání hodnoty z pole	71
25.1.3 Zápis do pole	71
25.1.4 Procházení pole	71
25.2 Vícerozměrná pole	72
25.3 Řadící (sorting) algoritmy	72
25.3.1 Klasifikace algoritmů	73

1 Historie počítačů

1.1 Počítač

- zařízení a stroje zpracovávající data pomocí vytvořeného programu
- hardware (fyzické součásti) a software (program)
- získávání dat vstupním zařízením, zpracování a uchování dat, reprezentace výsledku výstupním
- notebooky, desktopy, mikrocontroller (ledničky, chytrá zařízení...), auta, ...

1.2 Historie

- *"počítač"* – člověk, co počítá, později počítací zařízení
- předchůdci počítače
 - Abacus
 - mechanické kalkulátory
 - děrnostítková zařízení
 - analytický stroj

1.2.1 Abacus

- korálkové počítadlo
- 3 000 př. n. l., Malá Asie

1.2.2 Mechanický kalkulátor

- Wilhelm Schickard, 17. stol
- ozubená kolečka z hodin
- sčítání a odčítání šesticiferných čísel

1.2.3 Děrné štítky

- 1801 – Joseph Marie Jacquard – využití v tkalcovském stavu
- milník v počítačích – možné vytvářet a uchovávat a znova použít instrukce
- 1890 – Herman Hollerith, USA – zjednodušení administrativy (sčítání lidu), základ IBM

1.2.4 Analytický stroj

- 1833 – Charles Babbage – první návrh
- 50 místní čísla, fixní desetinná čárka, parní pohon
- nikdy nesestaven

1.3 Generace

1.3.1 Nultá generace

- elektromechanické stroje, využití relé
- posun vývoje za druhé světové války

Z1

- 1938 – první počítač
- dvojková soustava, děrné pásky

Colossus

- 1943, Thomas H. Flowers
- použit na rozluštění německých kódů za druhé světové války (Enigma)
- vakuové elektronky

Mark I. – ASCC

- první IBM počítač
- děrné pásky, 24 stop ve skupinách po třech (2 adresy + instrukce)
- desítková soustava, pevná desetinná čárka

1.3.2 První generace (1945 – 1950)

- použití elektronek
- počítače drahé, neefektivní, vysoký příkon

ENIAC

- 1944, Pensylvánie
- smyčky a podmíněné skoky, Turing complete
- 5000 součtů za minutu
- energeticky náročný, poruchový, drahý provoz

MANIAC

- inspirace ENIACem
- matematické výpočty fyzikálních dějů, vývoj jaderné bomby

1.3.3 Druhá generace (1951 – 1964)

- využití tranzistorů → vylepšení počítačů
 - zmenšení, větší výkon, efektivita a spolehlivost
- pronikání počítačů do běžného života
 - obchody, administrativa, skladování dat...
- děrné pásky, štítky, magnetické pásky
- první operační systémy
- jazyky symbolických adres
- ”vyšší” programovací jazyky – COBOL, FORTRAN, ALGOL...

1.3.4 Třetí generace (1965 – 1980)

- integrované obvody
- výkon počítače zhruba druhá mocnina jeho ceny
- vykonávání více procesů najednou (jeden program čeká na I/O, druhý používá procesor)

IBM System 360

- pevná i proměnná délka dat
- průlom v praktickém a komerčním využití, tisícové série

1.3.5 Čtvrtá generace (od 1980)

- mikroprocesory a osobní počítače
- snížení počtu obvodů na základní desce, zvýšená efektivita, menší rozměry, vyšší rychlosť
- ústup střediskových počítačů (mainframe)
- IBM, éra DOS, GUI
- exponenciální růst ceny – není nejlepší koupit nejvýkonnější počítač
- vznik clusterů
- vznik internetu, distribuovaných systémů

- multiprocesory

1.4 Data v počítači

- reprezentace pomocí 1 a 0 (zapnuto/vypnuto) – byty [b] (kb, Mb, Gb,...)
- seskupovány po 8 – byty [B] (kB, MB, GB,...)
- data uchovány podle různých řádů a systémů – formáty souborů, kódování jazyků (ASCII vs UTF8) atd.
- metadata
 - data o dalších datech / o souboru
 - specifikace MIME typu, čas, permissions, creator, ...
- konverze dat – převod mezi formáty
 - komprese dat – konverze s cílem zmenšit velikost souborů, lossy/lossless compression

1.4.1 Ukládání dat

- dnes používané – RAM, disky (HDD, SSD, USB disky, paměťové karty), disk array (RAID), cloud

Magnetické

- uchování dat pomocí změny vzoru a orientace magnetizace
- HDD, floppy disky, magnetické pásky, magnetické proužky

Optické

- využití laseru pro zápis a čtení dat
- zaostřený laser na optický disk
- ekologický způsob uchování
- CD, DVD, Blu-ray

Mechanické

- uchování dat mechanickou změnou systému
- přehození páky, vytvoření díry...
- děrné pásky

Elektrické

- uchování dat za pomoci logických bran
- flash memory (USB, SSD...)

2 Informace v počítači

2.1 Data v počítači

- reprezentace pomocí 1 a 0 (zapnuto/vypnuto) – byty [b] (kb, Mb, Gb,...)
- seskupovány po 8 – byty [B] (kB, MB, GB,...)
- data uchovány podle různých řádů a systémů – formáty souborů, kódování jazyků (ASCII vs UTF8) atd.
- metadata
 - data o dalších datech / o souboru
 - specifikace MIME typu, čas, permissions, creator, ...
- konverze dat – převod mezi formáty
 - komprese dat – konverze s cílem zmenšit velikost souborů, lossy/lossless compression
- ukládání – RAM, disky (HDD, SSD, USB disky, paměťové karty), disk array (RAID), cloud

2.2 Číselné soustavy

- soustavy specifikující reprezentaci čísel
- báze r – číslo specifikující počet znaků soustavy (desítková soustava – $r = 10$)
- řád n – hodnota řádů je r^n (stovky – 10^2)
- hodnota čísla

$$N = \sum_{i=-l}^{k-1} n_i \cdot r^i = n_{k-1}r^{k-1} + n_{k-2}r^{k-2} + \dots + n_0r^0 + n_{-1}r^{-1} + \dots + n_{-l+1}r^{-l+1} + n_{-l}r^{-l}$$

2.2.1 Dvojková (binární) soustava

- báze 2 – čísla 0, 1
 - každá cifra odpovídá mocnině dvou

Převody

Dvojková → Desítková

- každou cifru vynásobíme danou mocninou základu a hodnoty sečteme

$$(11010)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = (26)_{10}$$

Desítková → Dvojková

- číslo vydělíme 2, zapíšeme zbytek po dělení a dělíme dál, dokud nedělíme nulu

pořadí dělení	7.	6.	5.	4.	3.	2.	1.	číslo
výsledky dělení	1	3	6	13	26	53	107	214
rozklad na součin	$2 \cdot 0 + 1$	$2 \cdot 1 + 1$	$2 \cdot 3$	$2 \cdot 6 + 1$	$2 \cdot 13$	$2 \cdot 26 + 1$	$2 \cdot 53 + 1$	$2 \cdot 107$
zbytky dělení	1	1	0	1	0	1	1	0

Záporná čísla

Přímý kód

- první bit je znaménkový
 - $(00000001)_2 = (1)_{10}$
 - $(10000001)_2 = (-1)_{10}$
- potřebné odlišné algoritmy pro sčítání a odčítání
- dvě reprezentace nuly

Inverzní kód

- záporné číslo zaznamenáno jako binární negace
- dvě nuly
- různé algoritmy pro sčítání a odčítání

Doplňkový kód

- záporné číslo zaznamenáno jako binární negace (výměna 0 a 1) + 1
- podle úvodního bitu lze rozepnout kladná/záporná čísla
- využití přetečení
 - $00000000 - 00000001 = 11111111$
- jediná reprezentace nuly
- stejné algoritmy pro sčítání a odčítání
- zachována komutativnost
- používané v počítačích

- příklad: pokud 00001101 je binární vyjádření čísla 13, pak 13 se vypočte jako $\text{NOT}(00001101) + 1 = 11110010 + 1 = 11110011$

$$\begin{aligned} 20_{10} - 13_{10} &= 20_{10} + (-13)_{10} = 00010100_2 + 11110011_2 = \\ &= 1\ 00000111_2 = 7_{10} (\text{po odříznutí přeteklého devátého bitu}) \end{aligned}$$

Desetinná čísla

- tvořeny pomocí zlomků
- mocniny za desetinnou čárkou mají zápornou hodnotu → součet zlomků

$$\begin{aligned} (0,1011)_2 &= 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 2 \cdot 2^{-3} + 1 \cdot 2^{-4} = \\ &= 0 \cdot 1 + 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16} = \\ &= 0,5 + 0,125 + 0,0625 = 0,6875 \end{aligned}$$

2.2.2 Šestnáctková (hexadecimální) soustava

- báze 16 – čísla 1, 2, 3, 4, 5, 6, 7, 8, 9, A(10), B(11), C(12), D(13), E(14), F(15)
- jednoduchý převod mezi binární a hexadecimální soustavou
- častý zápis v programování – 0xE1

Převody

Šestnáctková → Desítková

- stejně jako u dvojkové soustavy, pouze násobíme mocninou 16

Desítková → Šestnáctková

- stejně jako u dvojkové soustavy, pouze dělíme 16

Šestnáctková → Dvojková

- každé 4 bity (nibble = 4 bity = polovina bytu) odpovídají jednomu řádu hex-soustavy ($2^4 = 16$)
 - 1 = 0001
 - 2 = 0010
 - 3 = 0011
 - ...
 - E = 14 = 1110
 - F = 15 = 1111
- z každé jedné cifry určíme dané 4 bity
- 2 cifry hex odpovídají bytu
- příklad: 0xE1 = 1110 0001

Dvojková → Šestnáctková

- opačný způsob – z každých 4 bitů určíme odpovídající cifru hex
- příklad: 1011 0101 = 0xB5

2.2.3 Osmičková (oktalová) soustava

- číselná soustava o základu 8 – čísla 1, 2, 3, 4, 5, 6, 7
- mocnina 2 – jednoduchý převod do binární soustavy
- použití v informatice – Linux permissions

Převody

Osmičková → Desítková

- stejné jako u dvojkové soustavy, pouze násobíme mocninou 8

Desítková → Osmičková

- stejné jako u dvojkové soustavy, pouze dělíme 8

Osmičková → Dvojková

- každé 3 byty ($2^3 = 8$) odpovídají jedné číslici osmičkové soustavy
 - 1 = 001
 - 2 = 010
 - 3 = 011
 - ...
 - 6 = 110
 - 7 = 111
- z každé cifry určíme 3 cifry binárního čísla a zapíšeme za sebe

Dvojková → Osmičková

- opačný způsob – převod každých 3 bitů na odpovídající cifru osmičkové soustavy
- příklad: $(101001)_2 = (51)_8$

Osmičková ↔ Šestnáctková

- 2 kroky – převod do binární (případně desítkové) soustavy a následné zakódování do dané soustavy
- převod přes desítkovou soustavu univerzální pro všechny soustavy

2.3 Jednotky informace

- základní jednotka – bit – 1 b
 - uložení jednoho on-off/true-false/1-0 stavu
- shromažďovány do skupiny po 8 – byte – 1 B = 8 b
- násobky
 - desítková soustava
 - * kilobyty – 1 kB = 10^3 B = 1 000 B
 - * megabyty – 1 MB = 10^6 B
 - * gigabyty – 1 GB = 10^9 B
 - * terabyty – 1 TB = 10^{12} B
 - dvojková soustava
 - * kibibyte – 1 KiB = 2^{10} B = 1 024 B
 - * mebibyte – 1 MiB = 2^{20} KiB = 2^{20} B
 - * gibibyte – 1 GiB = 2^{30} MiB = 2^{30} B
 - * tebibyte – 1 GiB = 2^{40} GiB = 2^{40} B

2.4 Záznamová média

- historicky
 - děrné pásky – záznam dat pomocí dér
 - diskety (floppy disk) – magnetické úložiště
- CD/DVD – optické disky / malá zrcadla, čtení/zápis pomocí disku
- HDD – Hard drive disk (pevný disk)
 - elektromagnetické úložiště
 - uvnitř točící se disky se čtecí hlavou
 - SATA protokol

- na dnešní poměry pomalé
- nízká cena za velkou kapacitu
- využití – archivace, uchování velkých dat (např. NAS)
- SSD – Solid state drive
 - uchování dat pomocí integrovaného obvodu (flash memory)
 - chybí mechanické části
 - SATA protokol
 - rychlý přístup k datům
 - vyšší cena na jednotku objemu
 - využití – instalace OS, programů, data potřebná rychlé čtení/zápis
- M.2 SSD
 - podobné SATA SSD
 - využívá M.2 slot
 - možné využití NVMe (na PCIe) protokolu – vyšší přenosová rychlosť
- USB flash drive
 - flash memory přes USB protokol
 - přenositelná zařízení
 - fyzický přenos dat
- Magnetické pásky
 - uchování dat pomocí magnetismu
 - nízká cena
 - dnešní využití pro archivaci, zálohy

3 Booleova algebra

- algebraická struktura se dvěma binárními a jednou unární operací
- zobecnění vlastností množinových a logických operací
- dvouprvková Booleova algebra
 - reprezentace pravdivostních hodnot a logických hodnot
- hodnoty proměnných – pravda/lež, true/false, 1/0

3.1 Zákony a pravidla

- asociativita
 - pro \wedge : $(x \wedge y) \wedge z = x \wedge (y \wedge z)$
 - pro \vee : $(x \vee y) \vee z = x \vee (y \vee z)$
- komutativnost
 - pro \wedge : $x \wedge y = y \wedge x$
 - pro \vee : $x \vee y = y \vee x$
- identita – vrácení původní hodnoty
 - pro \wedge : $x \wedge 1 = x$
 - pro \vee : $x \vee 0 = x$
- anihilace
 - pro \wedge : $x \wedge 0 = 0$
 - pro \vee : $x \vee 1 = 1$
- distributivita
 - \wedge přes \vee (dva zápis):
 - * $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
 - * $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
 - \vee přes \wedge (dva zápis):
 - * $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
 - * $x + (y \cdot z) = (x + y) \cdot (x + z)$
- absorpcie
 - $x \wedge (x \vee y) = x$

- $x \vee (x \wedge y) = x$
- idempotence
 - $x \wedge x = x$
 - $x \vee x = x$
- De Morganovo pravidlo (pravidla o vytvoření negace)
 - $\neg(x \vee y \vee z) = \neg x \wedge \neg y \wedge \neg z$
 - $\neg(x \wedge y \wedge z) = \neg x \vee \neg y \vee \neg z$

3

3.2 Základní logické operace

- výsledkem opět výrok
- hodnota výsledku závislá na hodnotách vstupu a druhu operace

3.2.1 Konjunkce / AND

- značka \wedge , nebo taky \cdot
- operace pravdivá, pokud oba výroky pravdivé

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Tab. 3.1: Implikace

3.2.2 Disjunkce / OR

- značka \vee , nebo taky $+$
- operace pravdivá, pokud alespoň jeden výrok pravdivý

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 3.2: Disjunkce

3.2.3 Implikace

- značka \Rightarrow
- x implikuje y , pokud z x nutně vyplývá y nebo je y již v x zahrnuto

x	y	$x \Rightarrow y$
0	0	1
0	1	1
1	0	0
1	1	1

Tab. 3.3: Implikace

3.2.4 Ekvivalence

- značka \Leftrightarrow
- x a y platí nutně zároveň

x	y	$x \Leftrightarrow y$
0	0	1
0	1	0
1	0	0
1	1	1

Tab. 3.4: Ekvivalence

3.2.5 Negace / NOT

- značka \neg
- hodnota operace opačná hodnotě výroku

x	$\neg x$
0	1
1	0

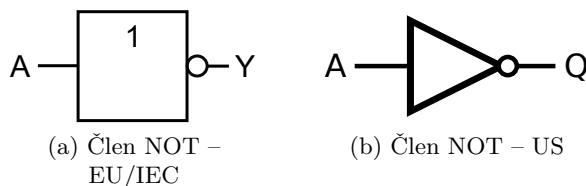
Tab. 3.5: Negace

3.3 Základní logické členy

- „hradla“
- prvek logických/elektrických obvodů
- vyčíslení logické funkce
- pomocí AND, OR a NOT možno sestavit jakýkoliv obvod

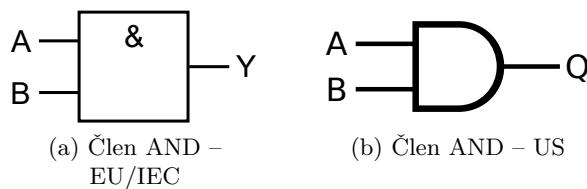
3.3.1 NOT

- realizace negace
- stejné hodnoty jako negace



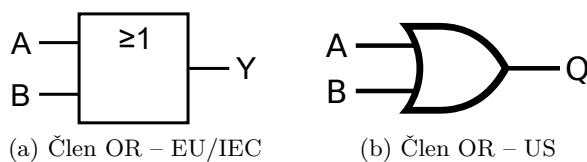
3.3.2 AND

- realizace konjunkce
- stejné hodnoty jako konjunkce



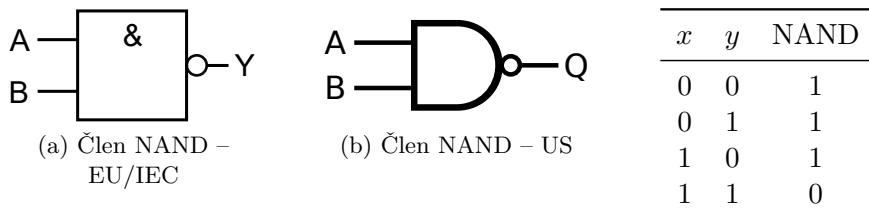
3.3.3 OR

- realizace disjunkce
- stejné hodnoty jako disjunkce



3.3.4 NAND

- převrácené (znegované) AND



3.3.5 NOR

- převrácené (znegované) OR

3.3.6 XOR

- „exklusivní OR“
- platné, pokud pouze jeden ze členů platný

3.3.7 XNOR

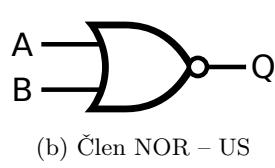
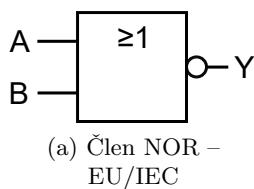
- negace XOR

3.4 Logická funkce

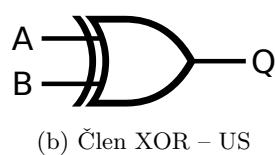
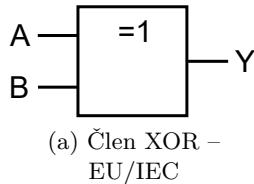
- funkce přijímající pravdivostní hodnoty jako vstup
- výstup také pravdivostní hodnota
- deterministická

3.4.1 Minimalizace logické funkce

- prováděna pomocí pravidel boolenové algebry



x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

- užitečné pro zmenšení obvodu

$$(A \vee B) \wedge (A \vee C) = A \vee (B \wedge C)$$

$$(A \wedge B) \vee (A \wedge C) = A \wedge (B \vee C)$$

$$A \vee (A \wedge B) = A$$

$$A \wedge (A \vee B) = A$$

$$A \vee (\neg A \wedge B) = A \vee B$$

$$A \wedge (\neg A \vee B) = A \wedge B$$

$$(A \vee B) \wedge (\neg A \vee B) = B$$

$$(A \wedge B) \vee (\neg A \wedge B) = B$$

$$(A \wedge B) \vee (\neg A \wedge C) \vee (B \wedge C) = (A \wedge B) \vee (\neg A \wedge C)$$

$$(A \vee B) \wedge (\neg A \vee C) \wedge (B \vee C) = (A \vee B) \wedge (\neg A \vee C)$$

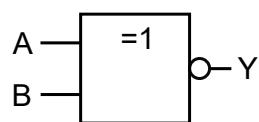
3.5 Kombinační obvod

- realizace logické funkce
- nemá paměť
- v počítači provádí boolenovou algebru na vstupních signálech a uložených datech
 - např. ALU (arithmetic logic unit)
- stavba na základě matematické funkce z logických členů

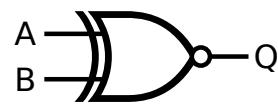
4 Sekvenční logický systém

4.1 Sekvenční logický systém

- též taky jen sekvenční obvod
- na rozdíl od kombinačních obvodů není závislý pouze na vstupních signálech ale také na pořadí zadání
- uchovávají předešlý stav – mají paměť
- při určení hodnoty výstupu nutno sledovat jak vstupy, tak stav paměti
- typy podle času změny výstupu a paměti

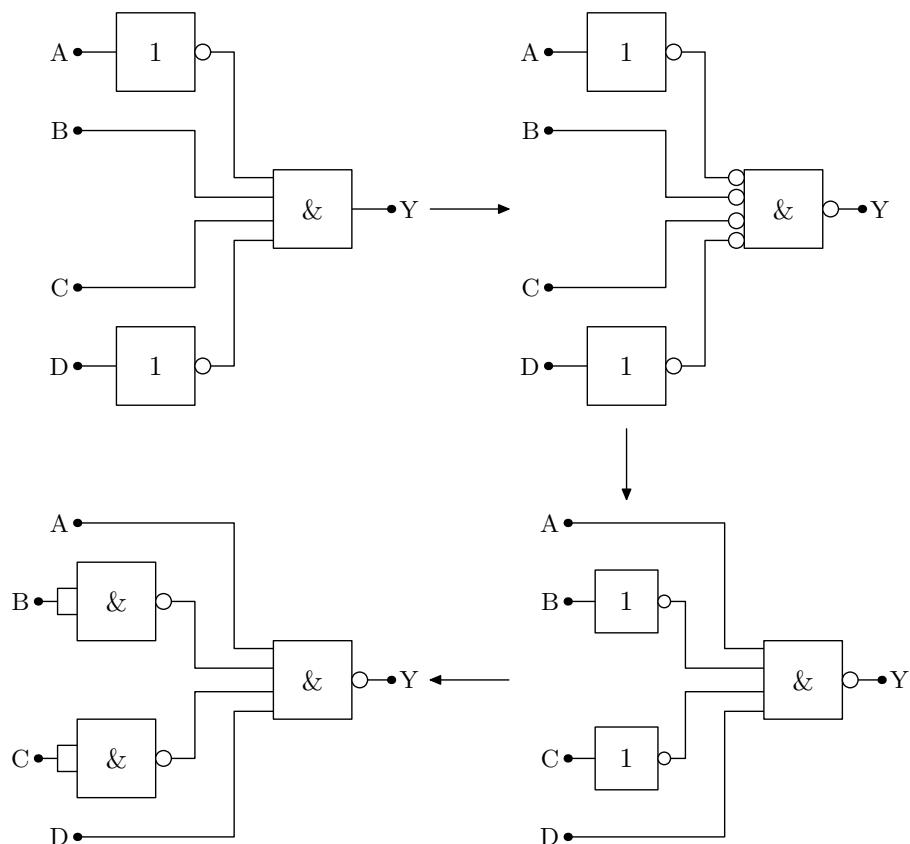


(a) Člen XNOR – EU/IEC



(b) Člen XNOR – US

x	y	XNOR
0	0	1
0	1	0
1	0	0
1	1	1



Obr. 3.1: Příklad kombinačního obvodu

- asynchronní – změna vstupu ihned ovlivní výstup; *latches*
- synchronní – řízeny vnějším synchronizačním signálem (*clock signal*), změna probíhá pouze při tiku hodin, *flip-flops*

4.1.1 Stavba

- kombinační a paměťová část
- paměťová
 - uchování vnitřního stavu
 - základem SR latch
 - většinou složena z NAND nebo NOR gate
- kombinační část
 - pomocná kombinační logika doplňující paměť
 - využití AND či NOT gate...

4.1.2 Návrh sekvenčního systému

Mealyho typ

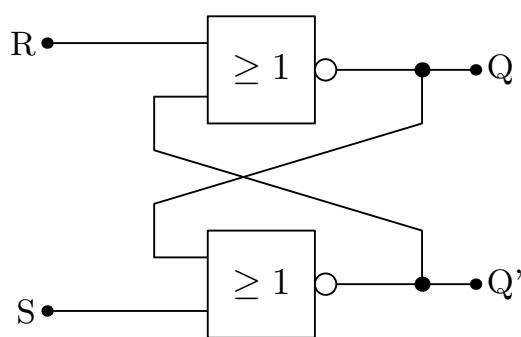
- výstup závislý na vstupu a vnitřním stavu
- rychlejší než Moor
- menší počet stavů (n vs n^2)

Moorův typ

- vstupní hodnoty se projeví až v následujícím stavu
- výstup závislý pouze na vnitřním stavu

4.2 SR Latch

- základní paměťový obvod
- sestaven z NAND nebo NOR gates
- 2 vstupy – S (set) a R (reset)
- 2 výstupy – Q a \bar{Q}
- set nastaví hodnotu Q na 1, reset na 0; opakované spouštění stejného vstupu nic nedělá
- oba vstupy zapnuté – nedefinovaný stav, latch se rychle přehazuje, konečný stav neurčitelný



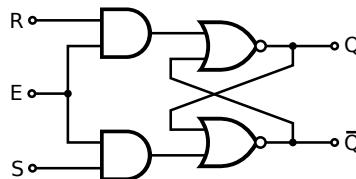
Obr. 4.1: SR latch za použití NOR gate.

4.3 SR latch with enable (gated SR latch)

- rozšíření SR latch o kombinační obvod
- přidání *enable* vstupu – povoluje vstup
 - s $E = 0$ není možné ovlivnit latch
 - s $E = 1$ vstupy projdou a změní stav

S	R	Q	\bar{Q}
0	0	Last	Last
0	1	0	1
1	0	1	0
1	1	—	—

Tab. 4.1: Pravdivostní hodnoty SR latch (Last – Last state – poslední stav)



Obr. 4.2: SR latch s enable vstupem

4.4 D latch

- základem gated SR latch
- dva vstupy – E (enable) a D
- vstupy S připojeno na D , R na \bar{D}
- bez enable vstupu by Q v podstatě jen kopírovalo stav D
- s enable vstupem možnost zapamatovat si poslední stav
- D flip flop – obvod používán jako paměťový modul

D	E	Q	\bar{Q}
0	–	Last	Last
1	0	0	1
1	1	1	0

Tab. 4.2: Pravdivostní hodnoty D latch

4.4.1 Rising edge D flip-flop

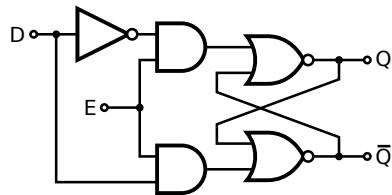
- uložení hodin pouze na začátku clock cyklu
- možnosti sestavení
 - master-slave – 2 D latches za sebou, u jednoho invertovaný clock signál
 - rising-edge detektor na clock signálu (viz 4.6)

4.5 JK latch

- modifikovaná gated SR latch
- zamezení nedefinovaného stavu

4.6 T latch

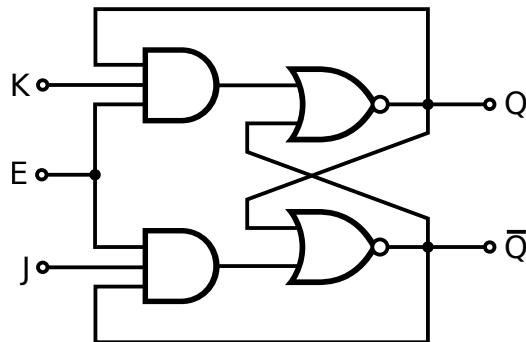
- modifikace JK latch
- J i K připojeny na $T \rightarrow$ stav J a K vždy stejný
 - $T = 0$ – latch si drží poslední stav
 - $T = 1$ – latch změní stav, toggles
- funguje jako přepínač



Obr. 4.3: D latch s enable signálem

J	K	Q	\bar{Q}
0	0	Last	Last
0	1	0	1
1	0	1	0
1	1	Toggle	Toggle

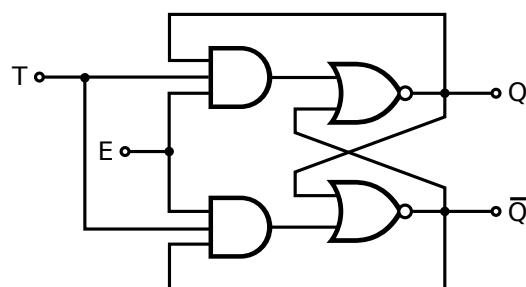
Tab. 4.3: Pravdivostní hodnoty JK latch



Obr. 4.4: JK latch

T	Q	\bar{Q}
0	Last	Last
1	Toggle	Toggle

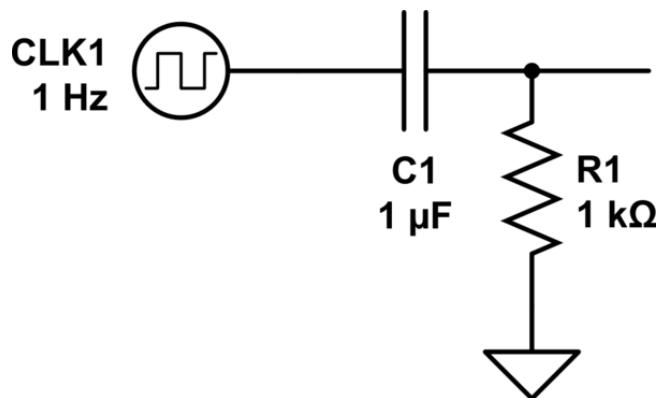
Tab. 4.4: Pravdivostní hodnoty T latch



Obr. 4.5: T latch

4.7 Flip Flop

- sekvenční obvod pracující dle hodin
- připojení hodin (CLK) na enable vstup
- nutno zapsat hodnotu jednou, ne ji zapisovat po celou dobu aktivace hodin → rising nebo falling edge
 - rising edge – k rychlé aktivaci dojde při změně signálu z 0 na 1
 - falling edge – k rychlé aktivaci dojde při změně signálu z 1 na 0
 - většinou využita rising edge



Obr. 4.6: Obvod pro dosažení rychlého zapnutí, možno přidat diodu pro odstranění *negativních pulzů*

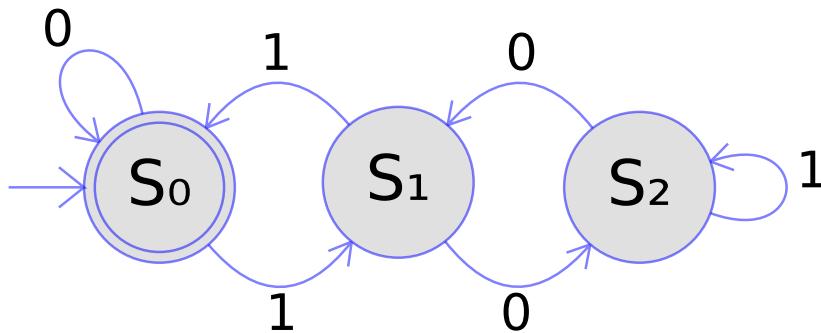
4.8 Konečný automat

- popis jednoduché počítače
- model systémů různých stavů
- přechod mezi stavů na základě vstupů
- vždy pouze v jednom stavu
- skladba
 - konečný počet stavů
 - konečný počet externích vstupů
 - konečný počet externích výstupů
 - explicitní specifikace přechodu mezi stavů
 - explicitní specifikace hodnoty výstupu
- části
 - logika dalšího stavu
 - registr stavu
 - * uložení dat na začátku clock cyklu
 - * data dostupná celý clock cyklus
 - logika výstupu
- příklad: výtahy, semafory, kombinační zámky...

5 Základní části počítače, operační systémy

5.1 Základní části počítače

- hardware – fyzické části počítače
- PC case – počítačová skříň
 - skříň pro uchování, ochranu a namontování jednotlivých počítačových komponentů
- základová deska
 - deska propojující všechny komponenty
 - uchovává v sobě BIOS – první spuštěný program, boot OS



Obr. 4.7: Symbolický nákres konečného automatu

- * uložení na ROM (read only memory)
- chipset
 - * north bridge – komunikace CPU a ostatních komponentů systému
 - * south bridge – připojen na north bridge, pomocná rozhraní
 - * Super I/O čip – připojen na south bridge, pomalé a legacy komponenty
 - sběrnice (bus)
 - * propojení komponentů na základní desce, „dálnice dat“
 - * interface pro zapojení přídavných karet
 - CMOS, CMOS baterie – vždy běžící hodiny
 - procesor (CPU – central processing unit)
 - „mozek počítače“
 - provádí výpočetní úkony, spouští programy, komunikuje s ostatními zařízeními
 - operační paměť (RAM – random access memory)
 - uchování dat během spuštění programu
 - pomalejší než L3 či jiná cache, ale mnohem rychlejší než disk
 - při zpracovávání se zde načtou data z disku pro rychlejší přístup
 - funkční pouze s energií, po vypnutí počítače smazání
 - disk
 - „persistent memory“
 - velkokapacitní dlouhodobé uchování dat
 - data zachována i po vypnutí počítače
 - HDD, SSD, CD, DVD
 - zdroj
 - zdroj elektrické energie pro komponenty
 - modulace 230 V střídavého napětí na různá nižší stejnosměrná napětí (3,3 V, 5 V, 12 V atd.)
 - rozšiřující karty
 - většinou na PCI(E) buse
 - grafická karta
 - * specializace na rendering
 - * velký počet cores o menším výkonu
 - * paralelizace
 - zvuková karta
 - * zvukový interface a porty
 - * dnes většinou integrovaná do motherboards
 - síťová karta atd.
 - * spojení s LAN, případně WLAN

5.2 Záznamová média

- viz sekce 2.4

5.3 Programové vybavení počítače

- software – programová část počítače

5.3.1 Možnosti programování / programovací jazyky

- nejnižší úroveň – executable code / strojový kód / machine code
 - interpretace CPU nebo GPU
 - čistá binární data měnící stav procesoru
- první úroveň abstrakce – programovací jazyk assembly
 - pro lidi již srozumitelné
 - velká kontrola nad systémem
 - možné měnit velmi low-level věci
- vyšší programovací jazyky
 - komplikované/interpretované na strojový kód a následně na binární kód
 - pro člověka srozumitelné
 - velká řada funkcí, u vyšších garbage collection, arrays, classes atd.

5.3.2 Typy softwaru

- systémový software
 - firmware – software v hardwaru, ovládání hardwaru, low-level operace
 - microcode – specifikace pro CPU jak spouštět strojový kód
 - operační systém
 - * správa počítače, spuštění programů
 - * ovladače – způsob komunikace se zařízeními
- aplikace / aplikační software
 - desktop aplikace – prohlížeč, office, další programy, příp. aplikace v telefonu
 - webové aplikace – využití JavaScriptu, aplikace spuštěna v prohlížeči
 - server software – většinou bez GUI, hostování webových aplikací, provádění výpočetních úkonů atd.
 - plugin/extension – rozšíření/modifikace jiné aplikace

5.4 Operační systémy

- hlavní software
- řízení hardwaru, management aplikací, IO operace, alokace paměti atd.

5.4.1 Příklady

Unix

- původně v assembly, později v C
- používán na mainframech
- multitasking OS
- *Unix filozofie*
 - jeden program dělá jednu věc a dělá ji dobře
 - předpoklad, že output jednoho programu může být input dalšího; programy jsou schopny spolu komunikovat
 - programy by měly být schopné využívat text streams – univerzální komunikace
 - všechno je soubor
- základ několika dnešních operačních systémů

Unix-like systémy

- systémy pokračující v Unix filozofii a rozšiřující Unix
- Linux, BSD, macOS / OS X a další

Linux

- 1991, Linus Torvalds, původně vytvořen jako vedlejší projekt
- Unix-like systém bez Unix kódu
- open-source
- univerzální, podpora mnoha systémů
- servery, embedded systems, superpočítáče
- různé distribuce – Debian, Fedora, Arch, Ubuntu, Linux Mint, OpenSUSE
- základ Androidu či Chrome OS

BSD

- různé varianty – FreeBSD, OpenBSD, NetBSD...
- využívá původní UNIX kód, jeho modifikace a rozšíření
- využití jako webserver, první server fungoval na BSD
- původ mnoha dnešních internetových protokolů

macOS / OS X

- proprietary operační systém firmy Apple
- používán na všech jejich počítačích
- postaven na technologii vyvinuté v NeXT
- prolínání programů s linuxem
- větší zastoupení uživatelů – větší hardwarová kompatibilita

MS-DOS a Microsoft Windows

- proprietary OS od Microsoftu
- nejrozšířenější OS mezi stolními počítači a notebooky
 - většinou již předinstalován
- založen na MS-DOS
- velký důraz pro zpětnou kompatibilitu
- design hlavně pro x86-64 procesory, verze i pro ARM a IA-32 (x86)
- verze i pro servery

5.4.2 Komponenty

Kernel

- základní operace a kontrola hardwaru
- přístup k RAM / memory management, správa hardwaru pro programy, interrupts
- multitasking, správa CPU, program extension
- disk access a file systémy
- ovladače/drivers
 - kód pro ovládání a komunikaci s hardwarem

Networking

- umožnění připojení k síti
- síťové protokoly, hardware, aplikace využívající síť
- TCP/IP, SSH, DHCP, DNS...

User interface

- komunikace uživatele se systémem
- pomocí vstupních a výstupních zařízení

Command-line interface (CLI)

- interfacing pomocí příkazové řádky
- ve své podstatě pouze text
- přímé psaní příkazů s argumenty
- velké možnosti
- SH, Bash, ZSH, FISH...

Graphical user interface (GUI)

- grafická část systémů
- využití moderními systémy
- mnohé operační systémy dovolují nainstalovat a vytvořit jakékoliv GUI, hlavně Linux
 - desktop environments – GNOME, KDE Plasma, Cinnamon, XFCE, LXDE...
 - window managers – i3, xmonad, dwm, bspwm, awesome
- Windows – windows shell

6

6 Periferní zařízení počítačů

- zařízení mimo počítač, komunikace s ním
- vstup/výstup informací
 - vstupní jednotky – klávesnice, myš/touchpad, grafický tablet, skener, mikrofon, dotyková obrazovka, joystick, gamepad, trackball, čtečka karet...
 - výstupní jednotky – monitor, tiskárna, reproduktory, sluchátka, RGB osvícení

6.1 Vstupní zařízení

- pořízení informace a předání počítači
- změna vstupu počítače

6.1.1 Klávesnice

- základní vstup
- vstup textu, znaků či funkcí
 - písmena, čísla, znaky, šipky, F1–F12, ctrl, alt, shift, změna hlasitosti...
- různé rozložení klávesnice
 - ČR – QWERTZ
 - US – QWERTY
 - další – ASERTY, Dvorak...
- připojení – PS/2, USB, bluetooth, bezdrátové 2,4GHz
- kódování jazyků – ASCII, UNICODE (UTF-8, UTF-16, UTF-32) – univerzální sada
- typy – membránové, mechanické (red, brown, blue... switches), optické

6.1.2 Myš

- vstup polohy na obrazovce + 2 tlačítka a kolečko (+ macro klávesy)
- myš senzory – kuličkový, optický, laserový
- připojení – PS/2, USB, bluetooth, bezdrátové 2,4GHz
- parametry – DPI, odezva

Touchpad

- u notebooků
- „malá dotyková obrazovka“
- kapacitní technologie
- gesta

Dotyková obrazovka

- rezistivní
 - starší, pouze jeden dotek, špatná citlivost a rozlišení
 - 2 kovové vrstvy separované vzduchem, spojení vrstev při stlačení – obvod
- kapacitní
 - vodivá vrstva na povrchu, prst přeruší proud, multitouch, gesta

6.1.3 Grafický tablet

- podobný myši
- tablet a pero
- malování, retuše, manipulace obrázků
- manipulace jak s tužkou, citlivost na tlak, absolutní polohování
- parametry – DPI, odezva, kvalita, citlivost, příjemnost psaní

6.1.4 Skener

- převod 2D nebo 3D snímku do počítače – 2D a 3D skenery
- parametry – DPI, rozlišení, barevná kvalita, rychlosť

2D skenery

- převod obrázku do počítače, skenování dokumentů
- neviditelné záření nebo LED technologie
- parametr DPI – 600–10 000 DPI

3D skenery

- převod objektu do 3D modelu
- snímání odraženého světla
- optické, laserové, kontaktní sonda
- archeologie, lékařství, vývoj technologií, 3D tisk

6.1.5 Mikrofon

- záznam zvuku
- převod změny tlaku vzduchu na elektrický signál
- dynamické
 - membrána s elektromagnetem → elektrický signál
 - potřeba zesilovač
- kondenzační
 - uvnitř kondenzátor, změna vzdálenosti mezi pláty

Parametry

- směrové charakteristiky – všesměrová, kardiodní, superkardiodní, hyperkardiodní, osmičková/- bidirekcionální, úzce směrová
- frekvenční rozsah
- napájení – tonaderspeisung, fantomové napájení, 5V přes konektor
- citlivost; impedance
- poměr signál a šumu

6.1.6 Fotoaparát / Kamery

- záznam fotografií / videí / vstup videa
- světlocitlivý detektor, objektiv, uzávěrka
- záznam v pixelech
- snímače
 - CCD
 - * kvalitnější, dražší
 - CMOS
 - * menší, levnější, horší citlivost

Parametry

- rozlišení
- velikost senzoru
- poměr stran senzoru
- typ kamery – DSLR, Mirrorless

6.2 Výstupní zařízení

- 6
- výstup informace z počítače
 - předání informace uživateli

6.2.1 Monitor

- základní výstup obrazu, textu a grafiky
- připojen na grafickou kartu
- připojení
 - analog – analog, VGA
 - digitální – DVI, HDMI, DisplayPort

Technologie

- CRT – stará, katody emitující elektrony, a cívky – „střílení obrazu“
- LCD
 - tekuté krystaly, rozsvícení/zhasnutí pixelu, barva zajištěna filtry
 - backlight, blokace světla pixely → nedokonalá černá
 - TN – levný, rychlý update, možnost 144 Hz, malý zorný úhel, nepřesné barvy
 - VA – vertical alignment, lepší tmavé barvy než LCD
 - IPS – rozšířené barvy, dnes nejrozšířenější, nejlepší z LCD technologie, dobrý zorný úhel
- Plazmová obrazovka – elektrický výboj v plynu
- OLED, QuantumDot OLED
 - dokonalá černá – pixel vytváření vlastní světlo

Parametry

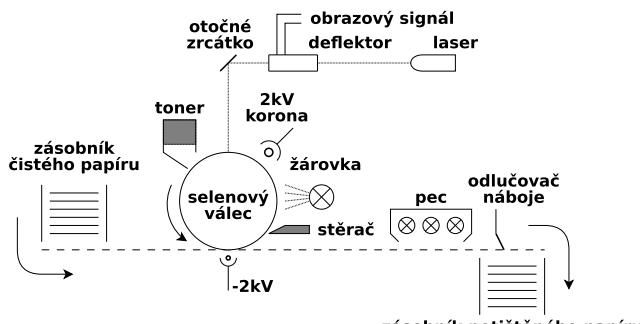
- velikost (uhlopříčka 15" – 42")
- připojení
 - analog – analog, VGA
 - digitální – DVI, HDMI, DisplayPort
- poměr stran – 16 : 9, 16 : 10, 4 : 3, 21 : 9
- rozlišení – Full HD ($1\ 920 \times 1\ 080$), QHD ($2\ 560 \times 1\ 440$), 4K
- frekvence – 50 Hz – 320 Hz

6.2.2 Tiskárna

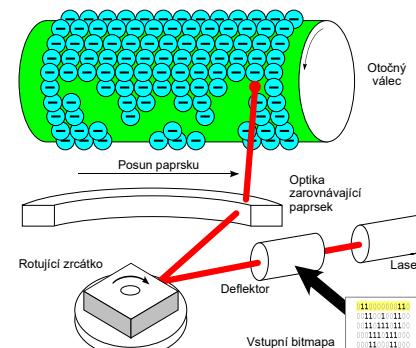
- vynesení obrazu na papír
- plotr
 - velkoformátová tiskárna
 - pero / inkoustová hlava, i řezací hlava
 - převážně vektorová grafika – definice cest pro hlavu
- 3D tiskárny
 - tisk 3D modelů
 - možnost mnoha filamentů

Typy tisku

- jehličková
 - otisknutí jehliček přes barvicí pásku na papír
 - kladky – spolehlivé, malé náklady, použití traktorového papíru
 - zápory – pomalé při tisku grafiky, grafika s omezenou paletou barev, malá kvalita tisku, hluk,
- inkoustová
 - vymrštování kapek ($\pm 35 \text{ pl} = 35 \cdot 10^{-12} \text{ l}$) vysokou rychlostí na papír
 - systém CMYK
 - typy
 - * termální – zahřátí inkoustu, vznik bubliny, vymrštění inkoustu
 - * piezoelektrický – piezoelektrické krystaly (mění objem vlivem el. napětí) – stažení komor, výstřik inkoustu
 - * voskové – princip termální tiskárny, místo inkoustu vosk, kvalitnější a pestřejší tisk,
 - kladky – klidnější provoz, kvalitnější tisk, barevný tisk, relativně nízká pořizovací cena
 - zápory – drahý inkoust, ucpávání trysek, pomalý proti laseru/LED, rozpustný ve vodě, omezena životnost inkoustu
- laserová
 - toner – jemný prášek
 - vykreslení obrázku laserem na světlocitlivý válec, nanesení toneru – uchycení na osvětlených místech, obtisk na papír, tepelná fixace
 - vybití náboje válce na místech ozářené laserem – přilnutí toneru pouze na tyto místa; na jiných místech odpuzován (stejná polarita, jako válec)
 - možnost nahradit laser LED diodami
 - kladky – vysoká kvalita tisku, rychlý tisk, nízké provozní náklady, nízká hlučnost
 - zápory – vyšší pořizovací cena, potřeba zahřát, nevhodné pro kvalitní fotografie



(a) Schéma



(b) Vybíjení válce

Obr. 6.1: Funkce laserové tiskárny

Parametry

- rozlišení – DPI (dots per inch) – 150 dpi – 1 200 dpi
- připojení – USB, Ethernet, WiFi
- rychlosť tisku
 - ppm (pages per minute) – tisk stránek textu
 - ipm (images per minute) – tisk komplexných stránek s obrázky a grafikou
 - většinou rychlosť okolo 15 ppm pro černobílý tisk

6.2.3 Zvukový výstup

- sluchátka, reproduktory
- výstup ze zvukové karty
- možnosť zesilovače, mixážního pultu...
- připojení
 - analog – 3,5 mm jack, 6,35 mm jack
 - digitální – bluetooth
- parametry – hlasitost, přesnost, frekvenční rozsah, impedance, napětí...

7 Počítačové sítě

- set počítačů sdílející informace nalezeny na nebo poskytující síťovými zařízeními
- komunikace mezi počítači díky digitálním protokolům
- osobní počítače, servery, síťové komponenty...

7

7.1 Rozdělení

7.1.1 Podle typu připojení

- více možností připojení k síti

Drátové sítě

- připojení k síti pomocí fyzického kabelu
- více možností kabelů
 - koaxial – hlavně televize, kanceláře, pracoviště; dnes již méně rozšířen
 - již existující vedení (coaxial, telefon, rozvod elektřiny)
 - twisted pair – 4 páry měděných drátů, až 20 Gb/s, využíván pro Ethernet, shielded varianta, různé kategorie (CAT 5, CAT 5E, CAT 6, CAT 7)
 - optický kabel – kabel vede světlo místo elektřiny, vysoká rychlosť rychlosť, možnosť přenosu na velké vzdálenosti (oceány)

7.1.2 Podle velikosti

- nanoscale
- NFC (near-field)
- BAN (Body Area Network)
- PAN (Personal Area Network)
- LAN (Local Area Network)
- CAN (Campus Area Network)
- MAN (Metropolitan Area Network)
- RAN (Radio Area Network)
- WAN (Wide Area Network)

LAN

- propojení malé sítě (domácnost, škola, firma...)
- Ethernet a WiFi (WLAN – wireless wide area network)
- propojení počítačů, připojení k NAS, tiskárny...
- dříve protokol IPX/SPX, dnes TCP/IP

VLAN

- virtuální LAN uvnitř sítě
- tvořena softwarem
- izolování částí sítě

WAN

- síť největšího pokrytí
- spojení pouze největších uzlů, dále již podsítě
- nejznámější příkladem internet

PAN

- síť krátkého dosahu
- Bluetooth, IrDA, USB, FireWire
- možno využít pro sdílení internetu

MAN

- spojení města do sítě

7.2 Síťové modely

- rozdělení sítě do vrstev
- komunikace v menších, jednodušších krocích
- standardizace komunikace mezi zařízeními

7.2.1 RM OSI

- *reference model Open Systems Interconnection*
- také nazýváno RM ISO/OSI (ISO – model mezinárodně standardizován)
- protokol o 7 vrstvách, n -tá vrstva využívá funkcí vrstvy $n - 1$
- výměna protokolových datových jednotek (PDU – protokol data unit) na každé vrstvě mezi dvěma entitami
 - PDU obsahuje SDU (service data unit) společně s headers a footers

Posílání dat

1. data pro přenos zkompletována na vrstvě n do PDU
2. PDU posláno do vrstvy $n - 1$, kde je přijímáno SDU
3. na vrstvě $n - 1$ se k SDU přidají headers a footers → PDU vrstvy $n - 1$
4. opakování procesu do vrstvy 1, kde jsou data přenesena
5. při přijímání jsou data přeneseny vrstvami jako série SDUs a přitom zbavována dat jednotlivých vrstev až do nejvrchnější vrstvy

7.2.2 TCP/IP

- *transmission control protocol / internet protocol*
- specifikace end-to-end komunikace
- specifikace paketizace dat, adresování, přenos, routování, a přijímání
 - paket – jednotka informace posílána sítí, obsahuje data a metadata
- zjednodušený RM OSI model

	Layer	Protocol data unit (PDU)	Function
Host layers	7 Application	Data	High-level APIs, including resource sharing, remote file access
	6 Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5 Session (Relační)		Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
Media layers	4 Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
	3 Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2 Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1 Physical	Bit, Symbol	Transmission and reception of raw bit streams over a physical medium

Tab. 7.1: Vrstvy tvořící OSI model

- funkce rozdělena do 4 vrstev
 1. vrstva síťového rozhraní
 2. síťová vrstva
 3. transportní vrstva
 4. aplikační vrstva

Aplikační vrstva (application layer)

- vrstva aplikací a procesů
- vytvoření uživatelských dat a předání dat dalším aplikacím na jiném, či stejném, hostově
- protokoly pro přenos dat – FTP, DNS, DHCP, Telnet, SMB, NTP, NFS...

Transportní vrstva (transport layer)

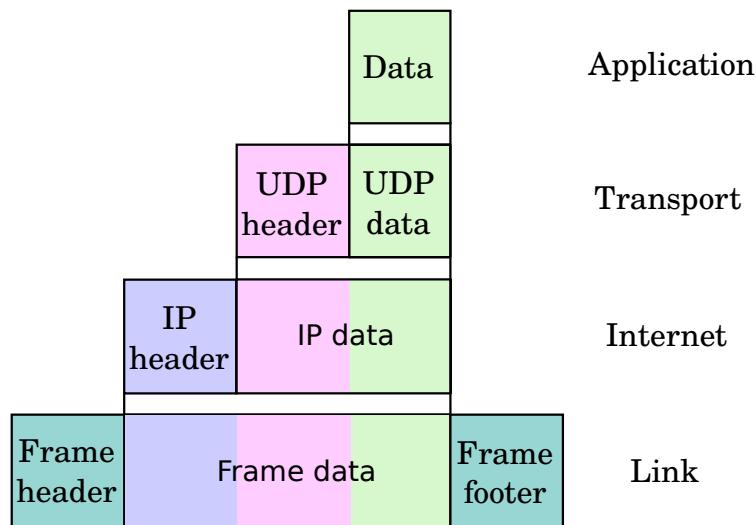
- host-to-host komunikace na lokální či vzdálené síti spojené routerem
- kanál pro komunikační potřeby aplikací
- spolehlivý (TCP) a nespolehlivý přenos (UDP) datagramů

Síťová vrstva (internet layer)

- předávání datagramů dalšímu hostovi
- navázání internetového spojení
- definice adres a routovacích struktur
- hlavní protokol – IP; funkce IP routeru

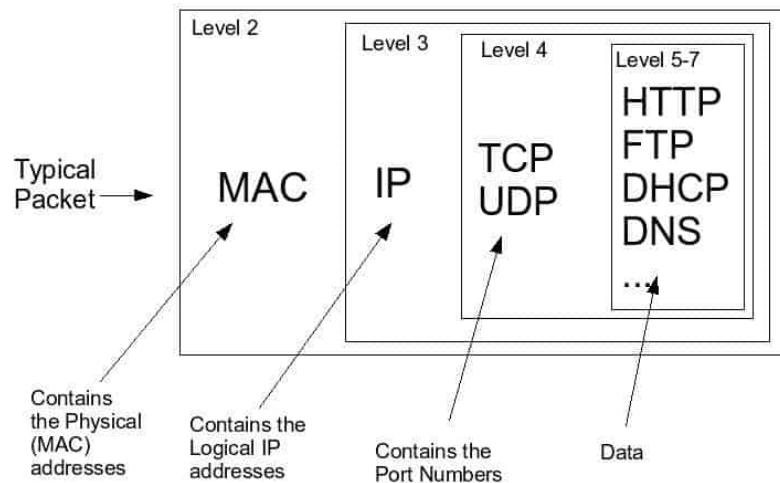
Vrstva síťového rozhraní (link layer)

- síťové metody v lokální síti, komunikace bez přerušení routerem
- protokoly popisující topologii sítě
- přístup k fyzickým hostům



Obr. 7.1: Zabalení TCP dat

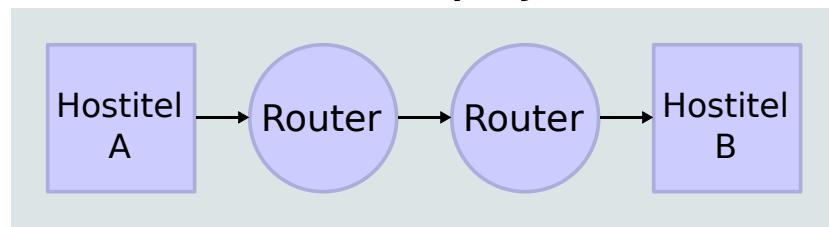
7



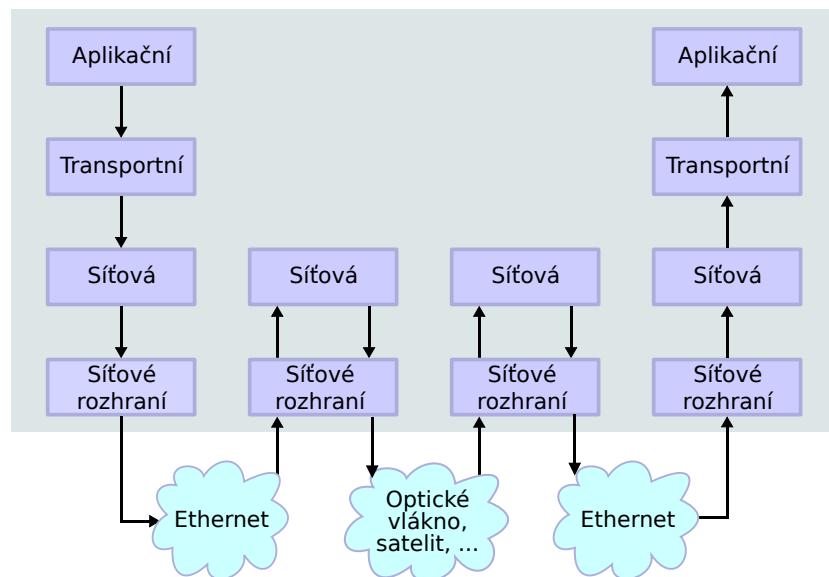
Obr. 7.2: Paket dat

- Ethernet, Token ring, FDDI

Síťová spojení



Architektura TCP/IP



Obr. 7.3: Topologie sítě při TCP přenosu

7.3 Síťové prvky

7.3.1 Router

- spojení jedné či více sítí
- využití IP protokolu
- dneska často spojení všech funkcí v jednom zařízení – modem, switch, AP, router a NAT v jednom
- vytváření podsítí
- routing tables pro směrování příchozích dat ke správným zařízením
- směrování paketů na základě síťové adresy v hlavičce

7.3.2 Modem

- převod analogového a digitálního signálu
- přenos digitálních dat po analogových trasách (telefon, koaxial, rádiový přenos)
- point-to-point packet routing na základě IP hlavičky packetu

TCP/IP	RM OSI
Aplikační vrstva	Aplikační vrstva
	Prezentační vrstva
	Relační vrstva
Transportní vrstva	Transportní vrstva
Síťová vrstva	Síťová vrstva
Vrstva síťového rozhraní	Linková vrstva
	Fyzická vrstva

Tab. 7.2: Porovnání TCP a OSI modelu

7.3.3 Switch

- síťový přepínač
- propojení několika částí sítě či zařízení k sobě
- přenosání signálu do cílového směru
- směrování paketů na základě síťové adresy v hlavičce paketu

Hub

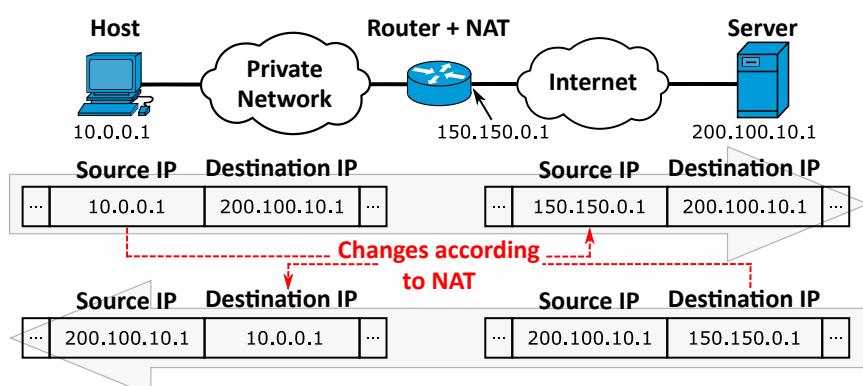
- předchůdce switch, posílá data do všech připojených směrů

7.3.4 Wireless Access Point (WAP, AP)

- zařízení vytvářející a umožňující připojení k bezdrátové síti
- funguje jako bridge
- zabezpečení – WEB, WPA, WPA2 (WPA-Personal, WPA-Enterprise)

7.3.5 NAT – Network address translation

- překlad lokálních a veřejných IP adres
- využíváno ISP pro zmenšení potřeby počtu nových IPv4 adres



Obr. 7.4: Funkce NATu

7.4 Protokoly

- TCP a UDP
 - navazují spojení mezi hostem a cílem
 - TCP
 - * naváže komunikace, přenese data, uzavře komunikace; spolehlivý přenos
 - UDP
 - * pouze přenese data, může vysílat na celou síť, nespolehlivý přenos
- IP (IPv4, IPv6)
 - adresování klientů a routování packetů
 - funkce společně s TCP pro vytvoření komunikačního modelu
 - IPv4
 - * jednodušší verze
 - * 32 bitové adresy
 - * komplexní, možné errory
 - * dnes již došly na veřejném internetu IP adresy
 - IPv6
 - * rozšíření IP protokolu
 - * 128 bitů na adresy
 - * efektivnější a bezpečnější než IPv4
 - * dnes se již plně nepřešlo, hlavně na lokálních sítích
- DHCP
 - protokol zajišťující přiděl IP adres
 - automatické přidělování, obnovení a znovupoužití IP adres
 - hosti nemají nutně statickou IP
 - při připojení zařízení vyšle do sítě požadavek a DCHP server mu na jeho MAC adresu odpoví
- DNS
 - *domain name system*
 - systém pro přidělování jmen a domén IP adresám
 - client si vyžádá po DNS serveru IP adresu dané destinace a dostane ji
 - př.: `google.com` → 142.250.186.78
- protokoly pro přenos souborů
 - připojení na jiný počítač/server a získání souboru
 - FTP – File Transfer Protocol
 - * univerzální, možnost obnovení spojení, chybí šifrování
 - SMB – Server Message Block
 - * sdílení souborů na lokální síti pro Windows
 - * Linux provider Samba
 - NFS – Network File System
 - * protokol pro sdílení souborů na lokální síti, podporovaný Unix systémy
- HTTP(S)
 - Hyper Text Transfer Protocol
 - přenos HTML dokumentů a obecně webových stránek
 - HTTPS – šifrováno
- mailové protokoly – POP3, SMTP, IMAP

7.5 Tok dat v síti

- síť tvoří graf
- vrcholy – síťové křižovatky (routery, switchs, NATy...) a jednotlivá zařízení; hrany – propojení mezi síťovými zařízeními
- každá hrana ohodnocena podle rychlosti spojení
- algoritmy pro nalezení nejrychlejší a nejkratší cesty

8 Internet

- systém propojených počítačových sítí
- komunikace pomocí TCP/IP

8.1 Připojení k internetu

- zprostředkovatel – ISP (internet service provider)
- telefonní linka
 - analogový signál
 - konvertování přes modem
 - zastaralé
- kabel – Ethernet, optické kably
- bezdrátově – satelity (starlink), mobilní síť (4G – 60 Mb/s; 5G – 600 Mb/s), WiFi
- elektrické rozvody

8.2 Možnosti internetu

8.2.1 WWW

- *world wide web*
- informační systém, informace přenášeny přes HTTP
- dostupné přes webový prohlížeč
- webové stránky uloženy na webových serverech
- přístup pomocí Uniform Resource Locators (URLs)
- HTML, JS, CSS, PHP...

8.2.2 Email

- přenos emailových zpráv
- příjemce, odesílatel, předmět, samotná zpráva, přílohy
- přenos pomocí POP3 (příjem) a SMTP (odesílání) nebo IMAP (přístup k serveru)

8.2.3 VoIP

- Voice over IP
- hlasová komunikace pomocí IP
- možno zobecnit na telefonní utilities přes internet (hlas, zprávy, dnes i video...)

8.2.4 Přenos souborů

- přenos souborů přes internet
- FTP, SMB, NFS

8.3 IP adresa

- adresa označující umístění na síti
- statická nebo přidělaná DCHP serverem
- správa protokolem IP
- IPv4
 - jednoduší verze
 - 32 bitové adresy – 0.0.0.0 – 255.255.255.255
 - komplexní, možné errory
 - dnes již došly na veřejném internetu IP adresy
- IPv6
 - rozšíření IP protokolu

- 128 bitů na adresy – 2001:db8:1234::f350:2256:f3dd
- efektivnější a bezpečnější než IPv4
- dnes se již plně nepřešlo, hlavně na lokálních sítích
- některé IP vytrábeny pro lokální síť
 - class A – 10.0.0.0 – 10.255.255.255
 - class B – 172.16.0.0 – 172.31.255.255
 - class C – 192.168.0.0 – 192.168.255.255

8.4 Mac adresa

- unikátní číslo NICu (network interface controller)
- 6 dvojciferných hexadecimálních čísel
- identifikace zařízení na lokální síti
- dříve přiděleno výrobcem; dnes výrobci přidělena první polovina, druhou určuje sám
- dnes lze softwarem měnit
- protokol IEEE 802

8.5 Podsítě

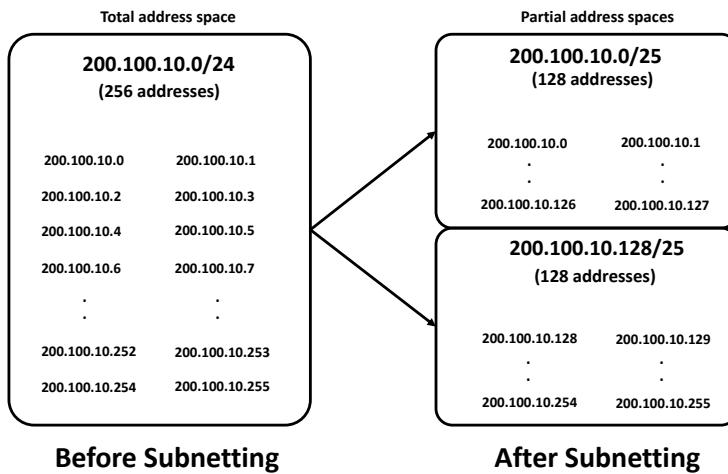
- označení pro samostatnou část sítě uvnitř větší sítě
- určení IP adresy pomocí masky sítě
 - určuje, kolik bitů je konstantních a které jsou možné využít pro adresování
 - značená počtem bitů za adresou (např. /24), nebo jako binární číslo (255.255.255.0)
- využito pro separování zařízení na samostatné sítě
- poslední IP rezervována pro subnet broadcast pro všem uživatelům
- příklad
 - Prefix sítě 192.168.88.0/24 – host 192.168.88.1–192.168.88.254
 - * broadcast na 192.168.88.255
 - Prefix sítě 10.5.24.128/25 – host 10.5.24.129–10.5.24.254

	Binární zápis	Tečka-decimální zápis
IP adresa	11000000.00000000.00000010.10000010	192.0.2.130
Maska podsítě	11111111.11111111.11111111.11000000	255.255.255.192
Prefix sítě	11000000.00000000.00000010.10000000	192.0.2.128
Číslo hosta	00000000.00000000.00000000.00000010	0.0.0.2

Tab. 8.1: Příklad zařízení s IP adresou 192.0.2.130/26

8.6 ARP

- *address resolution protocol*
- definován v RFC 826 v roce 1982; v rámci TCP/IP
- protokol k získání linkové adresy (např. MAC adresa), používán IPv4
 - IPv6 – NDP (neighbor discovery protocol)
- funguje v rámci jedné podsítě, využito při posílání dat neopouštějící podsítě
- práce na linkové vrstvě (vrstva síťového rozhraní)
- posílání dat
 - host 1 se podívá na DNS na IP hostu 2
 - host 1 se podívá do své routovací tabulky na IP hostu 2
 - IP hostu 2 je nalezeno v cachnuté tabulce
 - * host 1 pošle data na MAC adresu hostu 2
 - IP hostu 2 není v cachnuté tabulce



Obr. 8.1: Příklad rozdělí sítě do dvou podsítí

- * host 1 vyšle broadcast na **ff:ff:ff:ff:ff:ff** (přijímáno všemi hosty na síti) s požadavkem o MAC adresu IP hostu 2
- * host 2 odpoví svojí MAC adresou a přitom může přidat hosta 1 do své routovací tabulky

8.7 Směrování (routing)

- určení cesty datagramů (resp. paketů) po síti ke svému cíli
- snaha optimalizace rychlosti a efektivnosti
- zajišťováno síťovou vrstvou ISO/OSI
- cesty mnohdy velmi složitá – řešení vždy pouze jedno kroku / komu dalšímu předat paket
- cesta nalezena na základě směrovacích tabulek

8.7.1 Směrovací tabulky (routing table)

- udržují záznamy o cestách na různá místa sítě
- možné specifikovat ručně, ovšem většiny plněny routing protokolem
- uložena, používána a i vytvářena v aktivních uzlech sítě
- záznamy seřazeny sestupně podle síťové masky (nejkonkrétnější IP adresa / nejdélší maska nahore)
 - poslední záznam – nejbližší router k internetu (default gateway)
- informace záznamu (řádku tabulky)
 - cíl
 - maska podsítě
 - brána
 - síťové rozhraní
- vznik směrovací tabulky
 - statické směrování – tabulka vytvořena administrátorem, neměnná
 - dynamické směrování – reaguje na změny v síti, mění tabulku, využití směrovacích algoritmů

8.7.2 Směrovací algoritmy

- algoritmy zodpovědné za hledání cesty na síti

Distance vektor

- využití Bellman-Fordova algoritmu
- každé spojení nodes má svoji váhu
- nodes posílají pakety po trase s nejmenší sumou vah

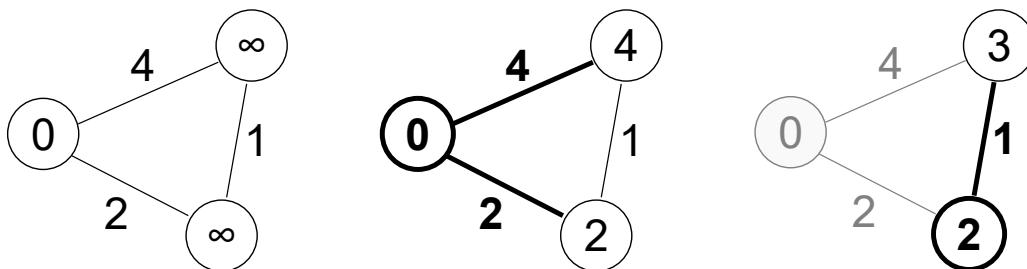
- vytváření spojení
 - při zapnutí node ví pouze své přímé sousedy a cenu, za jakou se k nim dostane
 - každá node si se svým sousedem vyměňuje informace o dalších nodes, o kterých ví (výměna routing tables)
 - jestliže přijatá cesta je efektivnější než již známá → zapsání do vlastní tabulky
 - při přidání či ubrání node → update informací
- časem konvergence nejkratších cest v celé síti

Link state

- node zaplaví síť requestem o sousedech každé jiné node
- z přijatých informací sestaví weighted graph
- na základě grafu zjistí pomocí shortest path algoritmu nejkratší cestu ke každé node
- z výsledku vytvoří tree graph s rootem v momentálním zařízení – cesta k jakémukoliv zařízení je poté pouze cesta tímto stromem

Dijkstrovovův algoritmus

- algoritmus používán k hledání cesty v grafu s váhami spojení
- nalezne nejlepší cestu mezi zdrojovou node a všemi ostatními
- postup při hledání nejkratší cesty
 1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
 2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. The tentative distance of a node v is the length of the shortest path discovered so far between the node v and the starting node. Since initially no path is known to any other vertex than the source itself (which is a path of length zero), all other tentative distances are initially set to infinity. Set the initial node as current.[15]
 3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be $6 + 2 = 8$. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, the current value will be kept.
 4. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
 5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
 6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new current node, and go back to step 3.



Obr. 8.2: Průběh Dijkstrovova algoritmu na malém grafu

9 Textové editory

- programy pro vytváření a úpravu textu

9.1 Druhy

- Základní
 - čistý text
 - notepad, vi, nano
- Programátorské
 - čistý text, rich text, code highlighting
 - notepad++, vim (s konfiguracemi), VS Code
- Publikační
 - formátování textu, vkládání obrázků a tabulek, hromadná korespondence...
 - Microsoft Word, LibreOffice Writer
- Profesionální
 - k tisku časopisů a dalších
 - Adobe InDesign
- Online
 - možnost pracovat s nimi na internetu, ukládání na cloud
 - Google docs

9.2 Formáty dokumentů

- .txt – plain text
- .htm, .html – webové stránky
- .doc, .docs – dokumenty Word
- .tex – L^AT_EX dokument
- .md – Markdown syntax

9.3 Vzhled dokumentu

- určen několika elementy
- určuje charakter práce

9.3.1 Obsah dokumentu

- různé prvky nutné
- závislé na typu dokumentu
- úvodní strana
 - název práce, autor, druh, rok vytvoření, název instituce
- anotace
 - stručná obsah v několika větách
 - v jazyce práce i anglicky
- klíčová slova
 - slova charakterizující práci, zhruba 4–6
- obsah práce
- seznamy
 - tabulek
 - obrázků
 - grafů
 - zkratek a značek...
- vlastní text
- přílohy
 - obrázky, tabulky, grafy

- další externí zdroje

9.3.2 Formát textu

- různé změny vzhledu
- **bold**/*italic*/underline
- barva textu, pozadí
- zarovnání textu
 - vlevo/vpravo (na levý a pravý praporek)
 - doprostřed
 - do bloku

9.3.3 Font (písmo)

- různé druhy písma, různě vhodné
- patkové
 - obsahuje „patky“ a různé dekorace
 - vhodnější na tisk
 - Times New Roman, Computer Modern
- bezpatkové
 - neobsahuje ozdoby, čisté písmo
 - vhodnější pro konzumaci z obrazovky
 - Arial, Helvetica, Roboto...
- ozdobné
 - písmo s velice ozdobnými elementy
 - napodobování rukopisu, kaligrafie
 - papyrus, hello
- symbolické
 - font tvořen symboly
 - wingdings

9

9.3.4 Odstavce

- řazení textu do odstavců
- mezi odstavci není mezera
- první slovo odsazeno
- první odstavec po nadpisu neodsazen

9.3.5 Seznamy

- odrážkový seznam
 - jako odrážka nějaký znak
 - úrovně od sebe odlišeny odsazením a odrážkou
- číslovaný
 - automatické číslování
 - různé styly počítaní – numerické, abecední, římské číslice...

9.3.6 Nadpisy

- dělení textu
- několik úrovní – nadpis, podnadpis, podpodnadpis...
- udržování stejného stylu pro všechny nadpisy

9.3.7 Sloupce

- dokument nebo část dokumentu možno rozložit do sloupců
- zlepšení čitelnosti v některých případech
- věděcké publikace

9.4 Hromadná korespondence

- způsob vygenerování souborů podle záznamů konkrétních osob
- využití importované tabulky
- automatické vyplnění dat

9.5 Záhlaví, zápatí (header a footer)

- přenášení informace mezi stránkami
- většinou číslo strany, autor, název práce, název instituce, rok vydání, název momentální kapitoly...
- word – 2x kliknutí na záhlaví/zápatí; vložení různých elementů
- LATEX – fancyhdr package

9.5.1 Číslování stránek

- samostatné; číslo stránky / celkový počet stránek
- důležité pro orientaci
- většinou vlevo dole nebo dole na vnější straně dokumentu

9.6 Oddíly

- způsob Wordu na to, jak oddělit sekce dokumentu od sebe
- lepší pro práci s textem, kdy se můžeme oddílu věnovat individuálně
- Karta rozložení – vzhled stránky – konce

9.7 Automatické číslování

- Karta Reference → Titulky → Vložit titulek → Automatický titulek →
 - bitmap image (obrázek)
 - tabulka aplikace MS Word (tabulka)

9.8 Generování seznamů

9.8.1 Obsah

- k lepší přehlednosti dokumentu
- automatická generace z označených nadpisů
- Reference → Obsah

9.8.2 Rejstřík

- výtah důležitých slov, kde je lze nalézt
- Reference → Rejstřík → vložit rejstřík
- slova musí být označena (Reference → Rejstřík → označit položku)

9.8.3 Seznam obrázků a tabulek

- obrázky a tabulky musí mít tituly
- Reference → Titulky → Vložit seznam

13 Databáze MS Access, SQL

13.1 Databáze

- organizovaná kolekce dat
- uložení a přístup elektronicky
- uloženy jako soubor (malé DB), nebo hostovány na počítačových clusterech či v cloudu (velké DB)
- různé designy, query jazyky, bezpečnost, způsob reprezentace...
- typy
 - Hierarchická databáze
 - Sítová databáze
 - Relační databáze
 - Objektová databáze
 - Objektově relační databáze
- MySQL, MongoDB, MariaDB, MS Access...

13.1.1 MS Access

- database management system (DBMS)
- kombinace relační databáze s GUI
- součástí MS Office
- uložení databáze ve vlastním formátu
- podpora VBA

13.2 Struktura

- databáze – systém uchovávající všechna data a tables
- table
 - objekt databáze
 - uchovává samotná data ve specifikovaných sloupcích (fields)
 - každý řádek jeden záznam (entry)
 - primary key
 - * sloupec udávající primární klíč / id záznamu
 - * unikátní pro každý záznam
 - * specifikován v jiných tables při propojování dat
 - foreign key
 - * klíč ukazující na entry dat v jiném tablu
 - * např.: objednávka 1 patří zákazníkovi 2
- relace
 - specifikace souvislostí mezi tably
 - foreign key v table1 ukazuje na primary key v table2
 - použití slučování – **INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN**
- view
 - virtuální table
 - výsledek **SELECT** příkazu na reálných datech

```

1 | CREATE TABLE `persons` (
2 |   `id` INT(11) UNIQUE NOT NULL AUTO_INCREMENT,
3 |   `last_name` VARCHAR(255) CHARACTER SET utf8 NOT NULL,
4 |   `first_name` VARCHAR(255) CHARACTER SET utf8 NOT NULL,
5 |   `born_name` VARCHAR(255) CHARACTER SET utf8 NULL DEFAULT NULL,
6 |   `created` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
7 |   PRIMARY KEY (`id`)
8 |

```

Kód 1: Vytvoření tablu

13.2.1 Formáty dat/datotypy

- pro každé field potřeba specifikovat data typ
- typy textu, čísel, binárních dat, času a dat...
- SQL (MySQL) data typy – `CHAR(size)`, `VARCHAR(size)`, `BINARY(size)`, `TEXT(size)`, `BLOB(size)`, `MEDIUMTEXT`, `MEDIUMBLOB`, `LONGTEXT`, `ENUM(val1, val2...)`, `BIT`, `BOOL`, `SMALLINT(size)`, `INT(size)`, `FLOAT(p)`, `DATE`, `TIMESTAMP`...
- MS Access data typy – text (MySQL – `TINYTEXT`), memo (MySQL – `TEXT`), byte, integer (MySQL – `TINYINT`), long, single, double, currency, date/time, yes/no, ole object (obrázky, audio, binární data...), hyperlink, lookup wizard (MySQL – `ENUM`)...
- computed column/field – sloupec počítán z jiných sloupců
 - SQL – `ALTER TABLE orders ADD final_price AS item_count * item_prize`
 - MS Access – vytvoření vzorce pomocí expression builderu

13.3 SQL

- standardizovaný programovací jazyk pro používání (hlavně relačních) databází
- původně vytvořeno roku 1970
- vytváření databází, zápis dat, čtení (query) dat, mazání dat, vytváření souvislostí...
- specifikace úkonu pomocí speciálních klíčových slov, pevná syntaxe

13.3.1 Příkazy

- `SELECT` – čtení dat z tablu
- `UPDATE` – změna uložených dat
- `DELETE` – smazání řádku
- `INSERT INTO <tableName>` – přidání řádku
- `CREATE DATABASE <dbName>` – vytvoření nové databáze
- `ALTER DATABASE <dbName>` – modifikace databáze
- `CREATE TABLE <tableName>` – vytvoření nového tablu
- `ALTER TABLE <tableName>` – změna tablu
- `DROP TABLE <tableName>` – smazání tablu
- `CREATE INDEX <indexName> ON <tableName> (column1, column2...)` – vytvoření indexu tablu (sloupce tablu)
- `DROP INDEX <indexName>` – smazání indexování tablu (sloupce tablu)

13.3.2 SELECT

- použito pro čtení dat z tabulky
- syntax
 - `SELECT column1, column2, ... FROM tableName;` – export specified columns
 - `SELECT * FROM tableName` – select all columns

WHERE

- specifikace podmínky, pro které se má příkaz aplikovat
- `SELECT * FROM persons WHERE firstName = 'Karel'`
- možno pomocí `AND` a `OR` použít více podmínek
- používáno i při `UPDATE` a `DELETE`

ORDER BY

- řazení dat
- `SELECT * FROM persons ORDER BY firstName`

13.4 Práce s MS Access

13.4.1 Návrh databáze

- vytvoření tables s příslušným obsahem dat a fields
- vytvoření příslušných relací mezi tables
 - Database Tools → Relationships

13.4.2 Formuláře

- způsob vnější integrace s daty
- více user-friendly způsob zapisování/aktualizace/mazání dat
- Create → Form
- změna vzhledu ve form designeru

13.4.3 Třídění a vyhledávání dat

- řešeno pomocí querries
- možnost zobrazit data, listovat různá data z více tabulek
- specifikace zobrazených sloupců, jejich podmínek a řazení
- možno vytvářet přímo SQL query

14 Obrazová informace

- informace obrazu v počítači
- záznam bitmapou nebo vektorem

14.1 Digitální obraz

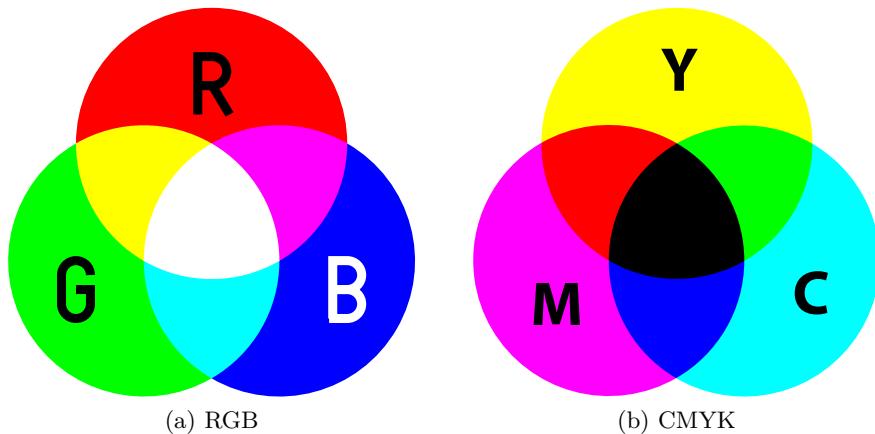
- reprezentace obrazové informace
- 2D čtvercová síť – pixely

14.1.1 Pixely

- svítící bod na monitoru / bod obrazu
- tvořen subpixely
 - červený, zelený a modrý subpixel – RGB

14.1.2 Barevné modely

- způsoby reprezentace barvy
- různé modely mají různé využití



Obr. 14.1: Porovnání RGB a CMYK modelu

RGB

- Red, Green, Blue
 - adititivní způsob míchání barev – všechny hodnoty na maximum dají bílou
 - zápis v hodnotách 0–1 nebo 0–255, možné i hexadecimálně
 - trojice čísel – (0, 49, 255), #0039FF
 - použití
 - zobrazování barev displeji
 - reprezentace barev v grafický programech

CMYK

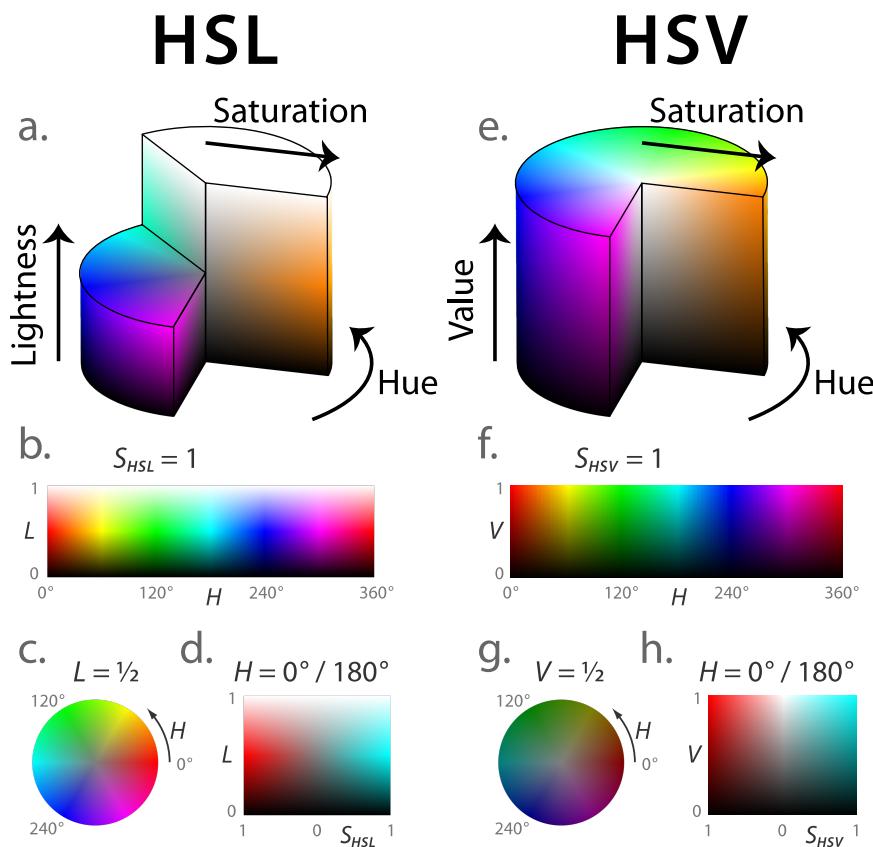
- Cyan, Magenta, Yellow, Key (azurová, purpurová, žlutá, černá)
 - subtraktivní míchání barev – barvy na maximum znamenají černou
 - zápis většinou v procentech
 - čtverice číslic – (0 %, 56 %, 99 %, 2 %)
 - použití – tisk

HSV

- Hue, Saturation, Value
 - reprezentace barvy jako odstínu, sytosti a světlosti / hodnoty
 - odstín
 - reprezentace jako úhel
 - udává odstín barvy
 - sytost
 - sytost barvy
 - sytost 0 – šedá, sytost 255 – plná barva
 - světlosť / hodnota
 - tmavost barvy
 - analogie ke svícení na barvu světlem
 - minimum – černá, maximum – plná barva
 - použití – grafické programy (jednodušší vybírání barev pro uživatele)

HSL

- Hue, Saturation, Lightness
 - podobná HSV, ale světlost značí polohu mezi černou a bílou, nejsvětlejší barvy v $L = 1/2$



Obr. 14.2: Porovnání HSL a HSV módu

14.1.3 Barevná hloubka (Color depth)

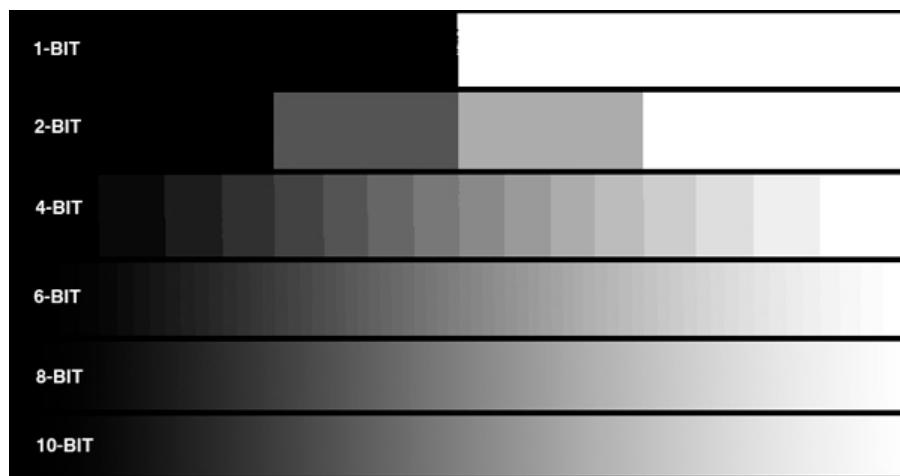
- vyjádření, kolik bitů je použito pro záznam barvy
- 1 bit – 2 barvy
- 8 bitů – 256 barev
 - retro konzole
- 24 bitů – 16 777 216 barev (True color)

14.1.4 Rozlišení

- udává počet pixelů na obrazovce
- udáváno v poměru $x \times y$
- standardní rozlišení (16:9)
 - HD / 720p – 1280×720
 - FullHD / 1080p / 2k – 1920×1080
 - 4k – 3830×2160
- poměr stran
 - poměr šířky a výšky, důležitý při zvětšování počtu pixelů
 - dnešní standard 16:9, také 16:10, dříve 4:3 displeje
 - formáty fotek také 1:1 nebo 3:2

14.1.5 DPI

- dots per inch
- hustota pixelů na jeden palec (2,54 cm)
- fyzické rozlišení obrazu
- moje používaný taky např. u citlivosti myší



Obr. 14.3: Porovnání různých barevných hloubek

- standard – 72 DPI, 96 DPI, 120 DPI, 300 DPI...



Obr. 14.4: Porovnání různých hodnot DPI a stejně velkém obrazu

14.2 Rastrová/bitmapová grafika

- obraz tvořen pixely v mřížce
- závislá na rozlišení
- u některých možnost pomocí algoritmů převést na vektor
- více běžná než vektorová
- fotografie, obrázky, kresby, scan...
- výhody
 - snadná tvorba
 - více detailů
- nevýhody
 - ztráta kvality při změně velikosti
 - velká velikost souborů při vysokém rozlišení
- různé editory
 - Adobe Photoshop, GIMP, Krita, Affinity Photo...
- různé formáty
 - jpg/jpeg, png, gif, raw, tiff...

14.2.1 Komprese

- zmenšování velikosti souborů
- ztrátová a bezztrátová
 - ztrátová – vynechání malých detailů, nevratná, jpg
 - bezztrátová – zanechává všechna data, png, gif, raw

Příklady metod komprese

- run-length encoding – optimalizace opakujících se dat
- huffman coding – převod patternů na bitové symboly s délkou v závislosti na frekvenci, využití binárního stromu
- diskrétní kosinová transformace – potlačení rozdílů v blízkých barvách

14.3 Vektorová grafika

- obraz tvořen matematicky definovanými tvary a křivkami
- nezávislé na rozlišení, rendering do konečného obrazu
- loga, ilustrace, technické nákresy...
- výhody
 - libovolná velikost
 - bezztrátovost dat
 - jednoduchý převod do rastru
 - paměť nezávislá na velikosti
- nevýhody
 - složitější výroba
 - méně detailů
- editory
 - Adobe Illustrator, Inkscape, Corel draw...
- formáty
 - eps, pdf, svg, ai (Adobe Illustrator), cdr (Corel draw)

17 Viry, WWW, FTP

17.1 Viry

- počítačové programy s cílem napadnout systém a dále se rozšířit
- motivace – profit, šíření zprávy, zábava, ukázání chyby, sabotáž systémů (DDoS)...
- využití hostovacího programu
 - computer worm nepotřebuje externí program
- většina virů směřována na MS Windows
- přenos přes soubory, po síti, USB flash disky...
- antiviry – programy chránící před viry

17.1.1 Historie

- 1949 – John von Neumann – Teorie sebereplikujícího se automatu
- 1982 – *Elk Cloner* – první vir ve veřejnosti
- 1984 – Fred Cohen – definoval pojem virus

17.1.2 Druhy virů

- spyware – vir vytvořen za účelem shromažďovat data o uživateli bez jeho vědomí
- adware – zahlcení počítače reklamami
- malware – software s účelem jakkoliv uškodit počítači
 - viry, červy, Trojské viry, Spyware, Adware, Ransomware...
- rozdělení podle hostitelů
 - spustitelné soubory – com, exe, elf...
 - boot sektry disket a disků, MBR (master boot record)
 - dávkové soubory a scripty – bat, sh...
 - makra v MS Office
 - speciální scripty aplikací
- rozdělení podle způsobu činnosti

- rezidentní viry – program nepokračuje, zůstává v RAM, infikuje soubory, se kterými uživatel pracuje
- nerezidentní viry – vir začne infikovat hned po spuštění vše, co najde
- stealth viry
 - * snaha maskovat
 - * při kontrole antivirem vrátí aplikaci původní data
 - * dnes lehce odhalitelné
 - * hlavně pro MS-DOS, dnes využití rootkitů
- makro viry
 - * makra kopírována z dokumentu do dokumentu
 - * šíření v office souborech
 - * dnes již ne moc rozšířené

Ransomware

- šifrování dat na pevném disku
- uzamčení systému s výhružnou zprávou
- požadavky výkupného (většinou krypto – BTC...)
- zrod v Rusku, dnes již mezinárodní
- distribuce pomocí trojského koně
- typy
 - file coders (CryptoLocker, CTB-Locker...)
 - scareware – falešné antiviry
 - screen lockers
 - doxing – vyhrožování osobními údaji
 - phising

Spacefiller virus

- hledá volné místo v kódu namísto připojení na konec programu
- nezvětší se velikost souboru – těžší identifikovatelnost

Trojský kůň

- přetvářka za užitečný software
- práce v utajení
- editace uživatelských souborů, blokace souborů, DDoS tool, zpomalení počítače
- řešení – reinstalace systému

DDoS attack

- distributed denial of service
- vyřazení služby/serveru zahlcením daty a požadavky
- znepřístupnění/zpomalení služby ostatním
- DoS – posílání z jednoho počítače, DDoS – posílání požadavků ze sítě počítačů

17.1.3 Prevence

- ověřené antiviry
- ověřování pochybných mailů a dalších zpráv
- kontrola aktualizací programů
- vyhýbání se pochybným stránkám
- vyhýbání se pirátství
- zálohy počítače
- použití bezpečnějšího systému

Antiviry

- programy bojující proti virů
- kontrola souborů a programů
- Kaspersky, Bitdefender, Norton, McAfee, ESET, Windows Defender, AVG...

17.2 FTP

- File Transfer Protocol
- protokol pro přenos souborů, komunikace se serverem
- využití TCP/IP
- přenos velkých souborů, základ infrastruktury firem...
- zabezpečení přes SSL/TLS
- přenos také přes SSH
- multiplatform
- možný přístup přes webový prohlížeč nebo prohlížeč souborů
- nevýhody – přístupové údaje přenášeny jako plain text

17.3 WWW

- World Wide Web
- informační systém pro sdílení dat a webových stránek
- www. – subdoména
 - používána hlavně při začátcích internetu, dnes přesměrovává na samotné stránky
- přenos dat přes HTTP(s)
- hostování souborů na web serverech
- přístup přes webové prohlížeče
- odkaz – adresa webové stránky
- cesta adres získávána přes DNS

18 WWW stránky / HTML

- stránky na WWW obsahující hyperlinky na sebe navzájem
- vytvářeny jednotlivci, firmami, vzdělávacími institucemi, státem...

18.1 HTML

- HyperText markup language
- základní jazyk pro vytváření webových stránek
- používáno společně s CSS a JS
- interpretace a rozložení textu, obrázků a dalších resources
- rendering webovým prohlížečem
- založeno na SGML (Standard General Markup Language), podobné XML
 - využití párových a nepárových tagů
- dnes HTML5

18.1.1 Historie

- 1989 – Tim Berners-Lee navrhl v CERNu internet-based hypertext system; 1990 – první webový prohlížeč a webový server
 - s Robertem Cailliau žádali, ale nedostali, funding
- 1991 – první veřejná specifikace HTML

18.2 Editorial

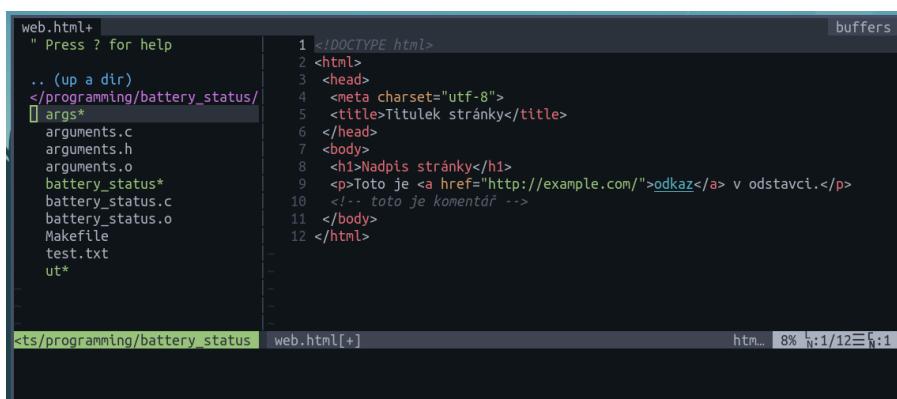
- programy pro vytvoření webových stránek
 - 2 typy
 - WYSIWYG
 - editory zdrojového kódu

18.2.1 WYSIWYG

- What You See Is What You Get
 - editory, kdy uživatel přímo vidí a edituje konečný výsledek
 - využití GUI
 - obchází nutnost editovat kód
 - [webnode.cz](#), [wix.com](#), [estránky.cz](#)
 - Content Management Systems (CMS)
 - systémy pro management obsahu stránek
 - WordPress, Joomla, Drupal

18.2.2 Editory zdrojového kódu

- obyčejné textové editory, většinou s přídavnými funkcemi
 - editace přímo zdrojového kódu (`.html` soubory)
 - pro výsledek nutno otevřít web v prohlížeči
 - otevření HTML dokumentu v prohlížeči
 - vytvoření lokálního serveru
 - Notepad++, VS Code, Atom, (Neo)vim, Emacs...



Obr. 18.1: Příklad textového editoru pro zdrojový kód (Neovim)

18.3 HTML dokument

- vždy nutnost základních tagů
 - pokud chybí, prohlížeč si je doplní
 - `<!DOCTYPE html>` – specifikace HTML formátu
 - `<html>...</html>` – začátek a konec HTML dokumentu
 - `<head>...</head>`
 - hlavička dokumentu
 - specifikace obsahu, který uživatel nevidí; metadata
 - `<body>...</body>`
 - tělo dokumentu
 - obsah prezentován uživateli

```

1 <!DOCTYPE html>
2 <html>
3   <head></head>
4   <body>
5     Hello World
6   </body>
7 </html>
```

18.3.1 Tagy v <head>...</head>

- <**title**>...</**title**> – Titulek stránky zobrazený v liště
- <**style**>...</**style**> – CSS styl stránky přímo v HTML
- <**link**> – link k externí resource (CSS, JS...)
- <**meta**> – specifikace metadat stránky
 - charset – HTML encoding
 - name – jméno specifikovaných metadat
 - content – specifikace hodnoty v name atributu
 - http-equiv – HTTP header pro informace/hodnotu v content atributu

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Metadata</title>
5     <link rel="icon" href="https://cdn-icons-png.flaticon.com
6       /512/1051/1051277.png">
7     <meta charset="utf-8">
8     <meta name="keywords" content="HTML">
9     <meta name="description" content="Web pro ukazku metadat">
10    <meta name="author" content="Adam Krska">
11  </head>
12  <body></body>
13 </html>
```

18.3.2 Tagy v <body>...</body>

Sekce

- <**h1**>...</**h1**>, <**h2**>...</**h2**>, <**h3**>...</**h3**>,... – úrovně nadpisů
- <**p**>...</**p**> – oddělení odstavce
- <**div**>...</**div**> – neviditelné rozdělení obsahu
- <**br**> – break line

Formátování

- <**b**>...</**b**> – tučný text
- <**i**>...</**i**> – italic text
- <**u**>...</**u**> – podtržený text
- <**tt**>...</**tt**> – strojní styl
- <**del**>...</**del**> – smazaný text
- <**font**>...</**font**> – specifikace fontu
 - atributy size, color, face...

Logické formátování

- <**strong**>...</**strong**> – důležitý text
- <**blockquote**>...</**blockquote**> – citace
- <**cite**>...</**cite**> – citace

- <**q**>...</**q**> – text s uvozovkami
- <**em**>...</**em**> – kurzíva
- <**a**>...</**a**> – odkaz
 - href – cesta odkazu
 - target – místo otevření odkazu

```

1 1<!DOCTYPE html>
2 2<html>
3 3  <head>
4 4    <meta charset="utf-8">
5 5  </head>
6 6  <body>
7 7    <h1>Hlavní nadpis</h1>
8 8    <h2>První podnadpis</h2>
9 9    <p>Obsah <i>prvního </i> odstavce.</p>
10 10   <p>Obsah <del>prvního</del> druhého odstavce.</p>
11 11   <h2>Druhý podnadpis</h2>
12 12   <p>Odstavec po <b>druhém </b>podnadpisu<br> na více řádích s odkazem na
13 13     <a href="https://google.com">Google</a>
14 14   </body>
15 15 </html>
```

Obrázky

- <**img**> – tag pro vložení obrázku
 - src – zdroj obrázku
 - width, height – rozměry
 - alt – alternativní text
 - loading, longdesc, ...

Tabulky

- <**table**>...</**table**> – specifikuje tabulku
- <**caption**>...</**caption**> – popis tabulky
- <**tr**>...</**tr**> – řádek tabulky
- <**th**>...</**th**> – buňka nadpisu
- <**td**>...</**td**> – buňka tabulky
 - colspan – šířka buňky ve sloupcích
 - align – zarovnání

Seznamy

- <**ul**>...</**ul**> – block seznamu
- <**il**>...</**il**> – item seznamu

Iframe

- <**iframe**>...</**iframe**> – zobrazení obsahu jiné stránky
 - width, height – velikost
 - src – URL zdroje
 - allow – povolení funkcí
 - loading, name...
- sociální sítě (Youtube, Facebook, Instagram), embedded mapy...

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>Tabulka</title>
6   </head>
7   <body>
8     <table frame="box" rules="all">
9       <tr>
10         <th colspan=2>Jmeno</th>
11         <th>Vek</th>
12       </tr>
13       <tr>
14         <td>Alena</td>
15         <td>Novakova</td>
16         <td>46</td>
17       </tr>
18       <tr>
19         <td>Jiri</td>
20         <td>z Podebrad</td>
21         <td>56</td>
22       </tr>
23       <tr>
24         <td>Evzenie</td>
25         <td>Slezakova</td>
26         <td>69</td>
27       </tr>
28     </table>
29   </body>
30 </html>
```

18.4 Prohlížeče HTML dokumentů

- prohlížení webových stránek ve webových prohlížečích
- stáhnutí všech potřebných dokumentů a scriptů a vyrenderování stránky
- Chromium based – Google Chrome, Edge, Brave, Vivaldi; ostatní – Firefox, Safari
- převaha chromium based prohlížečů
- kvůli cache a renderování dnes vyšší nároky na výpočetní výkon, hlavně při více webových oknech

19 Algoritmus

- konečná sekvence podrobně definovaných instrukcí řešící určitý problém
- výpočty, zpracování dat, automatické rozhodování
- možno použít stejný algoritmus s různými vstupními daty

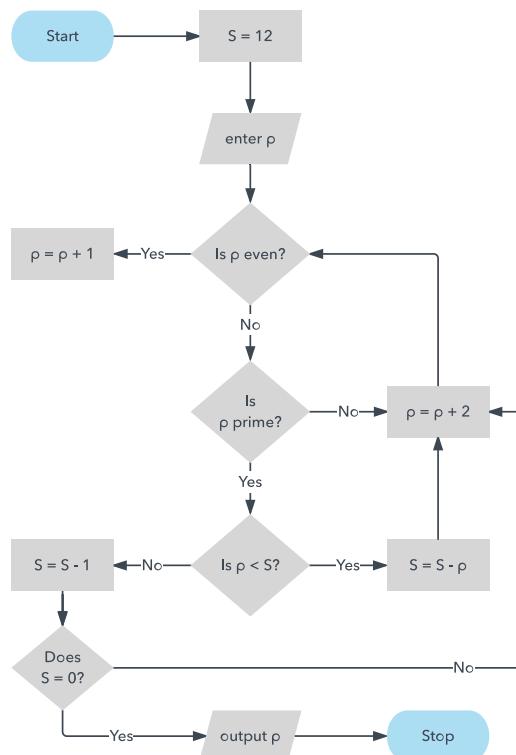
19

19.1 Vlastnosti algoritmu

- **správnost** – výsledek algoritmu musí být správný
- **resultativnost** – algoritmu v konečném čase dosáhne řešení a skončí
- **determinovanost** – v každém kroku jednoznačně určeno pokračování algoritmu
- **hromadnost** – možno algoritmus použít na řešení obecné úlohy
- **opakovatelnost** – se stejnými vstupními daty má algoritmus stejný výsledek

19.2 Zápis algoritmu

- slovní – slovní popis postupu v mluvené řeči
- grafický diagram – vývojový diagram, kroky zakresleny pomocí značek
- matematický – zápis pomocí veličin a rovnic
- v programovacím jazyku – vyjádření v programovacím jazyce pomocí funkcí, keywords, proměnných...



Obr. 19.1: Grafický diagram algoritmu

19.3 Rozdělení algoritmů

19.3.1 Podle implementace

Rekurzivní / iterativní

- rekurzivní
 - volá sám sebe dokud není splněna podmínka
 - rozdělení problému na části a postupné řešení
 - všechna potřebná data specifikována jako parametry funkce
- iterativní
 - opakování kódu v cyklech
 - použití externích datových struktur pro přenos dat mezi cykly
- každý problém možno přepsat z rekurze do iterace a naopak

Logický

- kontrolovaná logická dedukce
- specifikace komponentů jako axiomů, ty následně použity v dedukci

Sériové a Paralelní

- sériový alg.
 - spouští instrukce postupně jednu po druhé
- paralelní
 - umožněno spuštění instrukcí na více jádrech/procesorech najednou
 - rozdělení problému na symetrické či nesymetrické podproblémy
 - možno použít pouze při zpracování nezávislých dat
 - využití distribuovaných systémů

Deterministické a nedeterministické

- deterministické – přesné rozhodnutí v každém kroku
- nedeterministické – řešení problému hádáním, zavedení náhody

Přesné a přibližné (approximate)

- přesné alg. – řešení problému najdou přesně
- přibližné
 - řešení problému pouze approximují (deterministicky či náhodně)
 - řešení problému neřešitelných přesně či značně rychlejší řešení
 - Monte Carlo, Runge Kutta

19.3.2 Podle filozofie designu

Brute-force

- naivní metoda řešení problému
- vyzkoušení každé možnosti a nalezení té nejlepší

Divide and conquer (rozděl a panuj)

- opakování rozdělení problému do jednoho či více menších problémů, dokud nejsou problémy dostatečně malé na vyřešení
- typicky rekurzivní
- merge sort, binary search (decrease and conquer)

Vyhledávání a výčty

- problém vyjádřen grafem/stromem
- využití při hledání cest nebo např. hledání možnosti při hraní šach
- search algorithms...
- backtracking
 - více řešení vytvářeno postupně
 - zahození nevhodujících
 - vrácení k poslední validní hodnotě

Algoritmy s náhodou

- některé rozhodnutí (pseudo)náhodně
- hledání přibližných řešení
- Monte Carlo, Las Vegas algoritmus

Redukce komplexity

- transformace složitého problému na více známý, řešitelný problém
- cílem najít algoritmus, jehož složitost není přesahována složitostí algoritmu řešící známý problém
- např. medián neseřazeného seznamu
 1. seřazení seznamu – expensive
 2. nalezení mediánu – cheap

19.3.3 Optimalizační problémy

Lineární programování

- nalezení maxima/minima lineární funkce
- určitý počet proměnných na množině popsané soustavou lineárních nerovnic

Dynamické programování

- problém složen z podproblémů, které se překrývají (mají stejná řešení)
- zapamatování výsledků
- vyhnutí se přepočítávání řešení, které již byly vypočítány
- např. floyd-washallův algoritmus vyhledávání nejkratší vzdálenosti v grafu

The greedy method

- podobný dynamickému programování, ovšem pracuje již se získaným výsledkem
- snaha vylepšit řešení pomocí malý změn
- možno najít optimální řešení nebo taky jen lokální minimum
- Huffmanův strom, Kruskal, Prim, Sollin

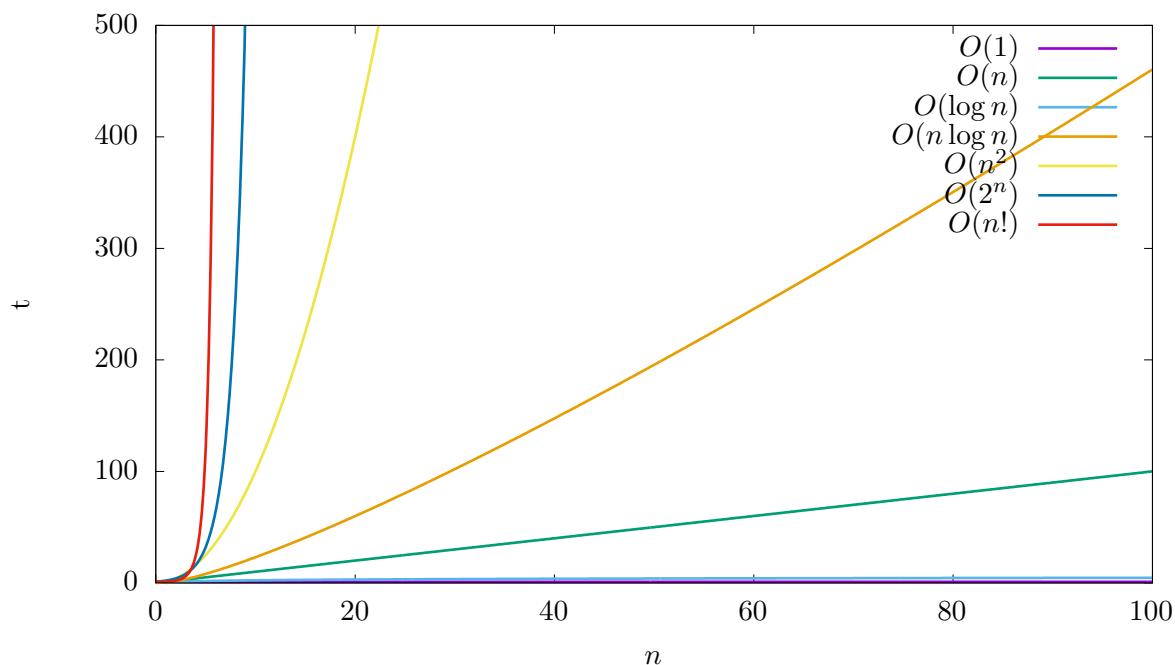
Heuristická metoda

- nalezení řešení blízké optimálnímu řešení
- v případech, že hledání optimálního řešení není praktické
- dostávání se blíže a blíže výsledku
- při nekonečném čase nalezení optimálního řešení
- local search, tabu search, genetické algoritmy

19.4 Komplexnost algoritmu

- vlastnost udávající výpočetní náročnost
- především sledován čas potřebný na spuštění, někdy i velikost potřebné paměti
- vyjádřeno pomocí Big-O notace
 - vyjádření nejhorské/nejdélší kalkulace
 - zápis pomocí matematické funkce; n – počet prvků
 - brán v potaz pouze nejrychleji rostoucí komponent

- * z předpokladu $n \rightarrow \infty$
- * $O(n^2 + n) = O(n^2)$
- související veličiny
 - big-omega $\Omega(n)$ – nejlepší případ
 - bit-theta $\Theta(n)$ – průměrný případ



Obr. 19.2: Porovnání průběhu jednotlivých funkcí

19.4.1 Příklady komplexnosti

- konstantní čas – $O(1)$ – získání array elementu, hledání v hashmapě
- logaritmický čas – $O(\log n)$ – binary search
- lineární čas – $O(n)$ – projití všech prvků v poli
- polynomiální čas – $O(n^2), O(n^3)$ – bubble sort ($O(n^2)$)
- exponenciální čas – $O(x^n)$ – brute-force search

20 Vývojový diagram, programovací jazyky

20.1 Vývojový diagram

- grafické vyjádření procesu (algoritmus, postup práce)
- reprezentace kroků pomocí tvarů a šipek
- analýza procesu, návrh, dokumentace
- programy – MS Visio, Lucid chart, Inkscape
- preferovaný směr shora dolů, zleva doprava

20.1.1 Start a konec programu

- vyznačeno oválem
- šipky ukazují směr pokračování procesu

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	
Quicksort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
Mergesort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Timsort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Heapsort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$
Bubble Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Tree Sort	$\Omega(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$
Shell Sort	$\Omega(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
Bucket Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
Radix Sort	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
Counting Sort	$\Omega(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$
Cubesort	$\Omega(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$

Obr. 19.3: Porovnání složitosti řadících algoritmů

20.1.2 Dílčí krok

- obdélník
- popis dílčí akce, kroku



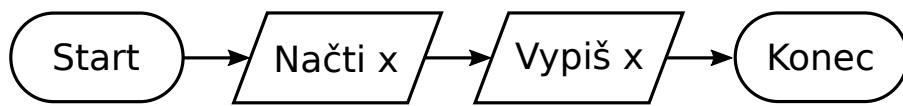
Obr. 20.1: Diagram jednoho kroku

20.1.3 Podprogram

- obdélník se svislými čárami
- více kroků označených jedním symbolem
- vyjádření funkce

20.1.4 Vstup/výstup

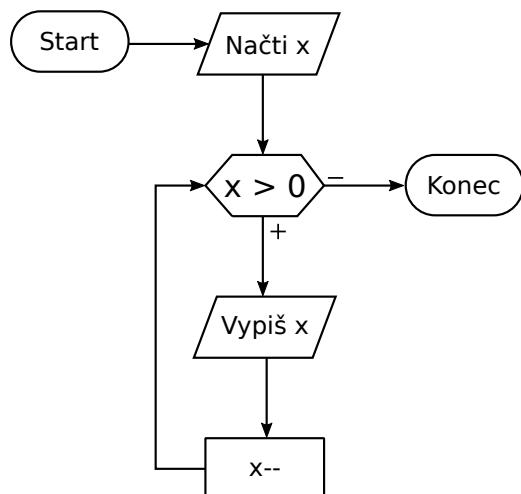
- rovnoběžník
- komunikace s uživatelem



Obr. 20.2: Diagram se vstupem a výstupem

20.1.5 Cykly

- šestiúhelník
- probíhají pouze za platné podmínky, jinak se pokračují dál

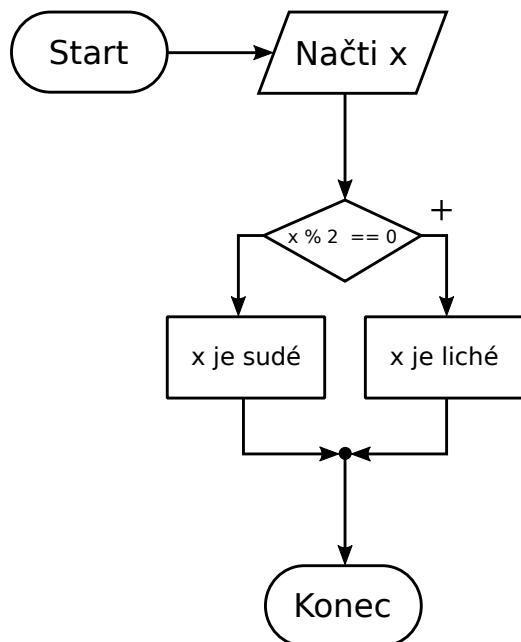


Obr. 20.3: Diagram cyklu

20

20.1.6 Podmínka

- kosočtverec
- pokud podmínka, splní se kroky



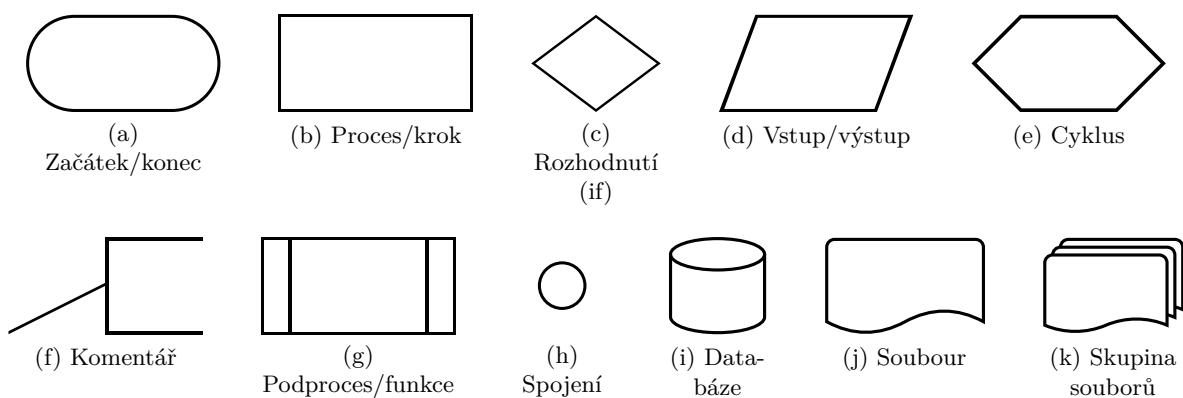
Obr. 20.4: Diagram podmínky

20.1.7 Spojovací značka

- kruh
- spojení více toků procesu do jednoho

20.2 Programovací jazyky

- způsob zapsání kódu a jeho následné použití v počítači
- rozdělení na vyšší a nižší



20.2.1 Nižší programovací jazyky

- strojový kód či mu velice blízko
- jednoduché na převod do binárního záznamu
- základ vyšších jazyků
- přímá kontrola nad registry, pamětí, pointery...
- strojový kód, assembly, občas za něj považované i C

20

20.2.2 Vyšší programovací jazyky

- použití přirozeného jazyku
- velké použití abstrakcí
- překládány do nižších jazyků
- velký počet předdefinovaných funkcí
- menší kontrola nad samotným hardwarem

20.2.3 Procedurální vs neprocedurální

- procedurální/imperativní
 - popis výpočtu pomocí příkazů, určení přesného postupu
 - Fortran, ALGOL, BASIC, C
- neprocedurální
 - specifikace cíle namísto postupu získání cíle
 - SQL, PROLOG, LIPS

20.2.4 Strukturované vs objektově orientované

- strukturované – jeden vstup a výstup, vytvoření algoritmu z řídíc struktur a funkcí
 - C, Pascal
- objektově orientované – využití objektů, jejich funkcí, dědičnosti...
 - Java, C#
- kombinované – strukturované programování s podporou objektů
 - Python, JavaScript, C++

20.2.5 Funkcionální vs logický

- funkcionální
 - využití matematických funkcí
 - využití lambda funkcí
 - SQL, Mathematica, Haskell
- logický
 - použití matematické logiky k programování
 - PROLOG

20.2.6 Kompilované vs interpretované

- komplikované
 - jazyky přeloženy do binárního souboru, který je následně spuštěn
 - C, C++, Rust
- interpretované
 - interpreter interpretuje jazyk v reálném čase
 - flexibilnější, ale pomalejší
 - chyby jsou zachyceny až při spuštění kódu
 - Python, JavaScript, Java, R

20.2.7 Statické vs dynamické

- statické programovací jazyky
 - typ proměnných je znám v čase komplikace
 - typ proměnných zadán programátorem (Java, C, C++) nebo odvozen komplikátorem (Haskell, Rust)
- dynamicky psané programovací jazyky
 - typ dat specifikován pro konkrétní hodnotu, ne pro proměnnou
 - možno změnit typ proměnné v průběhu programu
 - Python, JavaScript, PHP, Ruby

21 Kompilace, linkování, proměnné

21.1 Překlad zdrojového kódu na stroj

- není možné zdrojový kód přímo spustit → potřeba převést na strojový kód
- komplikace a linkování

Zdrojový kód → cílový kód	Překladový nástroj	Poznámka
jednotka1.c → jednotka1.asm, jednotka2.c → jednotka2.asm, ...	kompilátor jazyka C	překlad do assembly či dnes přímo do objektového kódu
jednotka1.asm → jednotka1.obj, jednotka2.asm → jednotka2.obj, ...	assembler	vytvoření objektového kódu
jednotka1.obj + jednotka2.obj + ... + knihovna1.lib + knihovna2.lib + ... → output	linker	spojení objektových kódů a kódu z knihoven do spustitelného kódu

Tab. 21.1: Průběh komplikace

21.1.1 Kompilace

- vytváření objektových souborů z zdrojových souborů
- prováděno překladačem (komplikátor)
 - gcc, clang, intel c++ compiler, mono, Gc, javac OpenJDK...
- několik kroků, např
 - line reconstruction
 - preprocessing
 - lexikální analýza
 - syntaktická analýza / parsování

- sématická analýza
- další – optimalizace, analýza, vytváření kódu...
- typy
 - source-to-source – překlad high-level jazyka do high-level jazyka
 - bytecode – komplikace do assembly teoretického stroje
 - just-in-time compilers – komplikace během spuštění; Python, JavaScript, Java, .NET...
 - hardware compilers
 - assembler – převod assembly do strojového kódu
 - ...
- spuštění
 - z IDE
 - ruční spuštění příkazů – cmd, bash
 - automatické systémy – make, cmake

21.1.2 Linkování

- seskupení objektových souborů a vytvoření jednoho spustitelného souboru, knihovny nebo objektu
- prováděno linkerem
- hledání dependencies, importování libraries, nastavení entry pointu...
- statické a dynamické linkování
 - statické – všechn potřebný kód v rámci executable, větší velikost souboru
 - dynamické – některé symboly nalezeny až při spuštění programu v operačním systému, menší velikost, potřeba nainstalovat externí knihovny

22

21.2 Proměnné

- způsob uložení dat v programu
- na hodnotu ukazuje symbolické jméno
- specifikace typu dat – int, string, float, char, bool, double...
- konstanty – keyword `const` – nelze měnit v průběhu programu
- `signed`, `unsigned` – specifikace čísel s/bez znamének, keyword před data typem
- další data typy, než ty zmíněny v tabulce 21.2, definovány ve standardních C knihovnách

21.2.1 Specifikování proměnné v C

- formát „<datatype> <variable name>;“
 - `int i;`
 - `char c, ch;`
 - `float f=0.5f, ch;`

21.2.2 Uchování dat v programu

- při specifikování proměnné se alokuje místo v paměti
- čtení/zápis dat následně z paměti na místo adresy
- pointer – adresa proměnné

22 Vstup/výstup, podmínky

22.1 Vstup/výstup do/z programu

- tzv. IO (input output) operace
- způsob komunikace programu se systémem, uživatelem a dalšími programy
- funkce poskytnutý knihovnou `stdio.h`

Typ	Vysvětlení	Min. velikost (bit)	Formátování
<code>char</code>	nejmenší adresovatelná jednotka uchovávající znak; vnitřně číslo	8	<code>%c</code>
<code>signed char</code>	stejně jako <code>char</code> , garance znaménka, hodnoty $\langle -127, +127 \rangle$	8	<code>%c</code>
<code>unsigned char</code>	stejně jako <code>char</code> , garance bez znaménka, hodnoty $\langle 0, 255 \rangle$	8	<code>%c</code>
<code>short, short int</code>	krátký integer typ	16	<code>%hi, %hd</code>
<code>int, signed, signed int</code>	uchování čísla, rozšíření počtu bitů v operačních systémem se širším busem	16	<code>%i, %d</code>
<code>unsigned, unsigned int</code>	uchování čísla bez znaménka	16	<code>%u</code>
<code>long, long int</code>	delší <code>int</code> , v moderních systémech shodné s <code>int</code>	32	<code>%li</code> (signed), <code>%lu</code> (unsigned)
<code>long long, long long int</code>	delší <code>long</code>	64	<code>%lli, %llu</code>
<code>float</code>	reálné číslo s desetinnými čísly, na většině systémech 32 bitů		<code>%f</code>
<code>double</code>	větší reálné číslo s desetinnými čísly, na většině systémech 64 bitů		<code>%lf</code>
<code>long double</code>	velké reálné číslo s desetinnými čísly; 80, 96 nebo 128 bitů		<code>%Lf</code>

Tab. 21.2: Data typy v C standartu

22.1.1 Uživatelský vstup

- čtení vstupu z `stdin`¹
- funkce `getchar(<variable>);`, `gets(<variable>);`, `scanf(<format>, <pointer>);`
 - `getchar` – načtení jednoho znaku
 - `gets`
 - * načtení stringu s mezerami, nelze číst čísla
 - * konec inputu novým řádkem nebo EOF
 - * unsafe, chybí ochrana před buffer overflow
 - `scanf` – načtení vstupu v zadáném formátu, konec inputu mezerou, `\n` nebo EOF

```

1 #include <stdio.h>
2
3 int main(){
4     // init variables
5     char character, word[20], sentence[20];
6     int number;
7
8     // different input methods
9     getchar(character);
10    gets(sentence);
11    scanf("%s %d", &word, &number);
12
13    return 0; // exit
14 }
```

Kód 2: Načtení vstupu od uživatele

22

22.1.2 Výstup programu

- výpis textu do `stdout` nebo `stderr`
 - `stdout` – standard stream pro output
 - `stderr` – standard stream pro errory
- funkce `putchar`, `puts`, `printf`, `fwrite`
 - `putchar(<variable>);` – vypsání jednoho znaku
 - `puts(<variable>);`
 - * funkce z knihovny `inlistc`
 - * prostý výpis proměnné do `stdout` bez formátování
 - * na konci automaticky `\n`
 - `printf(<string and format>, <variable>);`
 - * interpretace prvního stringu jako formátu
 - * následné proměnné specifikují hodnotu dat ve formátu
 - * nekončí automaticky novým řádkem
 - `fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);`
 - * `*ptr` – buffer na vypsání, `size`, `nmemb` – délka a počet dat
 - * zápis binárních dat do souboru nebo streamu

22.1.3 Formát printf/scanf

- využití formátovací značek pro definici formátu outputu – `%d`, `%i`, `%u`, `%c`...
- zápis do stringu
- formátovací značky různé pro každý data typ (viz otázka 21)

¹standard stream pro vstup v terminálu

```

1 #include <stdio.h>
2
3 int main() {
4     int a,b,c;
5
6     printf("Zadejte 3 cisla: ");
7     scanf("%i %i %i", &a, &b, &c);
8
9     printf("Sumy kazde dvojice: %i %i %i", a+b, b+c, a+c);
10
11    return 0;
12 }

```

Kód 3: Použití printf a scanf

22.2 Podmínky

- způsob spouštění kódu za pomocí podmínek
- příkaz **if** (rule){...} **else** {...}
- pokud je podmínka splněna, je spuštěn kód v bloku; pokud není splněna, blok je přeskočen
- else** blok spuštěn pouze za nedodržení podmínky
- podmínka – boolenový výraz

22.2.1 Booleovské výrazy

- v C použít **int**, případně **bool** ze **stdbool.h**
- true** – nenulová hodnota (nejčastěji 1), **false** – 0

Znak	Význam
==	rovnost
!=	nerovnost
<	menší než
<=	menší rovno
>	větší než
>=	větší rovno
&&	AND
	OR
!	negace

Tab. 22.1: Logické operátory v C

```

1 #include <stdio.h>
2
3 int main () {
4     int i;
5     printf("Zadejte cislo: ");
6     scanf("%d", &i);
7     if (i % 2 == 0) puts("Sude");
8     else puts("Liche");
9     return 0;
10 }

```

Kód 4: Příklad programu s podmínkou a if

Operátor	Směr vyhodnocení
! ++ -- - +	←
* / %	→
+ -	→
< <= >= >	→
== !=	→
&&	→
	→
? :	←
= += -= *= ...	→
,	→

Tab. 22.2: Priorita vyhodnocování logických výrazů

22.2.2 Kombinace podmínek

- kombinování podmínek za pomocí logických operací `&&` a `||`
- první vyhodnocení podmínek, následně vyhodnocení kombinací (viz tab. 22.2)
- vyhodnocení zleva doprava
- logické výrazy možno skládat, závorky pro přednost...
- pokud v AND je první argument 0, další hodnoty již nejsou vyhodnoceny

```

1 | int compare(a,b,c) {
2 |   if (a == 0) return 0;
3 |   if (a < b) return -1;
4 |   if (a >= b) return 1;
5 |   if ((a == c) && (b < c) || !(c == 1)) return 2;
6 |   return 3;
7 |

```

Kód 5: Příklady boolenových operací

23 Cykly

- způsob opakování kódu
- „iterace“ – jedno spuštění kódu
- 3 typy
 - `for` – opakování kódu n -krát
 - `while` – spuštění kódu za platné podmínky
 - `do while` – vykoná kód a za platné podmínky jej spustí znovu
- související keywords
 - `break`; – ukončí cyklus, pokračuje kódem po bloku cyklu
 - `continue`; – ukončí započatou iteraci, skočí na další iteraci cyklu

23.1 For loop

- cyklus používán pro vykonání známého počtu opakování
- použití – procházení pole, vykreslování, vypisování řádků...
- syntaxe – `for(init; statement; increment){...}`
 - `init` – výraz vyhodnocen za začátku, většinou zavedení proměnné indexu
 - `statement` – podmínka, za které se iterace spustí
 - `increment` – příkaz po vykonání iterace, většinou přičtení k indexu
 - výrazy nemusí být vůbec uvedeny či mohou být jinde, nicméně uvedený formát je standard

```

1 | for (int i = 1; i <= 100; i++) {
2 |   if (i % 2 == 0) printf("Number %i is even\n", i);
3 |   else printf("Number %i is odd\n", i);
4 |

```

Kód 6: Použití for cyklu

23.2 While cyklus

- syntaxe – `while(condition)`
- iteruje, dokud je podmínka platná
- testování podmínky před cyklem
- použití, pokud je podmínka během cyklu změněna
- možnost zapsat jako `for(;condition;)`

```

1 | int i = 1;
2 | while (i <= 100) {
3 |   if (i % 2 == 0) printf("Number %i is even\n", i);
4 |   else printf("Number %i is odd\n", i);
5 |   i++;
6 |

```

Kód 7: Použití while cyklu

23.3 Do while cyklus

- podobný jako while cyklus
- testuje podmínu až na konci
- zaručeno, že proběhne alespoň jednou
- vhodný na vstup několika hodnot (zadání další hodnoty záleží na předchozí)

```

1 | int i = 1;
2 | do {
3 |   if (i % 2 == 0) printf("Number %i is even\n", i);
4 |   else printf("Number %i is odd\n", i);
5 |   i++;
6 | } while (i <= 100);

```

Kód 8: Použití do while cyklu

23.4 Vnořený cyklus

- cyklus v cyklu
- způsob opakování kódu v rámci jedné iterace
- vykreslování 2D obrazců, procházení 2D pole, určení sumy pro n členů...

24 Práce se soubory, struktury, pole

24.1 Práce se soubory

- typy souborů
 - textové soubory – soubory obsahující plain text
 - binární soubory – obsahují binární data nemožné otevřít v obyčejném editoru
- operace

```

1 #include <stdio.h>
2
3 int main() {
4     int n, sum;
5     printf("Type n: ");
6     scanf("%i", &n);
7
8     for (int i = 1; i <= n; i++) {
9         sum = 0;
10        for (int j = 1; j <= i; j++) sum += j;
11        printf("1 - %2i: %2i\n", i, sum);
12    }
13    return 0;
14 }
```

Kód 9: Příklad vnořeného cyklu – Suma od 1 do n

24

```

1 #include <stdio.h>
2
3 int main () {
4     int size, num;
5
6     printf("Type in size (odd number) and number of images: ");
7     scanf("%d %d", &size, &num);
8
9     for (int y = 0; y < size; y++) {
10        for (int i = 0; i < num; i++) {
11            for (int x = 0; x < size; x++) {
12                if (x == size/2 || y == size/2 || x == y || x == size-y-1)
13                    putchar('*');
14                else
15                    putchar('_');
16            }
17        }
18        putchar('\n');
19    }
20 }
```

Kód 10: Vypisování 2D obrazců za pomocí cyklů

- vytvoření souboru
- otevření existujícího souboru
- zavření souboru
- čtení a zápis dat do souboru
- proměnná souboru – pointer na souboru `FILE *fptr`

24.1.1 Otevírání souboru

- syntaxe `fptr = fopen("filepath", "mode");`
 - `"filepath"` – cesta k souboru
 - `"mode"` – mód otevření souboru
- otevření datového streamu
- file open modes
 - `r`, `w`, `a` – čtení, zápis a připsání na konec
 - `r+`, `w+`, `a+` – čtení a zápis (soubor musí existovat), čtení a zápis (přepsání nebo vytvoření)
 - `b` – binární soubor (`rb`, `rb+`, `wb`, `wb+`, `ab`, `ab+`)

24.1.2 Zavření souboru

- `fclose(fptr);`
- zavření souboru / streamu po dokončení operací, připsání EOF
- pointer zahozen

24.1.3 Čtení ze souboru

- `fscanf(fptr, "format", &variable)` – načtení textového souboru
 - `fptr` – file pointer
 - `"format"` – formát textu
 - `&variable` – adresa proměnné pro zapsání dat
- `fread(&variable, dataSize, dataCount, fptr)` – načtení textového souboru
 - `dataSize` – velikost dat v bytech (často `sizeof(datatype)` – `sizeof(int)...`)
 - `dataCount` – počet dat o velikosti `dataSize`, použito při načítání pole

24

24.1.4 Zápis do souboru

- `fprintf(fptr, "format", variable)` – vypsání textu do souboru
- `fwrite(&variable, dataSize, dataCount, fptr)` – zápis binárních dat
- připsání/zapsání/přepsání dat do souboru

24.2 Struktury

- keyword `struct`
- proměnné/struktury uchovávající více dat/proměnných v sobě
- zjednodušení a seskupení dat
- příklad – struktura Kniha
 - název
 - autor
 - žánr
 - rok vydání
- různé metody definovaní
 - `struct StructName {...};`
 - definuje `struct` name
 - inicializace proměnné – `struct StructName variableName;`
 - `struct StructName {...} variableName;`

```

1 #include <stdio.h>
2
3 /* Calculate average of values in input.csv and write the output to
4  output.csv*/
5
6 const char inputFilePath[] = "./input.csv";
7 const char outputPath[] = "./output.csv";
8
9 int main () {
10     int values[3];
11
12     FILE *inputFile = fopen(inputFilePath, "r");
13     if (inputFile == NULL) {
14         puts("Input file not found!");
15         return(1);
16     }
17
18     fscanf(inputFile, "%d,%d,%d", &values[0], &values[1], &values[2]);
19     fclose(inputFile);
20
21     float avg = (values[0]+values[1]+values[2])/3.0f;
22
23     FILE *outputFile = fopen(outputFilePath, "w");
24     fprintf(outputFile, "Average: %f\n", avg);
25     fclose(outputFile);
26
27     return 0;
28 }
```

Kód 11: Načítání a zapisování do souboru

```

1 struct book {
2     char name[50];
3     char author[50];
4     char genre[50];
5     int year;
6 };
```

Kód 12: Příklad struktury

- * definuje variableName typu **struct StructName**
- * zkrácení předchozí definice do jednoho příkazu, nevhodné pro opakování využití v kódu
- **typedef struct {..} DatatypeName; / typedef struct StructName {..} DatatypeName;**
 - * inicializace proměnné – datatypeName variableName;
 - * definice „vlastního datotypu“
 - * obchází potřebu při inicializaci psát **struct**
 - * optional structName
- inicializace proměnné s hodnotami – **struct StructName structVar = insideVar1, insideVar2;**
- přístup k hodnotám – **structVar.insideVar1 = value;**

```

1 // -- Define struct Point --
2 struct Point {
3     int x,y;
4 };
5
6 struct Point p1 = {0,0}; // init p1 of type Point
7
8 // -- Declare p2 of type Point --
9 struct Point {
10    int x,y;
11 } p2 = {1,1};
12
13 // -- Define struct using typedef --
14 typedef struct {
15     int x,y;
16 } Point;
17
18 Point p3 = {-1,2}; // declare Point

```

Kód 13: Různé způsoby definování struktur

24.3 Pole

- jeden dlouhý list podobných hodnot o stejném data typu
- v paměti hodnoty uloženy hned po sobě
- možno udělat pole polí – 2D pole
- příklad – **int nums[] = {1,2,3,4,5};, char name[] = "David";, float values[10];**
- velikost pole předem daná – statické pole (v některý prog. jazycích i dynamická)
- začátek číslování od 0

name[0]	name[1]	name[2]	name[3]	name[4]
'D'	'a'	'v'	'i'	'd'

Tab. 24.1: Obsah **char** pole

24.3.1 Hodnota položky

- syntax – **arrayName[index]**
- čtení – **foo = arrayName[index];**
- zápis – **arrayName[index] = foo;**

24.3.2 Pole struktur

- pole složené ze struktur
- hodnota pole `arrayName[index].insideVar`

```

1 #include <stdio.h>
2
3 typedef struct {
4     float x,y,z;
5 } Point;
6
7 Point points[] = {
8     {1,4,2},
9     {4,2,3},
10    {2,3,6},
11    {3,1,2}
12 };
13
14 int main () {
15     for (int i=0; i<sizeof(points)/sizeof(points[0]); i++){
16         printf("x: %.2f, y: %.2f, z: %.2f\n",
17                points[i].x, points[i].y, points[i].z);
18     }
19 }
```

Kód 14: Vypsání pole struktur

25 Práce s polem, řadící algoritmy

25.1 Práce s polem

25.1.1 Inicializace pole

- v C vždy nutno specifikovat délku
 - specifikováním počtu – `int nums[5];`
 - specifikace listu dat – `int nums[] = {1,2,3,4,5};`
 - specifikace obou – `int nums[5] = {1,2,3,4,5};`

25.1.2 Získání hodnoty z pole

- specifikace jména pole a indexu
 - indexování pole začíná na 0
- `int value = nums[3];`
- `int i = 0; int value = nums[i];`

25.1.3 Zápis do pole

- hodnota buňky rovna zapisované hodnotě
- `nums[4] = 6;`
- `int value = 7; int index = 2; nums[index]=value`

25.1.4 Procházení pole

- využití `for` smyčky

```

1 #include <stdio.h>
2
3 int grades[] = {1,3,4,2,1,3,2,2,3,1};
4
5 int main() {
6     int arrayLength = sizeof(grades)/sizeof(grades[0]);
7     int sum = 0, min = 5, max = 1;
8
9     for (int i = 0; i < arrayLength; i++) {
10         int grade = grades[i];
11         sum += grade;
12         if (grade < min) min = grade;
13         if (grade > max) max = grade;
14     }
15
16     printf("Average grade: %.2f\n", (float)sum/arrayLength);
17     printf("Worst grade: %d\n", max);
18     printf("Best grade: %d\n", min);
19     return 0;
20 }
```

Kód 15: Příklad procházení pole

25.2 Vícerozměrná pole

- pole o více rozměrech, pole polí
- každý prvek pole je další pole
- `int matrix[height][width];`
- stejné čtení a zápis dat jako při jedné dimenzi

```

1 /* Usual declaration */
2 int abc[2][4] = {
3     {1, 2, 3, 4},
4     {5, 6, 7, 8}
5 };
6 /* Valid declaration*/
7 int abc[2][2] = {1, 2, 3, 4, 5, 6, 7, 8};
8 /* Valid declaration*/
9 int abc[][] = {1, 2, 3, 4, 5, 6, 7, 8};
10 /* Invalid declaration - you must specify second dimension*/
11 int abc[][] = {1, 2, 3, 4, 5, 6, 7, 8};
12 /* Invalid because of the same reason mentioned above*/
13 int abc[2][] = {1, 2, 3, 4, 5, 6, 7, 8};
```

Kód 16: Inicializace vícerozměrného pole

25.3 Řadící (sorting) algoritmy

- algoritmy zajišťující uspořádání dané sady (pole, seznam...)
- nejčastější numerické a abecední řazení
- v dnešní době důležitá efektivnost řadících algoritmů na velkých datech
- podmínky výstupu
 - výstup je monotónní – každý prvek není větší/menší než den předešlý (dle zadaných pravidel)
 - výstup je permutace – jsou zachovány všechny původní prvky
- pro optimální rychlosť data uložena ve strukturách s random přístupem, ne sekvenčním

		Druhý index j			
		matrix[0][0]	matrix[0][1]	matrix[0][2]	matrix[0][3]
První index i	matrix[1][0]	matrix[1][1]	matrix[1][2]	matrix[1][3]	
	matrix[2][0]	matrix[2][1]	matrix[2][2]	matrix[2][3]	
	matrix[3][0]	matrix[3][1]	matrix[3][2]	matrix[3][3]	

Tab. 25.1: Zobrazení 2D pole

25.3.1 Klasifikace algoritmů

- výpočetní složitost
 - nejlepší, nejhorší, průměrné
 - typicky dobrá složitost $O(n \log n)$, parallel sort $O(\log^2 n)$, špatný $O(n^2)$
 - optimální paralelní složitost $O(\log n)$
- využití paměti
 - některé algoritmy řadí v místě – $O(1)$
- rekurzivita – rekurzivní vs iterativní algoritmy, některé obojí (merge sort)
- stabilita – při souhlasných hodnotách je zachováno vzájemné pořadí udané vstupem
- metoda – vkládání, výměna, selekce, spojování...
 - výměnné – bubble sort, quick sort
 - selekce – cycle sort, heap sort
- sériový vs paralelní
- adaptivní – zda-li předtřídění ovlivní rychlosť algoritmu
- online – sort schopný třídit stálý tok dat, např. insertion sort

Jméno	Český název	Nejlepší	Průměrné	Nejhorší	Paměť	Stabilní	Metoda
Quicksort	rychlé řazení	$n \log n$	$n \log n$	n^2	$\log n$	Ne	záměna
Merge sort	řazení slučováním	$n \log n$	$n \log n$	$n \log n$	n	Ano	slučování
Heapsort	řazení haldou	$n \log n$	$n \log n$	$n \log n$	1	Ne	výběr
Insertion sort	řazení vkládáním	n	n^2	n^2	1	Ano	vkládání
Selection sort	řazení výběrem	n^2	n^2	n^2	1	Ne	výběr
Bubble sort	bublinkové řazení	n	n^2	n^2	1	Ano	záměna

Tab. 25.2: Porovnání řadících algoritmů

Seznam obrázků

3.1	Příklad kombinačního obvodu	14
4.1	SR latch za použití NOR gate.	15
4.2	SR latch s enable vstupem	16
4.3	D latch s enable signálem	17
4.4	JK latch	17
4.5	T latch	17
4.6	Rising edge obvod	18
4.7	Symbolický nákres konečného automatu	19
6.1	Funkce laserové tiskárny	25
7.1	Zabalení TCP dat	29
7.2	Paket dat	29
7.3	Topologie sítě při TCP přenosu	30
7.4	Funkce NATu	31
8.1	Příklad rozdělí sítě do dvou podsítí	35

```

1 #include <stdio.h>
2 #include <stdbool.h>
3
4 // Swap pointers
5 void swap(int *xp, int *yp)
6 {
7     int temp = *xp;
8     *xp = *yp;
9     *yp = temp;
10 }
11
12 // Function to implement bubble sort
13 void bubbleSort(int arr[], int n) {
14     for (int i = 0; i < n-1; i++) {
15         bool swapped = false;
16         for (int j = 0; j < n-i-1; j++) {
17             if (arr[j] > arr[j+1]) {
18                 swap(&arr[j], &arr[j+1]);
19                 swapped = true;
20             }
21             if (!swapped) break; // Break if already sorted
22         }
23     }
24
25 // Function to print an array
26 void printArray(int arr[], int size)
27 {
28     for (int i=0; i < size; i++)
29         printf("%d ", arr[i]);
30     printf("\n");
31 }
32
33 // Driver program to test above functions
34 int main() {
35     int arr[] = {11, 64, 34, 25, 12, 22, 90};
36     int n = sizeof(arr)/sizeof(arr[0]);
37     bubbleSort(arr, n);
38     printf("Sorted array: \n");
39     printArray(arr, n);
40     return 0;
41 }
```

Kód 17: Implementace bubble sortu v C

8.2	Průběh Dijkstrova algoritmu na malém grafu	36
14.1	Porovnání RGB a CMYK modelu	43
14.2	Porovnání HSL a HSV módu	44
14.3	Porovnání různých barevných hloubek	45
14.4	Porovnání různých hodnot DPI a stejně velkém obrazu	45
18.1	Příklad textového editoru pro zdrojový kód (Neovim)	49
19.1	Grafický diagram algoritmu	53
19.2	Porovnání průběhu jednotlivých funkcí	56
19.3	Porovnání složitosti řadících algoritmů	57
20.1	Diagram jednoho kroku	57
20.2	Diagram se vstupem a výstupem	57
20.3	Diagram cyklu	58
20.4	Diagram podmínky	58

Seznam tabulek

3.1	Implikace	10
3.2	Disjunkce	10
3.3	Implikace	10
3.4	Ekvivalence	11
3.5	Negace	11
4.1	Pravdivostní hodnoty SR latch (Last – Last state – poslední stav)	16
4.2	Pravdivostní hodnoty D latch	16
4.3	Pravdivostní hodnoty JK latch	17
4.4	Pravdivostní hodnoty T latch	17
7.1	Vrstvy tvorící OSI model	28
7.2	Porovnání TCP a OSI modelu	31
8.1	Příklad zařízení s IP adresou 192.0.2.130/26	34
21.1	Průběh komplikace	60
21.2	Data typy v C standartu	62
22.1	Logické operátory v C	64
22.2	Priorita vyhodnocování logických výrazů	65
24.1	Obsah <code>char</code> pole	70
25.1	Zobrazení 2D pole	73
25.2	Porovnání řadících algoritmů	73