1. **Informácia, údaj. Reprezentácia údajov v počítači. Základné Jednotky v informatike - bit, Byte, kB, MB, GB, Kbps. Digitalizácia textu, obrázkov a zvuku na počítači. Kódovanie, uchovávanie a spracovanie informácií v PC. Kódovanie textu, kódová tabuľka.**

Informatika je vedou, ktorá sa zaoberá, organizáciou, spracovávaním, uchovávaním a prenosom údajov, dát, informácií. Informácia ako základný pojem v informatike úzko súvisí s pojmami údaj a dáta.
Pod **informáciou** rozumieme ľubovoľnú správu, údaj, príkaz, dáta - (hodnoty, znaky), inštrukcie, príkazy, povely a pod, ktoré prinášajú **nové poznatky**. Údaj nie je nositeľom nových poznatkov. Za informácie považujeme tiež podnety a vnemy prijímané a vysielané živými organizmami.
Údaje sú zachytené pomocou abecedných a číslicových znakov. Reprezentácia údajov môže byť:
1/ *digitálna - nespojitá (skokovitá*), napr. digitálne zobrazovanie času,
2/ *analógová – spojitá*, napr. ručičkové hodinky bez skokovej sekundovej ručičky, kolísanie napätia v sieti
Z pohľadu výpočtovej techniky je základnou jednotkou 1bit (BInary digiT), bit (binary digit) ako zákl. jednotka informácie., čo je množstvo informácie vyjadrené hodnotou 0 alebo 1. Väčšie jednotky sú :

8 bitov = 1byte =1B - jedna slabika
^{10}B = 1024B = 1kB kilo
^{10} kB = 1024kB = 1MB mega
^{10}MB = 1024MB = 1GB giga

Bit je veličina, označujúca, či v danom okamihu bol alebo nebol zaregistrovaný elektrický impulz (signál), vyjadrujúci impulz jednotkového napätia.
Postupnosť 8 bitov, čiže 1 byte slúži na uloženie informácie o 1 znaku (písmene, číslici atď.). Napríklad 01000001 je znak "A". Hodnoty znakov sú dané tabuľkou ASCII (7bitová), alebo tabuľkami národných kódov (Latin2, Kamenických), ktoré sú založené na 8bitových kódovaniach.

Znak	Hodnota ASCII v desiatkovej sústave
"0"	48
"A"	65
"a"	97

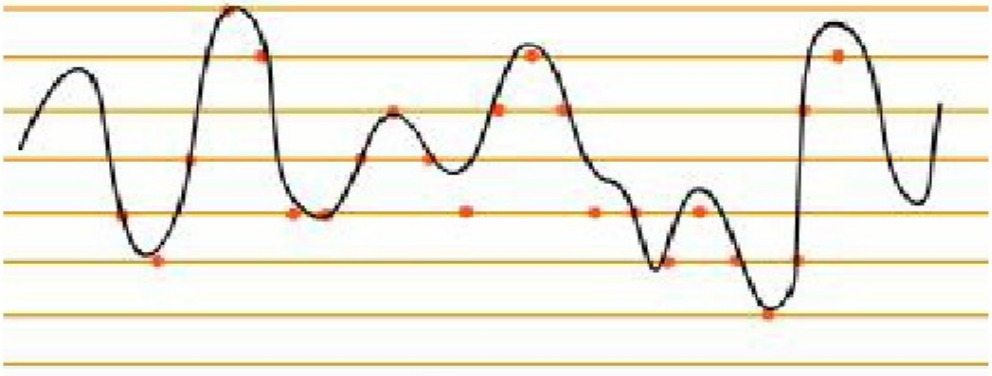
digitalizácia - prevod spojitých (analógových) údajov do binárneho kódu, v ktorom dokáže pracovať' počítač
Digitalizácia textu: Postupnosť 7 bitov alebo 8 bitov, čiže 1 byte slúži na uloženie informácie o 1 znaku (písmene, číslici atď.). Napríklad 01000001 je znak „A".
Hodnoty znakov sú dané tabuľkou ASCII alebo tabuľkami národných kódov (Latin2, Kamenických)
Digitalizácia obrázku
Rastrový formát.
Takýto obrázok je určený jednotlivými bodmi (pixel). Každý z týchto bodov má ako vlastnosť svoju farbu. Podľa počtu možných farieb každého bodu sa rastrové formáty delia na :
- monochromatické (čiernobiele)
- formáty v stupnici šedi
- formáty farebné

Bodový zápis :
Každý bod obrázku môže nadobúdať iba dve hodnoty, a to čierna a biela (1 alebo 0). Takéto číslo zaberá v súbore 1 bit a odtiaľ pochádza pravé označenie bitmapa. Používa sa pre perokresby a všade tam, kde vystačí rozlíšenie čierna - biela.
Čiernobiely polotónový obrázok. Tu už môže hodnota čísla reprezentujúceho bod nadobúdať hodnotu 0...255. Máme teda 256 variant. Každý variant reprezentuje jeden odtieň od čiernej po bielu - 256 stupňov šedi (grayscale). Jeden bod tohto obrázku zaberie 1 Byte.
256 farieb. Aj v tomto prípade je pre každý bod vyhradený 1 Byte. To znamená 256 variant. Tieto varianty však nie sú postupne radené odtiene, ale konkrétne, nenasledujúce farby.
24 bitová grafika - RGB. 16,7 mil. farieb. Tu je pre každý bod obrázku vyhradená trojica čísel. Každé číslo z trojice tak reprezentuje odtieň 0...255 v červenej (Red), zelenej (Green) a modrej (Blue). poskladaním týchto troch farebných odtieňov vzniká výsledná farba. Pre každý bod existuje teda 256^3=16 777 216 variant (farieb). 32 bitová grafika - CMYK. Princíp je podobný ako v predchádzajúcom prípade, máme však k dispozícii štyri škály.
Azúrová (Cyan), purpurová (Magenta), žltá (Yellow) a čierna (Black).

Editácia obrazu v rastrovom grafickom formáte sa prevádza individuálnym nastavovaním farby jednotlivých pixelov. Výhodou tohto formátu je veľmi realistické podanie grafiky. Nevýhodou sú vysoké nároky na pamäť pri uložení a zhoršenie zobrazenia pri zväčšovaní obrázku.

DIGITALIZACIA ZVUKU

Ľudský sluch rozpoznáva zvuky v rozsahu 20Hz - 16KHz, maximálne do 22KHz.
Skladby na hudobných CD bývajú uložené v kvalite 44100Hz a 16 bitov
Koľko krát za sekundu odoberáme vzorku= vzorkovacia frekvencia - **sample rate**
Koľko bitov použijeme na vzorku, koľkoúrovní snímame = bitová hĺbka - **bit rate**
Bitrate = veľkosť datového toku128 Kb/s býva hodnota pre zvuk
Variable Bitrate - variabilný datový tok



Príklad rozlíšenia s hĺbkou 3 bity ($2^3 = 8$ stavov)

D/A - digitálno-analógový prevodník
A/D - analógovo - digitálny prevodník
MPEG - Moving Picture Experts Group

Prevod zvuku:

1. Analógový zvuk sa prevedie na elektrický signál
2. Digitalizujeme elektrický signál (nastavenie sample a bit rate)
3. Zosilniť signál
4. Reprodukcia
5. Keďže takto spracovaný súbor je veľký, použijeme kódovanie (stratovú kompresiu), pričom sa primárne odstraňujú encoderom signály, na ktoré ľudský sluch nie je dostatočne citlivý (citlivosť v rôznych pásmach)

uchovávanie inf. - Pamäť a uchovanie dát

Vnútorná pamäť počítača je tvorená mikročipmi uloženými priamo na základnej doske počítača. Jej hlavnou zložkou je pamäť RAM. Rýchlejšia ako Ram pamäť je tzv. Cache pamäť, ktorá je uložená priamo v procesore alebo je integrovaná do základnej dosky počítača. K vnútornej pamäti počítača zaradujeme ja pamäť ROM.
Vonkajšiu pamäť počítača tvoria všetky periférne zariadenia, v ktorých môžu byť uložené počítačové dáta. pružný disk-floppy disk,pevný disk, Cd-rom, DVD, flash disky, mechanika ZIP 100 až 250 MB. Jaz drive 1GB.

Typy pamäti

Pamäť ROM je pamäť len na čítanie je mikročip integrovaný v základnej doske počítača, ktorý obsahuje základné programové inštrukcie tzv. BIOS slúžiace k otestovaniu integrity počítača po jeho zapnutí (kontrola grafickej karty, pamäte RAM a periférnych zariadení) a k zavedeniu operačného systému do počítača. Obsah pamäte ROM je nemenný a prenos dát jednostranný z pamäte ROM do mikroprocesora.
Pamäť RAM –Random Acess memory – sa zvykne nazývať aj operačnou pamäťou počítača pretože slúži výhradne na krátkodobé uchovanie dát súvisiacich s práve vykonávanou činnosťou počítača. Sú v nej uložené inštrukcie programov práve bežiacich v systéme a dáta, s ktorými tieto programy pracujú.
SIMM-single In-line menory module
DIMM-dual In-line memory module
SDRAM – Synchronious Dynamic RAM

Cache pamäť –vyrovnávacia pamäť procesora, ktorej rýchlosť korešponduje s rýchlosťou procesora lepšie a do ktorej si procesor vie ukladať dáta potrebné pre urýchlenie rôznych výpočtov. Vyrovnávacia pamäť sa používa aj pri mnohých periférnych zariadeniach –cache pamäť periférnych zariadení zvykne sa nazývať aj buffer.

Meranie pamäti

1 bit predstavuje základnú jednotku informácie , ktorá môže mať 2 rôzne hodnoty, najčastejšie označené ako 0 a 1. Počítač nespracováva bity jednotlivo ale v skupinách bitov pričom počet v skupine je vždy mocnina dvojky. 1 byte je názov pre skupinu 8 bitov. 1 byte môže mať 256 rôznych hodnôt

Kódovanie – znamená vzájomné priradzovanie abecied prirodzených jazykov do počítačovej abecedy a naopak. Kód je pravidlo ako kódovanie realizovať. Kódy teda slúžia k prevodu medzi človeku zrozumiteľnými údajmi a počítačovými údajmi. V súčasnosti sú najrozšírenejšie tieto kody

EBCDIC – extended binary coded decimal interchange code

ASCII – american standard code for information interchange

CCITT2 – comité consultatif international telégraphique et teléphonique

Jednoznačne najpožívanejším kódom v súčasnosti je ASCII kod. Ide o osembitový kod, ktorým je možné zobraziť teda 256 rôznych znakov. Táto abeceda obsahuje čislice 0-9 písmenka veľkej a malej abecedy znamienka a rôzne znaky, riadiace znaky a semigrafické znaky

Pre počítačové abecedy existuje medzinárodné dohoda ako využiť 256 znakov. Vznikli tak prevodné tabuľky kodov ktoré sú výhodné pre určité jazykové oblasti. Medzi najpoužívanejšie patria LATIN 1 – západoeurópska a LATIN 2 –stredoeurópska. Obidve tabuľky sú rozdelené na dve časti. Prvých 128 znakov je uznávaných ako štandard a pre anglicky hovoriace krajiny je táto sada postačujúca.

KODOVCIA TABULKA

Dec	Oct	Hex	Name	CTRL Description
65	101	41	(A)	CAPITAL LETTER A
66	102	42	(B)	CAPITAL LETTER B
67	103	43	(C)	CAPITAL LETTER C
68	104	44	(D)	CAPITAL LETTER D
69	105	45	(E)	CAPITAL LETTER E
70	106	46	(F)	CAPITAL LETTER F
71	107	47	(G)	CAPITAL LETTER G
72	110	48	(H)	CAPITAL LETTER H
73	111	49	(I)	CAPITAL LETTER I
74	112	4A	(J)	CAPITAL LETTER J
75	113	4B	(K)	CAPITAL LETTER K
76	114	4C	(L)	CAPITAL LETTER L
77	115	4D	(M)	CAPITAL LETTER M
78	116	4E	(N)	CAPITAL LETTER N
79	117	4F	(O)	CAPITAL LETTER O
80	120	50	(P)	CAPITAL LETTER P
81	121	51	(Q)	CAPITAL LETTER Q
82	122	52	(R)	CAPITAL LETTER R
83	123	53	(S)	CAPITAL LETTER S
84	124	54	(T)	CAPITAL LETTER T
85	125	55	(U)	CAPITAL LETTER U
86	126	56	(V)	CAPITAL LETTER V
87	127	57	(W)	CAPITAL LETTER W
88	130	58	(X)	CAPITAL LETTER X
89	131	59	(Y)	CAPITAL LETTER Y
90	132	5A	(Z)	CAPITAL LETTER Z

OK, 5 kreditov

2. Zvuk, video na počítači. Formáty zvukových súborov. Zaznamenávanie a prehrávanie zvuku a videa. Jednoduché animácie, multimediálne encyklopédie, výukové programy.

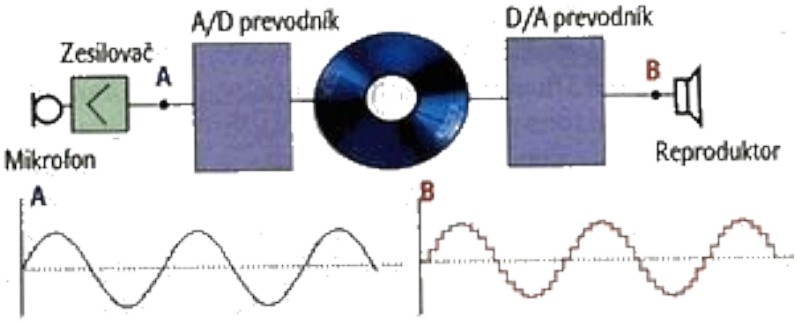
Zvuk – je pozdĺžne mechanické vlnenie s istou vlnovou dĺžkou, a teda zodpovedajúcou frekvenciou, s istou farbou a intenzitou (hlasitosťou).

Zvuk je spojitá - analógová informácia.
Harmonický zvuk (napr. komorné "a") môžeme znázorniť sínusoidou.
Počítače však nevedia spracovávať spojité informácie, preto bolo potrebné vymyslieť spôsob, ako zvuk digitalizovať - previesť na čísla.

Formáty zvuku
Tak, ako grafické údaje, tak aj zvukové údaje môžeme zakódovať pomocou rôznych formátov.

Súborové formáty môžeme rozdeliť do dvoch kategórií:
1. Formáty, ktoré zabezpečujú umelé vytvorenie zvuku
2. Formáty, v ktorých je uložený digitalizovaný zvuk (môžu byť stratové, teda vynechávať menej podstatné dáta, alebo sa môžu striktne snažiť o čo najvernejší záznam zvuku bez straty jeho kvality)
   MIDI (*Musical Instrument Digital Interface*)
   Tento formát je prvým typom - zabezpečuje umelé vytvorenie zvuku. Súbor MIDI je súhrn inštrukcií pre zvukový syntetizátor počítača, hudobného nástroja alebo iného zariadenia, podľa ktorých sa zvuk vytvára (rovnaké syntetizátory majú mobilné telefóny, preto sa hudba MIDI označuje tiež polyfonické zvonenie). Výhodou súborov MIDI je, že na minútu záznamu potrebujú oveľa menší úložný priestor ako ostatné formáty.
   Wave alebo **Waveform formát** (súbor s príponou .wav) vyvinuli spoločnosti IBM a Microsoft na použitie na platforme PC. Stal sa štandardom na záznam zvuku v operačných systémoch Windows. Metódou záznamu dát je najčastejšie Pulse Code Modulation (PCM), ukladajúca nekomprimované dáta (Existujú však i ďalšie spôsobí, ktoré Wave formát podporuje ako CCIT, GSM, g.723.1 ...). Princíp uloženia zvuku v tomto formáte popisuje digitalizácia zvuku:

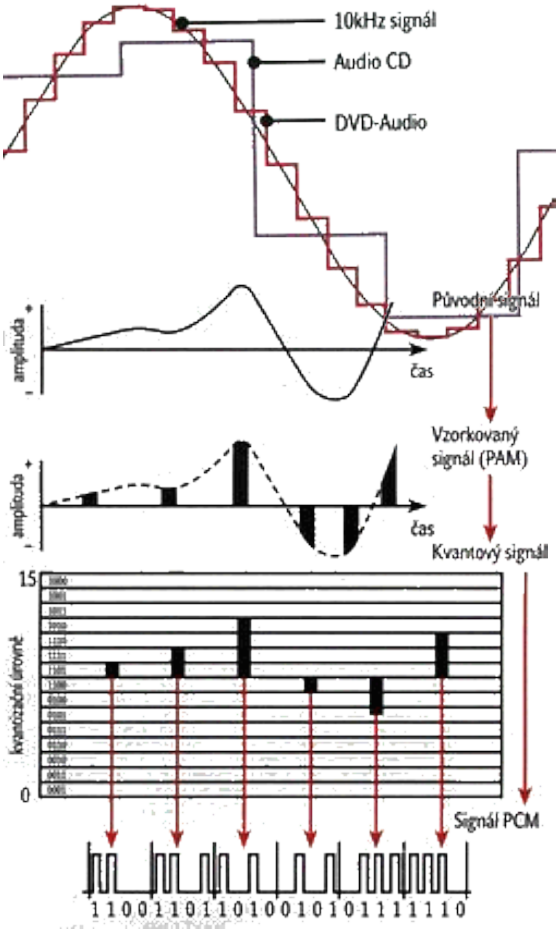
   Digitalizácia zvuku
   Prevod zvuku z analógovej podoby do digitálnej zabezpečuje A/D - analógovo - digitálny prevodník.



Skôr ako sa z analógového signálu stane PCM zvuk, musí najprv prejsť vzorkovaním, kvantovaním a kódovaním.

VZORKOVANIE: Vzorkovač zaznamenáva aktuálne hodnoty analógového signálu v pravidelných intervaloch s istou frekvenciou napr. pri frekvencii 10kHz sa zaznamená hodnota signálu 10 000 krát za sekundu. Vzniká signál PAM - pulzná amplitúdová modulácia.

Vzorkovacia frekvencia (Sampling rate)



Aby sa dal vzorkovaný signál PAM pri reprodukcii plne zrekonštruovať, musí byť splnené tzv. "Nyquistovo kritérium" : frekvencia, ktorou sa vzorkovanie prevádza, musí byť aspoň 2-krát vyššia ako frekvencia pôvodného signálu. Ak je počuteľný zvuk od 16 - 20 000 Hz, tak vzorkovacia frekvencia musí byť aspoň 40 kHz. V praxi sa vzorkuje s 10% navýšením, preto sa používa vzorkovacia frekvencia 44,1kHz.

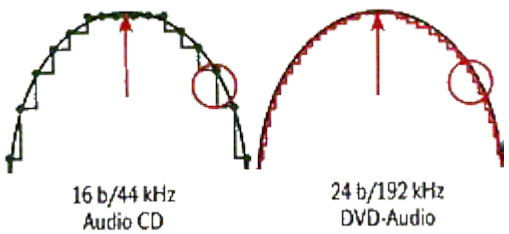
KVANTOVANIE: Kvantovaním sa namerané hodnoty "zaokrúhľujú" na najbližšiu úroveň amplitúdy každej vzorky, preto má digitálny signál na rozdiel od analógového schodovitý priebeh.

KÓDOVANIE: Pri kódovaní zvuku hudobného CD sa používa 16 bitové kódovanie - tzn. že každú vzorku zakódujeme 16 - ticou jednotiek a núl - všetkých možných napäťových úrovní signálu teda môže byť 216=65536 (hovor v telefóne je kódovaný 8 bitmi - rozlišuje 256 napäťových úrovní).
V prípade stereofónneho signálu sa používajú 2 kanály a výstupom sú 2 prúdy digitálnych hodnôt.

POROVNANIE

Kvalita digitálneho záznamu	**Vzorkovacia frekvencia**	**rozlíšenie, kanály**
Telefónna kvalita	11 025Hz	8 bit - mono
Rozhlasová kvalita	22 050Hz	8 bit - mono
CD kvalita	44 100Hz	16 bit - stereo
DVD kvalita	192 000Hz	24 bit - 5.1 surround sound

Čím je vyššia vzorkovacia frekvencia, tým kvalitnejší zvuk získame. Doteraz najkvalitnejší zvuk v CD kvalite so vzorkovacou frekvenciou 44,1kHz, 16bit stereo je prekonaný formátom DVD-Audio, kde vzorkovacia frekvencia je až 192kHz s 24 bitovým rozlíšením.



Ak chceme napríklad zakódovať 1 minútu stereo nahrávky s použitím vzorkovacej frekvencie CD kvality, tak takýto PCM zvuk bude zaberať 60x2x44100x16 bitov,

čo je asi 10MB pamäte. Takže zvuk vo formáte PCM je nepoužiteľný na vysielanie zvuku cez internet. Tento problém však riešia kompresné zvukové formáty, ktoré dokážu skomprimovať (zmenšiť) veľkosť zvukových dát, a tým i znížiť bitrate. Pri komprimovaní môže byť bitrate v nahrávke konštantný alebo sa môže dynamicky meniť, čím umožní dosiahnutie ešte lepšieho kompresného pomeru.

Ďalšie formáty zvuku
MPEG1 Layer 3 - veľmi populárny formát, ktorý všetci poznáme ako MP3. Je to stratový audioformát, schopný redukovať množstvo dát potrebných na záznam zvuku až desať násobne. To znamená, že na jedno CD sa zmestí desať bežných audio CD skomprimovaných do tohto formátu. Stratová kompresia spočíva vo využití vlastností ľudského sluchu na odstránenie nepočuteľných zvukov z nahrávky.

Windows Media Audio - konkurenčný stratový zvukový formát spoločnosti Microsoft (súbor s príponou .wma). Vznikol ako odpoveď na formát MP3 a ako priama konkurencia spoločnosti RealNetworks. Microsoft deklaruje kvalitu záznamu porovnateľnú s CD kvalitou už pri dátovom toku 64 kb/s, čím má prekonať MP3, ale nezávislé testy to nepotvrdzujú. Formát umožňuje vložiť do záznamu aj digitálne riadenie autorských práv, Digital Rights Management (DRM), preto ho na publikovanie nahrávok preferujú skôr vydavateľstvá. ☺

OGG Vorbis - rovnako veľmi kvalitný stratový formát. Jeho výhodou je to, že je OPEN SOURCE (tzn. je verejne k dispozícii zdrojový program a ktorýkoľvek programátor si ho môže upraviť podľa vlastných potrieb)

RealAudio je audioformát spoločnosti RealNetworks (súbor s príponou .ra). Vyžaduje si vlastný prehrávač zvuku. Je uspôsobený najmä na prenos zvuku pri nízkom bitrate. Často sa teda používa pri publikovaní cez internet pomocou tzv. prúdového audia.

Viackanálové formáty zvuku
AAC - MPEG-4 Advanced Audio Coding je veľmi kvalitná kompresia zvuku, ktorá umožňuje použiť takmer neobmedzený počet kanálov. Nevýhodou je, že nie je zadarmo. ☹

AC3 - Dolby Digital. Najviac sa používa na kódovanie zvukových stôp na DVD. Aj keď tento formát nedosahuje takú dobrú kvalitu, je často používaný.

DTS - Digital Theater Systems je ďalší formát určený pre DVD. Poskytuje vyššiu kvalitu než AC3, ale za cenu extrémne vysokého bitového toku.

Programy na prehrávanie zvuku
Jedným z najdôležitejších parametrov prehrávacieho programu je to, koľko zvukových formátov dokáže prehrať. Keď vznikol nový zvukový formát, bolo potrebné stiahnuť si novšiu verziu prehrávača, do ktorého výrobcovia pridali jeho podporu. Niektorí výrobcovia však tento problém vyriešili inak. Navrhli svoj prehrávač tak, aby dokázal prehrať akýkoľvek formát pomocou tzv. kodekov. Kodek je vlastne akýsi návod pre prehrávač, pomocou ktorého vie dekódovať daný zvukový formát. V operačnom systéme Windows sa nachádza práve takýto typ prehrávača s názvom Media Player. Okrem tohto prehrávača však existuje obrovské množstvo voľne dostupných prehrávačov, z ktorých najznámejšími sú WinAmp a Sonique. Súčasný výkon počítača však už umožňuje i prehrávanie

viacerých skladieb súčasne, takže môžeme pomocou prehrávača ako napríklad OTS Turntables mixovať skladby ako DJ na diskotéke.

Programy na nahrávanie zvuku a úpravu zvuku

Najjednoduchším programom pre nahrávanie zvuku je program "Nahrávanie zvuku", ktorý je súčasťou operačného systému Windows. Program ponúka aj jednoduché efekty, ako je zrýchlenie, či spomalenie záznamu, vloženie ozveny a zaujímavé spätné prehrávanie. Medzi vyspelejšie editory, ktoré umožňujú nahrávať a upravovať viac kanálov a sú dostupné zadarmo, patrí program Audacity. Umožňuje nahrávať neobmedzený počet kanálov, strihať, kopírovať a mixovať ich dohromady. Dokáže tiež zmeniť rýchlosť prehrávania a hlasnosť jednotlivých kanálov, odstrániť šum a výsledok uložiť v rôznych formátoch. Medzi Profesionálne nástroje, ktoré obsahujú viac ako 50 efektov na úpravu zvukových kanálov patria programy SoundForge od firmy Sony a Audition od firmy Adobe.

zvukový editor Audacity zvukový editor Sony SondForge zvukový editor Adobe Audition

VIDEO

Na spracovanie videa slúži mnoho programov. Niektoré slúžia na prevod formátu videa, iné umožňujú video i strihať, vkladať audio stopy alebo titulky ...

Spracovanie nami nahratého videa

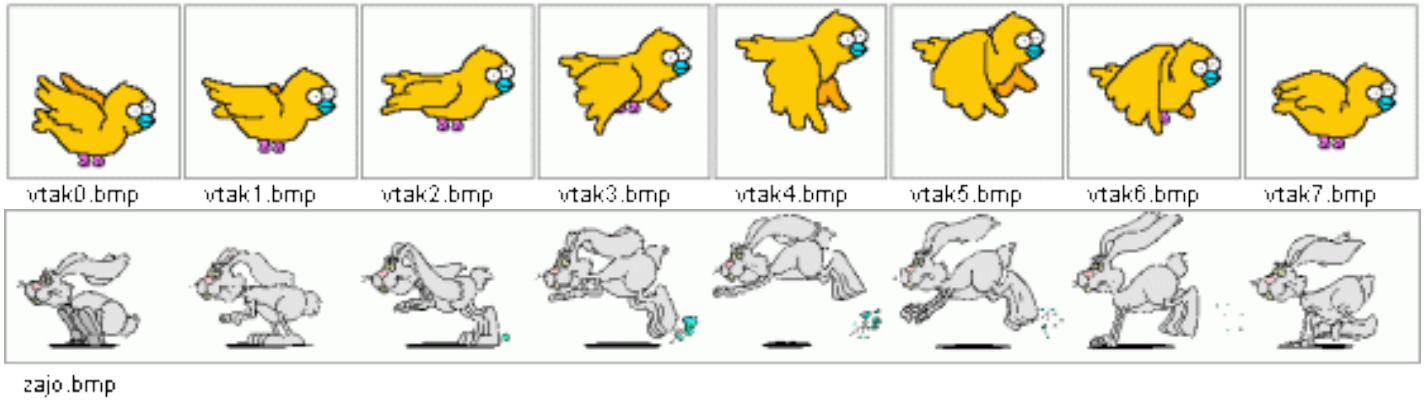
Ak chceme upravovať nami nahraté video na počítači, musíme si zaobstarať potrebné vybavenie, ktoré je závislé od toho, z akého zdroja budeme video spracovávať.

K digitalizačným, strihovým a televíznym kartám je okrem ovládačov pribalený i strihový softvér. Tento softvér umožňuje nastaviť kvalitu výsledného videa, najmä rozlíšenie a farebné kódovanie.
V prípade, že ku karte nemáte vyhovujúci softvér, môžete využiť program MS MovieMaker, ktorý je súčasťou systému Windows XP. Umožňuje základné funkcie ako nahranie videa z video zariadenia, import videa v rôznych formátoch, aplikovanie rôznych farebných a prechodových efektov, vloženie titulkov na začiatok a koniec videa a uloženie výsledného videa vo formáte WMV. Program tiež umožňuje urobiť video prezentáciu z fotografií. Hlavnou výhodou tohto programu je však to, že po nainštalovaní slovenského jazykového balíka (MUI alebo LIP) pre Windows je program celý vrátane pomocníka v slovenčine.

Pokiaľ chcete spracovávať video na profesionálnejšej úrovni, môžete si zaobstarať program Adobe Premiere Pro alebo rovno balík programov Adobe Production Studio, ktorého súčasťou je i spomínaný Adobe Premiere Pro.

Programy zadarmo – nahrávanie : Virtual Dub, Virtual VCR ...
Animácia - je to tzv. “oživovanie“ kreslených postavičiek alebo bábok
- dalo by sa povedať, že je to striedanie jednotlivých obrázkov, ktoré prebieha takou rýchlosťou, že to ľudské oko vidí ako plynulý pohyb

Príklady jednoduchej animácie:



Multimediálne encyklopédie
Encyklopédia je slovo gréckeho pôvodu a podľa slovníka cudzích slov znamená „vedecké dielo zahrňujúce poznatky jedného, prípadne mnohých odborov; náučný slovník.“
Slovo multimediálny znamená niečo ako „viac-prostriedkový“, teda v počítačovej terminológii „využívajúci väčšie množstvo prostriedkov na jednosmernú (prípadne obojsmernú) komunikáciu, ako je len grafický výstup.“
Multimediálna encyklopédia ale predovšetkým znamená obrovské množstvo informácií o danej téme, ktoré sú sprehľadnené a zotriedené.

Sila multimediálnej encyklopédie je ale v tom, čo obyčajné encyklopédie nedokážu, alebo čo poskytujú len v obmedzenej forme: obrovské množstvo obrázkov, animácií, zvukov, rôzne kvízmajstre, interaktívne mini hry a ďalšie súčasti ktoré sa nevyhnutne spájajú s používaním počítača.

Príklady multimediálnych encyklopédií: Eyewitness Encyclopedia of Science 2.0, MS Encarta

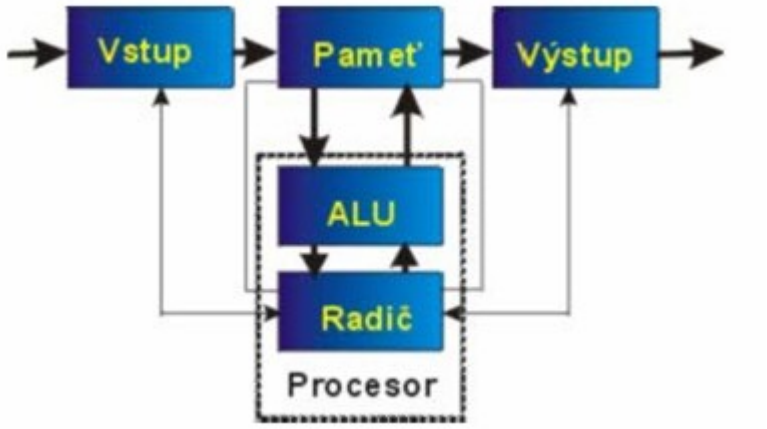
Výukové programy
- majú za úlohu nás o niečom informovať resp. nás niečo nové naučiť

- nie sú obyčajne také rozsiahle ako multimediálne encyklopédie ale sú skôr zamerane na konkrétne časti napr. mesta Slovenska, štáty sveta alebo tiež napr. matematika, fyzika, dejepis alebo iné

OK. 5 kreditov

3. História a vývoj počítačov. PC – popis, rozdelenie, von Neumannova štruktúra počítača. Centrálna jednotka, procesor, zbernice, prídavné karty, interné a externé pamäte, vstupné a výstupné zariadenia, princíp práce základných druhov prídavných zariadení.

Von Neumanova schéma
Von Neumannova schéma bola navrhnutá v roku 1945 americkým matematikom Johnom von Neumanom ako model samočinného počítača. Tento model s istými výnimkami zostal zachovaný dodnes.
Podľa tejto schémy sa počítač skladá z piatich hlavných modulov:



Operačná pamäť: slúži k uchovávaniu spracovaného programu, spracovávaných dát a výsledkov výpočtov
ALU: Aritmetic-logic Unit (aritmetická jednotka): jednotka vykonávajúca všetky aritmetické výpočty a logické operácie. Obsahuje sčítačky, násobičky (pre aritmetické výpočty) a komparátory (pre porovnávanie)
Radič: riadiaca jednotka ktorá riadi činnosť všetkých častí počítača. Toto zariadenie je ovládané pomocou riadiacich signálov, ktoré sú odosielané jednotlivým modulom. Reakcie na riadiace signály, stavy jednotlivých modulov, sú naopak odsielané späť radiču pomocou stavových hlásení
Vstupné zariadenia: zariadenia určené pre vstup programov a dát
Výstupné zariadenia: zariadenia určené pre výstup výsledkov, ktoré program spracoval

Vo von Neumannovej schéme je možné ešte vyznačiť ďalší modul ktorý vznikne spojením predchádzajúcich modulov: **Procesor**: Radič + ALU

Princíp činnosti počítača podľa von Neumannovej schémy:
1. Do operačnej pamäte sa pomocou vstupných zariadení cez ALU umiestni program, ktorý bude vykonávať výpočet.
2. Rovnakým spôsobom sa do operačnej pamäte umiestnia dáta, ktoré bude program spracovávať.
3. Prebehne vlastný výpočet, jeho jednotlivé kroky vykonáva ALU. Táto jednotka je v priebehu výpočtu spolu s ostatnými modulmi riadená radičom počítača. Medzivýsledky výpočtu sú ukladané do operačnej pamäte.
4. Po skončení výpočtu sú výsledky poslané cez ALU na výstupné zariadenie.

Základné odlišnosti dnešných počítačov od von Neumannovej schémy:
- Podľa von Neumannovej schémy počítač pracuje vždy nad jedným programom. Toto vedie k veľmi zlému využitiu strojového času. Je teda normálne, že počítač spracúva paralelne viac programov zároveň – tzv. multitasking.
- Počítač môže disponovať aj viacej ako jedným procesorom.
- Počítač podľa von Neumanovej schémy pracoval iba v tzv. diskrétnom režime.
- Existujú vstupné/ výstupné zariadenia I/ O devices, ktoré umožňujú aj vstup aj výstup dát (programu).
- Program sa do pamäte nemusí zaviesť celý, ale je možné zaviesť iba jeho časť a ostatné časti zavádzať až v prípade potreby.

Vstupné zariadenia

-zariadenia, ktoré slúžia pre zadávanie údajov do počítača
Napr. klávesnica, myš, scanner...

- Myš

Slúži na ľahké ovládanie PC .Mala pôvodne 2 tlačítka .Ľavé slúži na označenie objektu, kliknutím vyberieme a aktivujeme objekt (súbor). Dvojitým kliknutím otvorím požadovaný súbor, ikonu na pracovnej ploche. Podržaním tohto tlačítka a premiestnením myši po podložke premiestnim daný súbor, ikonu. Pravé sa používa okrem iného na vyvolanie kontextového menu. Na spodku myši sa nachádza gulička , ktorou je potrebné pohybovať po rovnej ploche pre vykonanie týchto operácii.

Druhy myší: -„Trackballova“ gulička –notebooky
-optická- miesto guličky je optický lúč, ktorý sníma pohyb na podložke
-Touchscreenová- ovládame ju prstom na monitore

V súčasnosti sa používajú 3-4 tlačidlové myši, funkcia tlačidiel je programovateľná.

Scanner

Vstup pre digitalizáciu obrázkov alebo fotografií či textu.
Obrázok sa nakopíruje do PC v tvare grafického súboru a textové súbory sú tiež uložené ako obrázok. Na ich premenu na textový súbor sa používa program OCR, ale nastáva niekoľko zlých prekladov. Pripojený je cez USB alebo SCSI port.

Výstupné zariadenia

-zariadenia prostredníctvom, ktorých nám počítač oznamuje výsledky

- Tlačiareň

Pre každú tlačiareň sa dodáva program ovládač(driver) ,ktorý musí byť nainštalovaný a riadi tlač. Najpoužívanejšie sú atramentové a laserové tlačiarne. Používali sa aj ihličkové tlačiarne.

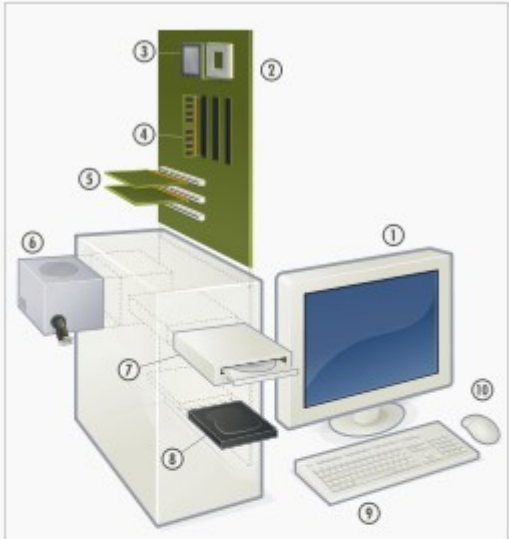
- Atramentová tlačiareň
- Laserová tlačiareň
- Ihličková tlačiareň
- Termotlačiareň

Atramentová tlačiareň

Zobrazuje znaky rozprašovaním drobných kvapôčiek atramentu na papier. Každá kvapôčka vytvára 1 bod a znaky sú opäť zobrazené z bodov. Rozdiel je v nanášaní atramentu a veľkosti bodu. HP má miešanie farieb v 1 bode – až 4krát- a veľkosť kvapky je 32 pikometrov. EPSON a iné miešajú pred tryskou a veľkosť bodu dosahuje 6-3 pikometre. Umožňuje kvalitnú tlač aj vo farbe. Používajú sa 4 farby: žltá, purpurová, čierna, tyrkysová. (CMYK – Cyan, Magenta, Yellow, blacK)

Laserová tlačiareň

Využíva na tlač laserovú technológiu. A osvetlených miestach sa stáva valec vodivým. Valec sa popráši práškom, ktorý sa prichytí na osvetlených miestach. Obraz sa potom prenesie z valca na papier. Umožňuje kvalitnú tlač, rýchlosť tlače.



Ihličková tlačiareň

Zobrazuje znak úderom ihličiek cez farbiacu pásku. Nevýhodou môže byť, že sú hlučné. Sú vhodné na textové výstupy vyžadujúce napr. vytváranie kópií.

Termotlačiareň

Využíva tlač na tepelne citlivý papier – registračné pokladnice.

CPU (skr. z angl. *central processing unit*, často prekladané ako *centrálna procesorová jednotka*) je hlavný procesor počítača. Interpretuje, vykonáva alebo spracuje inštrukcie alebo dáta programu vo forme strojového kódu. Dnes sú centrálne procesorové jednotky takmer vždy realizované vo forme mikroprocesora.

Rozširujúce karty. Sú to externé moduly počítača vo forme karty (doska plošného spoja veľkosti pohľadnice), ktorá má na spodnej časti zásuvné kontakty. Kontakty sa zasúvajú do zbernice matičnej dosky. Použitá karta musí zodpovedať typu zbernice na matičnej doske. Najčastejšie používanou rozširujúcou kartou je grafická karta, pomocou ktorej počítač zobrazuje na monitore. Ďalšími príkladmi sú sieťová karta (pripojenie do počítačovej siete),

zvuková karta (karta vytvárajúca zvuky), TV karta (karta pre príjem TV signálu) a pod. Viacero rozširujúcich „kariet“ – modulov sa pripája externe pomocou zbernice USB.
Zbernica je elektrický vodič, alebo častejšie sústava vodičov, ktoré prepájajú viaceré elektrické či elektronické zariadenia a slúžia na vzájomný prenos energie, alebo častejšie údajov, medzi nimi.
Jednou z najzákladnejších vlastností zbernice je počet bitov, ktoré sa v jednom okamihu prenášajú zbernicou (čo je dané počtom vodičov v zbernici). Iné vlastnosti zahrňujú rýchlosť zbernice, elektrické vlastnosti, fyzickú dĺžku, príp. určenie údajov prenášaných zbernicou (adresná, dátová, riadiaca zbernica).

História počítačov

Pojem bol používaný v priebehu 70. rokov 20. storočia. Ako zlomový dátum je uvádzaný január 1977, kedy vyšlo prvé číslo Personal Computing Magazine. Ale až s uvedením počítače IBM PC (IBM 5150) na trh v 1981 sa ustálilo označenie PC (alebo Personal computer) pre počítač s procesorom Intel x86 kompatibilným (tj. vnútornou architektúrou, komponentmi a programovým vybavením zlučiteľným) s týmto modelom.

je nutné podotknúť, že vývoj výpočtovej techniky v tej dobe bol veľmi prudký a práve platforma osobných počítačov sa stala hnacím motorom rozvoja tejto techniky. V tejto dobe boli na trhu aj ucelenejšie, technicky dokonalejšie a možno priekopníckejšie riešenia (Apple) ako IBM PC. Zásadnú úlohu však zohral fakt, že IBM zvolila otvorenú politiku, ktorá umožnila tretím výrobcom vyrábať komponenty pre PC. Politika drahého značkového výrobku, ktorú razila firma Apple nakoniec neuspela, pretože vďaka obrovskej konkurencii na trhu komponentov pre nasledovníkov IBM PC začal tento segment rásť raketovým tempom pri súčasnom poklese cien, takže výrobky Apple začali zaostávať aj po technickej stránke.

Prvý model IBM PC bol dodávaný procesorom Intel 8088, pracujúcom na frekvencii 4,77 MHz a s operačnou pamäťou RAM o veľkosti 16 alebo 64 kB (maximálne 256 KByte). Pre záznam dát sa používal kazetový magnetofón, neskôr osempalcová disketová mechanika.

Behom niekoľkých rokov sa objavili viaceré vylepšenia a tiež veľké množstvo výrobkov nadväzujúcich na platformu PC – periférií, tlačiarní monitorov a pod. Objavili sa prvé klony PC (PC kompatibilné počítače). V roku 1983 sa objavili prvé pevné disky o kapacite 10 MB, v roku 1986 prvé modely s procesorom 80286 (16bitová zbernica, adresovateľná RAM až 16 MB) a v roku 1989 s procesorom 80386. Došlo ku zmenšenie disketovej mechaniky na formát 5.25" pre /mini/diskety o kapacite 360 kB, k zvýšeniu kapacity 5.25" diskiet na 1.2 MB a nakoniec k ich náhrade 3.5" disketami o kapacite 1.44 MB.

Operačným systémom pôvodných osobných počítačov IBM PC a IBM PC kompatibilných býval takmer výlučne MS-DOS alebo jeho klon (PC-DOS, DR-DOS), ktorý bežal v textovom režime. Od roku 1984 sa ale začalo u osobných počítačov presadzovať grafické užívateľské rozhranie (GUI), najprv na počítačoch Apple Macintosh a neskôr na začiatku 90. rokov 20. storočia na počítačoch IBM PC kompatibilných v podobe MS Windows 3.0. Posledné roky 20. storočia znamenali rýchly nárast výkonu hardvérových komponentov i softvéru používaného v PC. Pribudli 80486, integrovaný matematický koprocesor sa stal samozrejmosťou. Následne prišla architektúra Intel Pentium, ktorá priniesla ďalšie zrýchlenie (1990 taktovacie frekvencie 10-20 MHz, 2000 okolo 100-200 MHz). Monochromatické monitory boli nahradené farebnými. Rozlíšenie stúplo na 1024 x 768 pix. Samozrejmosťou sa stala mechanika pre čítanie optických diskov CD ROM. Pribudla zvuková karta a modem. Ihličkové tlačiarne boli nahradené laserovými a neskôr sa objavili atramentové tlačiarne.

OK, 5 kreditov, gramatické chyby boli opravené ☺

4. Charakteristika operačného systému, jeho význam, druhy OS, porovnanie Windows - Linux. Základné príkazy OS (DOS, LINUX). Súbor, adresár, stromová štruktúra adresárov. Spravovanie zariadení, priečinkov, súborov v rôznych OS.

Operačný systém (OS) - je softvér, ktorý spravuje zdroje počítača a poskytuje programátorom rozhranie na prístup k týmto zdrojom. Operačný systém tiež spracúva systémové dáta a vstupy od používateľa a odpovedá alokovaním a spravovaním úloh a interných zdrojov počítača ako služby pre užívateľa. OS vykonáva základné úlohy ako kontrola a alokovanie pamäte, pridelenie priority systémovým požiadavkám, kontrola vstupných a výstupných zariadení, umožnenie pripojenia do siete a správa súborov. Operačné systémy môžme nájsť takmer vo všetkom, čo obsahuje integrované obvody, od

osobných počítačov, cez internetové servery, mobilné telefóny, hudobné prehrávače, routre, switche, herné konzoly, digitálne kamery, až po šijacie stroje či teleskopy.
Operačné systémy pre PC sa delia do viacerých skupín, najväčšie z nich sú:

1. **Microsoft Windows** – operačné systémy vyvíjané spoločnosťou Microsoft, pôvodne vznikli ako nadstavba pre MS-DOS, ktorý bol používaný v IBM PC. Novodobé verzie sú založené na novšom jadre Windows NT, ktoré bolo pôvodne zamýšľané pre OS/2 a prenesené od VMS. Windowsy bežia na x86, x86-64 a Itániových procesoroch. Staršie verzie bežali taktiež na DEC Alpha, MIPS, Fairchild (neskôr Intergraph) Clipper a PowerPC architektúrach. Dlhodobo patrí Microsoftu veľká časť celosvetového trhu s desktopovými operačnými systémami. Windows je tiež používaný na serveroch, pričom podporuje aplikácie ako webové či databázové servery. V uplynulých rokoch Microsoft investoval značnú časť peňazí na reklamu, výskum a vývoj aby ukázal, že Windows je schopný spustiť akúkoľvek aplikáciu, čo viedlo k výraznej akceptácii u mnohých firiem.
2. **UNIX a UNIXové (UNIX-like) systémy** - odvodená skupina operačných systémov, s niekoľkými hlavnými podkategóriami ako **System V**, **BSD** a **Linux**. Názov UNIX je ochrannou známkou The Open Group, ktorá pod ňou vydá akýkoľvek systém, ktorý bude spĺňať jej definície. Termín unixový (unix-like) je bežne používaný na označenie veľkej skupiny operačných systémov, ktoré sa podobajú pôvodnému Unixu. Unixové systémy bežia na širokej škále počítačových architektúr. Vo veľkom množstve sa využívajú ako servery v komerčnej sfére, ako aj pracovné stanice na akademickej pôde. Na Unixe je založený aj operačný systém používaný na počítačoch spoločnosti Apple **Mac OS X.**

Porovnanie:
Windows – komerčný operačný systém, pre ktorý je dostupné veľké množstvo softwaru. Je to najrozšírenejší OS na celom svete, dostupný v serverových aj desktopových verziách. Jeho hlavná výhoda je jeho jednoduchosť aj pre počítačovo menej zameraných ľudí. Medzi jeho hlavné nevýhody patrí rozšírenosť vírusov a iných škodlivých kódov pre tento OS (čo súvisí s tým, že na trhu OS ma Windows takmer monopolné postavenie) a nedostatočné množstvo softwarového vybavenia v základnom balíku.

Linux – je to slobodný (free) software (čo znamená, že je dostupný zdarmo, voľne šíriteľný a má otvorený zdrojový kód, ktorý je možné ľubovoľne meniť a dalej šíriť podľa podmienok licencie GNU GPL). Používa ho menšia skupina ľudí a väčšinou slúži ako serverový OS (narozdiel od Windowsu, ktorý sa používa prevažne na desktopoch). Medzi jeho výhody patrí obrovská flexibilnosť a výkonnosť. Kedže je oveľa menej rozšírený ako Windows, skupina vírusov pre tento OS je značne menšia, priam zanedbateľná. Medzi hlavné nevýhody patrí malá dostupnosť úzko špecializovaného softwaru a niektorých ovládačov. Na druhú stranu, základný balík obsahuje veľké množstvo výkonných nástrojov, ku ktorým existujú pre Windows takmer len komerčné varianty.

Súbor - je (najmenšie možné) logické zoskupenie údajov. Súbory sa ukladajú na pevný disk alebo do (operačnej) pamäte počítača. Sú usporiadané v stromovej štruktúre v adresároch. Adresár môže obsahovať súbory a adresáre, ktoré môžu obsahovať znova ďalšie adresáre a súbory. Súbor nemôže obsahovať adresáre ani ďalšie súbory, obsahuje údaje v dvojkovej (binárnej), resp. šestnástkovej (hexadecimálnej) číselnej sústave.
Adresár - v počítačoch programovo pridelená a sprostredkovaná oblasť na diskovom nosiči, ktorá je označená podľa predpísaných syntaktických pravidiel (podobne ako súbory) a väčšinou združuje logicky súvisiace súbory (a adresáre). Spravidla slúži na logické zoskupenie súborov a iných adresárov. Adresáre sú na pevnom disku usporiadané v stromovej štruktúre. Adresár môže obsahovať súbory a zložky, ktoré môžu obsahovať znova ďalšie zložky a súbory. Takto je možné vetviť teoreticky do nekonečna, prakticky po hodnotu danú obmedzením konkrétneho súborového systému.

Súborové systémy UNIXu
UNIX a UNIXové operačné systémy priraďujú každému zariadeniu názov zariadenia, ale takýmto spôsobom sa nepristupuje k súborom na zariadení. UNIX vytvorí virtuálny súborový systém, ktorý zaraďuje všetky súbory na všetkých zariadeniach do jedinej hierarchie. To znamená, že v UNIXe je jeden koreňový (root) adresár a každý súbor existujúci v systéme sa nachádza na niektorej jeho podúrovni.

Ďalej, koreňový adresár nemusí mať fyzické umiestnenie -- nemusí byť na prvom pevnom disku, dokonca ani len na lokálnom počítači. Ako koreňový adresár je možné použiť zdieľaný sieťový zdroj. Aby bol umožnený prístup k súborom na inom zariadení je potrebné informovať operačný systém, kde v hierarchii by sa mali objaviť. Tento proces sa nazýva montovanie (mounting) súborového systému. Napríklad, aby bol umožnený prístup k CD-ROM, treba povedať operačnému systému „Vezmi tento súborový systém z CD-ROM a nech sa objaví v adresári /mnt“. Uvedený adresár (v tomto prípade „/mnt“) sa nazýva mount point. Adresár „/mnt“ existuje na všetkých UNIXových systémoch a jeho zmysel je špecifický ako mount point pre dočasné médiá ako diskety alebo CD. Môže byť prázdny alebo môže obsahovať podadresáre ako mount pointy pre jednotlivé zariadenia. Vo všeobecnosti, iba administrátor (root) je oprávnený montovať súborové systémy. UNIXové operačné systémy často obsahujú softvér a nástroje uľahčujúce proces montovania a poskytujúce novú funkcionalitu. Pre niektoré z týchto stratégií sa zaviedol termín „auto-mounting“ (automatické montovanie) ako odraz ich účelu.

Súborové systémy Microsoft Windows
Microsoft Windows sa vyvinul zo skoršieho operačného systému MS-DOS (ktorý je zasa založený na CP/M-80, ktorý preberá významné myšlienky z ešte skorších systémov, menovite DEC) a pridáva myšlienky v oblasti súborového systému a používateľského rozhrania z iných systémov od svojho prvého uvoľnenia (UNIX, OS/2 atď.). Windows ako taký používa súborové systémy FAT and NTFS. Staršie verzie súborového systému FAT mali zásadné obmedzenia dĺžky názvu súboru a maximálnej veľkosti diskov a partícií.

NTFS zavedený so systémom Windows NT umožňoval riadenie prístupu založené na ACL. Hard linky, viaceré súborové toky, indexovanie atribútov, kvóta, kompresia a mout pointy pre iné súborové systémy (nazývané „junctions“) sú tiež podporované, ale sú nedostatočne zdokumentované.

Na rozdiel od mnohých iných operačných systémov používa Windows abstrakciu vo forme písmen diskovej jednotky na používateľskej úrovni na odlíšenie jednotlivých diskových partícií. Napríklad cesta „C:\WINDOWS\“ reprezentuje adresár „WINDOWS“ na partícii označenej písmenom C. Partícia C sa bežne používa pre prvú primárnu partíciu disku, z ktorého bootuje Windows. Táto tradícia sa stala taká pevne zakorenená, že v starších verziách systému existovali predpoklady, že partícia, kde je nainštalovaný Windows musí byť C. Tradíciu používania písmen na označovanie partícií môžeme vystopovať späť do MS-DOSu, kde boli písmená A a B rezervované pre disketové mechaniky. Tiež je možné na písmená mapovať sieťové disky. Keďže Windows používa na interakciu s používateľom grafické rozhranie, jeho dokumentácia používa pojmy ako priečinok, ktorý obsahuje súbory a je reprezentovaný grafickou ikonou priečinka.
Základné príkazy:
cd – zmena adresára (Windows aj Linux)
mkdir – vytvorenie nového adresára (Windows aj Linux)
rm (Linux) a **del** (Windows) – zmaže daný súbor
rmdir – zmaže daný adresár (Windows aj Linux)
dir – výpis obsahu daného adresára (Windows aj Linux)
copy – skopíruje daný súbor do určenej lokácie (Windows aj Linux)
move – presunie daný súbor do určenej lokácie (Windows aj Linux)
help – vypíše zoznam základných príkazov (Windows aj Linux)
exit – ukončí prácu v príkazovom riadku, zatvorí okno, prípadne odhlási používateľa zo serveru (Windows aj Linux)
Ok, môže byť, 5 kreditov

5. Vývoj zvoleného operačného systému. Popis prostredia, ovládanie, okno, ikona, aplikácia, ponuka štart, spúšťanie aplikácií, plocha. Vytvorenie nového zástupcu a zložky na ploche, nastavenie vzhľadu.
1985
Vznikol prvý **Windows v1.0** kde išlo vlastne o MS-DOS s GUI (Graphic User Interface) a obsahoval pár aplikácií (notepad, kalendár, kartové hry, kalkulačku, hodiny).

1987
Windows 2.0 s podporou Intel 286 CPU, vo verzii **2.03** už podporovali 386.
1990
Windows 3.0 (kódové označenie **NT OS/2**)ten pracoval už s 16 farbami !!! mal programového a súborového manažéra a tiež správu tlače.
1993
Windows NT 3.1 (kódové označenie **Janus**) tak toto bol prelom vo vývoji OS Windows išlo totiž o 32-bitový systém s podporou client/server aplikáciami. Taktiež sa tu objavila prvá verzia NTFS systému.
Windows for Workgroups 3.11 (kódové označenie **Snowball**) s podporou P2P siete a domén. Poprvý krát bolo možné použiť Windows v LAN PC a laptop. Ukladal konfiguráciu na server, obsahoval podporu pre Novell NetWare a RAS
1994
Windows NT Workstation 3.5 (kódové označenie **Daytona**)s podporou OpenGL, a podporou pre aplikácie s plnou podporou 32bit a dlhých názvov (255 znakov)
1995
Windows 95 (kódové označenie **Chicago**) kompletný desktopový systém ktorý obsahoval veci z Windows 3.1, Windows for Workgroup a MS-DOS. Integrovali tam plnú podporu TCP/IP, dialup a Plug and Play pre lahšie inštalácie. Bol updatovaný ako plus verzia.
1996
Windows NT 4.0 Workstation (kódové označenie **Cairo, Impala**)vylepšený NT systém, prakticky totožný s Windows 95 ale plnou podporou intranetu, siete a prístup k internetu či korporátnej siete. Stal sa základom pre Windows 2000
1998
Windows 98 (kódové označenie **Memphis**) Tu by som povedal že veľmi kontroverzný systém. Prakticky Windows 95 s podporou DVD, USB a smerovaný na užívateľa. Bohužiaľ veľmi nestabilný bez záplat.
1999
Windows 98 Second Edition update Windows 98 pridal niečo nové a to podporu multimédií, NetMeeting, DirectX 6.1, podporu ICS (Internet Connection Sharing) a kompatabilitu s Windows NT. Windows 98 SE bol jeden z najvydarenejších systémov od Microsoftu.
2000
Windows Milenium Edition (Windows ME) (kódové označenie **Georgia**) zameraný na hudbu, video a domáce siete. Obsahoval System Restore (Obnovu systému) pre uľahčenie riešenia problémov pre užívateľov. Obsahoval Windows Media Player 7 technológiu ma editovanie, ukladanie a zdielanie videa. Bol tu pokus o zlúčenie technológie Windows 95 a NT technológie.
Windows 2000 Profesional (kódové označenie **Asteroid**)sa stal ďaľším zlomom v Microsofte. Prevzal to najlepšie z predošlých systémov a pridal podporu Plug and Play na hardware, wirelles produkty, USB a IEEE1394 či infrared.
2001
Windows XP (kódové označenie **Whistler**)prišiel s novým GUI a tiež s podporou 64-bit architektúry a prestal podporovať DOS.
Je v 4 rôznych edíciach:
Home ktorý je primárne určený pre domácnosť (logicky Home=domov/dom) rozšírený o multimédiá (Movie maker, DVD maker a podobne ktoré síce sú súčasťou XP professional ale...) a ochudobnený o možnosť práce v doméne.
Media Center ktorý dokáže nahradiť alebo vylepšiť domáce kino. Bez problémovo sa dá ovládať iba diaľkovým ovládačom.
Professional je absolútne plná verzia obsahuje všetky veci s NT technológie a tiež bonusy Home verzie ktorá mala primárne nahradiť Windows 98 a ME.
Professional 64bit určený na 64 bitové procesory má výkon vhodný hlavne pre servery.
2003
Windows Server 2003 (kódové označenie **Whistler Server, SBS Bobcat**) vylepšili kompatabilitu s Unix systémom, práca v stabilnom serverovom prostredí a to čo je dôležité vylepšené prostredie na nasadenie infraštruktúry.
2007

Windows Vista (kódové označenie **Longhorn**)priniesla vylepšenia v oblasti jadra, stabili a bezpečnosti. Je relatívne krátko na trhu a ešte sa často objavujú nekompatabilné veci.
2008
Windows Server 2008 (kódové označenie **Longhorn Server, SBS Cougar**)je veľmi prekvapujúci. Vysoká stabilita, je lepšie zabezpečený proti prienikom z vonka, dokáže zachytiť vírusy. Nová štruktúra jadra (Mikrokernel) a spôsob bezpečnosti (Palladium) Trustwork computing. Využíva PowerShell.

Daľší plánovaný OS Windows (kódové označenie Blackcomb, Vienna, Windows Seven)

Pracovné prostredie OS Windows
Pracovné prostredie systému Windows, sa skladá z pracovnej plochy a hlavného panelu. **Pracovná plocha**, je miesto, kde sa nachádzajú základne ikony (**Tento počítač** – obsahuje zoznam pevných diskov a niektorých zariadení pripojených k PC, **Kôš** – miesto kam sa presúvajú vymazané súbory, **Miesta v sieti** – obsahuje dostupné lokácie v lokálnej sieti a dalšie) a súbory (prevažne odkazy) ktoré tam umiestni sám používateľ. **Hlavný panel** (nachádza sa väčšinou na spodnej hrane obrazovky) obsahuje **ponuku Štart** (tu sa nachádza ovládací panel systému, vyhľadávanie, obľúbené položky, nedávno otvorené dokumenty, odkazy na nainštalované programy a pod.), **panel rýchleho spustenia – QuickLaunch** (tento panel sa často nevyužíva, prípadne sa nachádza v inej časti obrazovky) obsahuje menšie množstvo odkazov, väčšinou len najčastejšie používané programy, **panel úloh** (ktorý zobrazuje aktuálne otvorené okná a umožnuje rýchle prepínanie medzi nimi), **panel jazyka** (ktorý ukazuje a umožňuje meniť aktuálne nastavenie jazyka klávesnice) a **oblasť oznámení**, ktorá zobrazuje ikony niektorých procesov bežiacich na pozadí (napríklad instant messenger, ovládač grafickej karty, antivírový program...), prípadne stav systému (pripojenie k miestnej sieti alebo internetu, stav fronty tlačiarní, aktualizácie systému Windows...) a aktuálny čas.
Systém Windows je naprogramovaný tak, aby ho mohli bezproblémovo používať aj menej zruční používatelia, takže väčšina úkonov sa dá vykonať intuitívne a väčšina funkcií systému funguje na princípe Drag&Drop (ťahaj a polož). Vytvorenie nového odkazu je možne buď týmto spôsobom, alebo cez sprievodcu a to spôsobom pravý klik na plochu –> nový –> odkaz. Druhý spôsob je tiež aplikovateľný aj na vytvorenie priečinka (pravý klik na plochu –> nový –> priečinok). Vzhľad operačného systému Windows s dá pomerne jednoducho upravovať. Dajú sa zmeniť prakticky všetky farby, fonty a aj tvary okien, prípadne tapeta pracovnej plochy, šetrič obrazovky, rozlíšenie monitora aj množstvo použitých farieb a to všetko len pravým kliknutím na plochu a zvolenia položky vlastnosti. Výraznejšie zmeny sa dajú dosiahnuť použitím rôznych utilít (napr. WindowsBlinds, ToggleMouse a pod.)
Ok, 5 kreditov

6. Charakterizujte skupinu užívateľských programov - aplikácií. Multiprogramový režim. Schránka, prenos údajov . Základné systémové nastavenia operačného systému. Požiadavky na hardware, pridanie nového hardwaru.

Charakterizujte skupinu užívateľských programov – aplikácií

- **Textový editor** (MS Word, Word, Writer a pod.) Práca s množstvom typov a veľkostí písma na vytváranie nadpisov a zvýrazneného textu. Vytváranie stĺpcov a rámčekov s textom, orámovanie celej strany a iné efekty .
- **Tabuľkový kalkulátor** (MS Excel, Quattro, Calc atď.) Je program na vytváranie tabuliek. Je to matica čísel (textu, vzorcov) medzi ktorými je možné vykonávať matematické a textové operácie.
- **Databázové systémy** (Microsoft Access, MySQL) Programy na vytváranie a správu databáz.
- **Prezentačný softvér** (Microsoft PowerPoint) Programy na vytváranie prezentácií, multimedálnych súborov určených pre prezentovanie činnosti, alebo veci. Je možné vkladať text, obraz, video, zvuk ...
- **Antivírusové programy a antispywarové** (NOD, AVG, Avast, Norton Antivirus) slúžia na ochranu počítača pred tzv. škodlivými programami - vírusami, trójskymi koňami, spyware a pod.

- **Správa súborov** (Windows Commander, Servant Salamander, Turbo Navigator) Programy na správu súborov. Umožňujú kopírovanie, prenášanie, náhľady, sledovanie zmien, vytváranie a správu adresárovej štruktúry.
- **Zálohovacie, archivačné a kompresné nástroje** (Nero Burning ROM, WinZip, WinRAR, Power Archiver, UltimateZip, ...) Programy pre vytváranie záloh, kompresiu dát a pod.
- **Multimediálne utility** (CD a DVD prehrávač, MP3player, prezerač grafických súborov) Programy pre prehrávanie a vytváranie multimediálneho obsahu.
- **Grafický editor** (Adobe Photoshop, Paint Shop Pro, Corel Draw!, Corel Photo-Paint)
- **Vývojové prostredia programovacích jazykov** (Turbo Pascal, Borland Delphi, MS Visual Basic, ...) Sú programy na vytváranie softvéru. Vyššie programovacie jazyky umožňujú naprogramovať aplikačný softvér˘. Súčasťou balíka sú ladiace programy a simulátory.
- **Internetové nástroje** (Internet Explorer, Netscape, off-line sťahovače www stránok, FTP, downloadery, HTML editory, ...) Nástroje na prezeranie obsahu internetu, sťahovanie, vyhľadávanie stránok, sťahovanie a odosielanie pošty.
- **Systémové nástroje** (ovládače, defragmentácia) Systémové nástroje sú programy na správu operačného systému, správu a údržbu softvéru.
- **Vzdelávacie programy** (encyklopédie, slovníky, prekladače, Route66)
- **Geografické informačné systémy a GPS** Programy pre prácu s GPS, mapové softvéry, navigačné softvéry.
- **CAD a 3D modelovanie** Programy pre kreslenie stavebným a strojárskych výkresov, modelovanie, 3D výstupy, renderovanie a pod.
- **Programy na oddych a relax (počítačové hry)**

Multiprogramový režim- (multitasking): je to vykonávanie viacerých úloh súčasne. Keďže väčšina počítačov má iba jeden centrálny CPU, multitasking sa simuluje tak, že procesor striedavo vykonáva časti jednotlivých procesov a navonok sa tvári, že sa vykonávajú súčasne.
Prepínanie medzi aplikáciami – prepnutie pomocou Alt+TAB. Keď je panel úloh zaplnený tlačidlami s viacerými programami, systém Windows poskytuje funkcie, ktorá môže spravovať veľký počet otvorených dokumentov a programových položiek. Funkcia zoskupovania tlačidiel na panely úloh funguje dvoma spôsobmi. Po prvé, tlačidlá dokumentov otvorených prostredníctvom jedného programu sú vždy zobrazené v rovnakej oblasti panela úloh, čo zjednoduší hľadanie. Po druhé, ak je prostredníctvom jedného programu otvorených veľa dokumentov, systém Windows skombinuje všetky dokumenty na jedno tlačidlo na panely úloh označené názvom programu. Trojuholník na pravej strane tlačidla označuje, že je v danom programe otvorených veľa dokumentov. Ak chcete prejsť na jeden otvorený dokument, kliknete na trojuholník na tlačidle panela úloh a potom v zozname kliknete na názov dokumentu.

Schránka- špeciálny program, ktorý slúži pre ukladanie objektov (textov, obrázkov a pod.) a ich prenos v rámci jednej aplikácie, alebo medzi aplikáciami v prostredí Windows. Obsah Clipboardu, čiže schránky, vytvoria funkcie Cut alebo Copy. Kópiu obsahu schránky vykoná funkcia Paste. Ak je obsah schránky prázdny, funkcia Paste je neprístupná.
-Schránka je vlastne pomocná pamäť vo Windowse pričom sa nachádza v hlavnej skupine.
-Odlišujeme aktuálny obsah schránky, čo je aktuálny stav clipboardu, ktorý sa po reštartovaní Windowsu vymaže. Preto sa aktuálny stav clipboardu dá uložiť do súboru a tak môžeme uložiť viac obrázkov, textov a pod.
-Keď prenášame napr. obrázok, tak môžeme preniesť celý alebo len časť.
Nastavenie prostredia- najjednoduchší postup pri nastavovaní prostredia je kliknutie pravého tlačidla myši na pracovnú plochu, kde zvolíme možnosť „vlastnosti“. Tým sa nám otvorí okno s nastaveniami obrazovky a pracovnej plochy. Prvá možnosť je nastavenie pozadia pracovnej plochy. Windows má predvolené pozadia, ale môžeme si dať ako pozadie vlastný obrázok (stiahnutý s internetu, nakreslený atď) na čo nám slúži tlačidlo „prehľadávať“ a pomocou ktorého sa dostaneme do adresára, v ktorom máme daný obrázok uložený.
Druhou možnosťou nastavenia je šetrič obrazovky. Môžeme si nastaviť po koľkých minútach sa šetrič spustí. Opäť má Windows aj svoje šetrič, ale takisto si k nim pridať svoje.
Ďalšou možnosťou je nastavenia celkového vzhľadu. Čiže farbu aktívneho a neaktívneho okna, typ a farbu písma, atď.

Poslednou možnosťou je nastavenie kvality farieb a rozlíšenie obrazovky. Windows XP má ešte možnosť „motívy“, kde si môžeme zvoliť motív XP, alebo Windows s klasickým nastavením a tiež motívy online, ktoré si stiahneme s internetu.
K týmto nastavenia sa môžeme dostať aj cez ponuku *štart – nastavenia - ovládací panel - obrazovka.*
Ovládací panel- obsahuje veľa špecializovaných nástrojov, ktorými je možné meniť vzhľad a správanie systému Windows.
Niektoré z týchto nástrojov umožňujú upraviť nastavenia tak, aby sa používanie počítača stalo zábavnejším. Použite napríklad nástroj „Myš“ na nahradenie štandardných ukazovateľov myši animovanými ikonami, ktoré sa hýbu na obrazovke. Použitím nástroja „Zvuky a zvukové zariadenia“ môžete nahradiť štandardné systémové zvuky zvukmi podľa vlastného výberu. Iné nástroje umožňujú nastaviť systém Windows tak, aby bolo používanie počítača jednoduchšie. Ak ste napríklad ľaváci, môžete použiť nástroj „Myš“ na zámenu tlačidiel myši tak, že primárne funkcie výberu a presúvania sa budú vykonávať pravým tlačidlom. Ak chceme otvoriť ovládací panel, klikneme na tlačidlo *štart – nastavenia – ovládací panel*. Windows XP má dva typy ovládacieho panelu. Prvé je klasické zobrazenie (napr.Win98) a druhé je zobrazenie kategórií (Windows XP).
Klávesnica- je hlavným spôsobom zadávania textu do počítača, hoci v budúcnosti ju možno nahradia programy na rozpoznávanie rukou písaného textu a reči. Klávesnica si po zapojení nevyžaduje žiadne úpravy ani nastavenie softvéru. Ak však chcete niektoré nastavenie zmeniť, môžete tak urobiť v ovládacom paneli v položke Klávesnica. Môžete nastaviť rýchlosť, ktorou sa znaky opakujú pri podržaní klávesu. Môžete nastaviť i čas oneskorenia, po uplynutí ktorého sa znak začne opakovať. Môžete takisto upraviť rýchlosť blikania kurzora. Väčšinu činností vykonávaných myšou môžete vykonávať aj pomocou klávesnice. Počas práce v systéme Windows používajte klávesové skratky ako alternatívu k používaniu myši. Pomocou klávesových skratiek môžete otvárať, zatvárať a pohybovať sa v ponuke Štart, na pracovnej ploche, v ponukách, dialógových oknách a webových stránkach. Klávesové skratky nám taktiež uľahčujú prácu s počítačom.
Myš- je špeciálne vstupné zariadenie, ktoré doplňuje klávesnicu. Počítač, resp. jeho základný operačný systém MS-DOS nie je schopný myš sám rozoznať a pracovať s ňou, a preto je nutné nainštalovať ovládač dodaný s myšou. Funkcie myši (ukazovateľ, dvojkliknutie, koliesko myši) môžeme nastaviť v ovládacom panely. Myš má primárne a sekundárne tlačidlo. **Primárne tlačidlo:** používa sa na výber a kliknutie na položky, na umiestnenie kurzora v dokumente a presúvanie položiek.

Sekundárne tlačidlo: používa sa na zobrazenie ponuky úloh a možností, ktorá sa mení v závislosti od oblasti, na ktorú kliknete. Kliknutie sekundárneho tlačidla myši sa nazýva kliknutie pravým tlačidlom myši.
Ľaváci si môžu primárne a sekundárne tlačidlá vymeniť v ovl. panely. Väčšina myší má v súčasnosti koliesko, ktoré slúži na jednoduchšie prechádzanie dokumentov. Koliesko sa môže tiež použiť ako tretie tlačidlo.
Dvojkliknutím primárneho tlačidla na ikonu sa otvorí program, hra, dokument, atď. v ovládacom panely môžeme nastaviť rýchlosť dvojkliku.
Miestne nastavenia- pomocou panelu „Miestne a jazykové nastavenie“ môžete nainštalovať na počítač viacero jazykov, ako napríklad hebrejčinu, arabčinu, japončinu, kórejčinu a západoeurópske jazyky (francúzštinu, španielčinu, nemčinu a mnohé iné.) Môžete si vybrať, ktorý z nich budete používať pri vytváraní dokumentu. Systém Windows potom sprístupní tabuľku znakov pre tento jazyk a vy môžete začať písať.

Požiadavky na hardware
Každá aplikácia-program má minimálne požiadavky na hardware, s ktorými ide daná aplikácia-program spustiť, a optimálne požiadavky na hardware pre optimálne používanie programov.
Pridanie nového hardwaru
Po vložení nového harwaru sa objaví diaógové okno Found Nw Hardware Wizard- pomocník pre pridanie nového hardwaru. Pomocou sprievodcu dokončíme inštaláciu..
OK, 5 kreditov, prehádzal som kapitoly

7. Dôvody budovania sietí. Charakteristika, popis, rozdelenie sietí, topológia sietí. Počítačová sieť Internet. Poskytovateľ služieb, spôsoby pripojenia do siete Internet.

Dôvody zavádzania počítačových sietí:
Počítačové siete prinášajú niekoľko zásadných výhod:

- Zdieľanie údajov: potrebné dátové súbory môže spracovávať viac používateľov súčasne; je to umožnené tým, že dátové súbory sú uložené na serveroch siete a pripojený používatelia majú k nim prístup
- Zdieľanie prostriedkov: umožňuje pracovným staniciam spoločne používať prostriedky siete, ktoré ponúkajú servery siete(zdieľanie diskov, tlačiarní...)
- Zvýšenie spoľahlivosti systému: napr. ak máme zdieľanú tlačiareň, tak v prípade poruchy môžeme ju nahradiť inou a systém môže pracovať ďalej; v prípade poruchy počítača môžeme pokračovať v práci na druhom, ak máme uložené potrebné súbory na servery...

Charakteristika:

Počítačová sieť je systém vzájomne prepojených a spolupracujúcich počítačov. Medzi týmito počítačmi môžeme prostredníctvom siete pohodlne a rýchlo prenášať informácie, na rozdiel od pomalších spôsobov (rôzne médiá,...)

Základné časti siete:

Sieť pozostáva z nasledujúcich základných častí:

- Hardvér (samotné počítače, smerovače, prepínače, koncentrátory a rozbočovače, sieťové mosty, meniče rozhraní, bezpečnostné, opakovače, modulátory/demodulátory, vysielače/prijímače, zábrany, káble)
- Softvér(Sieťový operačný systém, ...)

Druhy počítačov v sieti:

Podľa funkcie sa delia na:

- Pracovné stanice: samostatný počítač pripojený k sieti a tak môže využívať jej služby
- Servery: zabezpečuje chod siete, realizuje funkcie siete a poskytuje ostatným používateľom svoje prostriedky (pamäťové miesto na svojich diskoch, tlačiarne, plotre...)

ZÁKLADNÉ DRUHY POČÍTAČOVÝCH SIETÍ

Podľa typov pripojených počítačov:

- Homogénne: všetky pripojené počítače sú rovnakého druhu (operačné systémy môžu byť rôzneho druhu)
- Heterogénne: môžu obsahovať viacej druhov počítačov; napr. verejné dátové siete

Podľa rozlohy sietí:

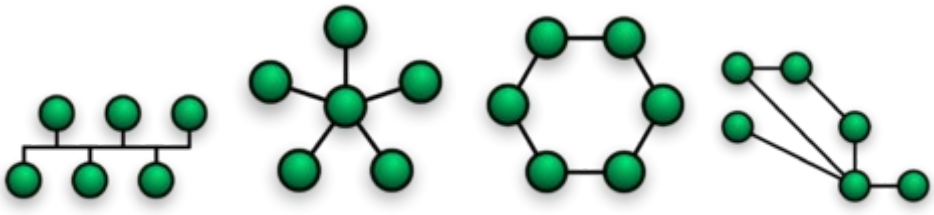
- Lokálne(LAN): nepoužívajú sa prostriedky pre diaľkový prenos údajov(modemy); max. vzdialenosti sú rádovo stovky metrov; väčšinou sa využívajú v jednej, resp. viacerých blízkych budovách
- Mestské (MAN): metropolitné, spájajú verejné inštitúcie, školy, ...
- Globálne(WAN): používajú prostriedky pre diaľkový prenos údajov; rozloha je podstate neobmedzená; spájajú sa počítače na väčších územiach(štát, kontinent...)

TOPOLÓGIA SIETÍ:

Kľúčovú úlohu v sieťach akéhokoľvek typu zohrávajú aktívne prvky. Ich úlohou je prepájať jednotlivé časti sietí, menia typy rozhraní, spôsoby komunikácie, zaisťujú bezpečnosť a riadia sieť. Sieť môže byť prepojená priamo z počítača do počítača, alebo s využitím týchto aktívnych prvkov.

Topológia = spôsob prepojenia počítačov

a) zbernica
b) hviezda
c) kruh
d) neobmedzená



Zbernicová **Hviezdicová** **Kruhová** **Neobmedzená(mesh)**

INTERNET:

Internet, hovorovo **net** alebo **sieť** je verejne dostupný celosvetový systém vzájomne prepojených počítačových sietí, ktoré prenášajú dáta pomocou prepínania paketov za použitia štandardizovaného **Internet Protocolu (IP**) a mnohých ďalších protokolov. Pozostáva z tisícok menších komerčných, akademických, vládnych a vojenských sietí. Slúži ako prenosové médium pre rôzne informácie a služby ako napr. elektronická pošta, chat a systém vzájomne prepojených webstránok a dokumentov **World Wide Webu (WWW**).

POSKYTOVATEĽ INTERNETOVÝCH SLUŽIEB (ISP):

Poskytovateľ internetových služieb **(ISP)** je spoločnosť, ktorá zvyčajne za poplatok poskytuje prístup na Internet. Najbežnejším spôsobom pripojenia k poskytovateľovi internetových služieb je pripojenie pomocou **telefónnej linky (telefonické pripojenie)** alebo **širokopásmové pripojenie (káblové alebo DSL).** Mnohí poskytovatelia internetových služieb poskytujú aj doplnkové služby, napr. e-mailové kontá, webové prehľadávače alebo miesto na vytvorenie webovej lokality.

Spôsoby pripojenie do siete Internet:

- Telefónne technológie(DIAL-UP, ISDN,DSL)
- Bezdrôtové(WLAN,GSM,...)
- Káblové technológie(televízne káble,...)
- Optické technológie(optické káble)

OK, 5 kreditov

8. Adresa URL, IP adresa, doménový systém. Služby klient-server. HTTP protokol. Prehľad základných služieb - finger, telnet, ftp, ping, traceroute.

URL (Uniform Resource Locator) - je skratka pre Uniform Resource Locator, zjednodušene povedané adresa lokácie na sieti internet. URL je taký URI (Uniform Resource Identifier), ktorý okrem identifikácie zdroja umožňuje jeho lokalizáciu pomocou opisu primárnej prístupovej metódy k nemu (napr. jeho "umiestnenia" v sieti).

URI (Uniform Resource Identifier) - doslova jednotný identifikátor zdroja, skratka URI, je kompaktný reťazec znakov používaný na identifikáciu alebo pomenovanie zdroja. Hlavný účel tejto identifikácie je umožniť interakciu s prezentáciami zdroja cez sieť, typicky cez World Wide Web, použitím špecifických protokolov. URI sú definované v schémach definujúc špecifickú syntax a asociované protokoly.

IP (Internet Protocol) adresa - je logický číselný identifikátor fyzického sieťového rozhrania (sieťovej karty) daného uzla (najčastejšie počítača) v sieti, ktorý komunikuje s inými uzlami prostredníctvom protokolu IP (napríklad Internet). IP adresa je (v protokole IP verzie 4) 32-bitové číslo, takže teoreticky existuje 4 294 967 296 (viac než 4 miliardy) možných adries. Je nepraktické a nepohodlné pracovať s takto zapísaným číslom, preto sa 32 bitov IP adresy delí na štyri 8-bitové čísla (číslo v rozsahu 0-255), ktoré sa zapisujú v desiatkovej sústave oddelené bodkou, napríklad 207.142.131.205.

DNS (Domain Name System) - je systém, ktorý ukladá prístup k informácii o názve stroja (hostname) a názve domény v istej distribuovanej databáze v počítačových sieťach ako internet. Najdôležitejšie je, že poskytuje mechanizmus získania IP adresy pre každé meno stroja (lookup) a naopak (reverse), a uvádza poštové servery (MX záznam) akceptujúce poštu pre danú doménu.

DNS poskytuje na internete všeobecne dôležitú službu, pretože kým počítače a sieťový hardvér pracujú s IP adresami, ľudia si vo všeobecnosti ľahšie pamätajú mená strojov a domén pri použití napr. v URL a e-mailovej adrese (obzvlášť nepríjemné by to bolo pri IPv6 adrese). DNS tak tvorí prostredníka medzi potrebami človeka a softvéru.

HTTP (Hypertext transfer protocol) – je protokol definujúci požiadavky a odpovede medzi klientmi a servermi. HTTP klient (označovaný ako user agent), ako webový prehliadač zvyčajne začne požiadavku nadviazaním TCP spojenia na určenom porte vzdialeného stroja (štandardne port 80). HTTP server počúvajúci na danom porte čaká, kým klient pošle reťazec s požiadavkou ako "GET / HTTP/1.1" (ktorý

žiada o zaslanie štartovacej stránky webservera) nasledovaný sériou hlavičiek podobných MIME opisujúcich detaily požiadavky a nasledovaných telesom ľubovoľných údajov. Niektoré hlavičky sú nepovinné, zatiaľ čo verzia HTTP/1.1 niektoré vyžaduje (ako názov stroja). Po prijatí požiadavky server pošle reťazec s odpoveďou ako "200 OK" nasledovanou hlavičkami spolu so samotnou správou, ktorej telo tvorí obsah požadovaného súboru, chybové hlásenie alebo iná informácia.

Architektúra klient-server je v súčasnosti jednou z architektúr, ktorá sa veľmi často používa v databázových technológiách a je ju možné nájsť vo viacerých variantoch. Základným predpokladom je, že aplikácia beží v počítačovej sieti a funkčnosť aplikácie je rozdelená do dvoch vrstiev:

- Klient
- Server

Pri komunikácii typu Klient - Server je jedna aplikácia typu Server a druhá typu Klient. Aplikácia typu server vystupuje v komunikácii ako poskytovateľ služieb, aplikácia typu klient zase využíva služby poskytované serverom. Jedná sa o centralizovaný prístup, pretože k jednému serveru sa pripája viac klientov a server vystupuje ako centrálny bod v takejto komunikácii. Všetky informácie o komunikujúcich uzloch sú uložené na serveri a klientom sú poskytnuté len informácie, ktoré nutne potrebujú na svoju činnosť (napr. zoznam pripojených používateľov).

Prehľad základných služieb
finger - zobrazuje informácie aký užívatelia sú pripojení k hostiteľskému počítaču. Pomocou tohoto príkazu je možné zistiť i podrobnejšie informácie o jednotlivých užívateľoch. Ak za príkaz uvedieme meno užívateľa, sú zobrazené informácie o danom uživateľovi aj pokiaľ nie je k hostiteľskému počítaču pripojený. Nie všetky servery príkaz finger automaticky umožňujú, niektoré ho síce používajú, ale s určitými zákazmi.

Príkaz je možné použiť s nasledujúcimi prepínačmi (finger [-slpm])
-s, stručný výpis
-l, kompletní výpis
-p, nezobrazujú sa informácie zo súborov .plan, .foward a .project
-m, u užívateľov sa zobrazuje iba užívateľské meno

ping - je nástroj používaný v TCP/IP protokoloch, pomocou neho sa vykonáva základný test, či funguje spojenie na určitého hostiteľa (server, router,...) z testovacieho hostiteľa. Po zadaní príkazu ping sú z testovacieho hostiteľa vysielané na cieľového hostiteľa ICMP pakety a testovaná odozva. Ide o analógiu sonaru u ponoriek (ping znamená v angličtine bzučať, zvoniť,...).

telnet - protokol a príkaz pre vzdialené terminálové prihlásenie k serveru v sieťach TCP/IP a internet. V súčasnosti sa nepoužíva, lebo je nešifrovaný a tým pádom vzniká zraniteľnosť. Šifrovanou obdobou protokolu telnet je SSH.

FTP (File Transfer Protocol) - je protokol aplikačnej vrstvy z rodiny TCP/IP, je určený pre prenos súborov medzi počítačmi, na ktorých môžu bežať veľmi rozdielne operačné systémy.

traceroute – je nástroj, ktorý sa používa k zisťovaniu tras, po ktorých sa datagramy IP dostávajú na cieľovú adresu. Príkaz traceroute používa pole TTL (Time-to-Live) protokolu IP a chybové správy ICMP a na ich základe zisťuje trasu medzi hostiteľmi v sieti. (obdoba príkazu pre Windows je tracert)
Ok, 5 kreditov

9. Kryptografia a šifrovanie dát. Digitálny podpis. Právne dôsledky. Bezpečnosť počítačovej siete.

Kryptografia je oblasť kryptológie, ktorej cieľom je skúmanie a navrhovanie šifrovacích systémov, ktoré budú spĺňať určité podmienky ako autenticitu, dôvernosť, dostupnosť, integritu a pôvod dát. Ich úlohou je teda urobiť určitý obsah správy nečitateľným v prípade jeho zachytenia treťou osobou.

Substitučné šifrovacie systémy
Substitučná šifra nahrádza každý znak otvoreného textu iným znakom šifrovaného textu. Aby príjemca získal otvorený text, musí na zašifrovaný text použiť inverznú substitúciu. V klasickej kryptografii existujú štyri typy substitučných šifier:

1. Jednoduchá substitučná šifra (monoalfabetická šifra) je taká šifra, v ktorej sa každý znak otvoreného textu nahrádza príslušným znakom šifrovaného textu.
2. Homofónna substitučná šifra sa podobá jednoduchému substitučnému systému. Jeden znak otvoreného textu môže byť nahradený jedným znakom z niekoľko možných znakov šifrovaného textu. Znak "A" by mohol byť nahradený napr. 5, 13, 25 alebo 56, "B" napr. 7, 19, 31 alebo 42 atď.
3. Polygramová substitučná šifra je taká, v ktorej šifrovanie prebieha medzi skupinami znakov. Napríklad skupina "ABA" môže byť nahradená skupinou "RTQ", "ABB" skupinou "SLL" atď.
4. Polyalfabetická substitučná šifra sa skladá z niekoľkých jednoduchých šifier. Napríklad túto šifru môže tvoriť päť rôznych, jednoduchých substitučných šifier; jednotlivé šifry sú postupne aplikované na jednotlivé po sebe idúce znaky otvoreného textu. Medzi substitučné šifry patrí napríklad Caesarova šifra alebo Vernamova šifra.

Transpozičné šifrovacie systémy
Pri transpozičných šifrách sa znaky otvoreného textu nemenia, mení sa iba ich usporiadanie. Pri jednoduchej stĺpcovej transpozičnej šifre sa otvorený text zapisuje horizontálne na štvorčekovaný papier a zašifrovaný text sa získava jeho vertikálnym čítaním. Pri dešifrovaní sa šifrovaný text zapíše vertikálne opäť na štvorčekovaný list papiera rovnakej šírky a číta sa horizontálne.

Pretože znaky šifrovaného aj otvoreného textu sú rovnaké, bude sa frekvenčný výskyt jednotlivých znakov šifrovaného textu približne rovnať frekvenčnému výskytu znakov v príslušnom národnom jazyku. Navyše je tento systém náročný na pamäť a niekedy aj obmedzuje dĺžku správ. Sem patria napríklad: Cardanova mriežka, Richelieuova šifra.

Symetrické šifrovacie systémy
V základnom delení bolo spomínané, že medzi tradičné šifrovacie systémy patria aj symetrické šifry. Využívajú vhodne zvolené kombinácie substitučných a transpozičných algoritmov. Tento šifrovací systém sa nazýva symetrický preto, lebo šifrovací kľúč je možné odvodiť z dešifrovacieho a naopak. Väčšina symetrických algoritmov má rovnaký šifrovací aj dešifrovací kľúč. Tieto algoritmy označované ako algoritmy tajného kľúča vyžadujú, aby sa príjemca aj odosielateľ vopred dohodli na kľúči, ktorý budú používať. Bezpečnosť je založená na utajení kľúča. Ak sa používa dlhšiu dobu, vzniká nebezpečenstvo, že kľúč padne do nepovolaných rúk a obsah utajovaných informácií sa prezradí. Preto sa kľúče často obmieňajú - nazývame ich aj "kľúče pre sedenie" (session - relácia).

V dnešných algoritmoch je typická dĺžka 64 bitov. V minulosti sa otvorený text šifroval po jednom znaku. Najbežnejšie používané symetrické algoritmy sú Rivest Cipher 5 (RC5), Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA), Blowfish Skipjack a AES.

Asymetrická kryptografie
Asymetrická kryptografia je skupina kryptografických metód, v ktorých sa pre šifrovanie a dešifrovanie používajú odlišné kľúče.
Okrem očividnej možnosti pre utajenie komunikácie sa asymetrická kryptografie používa aj pre elektronický podpis, tzn. možnosť preukázať autora dát.
Šifrovací kľúč pre asymetrickú kryptografiu pozostáva z dvoch častí:
jedna časť sa používa pre šifrovanie správ (a príjemca správy ani túto časť nemusí poznať), druhá pre dešifrovanie (a odosielateľ šifrovaných správ ju spravidla nepozná). Je vidieť, že ten, kto šifruje, nemusí s dešifrujúcim príjemcom správy zdieľať žiadne tajomstvo, čím eliminujú potrebu výmeny kľúčov; táto vlastnosť je základná výhoda asymetrickej kryptografie.

Najbežnejšou verziou asymetrickej kryptografie je využívanie tzv. verejného a súkromného kľúča: šifrovací kľúč je verejný, majiteľ kľúča ho voľne uverejní, a ktokoľvek ním môže šifrovať jemu určené správy; dešifrovací kľúč je súkromný, majiteľ ho drží v tajnosti a pomocou neho môže tieto správy dešifrovať. (Existujú i ďalšie metódy asymetrickej kryptografie, v ktorých je treba i šifrovací kľúč udržiavať v tajnosti.)
Je zrejmé, že šifrovací kľúč **e** a dešifrovací kľúč **d** spolu musia byť matematicky zviazané, avšak nevyhnutnou podmienkou pre užitočnosť šifry je praktická nemožnosť so znalosťou šifrovacieho kľúča vypočítať dešifrovací.

Matematicky tu asymetrická kryptografia postupuje nasledujúcim spôsobom:
Šifrovanie $c = f(m, e)$
Dešifrovanie $m = g(c, d)$
V princípe sa môžu šifrovacie a dešifrovacie funkcie líšiť, spravidla sú však matematicky prinajmenšom veľmi podobné.

Asymetrická kryptografia je založená na tzv. jednocestných funkciách, čo sú operácie, ktoré sa dajú ľahko vykonať iba v jednom smere: z vstupu ide ľahko vypočítať výstup, z výstupu je však veľmi zložité nájsť vstup. Najbežnejším príkladom je napríklad násobenie: je veľmi ľahké vynásobiť dve i veľmi veľké čísla, avšak rozklad súčinu na činiteľov (tzv. faktorizácia) je veľmi náročný. (Na tomto probléme je založený napr. algoritmus RSA.) Patrí sem napr: RSA, ElGamal, Kryptografia nad eliptickými krivkami, Kryptografia s Lucasovými funkciami, Diffie-Hellman, Digital Signature Algorithm, Merkle-Hellman (už zastaralý)

Digitálny podpis
Za elektronický podpis sa v širšom význame považuje aj prosté nešifrované uvedenie identifikačných údajov (napríklad mena a adresy, názvu a sídla, rodného alebo iného identifikačného čísla atd.) na konci textu v elektronickej podobe, ktoré zaručuje identifikáciu označené osoby, avšak nie integritu podpísaného dokumentu ani autenticitu podpísaného.
Zaručený elektronický podpis je elektronický podpis v takej forme, ktorá, spravidla kryptografickými metódami, zaručuje i integritu dokumentu a autenticitu podpísaného. Pre niektoré účely je naviac vyžadovaný zaručený elektronický podpis iba s predpísanými typmi certifikácie, teda „založený na kvalifikovanom certifikáte“. Zaručený elektronický podpis zaisťuje:
autenticitu (nepopierateľnosť) – ide preukázať, že autorom je skutočne ten, kto je pod dokumentom podpísaný, autor nemôže poprieť, že dokument podpísal.
integritu dokumentu – ide preukázať, že po podpísaní nedošlo k žiadnej zmene, súbor nie je poškodený (ani zámerne, ani omylom), niekedy má aj funkciu časového razítka, teda preukazuje dátum a čas podpísania dokumentu.

Zaručený elektronický podpis je aplikáciou kryptografie s verejným kľúčom. Pri podpisovaní sa používa tajný kľúč autora – keďže autor je jediný, kto k nemu má prístup, je zrejmá autenticita. Podpis potom môže overiť ktokoľvek pomocou autorovho verejného kľúča.
Z praktických dôvodov sa takto nespracováva celý dokument, ale len jeho hash, veľmi krátky (typicky niekoľko sto bitov) výťah vytvorený špecializovaným algoritmom z celého dokumentu. Tento hash sa potom zašifruje autorovým tajným kľúčom, čím vznikne podpis. Overenie podpisu potom spočíva v dešifrovaní hashu (podpisu) pomocou verejného kľúča autora, nezávislého výpočtu hashu z dokumentu a porovnania oboch hodnôt. Pokiaľ si odpovedajú, potom je podpis overený a dokument je považovaný za dôveryhodný. Autor nemôže poprieť svoje autorstvo, lebo k jeho tajnému kľúču nikto iný nemá prístup, a naopak, nikto iný nemôže zašifrovať hash dokumentu tak, aby po aplikácií autorova verejného kľúča vznikla správna hodnota. Dokument po podpísaní nemôže byť zmenený, lebo potom hash vychádza inak.

Právne dôsledky
V niektorých krajinách je alebo bolo domáce použitie kryptografie zakázané. Napr. Francúzsko do roku 1999 nepovoľovalo používanie kryptografie v domácnostiach. V Číne je stále na kryptovanie potrebné

povolenie. V niektorých krajinách sú zákony ešte prísnejšie napríklad v Bielorusku, Kazachstane, Mongolsku, Pakistane, Rusku, Singapure, Tunisku, Venezuele a Vietname.

Bezpečnosť počítačovej siete
Počítačová sieť je vždy taká bezpečná ako jej najslabší článok. Každá sieť by mala byť od internetu oddelená firewallom alebo proxy serverom, kvôli filtrovaniu portov a služieb, ktoré nie sú potrebné pre chod danej siete (napr. filtrovanie P2P, telnetu atd.), prípadne filtrovanie potencionálne nebezpečných adries. Každý počítač v sieti, by mal mat patričné zabezpečenie, hlavne antivírový program, firewall, prípadne nejaký software na odstránenie spyware atd. Jednotlivý používatelia by nemali používať konto s administrátorskými právami, pokiaľ to nie je vyslovene nutné a mali by byť hlavne opatrní, lebo najslabší článok v bezpečnosti je človek.
OK, 5 kreditov

10. Programy elektronickej pošty - mail, pine, IMP klient. Porovnanie, možnosti programov. Režim pošty. Reply. Forward. Prílohy

Elektronická pošta, niekedy tiež **e – mail,** je spôsob neintaraktívnej komunikácie. Väčšina správ sú napísané na počítači (vytvorené) a počítače, počítačové siete sú použité k ich prenosu na miesto určenia. Týmto miestom je tiež počítač či presnejšie povedané miesto na disku tohto špeciálneho počítača.

ADRESA
K tomu, aby naša správa dorazila ku svojmu adresátovi je potrebné poznať jeho adresu. Tejto časti adresy sa tiež hovorí **doménová časť** a skladá sa z tzv. **domén** navzájom oddelených bodkou.
Doména – je časť adresy, ktorá identifikuje umiestnenie počítača, na ktorý bude správa poslaná **.** Za poslednou bodkou sa nachádza tzv. Top Level Domain (TPL) vrcholová doména, ktorá môže byť odvodená od TPL štátu, alebo zamerania organizácie.
Napr. väčšina štátov na svete má svoju doménu. Je to tá časť adresy, ktorá sa píše na koniec adresy. Pre Slovensko je to **sk**, Česko **cz.** Doména **com** znamená, že ide o komerčnú organizáciu**.** Meno a doménová časť sú od seba oddelené zavináčom - **@.** Celá adresa môže vyzerať takto: **login@golem.gymzv.sk**, kde sk je TLD, gymzv je registrovaná doména druhej úrovne a golem je názov servera.

POŠTOVÝ KLIENT
Poštový klient je program, ktorý používame k odosielaniu o prijímaniu správ. Tento klient môže byť celkovo jednoduchý (UNIXový elm) program s obmedzenými schopnosťami a možnosťami, ako aj aplikácie, ktoré poskytujú užívateľovi nástroje nie len pre odoslanie a prijímanie pošty, ale tiež aj napr. rôzne operácie so správami a veľa ďalších. Takýmito poštovými klientami sú napr. IMP KLIENT, Outlook Expes apod.

- **Možnosti klienta, režim pošty**
  - Zobrazenie vybratej schránky – napr. inbox – došlá pošta
  - Zobrazenie obsahu správy
  - Vymazania správ
  - **Reply –** odpoveď na správu s možnosťou začleniť pôvodný text do odpovede,spiatočná adresa sa vyplní automaticky sama
  - **Forward** – distribúcia správy na ďalšie iné adresy
  - **Posielanie príloh – attachment** – ukladanie,pripojenie súborov rôznych formátov / word dokumenty, obrázky /
  - Uloženie príloh do súborov v adresári
  - Vytváranie položiek v address book

MAIL – umožňuje nám ovládanie režimu pošty cez príkazy v textovom režime
PINE - tak isto v textovom režime, program má hlavné menu + zodpovedujúcu nápovedu
základná obrazovka – 1. Riadok – názov programu

- Položky hlavného menu sú

- **HELP** – pomocné informácie
- **COMPOSE MESSAGE** - posielanie správ aj možnosť pripojenia príloh
- **MESSAGE INDEX** – hlavička, režim pošty
- **FOLDER LIST** – zoznam adresárov
- **ADDRESS BOOK** - vytvorenie prezývky
- **SETUP** – nastavenie konfigurácie a parametrov
- **QUIT -** ukončenie programu

POŠTOVÝ SERVER

Pokiaľ máme na svojom počítači nainštalovaný niektorý z poštových klientov, neznamená to ešte, že už môžeme automaticky komunikovať prostredníctvom elektronickej pošty. Okrem ďalších podstatných vecí je potrebné mať k dispozícii určitý priestor na tzv. **poštovom serveri. Poštový server** je počítač, na ktorý sú smerované všetky správy s určitým doménovým menom. A vtedy je podľa mena rozpoznané, do ktorého miesta na disku sa má daná správa uložiť. Tomuto miestu sa hovorí **poštová schránka**. Do príslušnej poštovej schránky ma prístup len ten, kto má tzv. účet na príslušnom poštovnom serveri a pre koho bola táto schránka zariadená. Účet u príslušného poštového serveru znamená právo prihlásiť sa k tomuto serveru pod určitým menom /heslom/ a používať jeho služby.

Protokoly pre prácu s e-mailom
E-mailové správy sú posielané e-mailovému serveru, ktorý ukladá prišlé správy v príjemcovom mailboxe. Užívateľ neskôr znovu získava tieto správy buď cez webový prehliadač, alebo cez e-mailového klienta, ktorý používa jeden z e-mailových protokolov.

SMTP - (Simple Mail Transfer Protocol) je internetový protokol určený na prenos správ elektronickej pošty medzi stanicami. Protokol zaisťuje doručenie pošty pomocou priameho spojenia medzi odosielateľom a adresátom; správa je doručená do tzv. poštovnej schránky adresáta, ku ktorej potom môže užívateľ kedykoľvek (off-line) pristupovať (vyberať správy) pomocou protokolov POP3 alebo IMAP
POP3 - (Post Office Protocol version 3) je internetový protokol, ktorý sa používa na sťahovanie emailových správ zo vzdialeného serveru na klienta. Jedná sa o aplikačný protokol pracujúci cez TCP/IP pripojenie. Adresa POP3 servera s poštou je `pop3.bla.gymzv.sk`, username a heslo je rovnaké ako pre Vaše konto. POP3 umožňuje dostať poštu z poštového servera na pracovnú stanicu, a tak pohodlne pracovať napr. s pripojenými súbormi (dokumenty, obrázky). Tento prístup má aj nevýhody:
[1]POP3 používa nechránenú komunikáciu, takže niekto môže Vaše heslo zachytiť a dostať sa k vašej pošte resp. kontu. V rámci fakultnej siete je toto riziko pomerne malé, ale podstatne rastie pri prihlasovaní sa z počítačov mimo nej (zahraničie, iný než fakultný modem).
[2] POP3 klient je štandardne nastavený tak, aby po prevzatí poštu zo servera zmazal. Nie je vhodný, pokiaľ sa pošta číta z rôznych počítačov.
IMAP - je protokol pre prístup k e-mailovým schránkam. V súčasnej dobe sa používa verzia IMAP4. Je optimalizovaný pre prácu v dlhodobo pripojenom režime, kedy správy zostávajú uložene na serveri a priebežne sa sťahujú, keď je treba. Zahrňuje podporu pre prácu viac pripojených klientov zároveň, uchovávanie stavov správ na serveri, podporu viacerých zložiek a prehľadávanie správ na strane serveru.
OK 5 kreditov

11. WWW server, www klient, browser. Verejné FTP servery, prenos súborov. Vyhľadávanie informácií, rozdelenie vyhľadávačov, spôsob práce. Spôsoby komunikácie na Internete, priama a nepriama komunikácia. Diskusné skupiny, ICQ, IRC, videokonferencie. Internetový obchod, internetové bankovníctvo, Netiketa na internete.

WWW server

Je to zariadenie,(HW+SW) pripojené k Internetu, prijíma a spracúva **požiadavky** (requests) od klienta (zariadenie, ktoré používa používateľ (user) na prehliadanie webu – napríklad internetový prehliadač (browser))

Jedná sa o druh počítača, na ktorom je nainštalovaný patričný program (software) ktorý umožňujú prijímať a vykonávať dane príkazy, napríklad posielanie súborov (files). Na serveroch môžu byť rôzne programy, ktoré slúžia na chod rozsiahlejšieho systému (interpretácia php, alebo asp scriptov, mod_rewrite podobne).
Väčšinou bežia na operačnom systéme Linux, alebo Windows, najznámejší systém pre webserver je **Apache**.

WWW klient

Zriadenie alebo program, ktorý je na strane používateľa a je schopný komunikovať so serverom. Môže sa jednať napríklad o **internetový prehliadač** (browser), alebo RSS čítačku a podobne.

Browser (internetový prehliadač)

Primárne slúži na **zobrazovanie html dokumentov**, moderné prehliadače však už zvládajú aj rôzne iné služby ako napríklad e-mailový klient a podobne. Je to program, ktorý komunikuje so serverom pomocou **http** (hypertext transfer protocol) a prijíma dáta, ktoré následné zobrazia používateľovi. (Opera, Firefox, Safari).

Verejné FTP servery

Slúžia na ukladanie a zdieľanie väčších dať, dajú sa využívať bez mena a hesla s anonymnom prihlásením.

Prenos súborov

Na prenos súborov existuje viacero protokolov. Medzi často používané patrí **FTP** (file transfer protocol). Jedná sa síce o starší, ale stále používaný protokol na prenos súborov medzi klientom a FTP serverom.
P2P – moderné protokoly na zdieľanie a prenos dát medzi užívateľmi. Ide o niekoľko rôznych protokolov, ktoré majú spoločné to, že v p2p sieti neexistuje server a dáta prúdia medzi používateľmi (ktorí sú si rovní – od toho názov p2p (peer to peer)). Protokoly: Bittorent, Limeware, FastTrack

Vyhľadávanie informácii

Medzi jednu z najdôležitejších veci v súčasnom internete patrí vyhľadávanie informácií, slúžia na to špecializované stránky tzv. **vyhľadávače a katalógy.**

Vyhľadávače

Vyhľadávajú informácie **fulltextovo**. Špeciálne programy nažívané tiež roboti (web crawler, webspider) prechádzajú weby a indexujú jednotlivé stránky. Databáza, ktorú nazbierajú títo roboti, sa spracúva a obsah sa triedi a boduje podľa obrovského množstva parametrov ako je napríklad výskyt kľúčových slov a podobne.

Jeden z najznámejších vyhľadávačov je **Google**. V minulosti niesol názov BackRub a bol naprogramovaný dvojicou študentov (Larry Page, Sergey Brin). Stránky hodnotí podľa množstva faktorov, napríklad podľa množstva, pozície a kvality kľúčových slov, podľa množstva a kvality webov odkazujúcich na daný web, pričom webom sú prideľované body (technológia sa nažíva **Pagerank** (nazvane podľa Lariho Pagea)). Verejnosti je prístupný Google Toolbar Pagerank, ktorý ma rozsah stupnice 0 – 10, PR 10 dosahujú len veľké weby napríklad: *Adobe.com*, *Google.com*, *Apple.com*.
Medzi známe vyhľadávače patrí okrem *Google.com* aj *Yahoo.com*, *Baidu.com* a Livesearch.com

Katalógy

Na vyhľadávanie informácii slúžia tiež katalógy, informácie sú v nich triedené do **kategórií** a väčšinou sú tam jednotlivé weby **pridávané ručné** administrátormi. Kategórie sú delené na podkategórie a tie na ďalšie podkategórie. K požadovanej stránke sa dá dostať preklikávaním sa, alebo použitím vyhľadávacieho pola, kde na základe zadaného kľúčového slova zobraz relevantné odkazy. Niekoľko slovenských katalógov: *Zoznam.sk, Surf.sk, Atlas.sk*

Komunikácia na internete

Komunikácia medzi ľuďmi je dôležitá odjakživa, avšak s nástupom moderných technológii sa výrazné mení.

Priama komunikácia

Účastníci sa na komunikácii podieľajú aktívne, komunikácia prebieha v reálnom čase.

Instant messaging

Jedná sa o okamžité prijímanie a posielanie správ. Používajú sa rôzne protokoly. Na Slovensku je pomerne rozšírené **ICQ** a **Jabber**, tieto protokoly majú väčšinou implementované aj prijímanie a odosielanie súborov, obsahujú rôzne pridané funkcie, ktoré majú zjednodušiť alebo ozvláštniť komunikáciu, napríklad smailíky, hry, a podobne.

IRC, Chat

Ide o realtime diskusiu, ale na rozdiel od IM tam komunikuje viac osôb spoločné na „sklo“ a medzi sebou môžu posielať súkromné správy. Napr. *Pokec.sk*

VoIP (Voice over IP), Videokonferencie

Ako vývoj pokračuje dopredu, na komunikáciu sa nepoužíva už len pišané správy, ale aj hlas a obraz. Účastníci potrebujú **web kameru, slúchadlá** a **mikrofón**. Komunikácia prebieha cez programy, ktoré podporujú daný typ prenosu, video a hlas zvládnu programy ako napr. Skype, ICQ.

Nepriama komunikácia

Užívatelia odošlú správu a ostatní reagujú na ňu neskôr (napr. Pri emailoch je bežnosť odpovedať do 2 pracovných dní). **E-maily, diskusné skupiny, fóra**.

Diskusné skupiny

Ide o skupinu užívateľov ktorí spolu komunikujú cez e-maily a systém ktorý to spravuje. Užívatel pošle e-mail na adresu automatu a ten to prepošle všetkým registrovaným. Diskusné skupiny sa rozdelujú podľa určitého zamerania (rybáry, linuxáci a podobne).

E-shopy

S nástupom moderných technológii sa rozširuje aj predaj výrobkov cez internet. Predáva sa v internetových obchodoch, sú to **webové aplikácie** kde návštevník pridáva tovar do virtuálneho **košíku**. Po nakúpení zaplatí napríklad kreditnou kartou, alebo špecializovaným systémom na platbu cez internet ako je napríklad paypal, paysec a podobne. Zákazník dostane e-mailom potvrdenie o objednávke.
Každý internetový obchod obsahuje základne prvky ako je košík, pridanie do košíka, detaily výrobku, úprava účtu a podobne.
Prevádzkovatelia obchodu vidia jednotlivé objednávky v administračnom systéme, môžu meniť ponuku výrobkov, ceny a podobne.
Každý spotrebiteľ by mal vedieť že do **14** dní ma právo výrobok vrátiť bez udania dôvodu.
Príklady: *hej.sk, alza.sk.*
K obľúbeným systémom patria tiež online aukcie, kde sa dá výhodné nakupovať a predávať. Predávajúci stanový podmienky a potencionálni kupci zvyšujú ponúkanú sumu.
(*ebay.com, aukro.cz*)

Internet banking

Je moderná forma bankovníctva. **Platiť** za služby a kontrolovať stav účtu môžeme odkiaľkoľvek a kedykoľvek cez internetovú aplikáciu banky. Môžeme zadávať platby, rôzne príkazy na úhradu a podobne.
Pri online bankovníctve je veľmi dôležitá bezpečnosť, preto so serverom komunikujeme šifrované cez https protokol. K účtu mame špecifické heslá a PIN kódy a na zrealizovanie platby existuje niekoľko bezpečnostných prvkov.

- **Grid karta** (je to karta, ktorá obsahuje množstvo kódov, pri každej platbe je potrebné zadať kód ktorý je na pozícii určenej systémom)
- **SMS potvrdenie** (pred zrealizovaním platby nám príde cez SMS kód, ktorý zadáme do systému na potvrdenie)
- elektronické **čítačky kariet** (čítačku kariet si prípoji užívateľ ku počítaču napríklad cez USB port a do nej vloží svoju platobnú kartu)
- pri veľkých sumách sa prevody potvrdzujú **telefonicky**

Pri používaní internet bankingu by sme mali používať bezpečný prehliadač stránok a nepoužívať verejne počítače (napríklad v internetovej kaviarni). Treba dávať pozor na **phishing** (podvod, kde páchateľ podvrhne falošnú stránku na vylákanie hesla) (*ib.vub.sk*)

Netiketa

Etiketa je akási pomyselná zbierka pravidiel a zásad, ktorá by sa mala v internetovom svete dodržiavať.

Pravidlá

Nikdy nezabúdajte, že na druhom konci sú ľudia a nie počítač. To čo anonymne napíšeme stroju by sme možno nikdy nepovedali dotyčnému do očí.
Dodržiavajte všetky pravidlá slušnosti z normálneho života. Čo je zlé v bežnom živote, bude určite nevhodné aj na internete.
Zistite si, kde sa nachádzate. Cez internet totiž komunikujete s ľuďmi z celého sveta. A čo je v jednej skupine na internete dovolené, iná to môže považovať za neprípustné. Politika, náboženstvo a iné rozporuplné témy by mali byť diskutované s maximálnou ohľaduplnosťou a taktom.
Majte ohľad k druhým. Nie každý má super rýchle pripojenie cez pevnú linku. Mnohý sa pripájajú z domu cez pomalý modem, za ktorý platia! Neposielajme teda zbytočné a zbytočne veľké e-mailové správy.
Nebuďte grobianom! Aj keď píšeme bez diakritiky (bez dĺžňov a mäkčeňov) snažme sa o správny pravopis. Publikovať nepravdivé informácie, alebo niekoho ohovárať tiež nie je vhodné. Nevydávajme za svoje prácu niekoho iného. Obrázky, texty a rôzne iné súbory sa z internetu dajú ľahko stiahnuť. Akoby sa vám páčilo, keby niekto iný vydával vaše dielo za svoje? Ak využijeme prácu iných, mali by sme spomenúť ich autorstvo.
Pomôžte, ak viete. Zaujíma vás nejaká téma a sledujete nejakú diskusiu k nej? Niekto z diskusnej skupiny má taký alebo onaký problém. Ak viete odpoveď, pomôžte. Nabudúce pomôže niekto vám. V diskusnej skupine platí zásada „Najprv počúvaj, až potom píš.“
Rešpektujme súkromie iných. Omylom vám prišla správa, ktorá vám nepatrí? Správajme sa tak, ako by sme chceli, keby niekto iný našiel našu poštu…
Nezneužívajme svoju moc a vedomosti. Užívatelia so špeciálnymi privilégiami, napr. správcovia serverov, ktorí majú prístup k pošte ostatných musia mať dôveru bežných užívateľov.
Odpúšťajme druhým chyby. Aj vy ste niekedy začínali. Nemusíme hneď reagovať výsmešne alebo so zlosťou.
Nešírte reťazové listy a poplašné správy HOAX typu pošli túto správu x ľuďom, pretože takýmto správaním spomaľujete internet. Upozornite i ostatných, že takéto správanie je nevhodné.
Nerozosielajte spam - správy s reklamným textom. Upozornite i ostatných, že takéto správanie je nevhodné.
Rešpektujte autorské práva iných. Nepublikujte cudzí text pod svojim menom, vždy uvádzajte meno pravého autora a zdroj odkiaľ je text prevzatý.
OK, 5 kreditov

12. Hypertextový dokument, hypermediálny dokument, linkovanie. Charakteristika jazyka HTML. Prehľad štandardných príkazov. Iné prostriedky pre tvorbu HTML dokumentu. Dynamické HTML. Vlastný projekt.

Skratka HTML je odvodená od *HyperText Markup Language* a slúži na označenie **značkovacieho** (~~nie programovacieho~~) jazyka používaného k tvorbe webovských dokumentov.

Tak ako väčšina jazykov aj HTML spadá pod určité štandardy, ktoré zostavuje a zastrešuje konzorcium **W3** (www.w3.org).

Počas vývoja webu, prešlo vývojom aj HTML a v súčasnosti sa na webe reálne môžeme stretnúť s **HTML 4.0** alebo **XHTML 1.0** príp. **1.1** (*eXtensible Hypertext Markup Language*) čo je vlastne jazyk vytvorený z *XML* a *HTML* a považuje sa za vyšší a lepší, pretože má prísnejšie štandardy ako bežné HTML.

Konzorcium W3 momentálne pripravuje a zostavuje nové verzie oboch jazykov, ktoré by mali byť do sveta uvedené ako *HTML5* a *XHTML2*.

Pôvodný význam HTML je popísať obsah dokumentu tak, aby prehliadač vedel rozlíšiť sémanticky rozdielne celky, teda napríklad aby vedel, že toto je obrázok, toto je text, toto je zoznam atď.

Jazyk HTML pozostáva (ako nám už názov napovedá) zo značiek, ktoré sa nazývajú **tagy**. Tagy sa zapisujú do ostrých zátvoriek „<“ a „>“ a môžeme ich rozdeliť na párové a nepárové.

- párové tagy

- V HTML je takmer každý tag párový a skladá sa zo začiatočného a ukončovacieho tagu. Ukončovací tag je taký istý ako začiatočný, líši sa len znakom „/“ po prvej ostrej zátvorke. Príklad: `<body>` a `</body>`

- nepárové tagy
  - Je ich iba malý počet a líšia sa tým, že nemajú ukončovací tag. Na ich ukončenie sa tiež používa znak „/“ ale tento krát je umiestnený pred poslednou ostrou zátvorkou. Príklad: `<hr />` (*pred „/“ musí byť medzera!*)

Väčšina tagov môže mať ešte aj určité atribúty. Atribúty sa zapisujú medzi „<“ a „>“ za názvom tagu. Každý atribút je vlastne dvojica pozostávajúca z názvu atribútu a jeho hodnoty. Zápis je nasledovný: `názov_atribútu="hodnota_atribútu"` (*hodnota musí byť v dvojitých úvodzovkách!*)

Posledná časť, z ktorej sa tag skladá je jeho obsah. To je vlastne všetko to, čo je umiestnené medzi začiatočný a koncový tag (obsah teda majú len párové tagy).

Kompletná syntax tagu je teda nasledovná:

```
<tag atribut1="hodnota1" atribut2="hodnota2">obsah tagu</tag>
<tag atribut1="hodnota1" atribut2="hodnota2" />
```

Hypertextový dokument je teda dokument (súbor) písaný v jednom z jazykov (HTML, XHTML) a má príponu *.htm* alebo *.html*.

Štandardná štruktúra HTML dokumentu vypadá nasledovne:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 ↵ Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; ↵
charset=windows-1250">
  <title>New page</title>
  </head>
  <body>

  </body>
</html>
```

Tag `<html>` označuje začiatok a koniec celého dokumentu. Pred ním sa uvádza ešte DOCTYPE definícia, ktorá určuje typ dokumentu. To znamená: v akom jazyku je písaný a na akej úrovni štandardov. Tu poznáme tri úrovne:

- **transitional** – menej prísna forma štandardu, akceptuje aj určité malé nezrovnalosti v kóde a zastaralé tagy
- **strict** – štandard v jeho čistej forme, je to teda najprísnejšia forma, ktorá neakceptuje žiadne chyby v kóde
- **frameset** – štandard používaný pre Frameset dokumenty (jednoducho povedané dokument v dokumente)

Tag <html> obsahuje ďalšie dva dôležité tagy. Tými sú `<head>` a `<body>`. Tag `<head>` (hlavička) obsahuje dôležité informácie o dokumente ako sú napríklad: autor, kľúčové slová, popis, linky na externé súbory, typ kódovania dokumentu a iné. Obsah tohto tagu nie je užívateľovi nijak zobrazený! Tag `<body>` následne obsahuje celé telo dokumentu – jeho obsah je to čo vidíme v prehliadačoch pri prezeraní webu.

Hypertextový dokument sa volá hypertextovým preto, lebo jednou z jeho najzákladnejších vlastností je, že sa môže odkazovať (linkovať) na iný dokument. Takýto odkaz, ktorý spája dva dokumenty sa nazýva link a tvorí sa tagom `<a>` (od angl. Anchor). Tradične bývajú odkazy na stránke nejakým

spôsobom odlíšené (defaultne je to modrá farba textu, podčiarknutie a pri nadídení myšou sa kurzor zmení na ukazujúcu ruku). Týmto sa dostávame k tradičným a najčastejšie používaným tagom v HTML:

- `<hx>`
  - kde x je z intervalu <1,6> a 1 znamená najvyššia úroveň. Tento tag označuje nadpis (angl. header) a používa sa na lepšie členenie textu v dokumente. V súčasnosti mu pripadá aj dôležitý sémantický význam, keďže jeho obsah je braný do úvahy pri vyhľadávaní na webe (platí približne len po 2. úroveň)
- `<p>`
  - označuje odstavec (angl. paragraph). Slúži na rozčlenenie textu do odstavcov.
- `<br />`
  - tag na ukončenie riadku (angl. break). Text za týmto tagom bude na novom riadku.
- `<hr>`
  - určuje horizontálnu čiaru (angl. horizontal rule) vhodnú na optické rozdelenie dokumentu
- `<ul>, <ol>, <li>`
  - Tieto tagy vytvárajú zoznam, pričom `<ul>` určuje nečíslovaný zoznam (zoznam tvorený zarážkami), `<ol>` určuje číslovaný zoznam (zoznam môže byť číslovaný arabskými alebo rímskymi číslicami alebo písmenami abecedy) a napokon `<li>` určuje položku zoznamu v oboch typoch zoznamov
- `<img />`
  - Tento tag slúži na vloženie obrázka do dokumentu. Má jeden povinný argument `src`, ktorý udáva zdroj obrázku. Ďalej má niekoľko nepovinných atribútov, ktoré by sa však vždy mali uvádzať. Sú to `width, height, alt` a `title`. Tie určujú šírku, výšku, alternatívny text (v prípade, že má užívateľ vypnuté zobrazovanie obrázkov) a titulok (krátky popis)
- `<table>, <tr>, <th>, <td>`
  - Sú 4 najzákladnejšie tagy na vkladanie tabuliek do dokumentu. Tag `<table>` určuje samotnú tabuľku, `<tr>` riadok tabuľky, `<th>` záhlavie tabuľky a `<td>` bunku tabuľky (angl. table row, table header a table data)

To by bol stručný zoznam najpoužívanejších tagov v HTML (pre ďalšie informácie viď odkazy na konci tejto otázky).

Na samotnú tvorbu HTML nám postačí textový editor (kľudne stačí notepad). Existujú však aj editory typu WYSIWYG (*What You See Is What You Get*). Sú to zjednodušené editori, pri ktorých užívateľ nemusí úplne poznať HTML. Stránku si proste akoby nakreslí a program mu vygeneruje kód (často nevalidný a nahustený – vrelo odporúčam **NEpoužívať**). Mimo týchto nástrojov máme samozrejme aj iné či už voľne dostupné (PSPad, 1st page) alebo platené, drahé a vyspelé programy (Adobe GoLive!).

HTML nám teda slúži len na značkovanie obsahu. Na úpravu jeho vzhľadu slúžia kaskádové štýly (*CSS – Cascading Style Sheets*).

Príklad CSS:

```
* {
   margin: 0px;
   padding: 0px;
}

body {
   font-size: 62.5%;
   font-family: Tahoma, "Lucida Grande CE", lucida, sans-serif;
}
```

```
h3.designed {
   color: #0C6E9C;
   font-family: Verdana, "Geneva CE", lucida, sans-serif;
   padding: 1.2em 0em;
}

.clear {
clear: both;
}

#container {
position: absolute;
bottom: 0px;
right: 0px;
width: 60px;
}
```

Samotné HTML je statický jazyk, ktorý vo svojej podstate neumožňuje interakciu s užívateľom. Na vytváranie dynamických webovských dokumentov v dnešnej dobe slúžia rozličné programovacie jazyky alebo doplnky. Medzi tieto jazyky môžeme zaradiť napríklad *JavaScript* a *Ajax* a medzi doplnky napríklad *Flash*. Na webe sa môžeme taktiež stretnúť s pojmom DHTML (*Dynamic HTML*), ktorý sa často používa na označenie webov, ktoré kombinujú HTML, CSS a JavaScript. Pri DHTML sa často využíva aj DOM (*Document Object Model*).

Príklad JavaScriptu:

```
function blankLinks() {
 if (!document.getElementsByTagName) return;
 var anchors = document.getElementsByTagName("a");
 for (var i=0; i<anchors.length; i++) {
   var anchor = anchors[i];
   if (anchor.getAttribute("href") &&
       anchor.getAttribute("rel") == "e")
   anchor.target = "_blank";
   }
  }
```

Na záver pár užitočných linkov:

- www.w3.org – kompletne spracované štandardy pre web
- www.jakpsatweb.cz – veľmi dobre a prehľadne rozpracované vytváranie html dokumentov (obsahuje HTML, CSS, JavaScript spolu s referenciami)
- www.w3schools.com – škola webu, obsahuje základné informáciách o takmer všetkých technológiách využívaných na webe, ich referencie a početné príklady
- www.javascriptkit.com – užitočný web s príkladmi a referenciami ohľadom JavaScriptu

OK, 5 kreditov

13. Digitalizácia obrazu. Snímanie obrázkov. Parametre snímania, parametre skenerov, digitálnych fotoaparátov. Formáty grafických súborov. Popis rastrového editora. Kreslenie, čiara, úsečka, obdĺžnik, elipsa, krivka, text, obrys a výplň, farba, farebný prechod. Úprava fotiek, zväčšovanie zmenšovanie, zaostrenie, vyhladenie, úprava horizontu, otočenie, zmena počtu farieb.

Existujú dva základné spôsoby zápisu grafickej informácie: Prvým z nich je pozeranie sa na obrázok ako na sieť (raster) veľmi malých štvorcov - pixlov. Uloženie grafickej informácie pomocou takéhoto prístupu voláme **rastrová grafika**. Druhý spôsobom je pozeranie sa na obrázok ako na zoskupenie objektov (alebo ich častí), ktoré sa dajú nakresliť pomocou matematických vzorcov a funkcií. Tieto objekty majú svoje vlastnosti ako polohu na obrázku, veľkosť, farbu, priehľadnosť povrchu, lesklosť povrchu... Tieto vlastnosti sú vstupnými parametrami (vektormi) matematických vzorcov a funkcií, pomocou ktorých sa objekty nakreslia. Uloženie grafickej informácie pomocou takéhoto prístupu sa nazýva **vektorová grafika**.

Rastrová grafika

Rastrové obrázky sú rozdelené na sieť myslených štvorčekov - pixlov. Rozmer každého obrázka je pre počítač počet pixlov na šírku x počet pixlov na výšku. Pre každý pixel (štvorček) je nutné okrem polohy (riadok a stĺpec) zakódovať aj farbu, resp. ďalšie parametre.
Ak je obrázok monochromatický (čierna a biela farba), kódovanie je jednoduché (1 - rozsvietený (biely) bod, 0 - nerozsvietený (čierny) bod).
Pri niektorých obrázkoch je však potrebné zaznamenať i odtiene, preto dve farby - čierna a biela - nestačia. Výhodné je zakódovať odtiene sivej farby pomocou ôsmich bitov tak, aby informácia o každom bode zaberala 1 B, t.j. jedno pamäťové miesto počítača. Čím bude bod svetlejší, tým väčšia hodnota sa do pamäte uloží. Preto bod čiernej farby bude uložený ako 0 a bod bielej farby ako maximálna možná hodnota – 255.

Kódovanie farieb

Prvé riešenie problému kódovania farieb, ktoré sa ponúka, je zobrať celý rozsah vlnových dĺžok a rozdeliť ho na niekoľko častí a do pamäte počítača uložiť poradové číslo farby, ktorej prislúcha určitá vlnová dĺžka, ďalej jej sýtosť a jas. Kódovanie farieb takýmto spôsobom využívajú farebné modely **HSB, HLS**. Oba modely uchovávajú informáciu o odtieni (Hue) a sýtosti (Saturation). Tieto modely sa odlišujú iba v tom, že prvý model používa jas (Brihtnes) a druhý používa svetlosť (Lightnes). V prvom modeli dostaneme čiernu farbu tým, že nastavíme jas na nulu a bielu tak, že jas nastavíme na maximálnu hodnotu (nezávisle od sýtosti). V druhom modeli dosiahneme čiernu, ak je svetlosť aj sýtosť 0 a bielu, ak je svetlosť aj sýtosť maximálna. Hlavným nedostatkom tohto problému je náročnosť vyrobenia svetla so zadanou vlnovou dĺžkou. Naštastie existujú aj iné riešenia problému kódovania farieb.

Farebný model **RGB** je najčastejšie využívané kódovanie farby bodu obrázka. Empiricky sa zistilo, že takmer všetky farby sa dajú vytvoriť zmiešaním ľubovoľných troch nezávislých farieb (t.j. že pomocou zmiešania dvoch nedostaneme tretiu). Najvýhodnejšie pre výrobu svetelných lúčov bolo použitie farieb červená (Red), zelená (Green), modrá (Blue). Aby sme vedeli vytvoriť 1,5 milióna farieb stačí, ak každú z týchto farieb rozdelíme na 115 oddtieňov, ktorých zmiešaním v rôznych pomeroch vzniknú všetky farby. Kvôli uchovaniu v pamäti počítača, je však výhodnejšie použiť až 256 odtieňov každej farby (8 bitov), čo nám umožní vytvoriť až 16 777 216 rôznych farieb. Pri takomto kódovaní je každá farba zakódovaná 24 bitmi, čo sú tri pamäťové miesta počítača. Pričom 255 0 0 je sýta červená farba, 0 255 0 je sýta zelená farba, 0 0 255 je sýta modrá farba, 0 0 0 je čierna farba a 255 255 255 je biela farba.

Farebný model **CMY** je model, ktorý používa doplnkové farby. V modeli RGB platilo, že ak zmiešame všetky tri základné farby s maximálnou sýtosťou, dostaneme bielu farbu. Takýto spôsob je výhodný pri obrazovkách monitorov, pretože tienidlo je čierne. Pri tlačiarňach sa však tlačí na biely papier, preto potrebujeme vziať také farby, pri ktorých, ak zmiešame ich najsýtejšie odtiene, dostaneme čiernu farbu. Takéto farby dostaneme, keď zoberieme doplnkové farby k farbám červená, zelená a modrá. Týmito farbami sú azúrová (Cyan), purpurová (Magenta) a žltá (Yellow). Kvôli tomu, že je lacnejšie vyrobiť čierny atrament ako ho miešať pomocou týchto troch farieb, sa k týmto farbám pridáva i samostatná čierna farba a tento model sa označuje tiež CMYK, kde posledné písmeno je odvodené od blacK - čierna .
Výhodou tohto formátu je tá, že výsledná farba CMY sa dá veľmi jednoducho získať z modelu RGB pomocou vzorcov:

$$C = (255 - R); M = (255 - G); Y = (255 – B)$$

Farebný model **YUV** slúži na zachovanie čiernobielej informácie pri televíznom vysielaní. Po vzniku farebného filmu nastal problém, ako zakódovať farbu tak, aby farebné filmy mohli pozerať i ľudia s čiernobielymi prijímačmi. Bolo potrebné zachovať pôvodnú čiernobielu informáciu a doplniť ju tak, aby vznikol farebný obraz. Farba je kódovaná tak, že k čiernobielej zložke Y, ktorú tiež nazývame svietivosť (luminance), pridáme dve farebné zložky UV farebnosti (chrominance), pričom zložka U udáva odtieň medzi modrou a žltou a zložka V udáva odtieň medzi červenou a žltou farbou. Takže čiernobiely prijímač berie do úvahy iba zložku Y, farebné prijímače za pomoci zložiek YUV získajú RGB kód farby pomocou jednoduchej transformácie:

$$R = Y + 1{,}403V;\ G = Y - 0{,}344U - 0{,}714V;\ B = Y + 1{,}770U$$

Grafické formáty

Všetky spomenuté farebné modely kódujú farbu troma nezávislými hodnotami. Najvýhodnejšie je teda každý pixel obrazu zakódovať pomocou troch pamäťových miest počítača - 3 Bajtov. Obrázok s rozmermi 1024x768 pixlov tak v pamäti grafickej karty zaberie 2 359 296 Bajtov. V minulosti kvôli cene pamätí sa na grafické karty montovali pamäte menších rozmerov, teda na kódovanie farieb sa použil menší počet Bajtov. Najstaršie počítače používali iba 16 farieb, to znamená, že každý bod bol zakódovaný 4 bitmi, neskôr sa začali vyrábať 256-farebné grafické karty (**VGA**), ktoré mali každý bod kódované 8 bitmi. Po zlacnení počítačových pamätí už bolo možné vyrábať karty **SVGA**, ktoré kódovali farby pomocou 16 bitov (dve pamäťové miesta) v režime **High Color** (vysoká farebnosť). V súčasnosti už grafické karty majú toľko pamäte, že bez problémov môžu kódovať každý bod 24 bitmi (tri pamäťové miesta) v režime **True Color** (pravá farebnosť). Popísaný spôsob uloženia obrázka (keď je každý bod kódovaný pomocou niekoľkých bitov - 4, 8, 16 alebo 24) sa používa formát, ktorý sa volá bitová mapa (**BitMaP**). Obrázky v takomto formáte sú v počítači uložené v súboroch s príponou **BMP**.

Veľká pamäťová náročnosť už teda nie je problém grafickej karty počítača, stále je však problém pri posielaní takýchto obrázkov prostredníctvom internetu, pretože obrázok s rozmermi 1024x768 pixlov v režime true color sa nezmestí ani na disketu. Preto sa na zníženie pamäťových nárokov používa paleta farieb a kompresia (stlačenie).
Paleta využíva skutočnosť, že na kreslených obrázkoch väčšinou nie je použitých viac ako 256 farieb. Zníženie pamäťových nárokov spočíva v tom, že očíslujeme všetky použité farby v obrázku číslami od 0 do 255, a potom kódujeme každý bod tak, že uvedieme poradové číslo farby v palete. Tým miesto troch pamäťových miest, každý bod zakódujeme len pomocou jedného pamäťového miesta.
Ďalší spôsob ako znížiť pamäťovú náročnosť obrázka, je použitie **kompresie**. Princíp kompresie spočíva v tom, že ak sa pixel s rovnakou farbou vyskytuje viac krát za sebou, do pamäte neukladáme jednotlivé pixely, ale uložíme koľko krát sa pixel danej farby vyskytol.
Niekedy však by sme v konečnom dôsledku zabrali ešte viac pamäte ako pôvodný obrázok. To, či sa má obrázok do pamäte uložiť skomprimovane alebo nie, sa rozhodneme pomocou tzv. Kompresného pomeru (veľkosť po komprimovaní / pôvodná veľkosť). Ak je kompresný pomer < 1, obrázok uložíme komprimovaný, v opačnom prípade ho uložíme v normálnej podobe.

Oba popísané spôsoby zníženia pamäťovej náročnosti používa formát **GIF**. Okrem týchto výhod, formát gif umožňuje označiť jednu z farieb na obrázku za priehľadnú. Ďalšou obrovskou výhodou tohto formátu je to, že dokáže uložiť animácie obrázkov. Naopak nevýhodou tohto formátu je obmedzenie 256 farbami.
Kompresiu využíva i formát **JPEG** (súbory s príponou **JPG**), tentokrát sa však jedná o tzv. stratovú kompresiu. Táto kompresia je založená na vynechávaní, niektorých málo viditeľných detailov. V praxi to znamená, že ak je niekde napríklad jedna svetložltá bodka uprostred veľkého bieleho poľa, jednoducho sa vymaže. Ďalej ak je niekde tenká čiara medzi dvoma plochami, tak sa farba tejto čiary upraví tak, aby sa jej farba dala vypočítať zložením farieb plôch, ktoré obklopuje. Po aplikovaní takýchto úprav sa aplikuje klasická kompresia, ktorá je tým pádom oveľa efektívnejšia. Výhodou tohto formátu je to, že používa všetky farby (true color) a dosahuje výborný kompresný pomer (zmenšenie). Je vhodný najmä na uloženie digitálnych fotografií. Nedá sa tu však nastaviť priehľadná farba.
Problém s priehľadnou farbou rieši formát **PNG**, ktorý spája výhody formátov GIF - priehľadná farba, bezstratová kompresia a JPEG - True Color farby, dobrý kompresný pomer, tento formát však nieje tak rozšírený ako JPEG.
Obrázok vo formáte **TIFF** je najvhodnejšou voľbou pre tlač, pretože nestráca na farebnosti.

Snímanie obrázkov

Snímanie obrázkov uskutočňujeme pomocou skenerov. Skenerov. Skenovanie je proces, pri ktorom sa digitalizuje daná predloha (obrázok, text a iné). Je to prevod z grafickej predlohy do digitálnej, ktorá je vhodná na ďalšie spracovanie na počítači. Poznáme viacero typov scanerov, od tužkových na snímanie textu, cez prieťahové scanery používané vo väčšine faxových zariadení až po špecifické typy, napríklad na snímanie diapozitívov a negatívov, knižné scanery či multifunkčné zariadenia.
Iným spôsobom zachytávania grafickej informácie je pomocou digitálnych fotoaparátov a kamier, kde obraz zachytený objektívom dopadá na elektronický snímač, ktorý celý obraz rozloží na milióny pixlov a informácie o jase, farbe a ostatných parametroch týchto bodov uloží do pamäte.

Programy na spracovanie grafickej informácie

Programy na spracovanie grafiky delíme do troch skupín na rastrové (niekedy tiež bitmapové) editory, vektorové editory a univerzálne prehliadače.

Medzi **rastrové editory** patria programy na spracovanie fotografií alebo na vytváranie kreslených obrázkov či animácií. Medzi najjednoduchšie rastrové editory patrí napríklad i program Maľovanie (Paint), ktorý je súčasťou operačného systému Windows. Vyspelejším editorom je grafický editor GIMP, ktorý je zadarmo. Z českých grafických editorov je známy editor Photo Studio od firmy Zoner. Najznámejšie komerčné editory sú PhotoShop od firmy Adobe, Paint Shop Pro od firmy Corel a Fireworks od firmy Macromedia.
Najznámejšie **vektorové editory** sú Corel Draw od firmy Corel, Illustrator od firmy Adobe a z českých je známy Callisto od firmy Zoner, ktorým tiež hovoríme DTP (Desktop Publishing) - programy na tvorbu plagátov, novín a pod. Medzi vektorové editory, ktoré umožňujú vytvárať animácie patria tiež 3D modelovacie nástroje, z ktorých je najznámejšie 3D studio MAX.
Univerzálne prehliadače pôvodne slúžili iba na prehliadanie čo najväčšieho počtu grafických formátov, dnes však obsahujú i niektoré základné funkcie grafických editorov ako orezávanie, zmena velkosti, odstránenie efektu červených očí, zmena farebnosti, jasu a kontrastu atď. Možno ich tiež využiť na prevod grafických informácii z jedného formátu do druhého. Najznámejšími prehliadačmi sú IrfanView, ACDsee a XnView.

Úloha 1

Na digitálnom fotoaparáte máme nastavené rozlíšenie, pri ktorom sa fotografie ukladajú ako bitmapové súbory s veľkosťou 600 KB s 24 bitovým farebným kódovaním. Ak prekonvertujeme takúto fotografiu do 256 farieb, akú bude mať približne veľkosť?

Riešenie:

Najprv musíme zistiť koľko bodov sa nachádza v obrázku. Vieme, že jeden bod obrázka je kódovaný 24 b, zaberá teda tri pamäťové miesta teda 3 Bajty (24:8=3). Ďalej vieme, že 1 kB = 1024 B, takže obrázok bude zaberať: 600 kB × 1024 = 614 400 B
Teraz zistíme, koľko bodov sa nachádza v obrázku:

614 400 B : 3 B = 204 800
veľkosť obrázka veľkosť bodu počet bodov.

Jeden bod 256 farebného obrázka sa dá zakódovať 2^n bitmi- použité je teda 8bitové kódovanie.
Veľkosť 256 farebného obrázka teda bude:

204 800 × 1 B = 204 800 B
počet bodov veľkosť bodu veľkosť obrázka

Teraz už len prevedieme výsledok na kB:

204 800 B : 1024 = 200 kB

Úloha 2

Katka má fotografiu triedy uloženú v bitmapovom súbore a chce ju poslať MMS-kou na Petrov mobilný telefón. Fotka má rozmery 256 x 180 pixelov (obrazových bodov) a je v nej použitých 256 farieb. MMS-ka môže mať maximálne 5 KB. Najmenej koľko MMS musí Katka poslať, ak chce poslať celú fotografiu rozloženú do viacerých MMS?

Riešenie:
Rozmery fotografie sú 256 x 180 pixelov, teda počet bodov v obrázku bude:

$$256 \times 180 = 46\,080$$

Každý bod 256 farebného obrázka zaberie 1 B pamäte (pozri úlohu 1), takže obrázok bude zaberať:

$$46\,080 \times 1\text{ B} = 46\,080\text{ B}$$

počet bodov veľkosť bodu veľkosť obrázka

Jedna MMS môže zaberať 5 kB čo je v B:

$$5\text{ kB} \times 1024 = 5\,120\text{ B}$$

Počet MMS vypočítame takto:

$$46\,080\text{ B} : 5\,120\text{ B} = 9$$

veľkosť obrázka veľkosť MMS počet MMS

Úloha 3

Na webovej stránke chceme publikovať animovanú fotografiu. Dve skoro rovnaké fotografie sa líšia iba mrknutím oka na jednej z fotografií. Vytvoríme z nich jediný súbor - animovaný obrázok. Ktorý z uvedených formátov použijeme?
(A) gif **(B)** bmp **(C)** jpeg **(D)** žiadny z nich
Riešenie
Správna odpoveď je **(A)** (pozri formát **GIF**).

Úloha 4

Obrázok v pamäti je zložený zo 600 x 800 farebných bodov. Najmenej koľko KB pamäte zaberie tento 6-farebný neskomprimovaný obrázok?

Riešenie
Obrázok pozostáva z 600 x 800 bodov. V obrázku je použitých 6 farieb, teda na zakódovanie jedného bodu budeme potrebovať nasledovný počet bitov:

2^n 6, n=2,58. Musíme teda použiť 3 bity

Ak chceme zistiť, koľko takýto obrázok zaberie pamäte, musíme deliť 8, pretože jedno pamäťové miesto (1 B) pozostáva s 8 bitov:

$$800 \times 600 \times 3 : 8\text{ B}$$

Výsledok je potrebné ešte previesť na kB, teda vydeliť 1024:

$$800 \times 600 \times 3 : 8 : 1024\text{ kB} = 800 \times 600 \times 3 : (8 \times 1024)\text{ kB}$$

OK, dobre, 5 kreditov

14. Kódovanie vektorovej grafickej informácie, výhody vektorových obrázkov. Použitie základných nástrojov vektorového grafického editora. Objekty a manipulácia s nimi, ich úprava. Práca s textom, kreslenie objektov - úsečky, polygóny, krivky, obdĺžnik, štvorec, elipsa a kružnica. Transformácia objektov, vlastnosti, obrys, jednofarebná výplň, farebný prechod, textúra, priesvitnosť a tieň. Práca s viacerými objektmi, zoskupovanie.

Vektorová grafika na rozdiel od rastrovej grafiky nekóduje jednotlivé body obrazca, ale určuje a opisuje geometrické útvary ako body, priamky, krivky, kružnice, mnohouholníky, ich farbu, hrúbku a iné parametre. Taktiež sa používa text. Používa pre znázornenie rôznych geometrických konštrukcií, ale aj pri vytváraní kresleného designu. Výhodou je, že všetky tvary sú zapísané matematicky a tak možno kresby ľubovoľne zväčšovať aj donekonečna bez straty kvality.

Ohraničené prvky môžu byť zobrazujúcou aplikáciou (programom) vyplnené farbou. Vyplňovanie sa obvykle deje nezávisle na obrysoch prvkov. Každému prvku môže prislúchať dve a viac farieb – jedna prislúcha obrysom, druhá vyplňovanej ploche. Farba pre vyplňovanie môže byť aj priesvitná. Niektoré formáty definujú takzvané farebné atribúty. Pokiaľ ide o farby nepriesvitné môžu tieto prvky obsahovať šrafovanie alebo tieňovanie, ktoré nazývame vyplňovanie atribúty. Formáty, ktoré nedovoľujú akékoľvek vyplňovanie, ho musia simulovať vzorovým vykresľovaním. Táto činnosť nielen že neprináša žiadanú kvalitu, ale zároveň podstatne zväčšuje veľkosť súboru.
Najpoužívanejšie programy na tvorbu a úpravu vektorovej grafiky sú Adobe Illustrator, Corel Draw a Inkscape. Adobe a Corel používajú vlastné formáty, Inkscape používa SVG, čo je otvorený štandard definovaný W3 konzorciom. SVG je štandard určený najmä pre web. Je kódovaný v XML zápise. Môže obsahovať tiež rôzne skripty.

Medzi ich nástroje patrí tvorenie úsečiek, kriviek, textu a rôznych mnohouholníkov. Ďalej môžeme upravovať už vytvorené objekty: meniť veľkosť, zahnutie, farbu, ohraničenie.
Taktiež môžeme použiť výber objektov.

Objektom môžeme priradiť rôznu výplň: jednofarebnú alebo gradientnú, rôznej úrovni priehľadnosti.
Zadanie je pomaly obšírnejšie ako vypracovanie, 2 kredity

15.Zošity, dokumenty, listy. Bunka, obsah bunky, formát bunky, blok buniek, štýl bunky. Formát tabuľky, kopírovanie tabuľky, údajov, formátu.

Zošit je súbor obsahujúci jeden alebo viacero pracovných hárkov, ktoré možno použiť na usporiadanie rôznych druhov súvisiacich informácií.
Pracovný **hárok**, resp. **list** je základný dokument, ktorý sa v programe Excel používa na uloženie údajov a na prácu s údajmi. Označuje sa aj ako tabuľka. Pracovný hárok pozostáva z buniek usporiadaných do stĺpcov a riadkov a je vždy súčasťou zošita.

Bunka je najmenšou štruktúrnou jednotkou programu Excel. Je to okienko, v ktorom sa udržujú dáta. Každá bunka má svoju adresu – reprezentuje ju písmeno stĺpca a číslo riadku, napríklad: A1, D27. Každá bunka môže byť naformátovaná podľa jedinečných pravidiel. Hocijaká bunka, alebo rozsah, čiže blok buniek (napríklad A1:A3; B2:D4) môže byť zvýraznený niekoľkými odlišnými spôsobmi napríklad použitím **bold** textu**,** farbou, fontom, veľkosťou textu. zarovnaním, orámovaním atď.

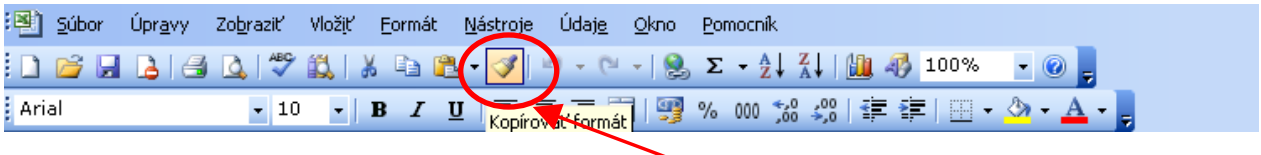
Štýl bunky – môžeme si nastaviť, aký typ dát daná bunka reprezentuje, napríklad: dátum, mena, percentá, zlomky, text, vlastné, samozrejme predvolený je všeobecný formát bunky. V jednotlivých štýloch bunky je možné nastavovať ich formát, teda aký majú mať bunky aj vzhľad.

Formát tabuľky – je formát bloku buniek, napríklad nižšie na obrázkoch je jedna tabuľka formátovaná 2 odlišnými spôsobmi.

Vypracovanie MO	ot. 1	ot. 2	ot. 3	ot. 4
Dzurenko Martin	34	40	53	56
Kredity				
Chrenko Matej	11	27	38	48
Kredity				

Vypracovanie MO	ot. 1	ot. 2	ot. 3	ot. 4
Dzurenko Martin	34	40	53	56
Kredity				
Chrenko Matej	11	27	38	48
Kredity				

Kopírovanie tabuľky – označíme si blok buniek ťahaním myšou, alebo pomocou klávesnice držaním klávesy „Shift" a pohybom smerových šípok. Skopírujeme vložením do schránky, nastavíme kurzor do miesta, kam chceme tabuľku kopírovať a prilepíme.



Kopírovanie formátu umožňuje skopírovať už použité formátovanie znova aplikovať pomocou pár kliknutí: označíme zdrojový text, klikneme na *ikonku štetca*, a ťahaním označíme bunky, na ktoré chceme formát aplikovať.

Hm, čo je tu, je OK, len sa bude ťažko rozprávať 10 minút ☹

4 kredity

16. Vzorec, funkcia. Tvar vkladaného vzorca a funkcie. Relatívny a absolútny odkaz. Kopírovanie vzorcov a funkcií.

Vzorec- zápis matematickej operácie, na jeho základe môžeme vykonávať výpočty v tabuľke.

Vzorec sa môže skladať z konštánt, operátorov, odkazov, funkcií a názvov, a všetko sa to môže kombinovať.

- vzorce tvorené iba **konštantami a operátormi** nepatria medzi dynamické vzorce, tzn. nemenia svoj výsledok v závislosti na obsahu inej bunky alebo buniek. Sú to napríklad vzorce :=24+26-18
- vzorce pozostávajúce z **odkazov na bunky a operátorov**. Ich výsledok závisí na iných bunkách a teda akonáhle zmeníte obsah buniek, zmení sa výsledok. Príklad vzorca =A1+A2+A3+C8
- ďalší typ vzorcov je tvorený excelovskými **funkciami**, ako napr. =ABS(E5)

Tvar vkladaného vzorca:

Excel interpretuje ako vzorec každý obsah bunky, ktorý sa začína znamienkom =

Po vložení znamienka *rovná sa*, sa v Poli názvov objaví rozbaľovací zoznam obsahujúci definované excelovské funkcie; tlačidlá všetkých zobrazených panelových nástrojov sa stanú nedostupnými. V prípade kliknutia na tlačidlo *rovná sa* sa okrem toho pod riadkom vzorcov objaví tzv. *okno vzorca*: šedý obdĺžnik s tlačidlami *OK*, *Storno* a *?* v ktorom sa nachádza text *Výsledok* =. Okno vzorca pomáha pri vytváraní vzorcov prostredníctvom funkcií listov a umožní priebežne sledovať výpočet vzorca.

Kopírovanie vzorcov- ak nie sú rovnaké hodnoty v stĺpci nad sebou, alebo sa opakuje časť tabuľky, môžeme údaje kopírovať. Štandardný spôsob kopírovania a presúvania umožňuje schránka. Program Excel ponúka aj špeciálne nástroje na presun a kopírovanie údajov. Využívanie týchto nástrojov je viazané na hranicu alebo vypĺňaciu úchytku aktívnej bunky, resp. označenej oblasti. Ak je v aktívnej bunke uložený text alebo číslo, vypĺňajú sa tie isté hodnoty. Ak je však v bunke uložený dátum, vypĺňajú sa do ďalších buniek dátumy nasledujúcich dní (aj mesiace a dni v týždni).

Relatívny odkaz:

Relatívny odkaz je odkaz bez dolárových znakov ($). Odkaz je interpretovaný ako vzdialenosť buniek vzhľadom k aktuálnej bunke, čo znamená, že je pri kopírovaní alebo presune v rámci listu či zošitu nemenená tak, aby táto vzdialenosť zostávala naďalej rovnaká.

Napr.: V bunke A3 chcete sčítať A1 a A2, tzn. bunky nad bunkou A3. Pokiaľ chcete aj v bunke B3 sčítať B1 a B2, môžete vytvoriť vzorec = B1+B2 alebo do bunky B3 skopírovať obsah bunky A3 (napríklad pomocou CTRL+C a CTRL+V). Po skopírovaní zistíte, že vzorec v bunke B3 znie =B1+B2. Excel totiž automaticky prispôsobil adresy buniek vo vzorci novej pozície v zošite, pretože si zapamätal, že má v bunke vytvárať vždy súčet dvoch buniek nachádzajúcich sa nad „vzorcovou“ bunkou.

Absolútny odkaz:

Absolútny odkaz odkazuje na určitú pevnú pozíciu v zošite a poznať to podľa toho, že sa pred adresami riadkov a stĺpcov nachádzajú znaky *$*. Pri kopírovaní alebo presune vzorca zostanú absolútne odkazy zachované.

Napr.: Pokiaľ by ste v predchádzajúcom príklade zadali vzorec ako =$A$1+$A$2 a potom ho skopírovali do bunky B3, sčítal by aj naďalej hodnoty buniek A1 a A2 (nie B1 a B2).

Relatívny a absolútny odkaz sa vo vzorci dajú samozrejme kombinovať.

Funkcia :

Funkcia je zjednodušená forma zložitého a neprehľadného vzorca. V Exceli existujú funkcie matematické, statické, databázové, logické, vyhľadávacie, informačné, finančné, textové, dátumové a časové. Z ich názvov je dostatočne zrejmé, akými výpočtami sa asi zaoberajú.

Dajme tomu, že sa v bunkách A2:A5 a B2:B5 nachádzajú dáta, z ktorých chceme v bunkách A7 a B7 zistiť priemer hodnôt príslušného stĺpca. Budeme teda musieť vložiť statickú funkciu PRIEMER pre výpočet priemeru.

Funkciu vložíme nasledovne: Označíme bunku do ktorej chceme funkciu vložiť. *Vložiť funkcia.*

Funkcie - sú pomenované, dopredu definované výpočty, ktoré možno používať vo vzorcoch. Všetky funkcie sú rozdelené do kategórií: všetko, finančné, dátum a čas, matematické, štatistické, vyhľadávacia, databáza, text, logické, informačné. Poznáme viac ako 300 funkcií. Ak poznáme presný názov a argumenty funkciu môžeme vložiť priamo do bunky. Nemusíme však poznať všetky mená, ale pri ich zápise môžeme využiť sprievodcu funkciami(príkaz vložiť/funkciu alebo kliknúť na fx na paneli. Objaví sa pravé okno sprievodcu, v ľavej časti ktorého sú uvedené skupiny príbuzných funkcií. Po výbere skupiny sa v prvom okne zobrazia mena funkcií. Ak nevieme, do ktorej skupiny patrí konkrétna funkcia, vyberieme VŠETKO. V druhom okne sprievodcu zadávame argumenty vybranej funkcie. Argumentmi funkcií môžu byť nielen konštanty a adresy buniek alebo oblastí, ale aj vzorce a ďalšie funkcie. Aj keď zadávame funkcie pomocou sprievodcu, môžeme argumenty obsahujúce adresy buniek zadávať vytyčovaním. V prípade, ak je argumentom iná funkcia, aj pre jej zadanie môžeme znova vyvolať sprievodcu funkciami pomocou rozbaľovacej ikony v ľavej časti vzorcového panela.

Medzi základné funkcie patrí: napr.

SUMA(slúži na sčítanie číselných hodnôt v bunkách)

MIN alebo MAX(po označení tabuľky s číslami vyhľadá najväčšie a najmenšie číslo)

CELÁ ČASŤ(po označení bunky s číslom odstráni desatinné čísla)

GENEROVANIE NÁHODNÝCH ČÍSEL (po zadaní určitého intervalu vygeneruje čísla z tohto intervalu do nami stanovených buniek),ak pridáme do riadku s funkciou generovanie náhodného čísla funkciu celá časť vygenerujú sa z daného intervalu len celé čísla.

Ak chceme ako argument použiť viacej úsekov označte v okne sprievodca funkcií prvý úsek, stlačíme klávesy SHIFT +F8 čím sa nastaví add) a označte ďalšie úseky. Vzorce s funkciami môžeme kopírovať a presúvať z tým, že platia všetky pravidlá pre prácu so vzorcami(viď kopírovanie vzorcov).

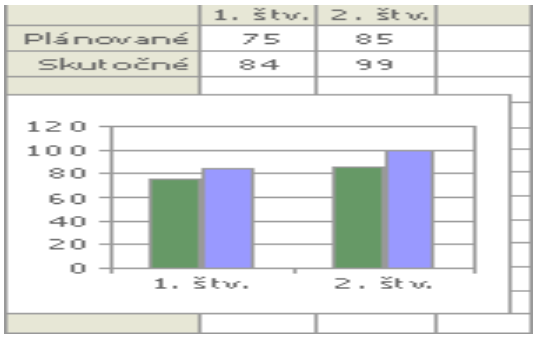
OK, môže byť, 5 kreditov

17. Označenie údajov v tabuľke pre graf, zobrazenie. Vytvorenie štandardného grafu pomocou sprievodcu. Grafický list. Základné operácie s grafom. Prvky grafu. Formátovanie grafu.

Grafy v programe Excel

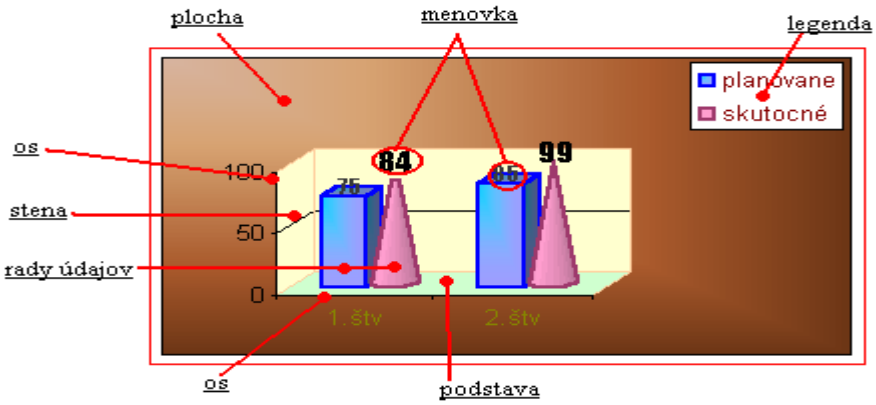
1. **Tvorba grafu**:

Ak chceme vytvoriť graf musíme mať najskôr vytvorenú tabuľku s údajmi. Ak nechceme graf vytvoriť zo všetkých údajov v tabuľke musíme si požadované stĺpce a riadky označiť. Môžeme ich označiť pred tvorbou grafu, alebo ich označíme počas tvorenia grafu sprievodcom. Ak údaje v tabuľke zmeníme, zmení (prispôsobí sa zmenám) aj graf.

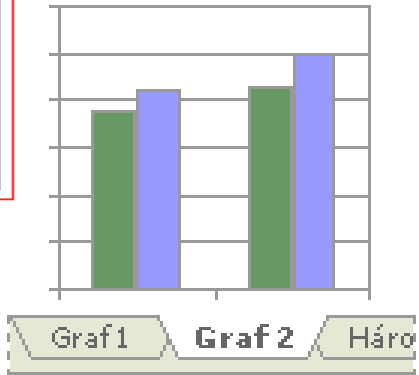


Zobrazenie - Graf môžete vytvoriť na samostatnom hárku alebo ako vložený objekt na pracovnom hárku.

- **Vložené grafy** je považovaný za grafický objekt a ukladá sa ako časť pracovného hárka, v ktorom bol vytvorený. Je výhodný ak chceme zobraziť alebo vytlačiť jeden alebo viac grafov s údajmi pracovného hárka.



ktorý obsahuje iba graf a má
omplexnejšie grafy zobraziť či
s hárkom chcete ušetriť miesto



2. Tvorba grafu pomocou sprivodcu:

V hornom menu si vyberieme z ponuky VLOŽIŤ- GRAF a zobrazí sa mi okno sprievodcu grafom.

1. Môžem si vybrať z rôznych typov(napr. stĺpový, pruhový, koláč...) a podtypov (umiestnenie vedľa seba, nad sebou, v priestore...), ktoré sú ponuke zobrazené, alebo si môžem vytvoriť aj vlastný typ.
2. Rozsah údajov- ak sme dopredu určili bunky v tabuľke, z ktorých chceme graf vytvoriť sú ich hodnoty už zaznamenané. V prípade, že sme tak neurobili, označila sa automaticky celá tabuľka. Ak nechceme použiť všetky údaje tabuľky musíme tieto údaje upresniť pomocou tlačidla, ktoré sa vyskytuje na konci riadku Rozsah údajov. V programe Excel sa pre názvy radov takisto používajú nadpisy stĺpcov a riadkov v údajoch pracovného hárka. Názvy radov sa zobrazia v legende
3. V treťom kroku môžeme pomenovať graf, určiť zobrazenie a umiestenie legendy(je to pole označujúce vzorky a farby priradené jednotlivým radom údajov alebo kategóriám v grafe) a menoviek údajov grafu.
4. V poslednom kroku si určíme umiestnenie(zobrazenie) grafu.

Aj po vytvorení môžeme graf meniť, upravovať.

3. Formátovanie grafu:

graf môžem ďalej upravovať, formátovať: formátovať môžeme všetky časti grafu. Obsah okna pre formátovanie sa mení podľa typu grafu(stĺpcový, koláč, atď.). Časť, ktorú chceme formátovať označíme a stlačením pravého tlačidla myši vyberieme možnosť formátovať, prípadne dvojklikom na označenej časti otvoríme okno na formátovanie.

Formátovanie plochy grafu- tu môžeme upravovať orámovanie, farbu plochy. Tlačidlom *efekty výplne* môžeme nastaviť rôzne farebné prechody, tieňovanie, prípadne vložiť obrázok. Ďalej sa dá nastaviť farba, druh, typ, veľkosť, efekty písma a vlastnosti presunu a veľkosti.

Formátovanie rady údajov- orámovanie, plocha, tvar(napr. pri stĺpcovom grafe- ihlan, valec, hranol,..), menovky údajov, poradie radov, možnosti(hĺbka, šírka medzery, hĺbka grafu).

Formátovanie stien, podstavy- farba a orámovanie

Formátovanie osí- štýl, farba, hrúbka, mierka, písmo, číslo(číslo, mena, čas, účtovnícke, ...), zarovnanie.

Formátovanie menovky- orámovanie, plocha, písmo, číslo, zarovnanie.

Formátovanie legendy- orámovanie, plocha, písmo, umiestnenie

OK, treba veriť, že to spolužiaci zvládnu aj s týmto stručným popisom, 4 kredity

18. Spracúvanie databázových informácií, triedenie, filtrovanie údajov, automatický filter, , rozšírený filter.

Program Excel umožňuje vykonávať základné databázové operácie. Sú to triedenie a filtrovanie. Najjednoduchší spôsob výberu požadovaných záznamov z databázy predstavuje automatický filter.

Pre prácu s databázami sú dôležité tieto pojmy:

pole: stĺpec v databáze (Základnou podmienkou je zabezpečiť, aby celé pole obsahovalo údaje toho istého typu.)

záznam: riadok v databáze

názov poľa: vrchný riadok databázy (záhlavie alebo hlavička tabuľky) obvykle obsahuje názov poľa, ktoré sa skladá z textu popisujúceho pole

Triedenie

Zmene poradia riadkov tabuľky sa najčastejšie podmieňuje ich usporiadaním podľa rastúcich alebo klesajúcich hodnôt určitého znaku (kritéria) uloženého v niektorom poli tabuľky (napr. usporiadanie mien osôb podľa abecedného poradia).

Na rýchle usporiadanie tabuľky podľa jedného kritéria vyberieme bunku daného stĺpca a kliknutím na ikonu štandardného panelu pre vzostupné alebo zostupné (od najväčšieho po najmenšie - klesajúce) usporiadanie tabuľku usporiadame.

Rozšírenie možností usporiadania tabuľky ponúka príkaz *Údaje/Zoradiť*. Zoraďovať je možné súčasne podľa troch kritérií. Ak potrebujeme zoradiť podľa viacerých kritérií, je možné zoraďovanie

opakovať. V takom prípade najprv zoradíme podľa menej dôležitejších kritérií a potom podľa kritérií dôležitejších. Je dôležité pri označovaní do bloku zahrnúť aj záhlavie (hlavičku), pretože v poliach definujúcich podľa čoho sa zoraďuje sa zobrazia miesto anonymných názvov stĺpcov(A,B,C) priamo názvy polí. Prepínače vzostupne a zostupne určujú ako sa bude podľa daného kritéria zoraďovať. Oddiel *Rozsah údajov* umožňuje dodatočne zvoliť, či vybraná oblasť obsahovala záhlavie (hlavičku) alebo nie. Tlačítko *Možnosti...* vyvoláva dialógové okno *Zoradenie-možnosti*, pomocou ktorého môžeme nastaviť ďalšie parametre napr. či má rozlišovať malé a veľké písmená (štandardne to nerozlišuje).

Pri zoraďovaní:
Logická hodnota FALSE je pri vzostupnom poradí pred hodnotou TRUE.
Časy a dátumy sa pri zoraďovaní vo vzostupnom poradí radia od najstaršieho k najnovšiemu.
Prázdne hodnoty sú posledné aj pri vzostupnom, aj pri zostupnom poradí.

Filtrovanie údajov

Niekedy potrebujeme zobraziť iba tie záznamy, ktoré spĺňajú určité podmienky. Dosiahneme to pomocou filtrovania databázy.

Existujú dva typy filtrov: Automatický filter
Rozšírený filter (nepatrí do mat. zadania)

Automatický filter

Vyberieme jednu bunku tabuľky a zadáme príkaz *Údaje/Filter/Automatický filter*. Pri názvoch polí sa v pravej časti zobrazia rozbaľovacie šípky, ktoré po kliknutí myšou zobrazia zoznam dostupných prvkov s nasledovným významom:

(Všetko)	zobrazené budú všetky údaje daného stĺpca
(Prvých 10...)	uplatňuje sa pri číselných údajoch. Po jeho zvolení sa zobrazí dialógové okno *Automatický filter - prvých 10*, pomocou ktorého je možné nadefinovať koľko prvých či posledných hodnôt či percent údajov z daného stĺpca bude zobrazených.
(Vlastné...)	umožňuje nastaviť dve kritériá pre daný filter. Medzi kritériami je možné zvoliť operátor A alebo ALEBO.
názvy položiek	zoznam jednotlivých položiek nachádzajúcich sa v danom stĺpci. Výberom patričnej položky sa nastavuje filter tak, že sú zobrazované len riadky ,ktoré v danom stĺpci obsahujú vybranú položku.
(Prázdne)	zobrazia sa iba riadky obsahujúce v danom stĺpci prázdnu bunku (ak sa v celom stĺpci nenachádza prázdna bunka, tento prvok sa ani nezobrazí v zozname)
(Nie sú prázdne)	zobrazia sa iba riadky obsahujúce v danom stĺpci neprázdnu bunku (ak sa v celom stĺpci nenachádza prázdna bunka, tento prvok sa ani nezobrazí v zozname)

Keď je filter aplikovaný, rozbaľovací ovládací prvok ▼ zmení farbu (farbu zmenia tiež čísla riadkov).
Filtrovať môžeme aj niekoľko polí súčasne.

Rozšírený filter

Pre filtrovanie údajov podľa viacerých kritérií sa používa **Rozšírený filter**.

Najskôr musíme zadefinovať **oblasť kritérií**, podľa ktorých budeme filtrovať. V prvom riadku sú názvy polí, ktoré napr. skopírujeme, v druhom sú hodnoty, podľa ktorých budeme filtrovať. Ak sú hodnoty vedľa seba sú spojené logickou spojkou "a", ak je jedna o riadok nižšie sú spojené logickou spojkou „alebo", prázdna bunka znamená výber všetkých hodnôt.

- Ako kritérium môže byť text (celý názov, alebo začiatočné písmená), číslo, reťazec, alebo odkaz na inú bunku.
- „*" zastupuje všetky znaky od danej pozície, „?" nahrádza ľubovolný znak,.

1. Kliknite na ľubovoľnú bunku v zozname.
2. Z ponuky **Data** zvoľte príkaz **Filter** a z jeho podponuky **Rozšírený filter**.
   - Ak chcete priamo v zozname zobraziť záznamy, ktoré vyhovujú kritériám aktivujte prepínač **Priamo v zozname.**

- Ak chcete filtrované dáta zobraziť niekde inde a pritom zachovať pôvodný zoznam, kliknite na prepínač **Kopírovať inam**, potom v poli **Kopírovať do** vyberte jednu bunku (skopírujú sa všetky kritériá), alebo skopírujete niektoré kritéria a len tie sa vyfiltrujú. Výsledky filtrácie môžu byť aj na inom liste.
- Do poľa oblasť kritérií zadajte odkaz na oblasť kritérií vrátane riadku, ktorý obsahuje názvy kritérií.

OK, 5 kreditov

19. Zásady editovania a úprav textu. Blokové operácie, formát písma, formát odstavca, štýl odstavca. Kontrola pravopisu, vyhľadanie a náhrada textových reťazcov.

Zásady editovania a úprav textu:

Musíme dodržiavať:

- Konce odstavcov
- Medzery
- Tvrdé medzery
- Tabulátory
- Pevné konce riadkov
- a ďalšie

Blokové operácie:

Práca s blokmi:

Blok je inverzne označená časť textu definovaná začiatkom a koncom. Môže to byť jediný znak, časť dokumentu alebo celý dokument. Je možné robiť s ním úpravy kopírovanie, presun a podobne.

Označenie bloku:

- **Písmená** od prvého písmena až po posledné ťahať
- **Slovo** 2xľavé tlačidlo myši
- **Veta** Ctrl + ľavé tlačidlo v ľubovoľnej časti vety
- **Odstavec** 3x ľavé tlačidlo na myši v ľubovoľnej časti odstavca
- **Riadok** myšou vo výberovom pruhu ľavé tlačidlo pred riadok
- **Celý dokument** 3x ľavé tlačidlo vo výberovom pruhu
- **Stľpcový blok** Alt + ťahať ľavé tlačidlo

Formát písma:

Zmena typu(fontu), rez(tučné, kurzíva...) veľkosti, farby a pod.

Všetky tieto zmeny môžeme nastaviť cez **formát, písmo.**

Formát odstavca(odseku): Formát odstavca tvoria vlastnosti, ktoré sa používajú pre celé odstavce, a nie len pre ľubovoľnú časť textu. Zmenu formátu môžeme urobiť v niekoľkých odstavcoch naraz, ak ich predtým označíme. Ak nie je žiadny odstavec označený, vykoná sa zmena v odstavci, v ktorom je kurzor.

Formát odstavca nastavujeme: cez menu **FORMAT – ODSTAVEC, Karta ODSADENIE A MEDZERY.**

Nastaviť sa dá: zarovnanie textu, riadkovanie, odsadenie vo zvislom smere, odsadenie vo vodorovnom smere, tabulátory.

Štýl odstavca:

ŠTÝL je množina formátovacích inštrukcií, ktorá mení vzhľad textu. Jeho výhodou je ľahká zmena formátov pre mnoho odstavcov naraz.

Vytvorenie štýlu odstavca zo vzorky textu:

- zapíšte príklad odstavca, ktorý chcete formátovať
- Vo voľbe FORMÁT – ODSTAVEC, FORMÁT PÍSMO nastavte parametre odstavca
- kurzor umiestnite na odstavec

- klepnite na rozbaľovací zoznam Štýl na formátovacom paneli nástrojov (1.zľava)
- v políčku ŠTÝL zapíšte meno štýlu a stlačte ENTER

Tvorba štýlu pomocou príkazu štýl:

- Umiestnite kurzor na odstavec, ktorý chcete zmeniť
- zvoľte FORMÁT – ŠTÝL – klepnite na tlačidlo NOVÝ
- zapíšte meno nového štýlu do poľa názov
- v zozname typ štýlu zvoľte ODSTAVEC (znak)
- stlačte tlačidlo FORMÁT – každému štýlu odstavca môžete priradiť nastavenie výberom jedného zo 7 príkazov (písmo, odstavce, … číslovanie)
- Pole ŠTÝL NASLEDUJÚCEHO ODSTAVCA umožní špecifikovať, aký štýl bude mať nasledujúci odstavce (aplikuje sa len keď vytvoríte nový prázdny odstavec – na už existujúce odstavce neplatí)
- Klepnite na tlačidlo KLÁVESOVÁ SKRATKA priraďte klávesovú skratku
- Stlačte PRIRADIŤ

Kontrola pravopisu:

Slúži ako pomôcka pri písaní dokumentu, keď nám uľahčuje čas tým, že kontroluje to čo píšeme v reálnom čase alebo až po dokončení písania dokumentu.

Pravým tlačidlom myši na slovník, ktorý sa nachádza na spodku dokumentu, MOŽNOSTI – tu si môžeme vybrať rôzne funkcie KONTROLY PRAVOPISU(automa. kontr. pravop., ...)

Vyhľadanie a náhrada textových reťazcov:

V menu **ÚPRAVY** – tlačidlo **HĽADAŤ**: v prvej lište máme na výber funkciu HĽADAŤ, ktorou vyhľadáme svoje zvolené slovo. V rozšírených vlastnostiach môžeme rozšíriť vlastnosti hľadania napr. o celé slová, foneticky podobné, rozlišovať malé a veľké slová,....

V ďalšej lište **NAHRADIŤ**, funguje v podstate rovnako ako hľadanie, ale na rozdiel a neho máme možnosť nájdené slovo nahradiť naším zvoleným, ktorý v prípade výskytu viacerých slov môžeme nahradiť naraz alebo postupne.

OK, 5 kreditov

20.Pohľady na dokument. Vzhľad stránky, záhlavie a päta, Náhľad. Šablóna, osnova, obsah a register. Hromadná korešpondencia.

Vloženie hlavičky a päty

cez ponuku **Zobraziť —> Hlavička a päta**

Zobrazí sa HLAVIČKA a panel s nástrojmi pre **Hlavičku a pätu**

Môžeme sem vkladať rôzne údaje napr. číslo strany, dátum, čas...

Pohľady na dokument (pre MS WORD 2003)

normálne zobrazenie

zobrazenie Webové rozloženie

zobrazenie rozloženie pri tlači

zobrazenie prehľadu

rozloženie na čítanie

Náhľad na stránky dokumentu

Šablóna

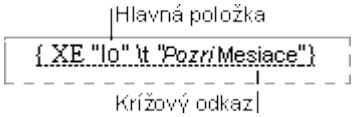
Každý dokument programu Microsoft Word je založený na šablóne. Šablóna určuje základnú štruktúru dokumentu a obsahuje nastavenia dokumentu, ako sú napríklad položky automatického textu, písmo, priradenia klávesov, makrá, ponuky, rozloženie strany, špeciálne formátovanie a štýly.Globálne šablóny vrátane normálnej šablóny obsahujú nastavenia, ktoré sú dostupné pre všetky dokumenty. Šablóny dokumentov, ako sú napríklad šablóny oznamov alebo faxov v dialógovom okne **Šablóny**, obsahujú nastavenia, ktoré sú dostupné len pre dokumenty založené na danej šablóne

Obsah

Text, ktorý chcete zahrnúť do obsahu, môžete označiť týmito spôsobmi:použitím štýlov nadpisov, vytvorením prehľadu dokumentu, vytvorením vlastných štýlov. Používanie vstavaných štýlov nadpisov programu Word na formátovanie dokumentov je zrejme najjednoduchší spôsob označovania textu. Stačí jednoducho formátovať text pomocou jednej z deviatich preddefinovaných úrovní nadpisov a vytvoriť obsah.Výhodami používania štýlov nadpisov sú jednoduchosť použitia a rýchlosť. Po označení textu je čas zhromaždiť všetky položky vo forme obsahu. Túto úlohu program Word vykoná za vás.Umiestnite kurzor na miesto, kde sa má zobraziť obsah – zvyčajne to bude začiatok dokumentu. V ponuke **Vložiť** potom ukážte na položku **Odkaz**, kliknite na položku **Register a zoznamy** a kliknite na kartu **Obsah**. Ak chcete použiť predvolené možnosti, vytvorte obsah kliknutím na tlačidlo **OK**.

Register

Register obsahuje termíny a témy, ktoré sa vyskytujú v určitom dokumente, spolu so stránkami, na ktorých sa nachádzajú. Ak chcete vytvoriť index, označte položky registra v dokumente a vytvorte index. Po prvom označení položky, program Microsoft Word pridá špeciálne pole položiek registra XE (položka registra) k dokumentu:



Môžete vytvoriť položku registra: pre jednotlivé slovo, frázu alebo symbol, pre tému, ktorá presahuje rozsah strán, odvolávajúcu sa na inú položku, napríklad „Preprava. *Pozri* Bicykle“. Ak ste už označili všetky položky registra, zvoľte úpravu registra a zostavte hotový register. Program Word potom zhromaždí položky registra, utriedi ich podľa abecedy, nájde odkazy na príslušné čísla stránok, nájde a odstráni zdvojené položky z tej istej strany a zobrazí register v dokumente. Program Word umiestni symboly ako je @, na začiatok registra. Ak si vyberiete formát registra, ktorý obsahuje nadpisy abecedných skupín, symboly sa zoskupia pod nadpisom # (znak čísla). Ak použijete doplňujúce možnosti prispôsobenia registra, môžete použiť polia. Môžete napríklad zostaviť register iba pre časť dokumentu.

Hromadná korešpondencia

Hromadná korešpondencia sa používa v prípadoch, keď chcete vytvárať kolekcie dokumentov, ktoré sú v podstate rovnaké, ale každý z nich obsahuje aj určité jedinečné prvky. Ak použijeme ako príklad list,

ktorý ohlasuje uvedenie nového produktu, potom logo vašej spoločnosti a text o produkte bude v každom liste rovnaký, ale adresát a oslovenie sa budú v jednotlivých listoch líšiť. Pomocou hromadnej korešpondencie môžete vytvoriť: **kolekcie štítkov alebo obálok** (adresa odosielateľa bude na všetkých štítkoch a obálkach rovnaká, ale adresát bude na každom štítku alebo na každej obálke iný), **kolekcie formulárových listov, e-mailových správ alebo faxov** (základný obsah je vo všetkých listoch, správach a faxoch rovnaký, ale jednotlivé listy, správy alebo faxy obsahujú aj informácie špecifické pre jednotlivých príjemcov, ako je napríklad meno, adresa alebo iné osobné údaje), **kolekcie číslovaných kupónov** (kupóny sú identické, ale ich čísla sú jedinečné).

Vytváranie listov, správ, faxov, štítkov, obálok alebo kupónov po jednom by vám trvalo niekoľko hodín. A práve preto je vhodnejšie použiť hromadnú korešpondenciu. Ak použijete hromadnú korešpondenciu, stačí vám vytvoriť jeden dokument s informáciami, ktoré sú vo všetkých verziách rovnaké. Potom len pridáte zástupné symboly pre informácie, ktoré sú v každej verzii jedinečné. O zvyšok sa postará program Word.

1. Spustite program Word

   Na základe predvoleného nastavenia sa otvorí prázdny dokument. Ponechajte ho otvorený. Ak ho zavriete, nebude možné vykonať ďalší krok.

2. V ponuke **Nástroje** ukážte na položku **Listy a korešpondencia** a potom kliknite na položku **Hromadná korešpondencia**.

   POZNÁMKA. Ak ide o program Word 2002, kliknite na ponuku **Nástroje**, ukážte na položku **Listy a korešpondencia** a potom kliknite na príkaz **Sprievodca hromadnou korešpondenciou**.

Otvorí sa pracovná tabla **Hromadná korešpondencia**. Pomocou hypertextových prepojení na pracovnej table potom prejdite jednotlivými krokmi spracovania hromadnej korešpondencie.

OK, 5 kreditov

21. Stĺpcová sadzba. Odrážky a číslovanie, tabelátory, tabuľky, textové pole. Grafika.

Stĺpcová sadzba

Táto funkcia textového editora MS Word umožňuje upraviť ľubovoľný text na novinové stĺpce, ktoré môžeme ďalej upravovať – meniť ich šírku, počet, vzhľad atď. Celý dokument alebo jeho časť môžeme písať vo forme novinových stĺpcov, kde text z konca jedného stĺpca pokračuje na začiatku stĺpca nasledujúceho. Stĺpce pritom môžu mať rôznu šírku. Dokonca môžeme meniť počet stĺpcov, a to i v rámci jednej stránky. Ak chceme aj na obrazovke vidieť stĺpce vedľa seba, musíme prepnúť do stránkového zobrazenia alebo do ukážky pred tlačou.

Možnosti vytvorenia stĺpcov

Klepneme na tlačítko Stĺpce na štandardnom paneli nástrojov a premiestnením ukazovateľa myši vyberieme požadovaný počet stĺpcov. Stĺpce sa vytvoria iba v oddieli, v ktorom je umiestnený kurzor. Ak dopredu vyberieme časť dokumentu a až potom klepneme na tlačítko Stĺpce na paneli nástrojov, vytvoria sa stĺpce rovnakej šírky iba vo vybranej časti dokumentu. Word automaticky vloží znaky konca oddielu pred a za vybranou časťou.
(Oddiel je časť dokumentu, v ktorej sa nastavia určité možnosti formátovania stránky. Nové oddiely sa tvoria kvôli zmene vlastností, ako je číslovanie riadkov, počet stĺpcov alebo záhlavie. Ak nevložíme koniec oddielu, Word považuje dokument za jediný oddiel.)

- Vytváranie stĺpcov nerovnakej šírky

Pomocou príkazu Stĺpce z ponuky Formát a v poli Predvoľby vyberieme Vpravo alebo Vľavo. Vľavo znamená, že ľavý stĺpec bude užší – rozdelenie medzi stĺpcami bude vľavo od stredu. Vpravo potom znamená, že užší bude pravý stĺpec.

- Zmena šírky stĺpca

V stránkovom zobrazení umiestnime ľavú alebo pravú zarážku stĺpca na vodorovnom pravítku a upravíme tak šírku stĺpca. Ak by boli vytvorené rovnako široké stĺpce, zmení sa šírka všetkých stĺpcov. Ak chceme zadať šírku každého stĺpca a každej medzistĺpcovej medzery, musíme zvoliť príkaz Stĺpce z ponuky Formát.

- Zmena stĺpcovej úpravy
Stĺpcovú úpravu môžeme meniť nasledujúcimi spôsobmi:

- **Zmena zalomenia stĺpca**. Word zarovnáva stĺpce na stránke automaticky. Existuje však tiež možnosť zalomiť stĺpce v ľubovoľnom inom mieste – napríklad medzi odstavcom alebo medzi obrázkom a odstavcom, ktoré chceme mať pohromade. Vykonáme to príkazom Vložiť – Koniec – Zalomenie stĺpca.
- Zmena šírky stĺpca a medzistĺpcovej medzery.
- Voľba medzi stĺpcami rovnakej a nerovnakej šírky.
- Zmena počtu stĺpcov.
- Pridaním zvislých čiar medzi stĺpce.
- Zarovnaním dĺžok stĺpcov na stránke. Na konci oddielu a na konci dokumentu sa obvykle text nerozdelí rovnomerne do všetkých stĺpcov. Ak vložíme na koniec posledného stĺpca tzv. priebežné zalomenie oddielu, rozdelí sa text rovnomerne do všetkých stĺpcov. Nový oddiel potom môže začať na tej istej stránke. Vložiť - Koniec – Koniec oddielu na rovnakej stránke.
- Kombinácia stĺpcov s obrázkami. Kombináciou obrázkov a novinových stĺpcov dáme svojmu dokumentu zaujímavejší vzhľad. Orámovaný obrázok pritom môže zaujímať časť šírky stĺpca, celú šírku stĺpca alebo môže zasahovať do niekoľkých stĺpcov.

ODRÁŽKY A ČÍSLOVANIE:

Pridanie odrážky alebo číslovania:

- Označte položky, ku ktorým chcete pridať odrážky alebo číslovanie.
- Na paneli s nástrojmi **Formátovanie** vykonajte jednu z nasledovných činností:
  1.Ak chcete pridať odrážky, kliknite na tlačidlo **Odrážky**
  2.Ak chcete pridať čísla, kliknite na tlačidlo **Číslovanie**

Odrážku je možné nahradiť obrázkom: **V ponuke** Formát **kliknite na príkaz** Odrážky a číslovanie **a potom kliknite na kartu** S odrážkami**. Kliknite na tlačidlo Obrázok ,vyberte si požadovaný obrázok a kliknite na tlačidlo** Vložiť klip 
Na karte **S odrážkami** po stlačení tlačidla **Prispôsobiť...** môžeme nastaviť znak a umiestnenie odrážky a textu
Na karte **Číslované** po stlačení tlačidla **Prispôsobiť...** môžeme nastaviť Formát čísiel, Štýl číslovania, umiestnenie čísla a textu
Na karte **Viacúrovňové** po stlačení tlačidla **Prispôsobiť...** môžeme takisto nastaviť Formát čísiel, Štýl číslovania, umiestnenie čísla a textu pre každú z úrovní.

Formátovanie tabuľky:

V tabuľke môžete pridávať okraje a tieňovanie, zlučovať bunky, rozdeľovať bunky a formátovať odstavce v bunkách pomocou väčšiny príkazov slúžiacich na formátovanie bežných odstavcov. V tabuľke je možné údaje zoraďovať a prevádzať výpočty.
Čiary mriežky, ktoré ohraničujú bunky tabuľky, sa nevytlačia. Sú to iba vodítka, ktoré vám vravia, v ktorej bunke práve pracujete. Ak chcete pridať skutočné okraje (ktoré sa vytlačia), vyberte bunku, viac buniek, stĺpec alebo celú tabuľku. Zvoľte *Formát Ohraničenie a tieňovanie.* Zobrazí sa dialógové okno *Bunky* a potom môžete formátovať bunky pomocou rovnakých možností ako keď používate dialógové okno *Ohraničenia a tieňovania* pre bežný text.

Úpravy výšky a šírky buniek:
Ak chcete nastaviť presnú výšku a šírku buniek, nemôžete k tomu používať myš. Musíte zvoliť *Tabuľka Výška a šírka bunky*. Výšku riadku môžete zmeniť z Auto na konkrétnu hodnotu zvolením Presne v rozbaľovanom zozname *Výška riadkov* a nastavením hodnoty v číselníku *Rozmer*. Hodnotu môžete zadať v centimetroch (cm), bodoch (b.) alebo v riadkoch (ř.) napísaním patričnej skratky za číslo. Rovnako môžete nastaviť šírku stĺpca a zmeniť veľkosť rozostupu medzi stĺpcami na karte *Stĺpec*. Kliknite na kartu *Stĺpec Predchádzajúci stĺpec* alebo *Ďalší stĺpec*, ak chcete prechádzať medzi stĺpcami. Zadajte pre každý stĺpec požadované hodnoty do číselníka v dialógovom okne.

Obrysy tabuľky a tieňovanie:

Ak chcete orámovať niektorú tabuľku, umiestnite kurzor do niektorej bunky alebo ju vyberte celú a zvoľte príkaz *Orámovanie a podfarbenie* v ponuke *Formát* a nalistujte kartu *Obrysy*

(prípadne *Podfarbenie* ak chcete zmeniť podfarbenie). Ak chcete orámovať len niektoré bunky, vyberte ich ešte pred uskutočnením príkazu *Orámovanie a podfarbenie*.

Zlúčenie buniek:

V prípade, že chcete zlúčiť bunky, stačí keď označíte bunky ktoré chcete zlúčiť a v ponuke *Tabuľka* označíte *Zlúčiť bunky*. Nová bunka bude mať rovnakú šírku ako bola šírka všetkých zlučovaných buniek.

Prevedenie textu do tabuľky a späť:

Ak chcete existujúci text previesť do tabuľky, stačí označiť text a zvoliť príkaz *Konvertovať text na tabuľku* v ponuke *Tabuľka*. Ak zvolíte príkaz *Konvertovať tabuľku na text*, prevediete tabuľu opäť na text.

Vloženie hlavičky a päty
cez ponuku **Zobraziť —> Hlavička a päta**
Zobrazí sa HLAVIČKA a panel s nástrojmi pre **Hlavičku a pätu**
Môžeme sem vkladať rôzne údaje napr. číslo strany, dátum, čas...

Členenie textu
Znak – elementárna jednotka textu – znak, číslica
Slovo – zložené zo znakov, oddeľuje sa medzerou
Veta – zložená zo slov a oddeľuje sa bodkou a medzerou
Riadok – má mäkké ukončenie(plynulý prechod na ďalší)
- tvrdé ukončenie (Enter alebo bez ukončenia Shift +Enter)

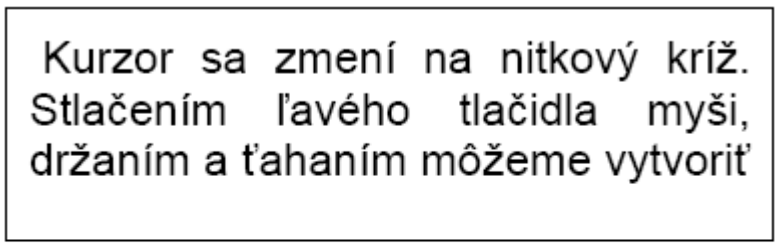
Odsek – text ukončený klávesou Enter

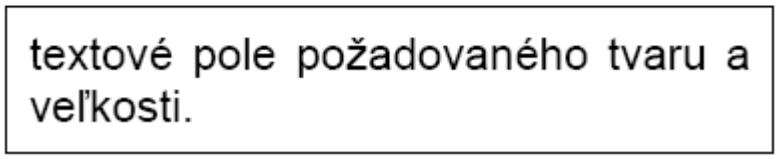
Blok – označená časť textu

TEXTOVÉ POLE

Vytvorenie textového poľa

- Ak chceme vytvoriť textové pole pre už napísaný text, označíme ho a zadáme príkaz VLOŽIŤ (INSERT) - TEXTOVÉ POLE (TEXTBOX).
- Ak chceme vytvoriť prázdne textové pole, nič neoznačujeme a zadáme príkaz VLOŽIŤ - TEXTOVÉ POLE. Kurzor sa zmení na nitkový kríž. Stlačením ľavého tlačidla myši, držaním a ťahaním môžeme vytvoriť textové pole požadovaného tvaru a veľkosti.





Po vytvorení textového poľa je okolo neho silná čiara a malé štvorčeky – *úchytky*. Tieto prvky sú netlačiteľné znaky. Ak chceme robiť s textovým poľom nejakú činnosť (presúvanie, meniť veľkosť ťahaním za úchytky) musí byť takto označené.

Prepojenie textových polí

Plynutie textu z jedného textového poľa do druhého: Označíme prvé z textových polí, klikneme na jeho ráme pravým tlačidlom myši a z plávajúceho menu vyberieme ponuku VYTVORIŤ PREPOJENIE TEXTOVÉHO POĽA (CREATE TEXT BOX LINK). Kurzor sa zmení na šálku. Klikneme ním v druhom textovom poli (vylejeme šálku).

Odstránenie textového poľa

Pole označíme kliknutím na jeho okraj, stlačíme klávesu DELETE a vymažeme ho aj s obsahom.

Zmena umiestnenia textového poľa

Klikneme na textové pole, nastavíme kurzor na jeho okraj mimo úchytky, kliknutím a ťahaním presúvame.

Formát textového poľa

Jednotlivé nastavenia pre textové pole (obrysy a tieňovanie, veľkosť, obtekanie textom...) robíme cez príkaz FORMAT - TEXTOVÉ POLE (TEXTBOX)

Vkladanie obrázkov

Obrázok je objekt obsahujúci grafiku, ktorý je možné vložiť do dokumentu. Obrázky nemusíme iba kresliť vlastnými silami. Nájdeme ich uložené v súbore a do textu ich vkladáme ako súbor, pretože pri vkladaní musí prebehnúť konverzia do tvaru, ktorý je možno vo Worde použiť.

Vloženie obrázku do textu nie je zložitá záležitosť. Do textu môžeme napríklad vložiť oddeľovač, ktorý je súčasťou clipartu dodávaného spolu s textovým editorom Word.

1. Presunieme kurzor na miesto, kam chceme obrázok vložiť.
2. Zadáme príkaz Vložiť – Obrázok.
3. Nastavíme adresár /zložku/ so súbormi clipart.
4. Prehliadneme si typy grafických súborov, ktoré môžeme vložiť, vo voľbe Typ zobrazených súborov. Ak nepoznáme typ súboru, môžeme zvoliť všetky grafické súbory.
5. Vyberieme súbor a klikneme naň dvakrát myšou.

Obrázok sa uloží na určené miesto v dokumente. Každý obrázok vložený do textu sa bude chovať tak, ako keby to bol odstavec textu. Dokonca jeden odstavec môže tvoriť obrázok i text – potom sa obrázok správa tak, ako keby to bol jeden znak. Preto môžeme použiť pre umiestnenie obrázku formát odstavca.

Z obrázkom možno prevádzať niektoré úpravy, predovšetkým zmeniť jeho veľkosť. Môžeme využiť myš a nastaviť rozmery alebo nastaviť pomerné veľkosti oproti pôvodnej veľkosti obrázku. Ďalšie úpravy, predovšetkým jeho umiestnenie, orámovanie a ďalšie môžeme prevádzať pomocou rámu.

OK, toto je dobrá práca

Ale na konci chýba odstavec o umeleleckom texte (vložiť – obrázok - wordart) a to sú pekné veci a veľa možností nastavenia i použitia, ktoré by si zaslúžili pozornosť v tejto maturitnej otázke. Preto prosím dopísať ešte jeden odstavec.

OK, 5 kreditov

22. Úloha prezentácie, oblasti použitia prezentácie, formy (riadená prednášajúcim, automatická). Zásady pri tvorbe prezentácie a jej predvádzaní

Slúži na tvorenie multimediálnych prezentácií. Najznámejší program je Microsoft Powerpoint. Jeho úlohou je za pomoci užívateľa vytvoriť prezentáciu, ktorá môže mať rôzne **funkcie:** poučiť, informovať, pobaviť, atď. Výstupom môže byť pc (monitor, alebo projektor), internet, alebo tlač.

Prezentáciu môžeme vytvoriť pomocou: 1.Sprievodcu - pomocou krátkych krokov vytvorí hlavnú kostru prezentácie.

2.Predpripravených šablón, ktoré poskytujú užívateľovi návrhy, ktoré predurčujú vzhľad celej prezentácie.

3.Prázdnej prezentácie, ktorá ponúka možnosť vytvoriť novú prezentáciu presne podľa predstáv daného užívateľa

Prezentácia sa skladá zo série snímok, na ktorých môžu byť umiestnené bloky textu, obrázky, tabuľky, grafy, animácie, videozáznamy a pod. každej jednotlivej snímke môžeme navrhnúť: pozadie, farebnú schému (nastavenie farieb pre text, pozadie, prepojenia a pod.), rozloženie, schému animácií, prechod snímky.

Tvorba prezentácie:

1) Založenie snímky - Powerpoint nám ponúkne rozvrhnutí snímkov
2) Výber farby pozadia
3) Vzhľad výplne – zvolíme príslušnú farbu alebo vzhľad výplne, tieňovanie farieb pomocou jednej alebo dvoch farieb. Rovnako si môžeme vybrať z palety už vopred pripravených tieňovaní.
4) Vkladanie textu - PowerPoint automaticky označuje textové polia, do ktorých môžeme zadávať text. Pri práci s textom si označíme text, s ktorým chceme pracovať, môžeme meniť farbu písma, veľkosť, font, štýl, zarovnanie.
5) Vkladanie objektov- podpora práce s objektmi (tabuľka, graf, obrázok, video, zvuk).
6) Úpravy objektov- jednotlivé objekty môžeme ešte ďalej upravovať. Napríklad pomocou 3D zobrazenia môžeme z kruhu robiť valec. Objekt môžeme otáčať, meniť spôsob osvetlenia, tvar a povrch.
7) Animácia objektov- časovanie: postupne si vyberáme objekty v poradí, v akom ich chceme mať zanimované. V tomto poradí sa postupne objavujú v okne *Poradie snímkov*. V okne *Spustiť animáciu* určíme či chceme daný objekt zanimovať. Ak áno, môžeme nastaviť prechod snímok stlačením tlačidla myši alebo sa snímka prepne automaticky po nastavených minútach. Pre každý objekt môžeme nastaviť efekty s akými sa zobrazia na snímke. V okne *Úvodná animácia a text* si vyberáme spôsob nástupu snímok a sprievodný zvuk.

Zásady tvorby prezentácie

V celej prezentácii by sa mali zachovať rovnaké vlastnosti všetkých snímok:

- Formát písma, spôsoby zarovnania textu
- Vzhľad jednotlivých úrovní zoznamov (odrážok)
- Farebná schéma snímky
- Vlastnosti textových polí
- Nastavenie animačných vlastností
- Päta snímky (číslovanie, názov prezentácie, …)

OK, 5 kreditov

23. Snímka, stránka, textové polia, text, tabuľku, obrázok a graf, pozadie snímky. Animácie objektov v snímke (poradie animácie objektu, efekt, časovanie a zvuk. Prechodové efekty medzi snímkami, riadenie prechodu na nový snímok, zvuk pri uvedení snímky.

PowerPoint slúži na tvorenie multimediálnych prezentácií. Najnovšou verziou je PowerPoint 2007. Výstupom môže byť PC (monitor alebo projektor), webstránka alebo tlač.
Snímka (=slide, frame) je časť prezentácie, ktorá obsahuje text a/alebo multimediálne dáta (graf, obrázky, video, zvuk).
Textové polia obsahujú text. Môžu mať rôzne rozloženie v rámci snímky.
Tabuľka obsahuje prehľadne rozložené dáta, ktoré môžu byť použité v grafe. Obsahuje stĺpce a riadky.
Graf obsahuje graficky zobrazené dáta. Jeho časti môžu používať rôzne vizuálne efekty. Existujú rôzne typy grafov: stĺpcový, pruhový, spojnicový, výsekový, plošný, atď..
Animácie sú pohyblivé prvky na snímkach, ktoré pridávajú dynamický charakter. Animácia každého prvku na snímke môže byť spustená samostatne alebo spolu.
Zvuk môže byť priradený každej akcii v prezentácii. Napr. pri prechode snímkov
Čo dodať, odpovedajúci, pomôž si sám ☹
2 kredity

24. Databázový systém. Databázový súbor, záznam. Objekty databázy, vytvorenie novej databázy. Vytvorenie tabuľky, základné operácie v tabuľke. Triedenie záznamov. Vyhľadávanie a náhrada textových reťazcov.

Na začiatok si definujeme potrebné pojmy:

- **databáza (DB)**
  - súbor účelovo zoradených, štruktúrovaných údajov, ktoré popisujú danú skutočnosť (tovar na sklade, inventár, adresár ľudí…)
- **databázový systém**
  - program na manipuláciu s databázou, príklady: Oracle, MS Access, MS SQL server, PostgreSQL, MySQL

Na nasledujúcich riadkoch sa budeme zaoberať DB systémom MS Access.

Objektmi DB v tomto programe sú:

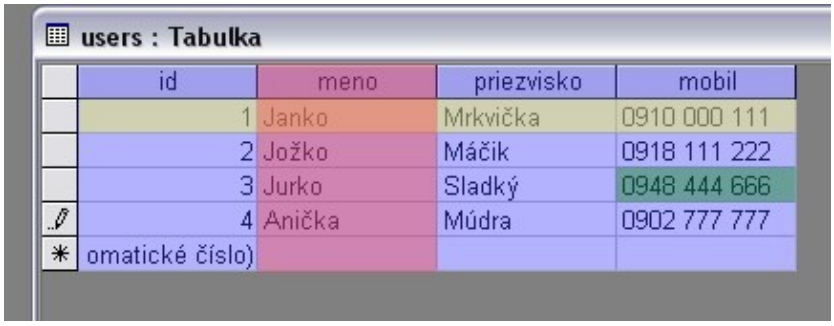
- tabuľky
- otázky (dotazy)
- formuláre
- zostavy
- stránky
- makrá
- moduly

Všetky tieto objekty sa ukladajú do jedného súboru s príponou *.mdb*

Tabuľka

Je to základný objekt DB. Obsahuje údaje týkajúce sa vždy jedného typu subjektu (človek, článok, tovar ...). Tabuľka sa skladá so záznamov. Jeden záznam je jeden riadok v tabuľke. V jednej tabuľke majú všetky záznamy jednotne definované vlastnosti – polia. Každé pole je určené názvom (meno, mobil, autor ...), typom (text, číslo, dátum, logická hodnota [boolean] ...) a dĺžkou.

- **tabuľka**
- **záznam**
- **vlastnosť (stĺpec, skupina polí)**
- **pole**

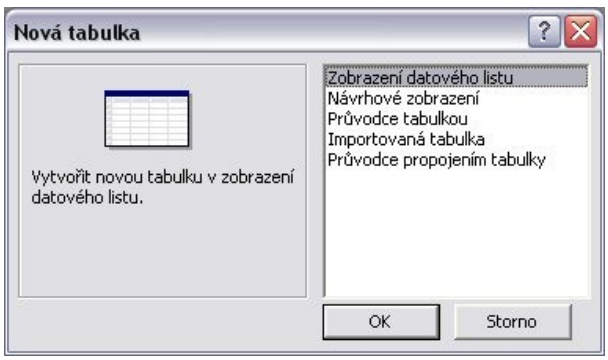


Dátové typy v MS Access: text, memo, číslo, dátum/čas, automatické číslo, mena, boolean, OLE objekt (odkaz na dokument, obrázok), hypertextový odkaz

Vytvárame tabuľku

V MS Access môžeme tabuľku vytvoriť 3 spôsobmi:

- v návrhovom prostredí
- pomocou sprievodcu
- vkladaním údajov (zobrazením dátového listu)



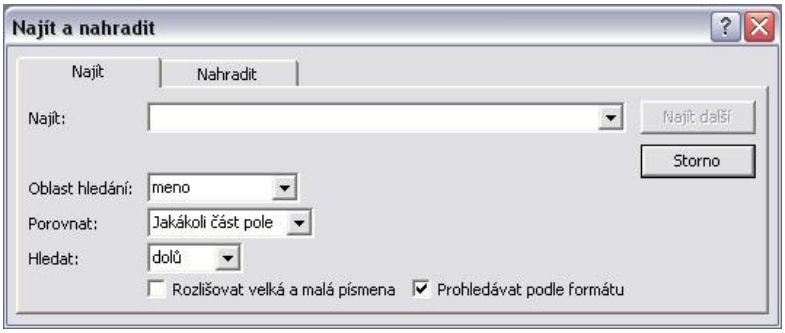
Základné operácie s tabuľkou

S tabuľkou sa v MS Access dajú robiť rôzne operácie. Môžeme meniť štruktúru tabuľky – treba však postupovať opatrne, pretože môže dôjsť k strate údajov. Ďalej môžeme meniť výzor tabuľky (väčšina úprav je realizovaná cez položku Formát v menu programu). Taktiež môžeme narábať so stĺpcami: pridávať, mazať, upravovať vlastnosti, upravovať ich šírku, môžeme ich skrývať alebo meniť ich poradie a iné (buď cez návrhové prostredie alebo dátový list)

Triedenie, vyhľadávanie a nahradzovanie textových reťazcov

Na spustenie daných funkcií programu môžeme využiť príslušné ikony alebo menu (Úpravy – Hľadať [Ctrl+F], Záznamy – zoradiť).

Záznamy môžeme triediť zostupne alebo vzostupne. Pred spustením triedenia sa kurzorom nastavíme na stĺpec podľa, ktorého chceme údaje triediť. Pokiaľ chceme, môžeme záznamy triediť podľa viacerých kritérií – stačí keď označíme viacero stĺpcov.



Vyhľadávanie funguje na tej istej báze. Kurzorom sa nastavíme na daný stĺpec, v ktorom chceme hľadať (prípadne označíme viacero stĺpcov alebo dokonca celú tabuľku). Zadáme aký reťazec chceme hľadať, aké podmienky má spĺňať (aká časť záznamu má byť porovnávaná s hľadaním reťazcom, či sa majú rozlišovať malé a veľké písmená ...) a môžeme dať hľadať.

Náhrada textových reťazcov funguje na tom istom princípe ako vyhľadávanie. Nahradzovať môžete po jednom (pri každom nájdenom zázname podľa kritérií sa môžete rozhodnúť či daný text nahradíte) alebo naraz (všetky vyhovujúce reťazce budú nahradené vami zadaným textom)

Vytvárame kvalitnú Databázu

Na záver ešte pár *bonusových* informácií. Pri vytváraní DB platí staré známe: *čas sú peniaze* ale v troška inom význame. Je dobré nepodceňovať čas venovaný návrhu DB, pretože to je jeden (ak nie aj jediný) z kľúčových momentov pri tvorbe DB. Vždy je totiž lepšie mať kvalitne navrhnutú DB od začiatku (keď sa budeme snažiť myslieť na všetky možné i nemožné prípady, ktoré môžu nastať počas jej existencie) ako potom neskôr strácať čas a peniaze pri jej dodatočnej (a často náročnej) úprave.

Normalizačné pravidlá

Sú to pravidlá používane pri tvorbe DB aby sa zaistila jej správnosť a optimalizácia. Existujú 3 normálové formy (NF) (niekedy môžete naraziť na číslo 6 ale posledné tri formy sa v praxi takmer nepoužívajú). Pokúsim sa tu teda v jednoduchej forme popísať prvé 3 normálové formy.

1. normálová forma

Obsahuje nasledujúce pravidlá:

- treba eliminovať stĺpce s rovnakým obsahom
- je potrebné vytvoriť tabuľku pre každú skupinu príslušných údajov
- každý záznam musí byť jednoznačne identifikovaný primárnym kľúčom

Po slovensky to znamená asi toto:

Ak máme tabuľku a v nej stĺpce:

titul, vydavateľstvo, rok, meno_autora1, meno_autora2, meno_autora3

je potrebné stĺpce *meno_autora1, meno_autora2, meno_autora3* odstrániť a vytvoriť nové záznamy pre každého autora. Ďalej táto forma vraví o tom, že každý záznam je potrebné označiť primárnym kľúčom – najčastejšie sa používa obyčajné číslo typu *integer* (udáva akoby poradie, v akom bol záznam vložení do DB)

2. normálová forma

- zakaždým keď sa opakujú obsahy niektorých stĺpcov v rôznych záznamoch, musíme tabuľku rozdeliť na viacero tabuliek
- tieto tabuľky prepojíme cudzími kľúčmi

Po aplikovaní 1. NF sa nám v tabuľke môže stať, že ak má kniha viacero autorov bude v nej uvedená presne toľkokrát koľko autorov má. Teda každý autor bude reprezentovať jeden riadok avšak ostatné údaje budú rovnaké (spolupracovali predsa na tej istej knihe). Teraz je vhodné našu tabuľku rozdeliť na viacero tabuliek, konkrétne tri. Jedna pre autorov (bude obsahovať meno autora a jeho primárny kľúč) a druhá bude pre vzťah autorov a knižiek (keďže jeden autor mohol napísať viacej knižiek a jedna knižka môže mať viacero autorov) – táto tabuľka bude obsahovať len 2 stĺpce (primárny kľúč autora a primárny kľúč knižky).

3. normálová forma

- stĺpce, ktoré nie sú priamo závislé na primárnom kľúči sa musia odstrániť a presunúť do samostatnej tabuľky

Táto forma sa podobá na 2. NF avšak má určitý logický odtienok. Môžeme si ju vysvetliť na príklade, že pri knihách uvádzame aj vydavateľstvo. Nastáva tu troška iná situácia ako u autoroch. Každé vydavateľstvo mohlo vydať viacero kníh ale každá kniha má práve jedno vydavateľstvo. To znamená, že v tabuľke s knižkami sa môžu názvy vydavateľstiev opakovať. Preto vytvoríme novú tabuľku pre vydavateľstvá a do pôvodnej budeme ukladať len primárny kľúč vydavateľstva. Pričom vidíme, že tabuľky týkajúce sa autorov a ich vzťahu s knižkami to nemalo žiaden efekt. Je to preto, že samotné množiny (knižiek a autorov) a (knižiek a vydavateľstiev) sú rozdielne a majú rozdielne prieniky.

Použitím NF sme teda získali z jednej tabuľky 4:

- knižky
- autori
- vztah_knizka_autor
- vydavateľstvá

Vzťahy

Pri používaní NF vznikajú tabuľky s rozličnými vzťahmi. Poznáme 3 typy vzťahov:

- 1:1
- 1:n
- m:n

1:1

Tento vzťah je veľmi neobvyklý, pretože ku každému záznamu existuje práve jeden ďalší záznam. Čo je na prvý pohľad nelogické, keďže tento vzťah sa dá uložiť v jednej tabuľke a nie je potrebné rozdeľovať ho do 2. Sú však situácie, keď sa takéto rozdelenie hodí. Ak máme napr. tabuľku zamestnancov, v ktorej uchovávame okrem ich identifikačného čísla, mena, priezviska aj iné osobné údaje je vhodné

z bezpečnostných a optimalizačných dôvodov túto tabuľku rozdeliť na 2. V jednej budú len základné, často požadované, údaje (*ID, Meno, Priezvisko*) a v druhej budú všetky extra (menej dôležité údaje).

1:n

Príklad pre takýto vzťah sú napríklad naše tabuľky *knižky* a *vydavateľstvá*. Ku každému vydavateľstvu (1) existuje viacero alebo žiadna knižka (n).

m:n

Ako príklad nám zase poslúžia naše tabuľky: *knižky* a *autori*. Jedna knižka mohla byť napísaná viacerými autormi a jeden autor mohol napísať viacero knižiek. Tento vzťah teda reprezentuje tabuľka *vztah_knizka_autor*.

OK, 5 kreditov

25. Význam vstupných formulárov, vytvorenie a použitie. Filtrovanie, spôsoby filtrovania.

Formulár

- umožňuje užívateľovi iný pohľad na záznamy databázy
- umožňuje prehľadnejšie pracovať s tabuľkou, ktorá môže byť značne rozsiahla a tým aj neprehľadná
- je nadstavbou tabuľky, ktorá zobrazuje dáta z tabuľky (alebo viacerých tabuliek) v inej, prehľadnejšej podobe
- najrýchlejšie a najjednoduchšie získame nový formulár vytvorením automatického formulára

Automatický formulár si užívateľ môže vybrať:

- stĺpcový – polia sú usporiadané pod sebou
- tabulátorový – polia sú usporiadané vedľa seba a je súčasne zobrazených niekoľko záznamov pod sebou
- dátový list – obsahuje zobrazenie dátového listu

Tvorenie dokonalých **formulárov**
Vytvoríme formuláre pomocou sprievodcov a následne upravíme v návrhovom zobrazení. Tento postup umožní veľmi ľahko a rýchlo vytvoriť väčšinu bežných formulárov

Filtrovanie, spôsoby filtrovania

- Záznamy - filter
- Ikony
- Pravé tlačidlo
    1. Podľa výberu (resp. Opačne Mimo výber)
    2. Podľa formulára
    3. Rozšírené filtrovanie - logické AND, OR (kritéria píšeme vo filtri vedľa seba, resp. pod sebou)

Pri filtrovaní môžeme používať 3 rôzne zástupné symboly: #, ?, *; takisto môžeme používať veľký počet zástupných výrazov, napríklad: (doplnit od p.Kv.)

Zastupne znaky:
between #1/15/1955# and #12/31/1955#
In("Praha","Brno","Bratislava")
Like "*hradec*"
> date() -30 nie su starsie ako 30 dni
like "N*" slova zacinajuce na N

Zásadný rozdiel medzi použitím filtrovania a dotazov je:
Filtrovanie – na okamžité hľadanie podľa špecifického kritéria, dotaz – na časté a opakované použitie vyhľadávania.

Velmi strucne, 4 kredity

26. Dotaz, popis a význam, spôsoby vytvorenia, použitie dotazov. Výstupné zostavy, náhľad, tlač.
Medzi najčastejšie požiadavky pri práci s databázou je vyhľadanie dát, ktoré spĺňajú určitú podmienku, alebo je potrebné vyhľadať súvislosť medzi dátami.

Dotaz je žiadosť o načítanie, vytvorenie, úpravu alebo odstránenie dát v databáze. Dotazy je možné vytvárať v návrhovom zobrazení alebo pomocou sprievodcu.

Dotazy umožňujú
- voliť polia, ktorých sa bude dotaz týkať
- zadať kritéria, ktoré majú záznamy spĺňať
- triediť zoznamy

Existuje niekoľko typov dotazov na rôzne účely. **Výberový dotaz** napríklad zobrazuje údaje. **Akčný dotaz** mení údaje vo svojom zdroji údajov alebo vytvára novú tabuľku. **Parametrický dotaz** vyzýva pri svojom spustení na zadanie kritérií.
Výberový dotaz sa používa na vytvorenie podmnožín údajov, ktoré možno použiť pre odpovede na konkrétne otázky. Tento dotaz možno použiť aj na poskytnutie údajov pre iné databázové objekty. Po vytvorení výberového dotazu možno daný dotaz použiť kedykoľvek je to potrebné.
Výberový dotaz je typ databázového objektu, ktorý zobrazuje informácie v údajovom zobrazení (údajové zobrazenie: okno, ktoré zobrazuje údaje z tabuľky, formulára, dotazu, zobrazenia alebo uloženej procedúry vo formáte riadkov a stĺpcov. V údajovom zobrazení je možné upraviť polia, pridať a odstrániť údaje a vyhľadávať údaje.). Dotaz môže získať údaje z jednej alebo viacerých tabuliek, z existujúcich dotazov alebo kombináciou obidvoch možností. Tabuľky alebo dotazy, z ktorých dotaz získava údaje, sa nazývajú zdroj záznamu.
Postup na vytvorenie výberových dotazov je rovnaký bez ohľadu na to, či vytvárame tieto dotazy pomocou sprievodcu alebo v návrhovom zobrazení. Vyberieme si zdroj záznamu, ktorý chceme použiť a polia, ktoré chceme zahrnúť do dotazu . Voliteľne môžete zadať kritériá na spresnenie výsledkov.

Po vytvorení výberového dotazu je potrebné daný dotaz spustiť a zobraziť výsledky. Spustenie dotazu je jednoduché – jednoducho otvoríme údajové zobrazenie. Daný dotaz môžeme potom v prípade potreby znova použiť, napríklad ako zdroj záznamu pre formulár, zostavu alebo iný dotaz.
Treba tu povedať, že kritériá umiestnené v jednom riadku pre viacero polí sa vyhodnocujú logickým AND a kritéria vo viacerých riadkoch sú vyhodnotené logickým OR.
Treba tiež vedieť aspoň základné nástroje pre tvorbu kritérií:
Zástupne znaky sú *, ?, #
Pravidlá pre tvorbu kritérií môžu byť nasledovné :
between #1/15/1955# and #12/31/1955#
In("Praha","Brno","Bratislava")
Like "*hradec*"
> date() -30 nie su starsie ako 30 dni
like "N*" slova zacinajuce na N

Zostava je najvhodnejším nástrojom pre výstup v tlačenej forme. Tlačiť je možné formuláre i tabuľky. Zostavu je možné vytvoriť na základe tabuľky alebo dotazu. Pre zostavu je potrebné zadať objekt a polia, ktoré sa budú tlačiť . Zostava môže mať charakter stĺpcov, riadkov, formulára. Vytvorenú zostavu treba doladiť v návrhovom zobrazení. Najčastejším problémom je stanovenie nesprávnej šírky riadku pre tlač.
OK, doplnil som potrebné, 4 kredity

27. Typy softvéru. Aplikačné programy (textové a grafické editory, programovacie jazyky, výukové programy, multimediálne programy, počítačové hry, komunikačné programy, antivírové programy, diagnostické a špeciálne programy) .

Softwer sa rozdeluje na tri základné typy:

- **Systémový software**: Slúži na beh systému a hardwaru, je to napríklad operačný systém, ovládače zariadení (tlačiarne, grafické karty, modemy..), servre a rôzne nastroje.
- **Programovací software**: Slúži na vývoj softwaru, zahŕňa rôzne nástroje ako je textový deitor, kompilér, debugger, linker a podobne. Existuje vela typov týchto programov v závislosti na danom programovacom jazyku.
- **Aplikačný software**: slúži pre užívateľov na vykonávanie určitých činností, od práce, vzdelávania až po zábavu

Aplikačný software

Textový editor

Slúží na vytváranie a editáciu textových dokumentov. Môžu to byť jednoduché programy ako je napríklad notepad. V ňom nemáme žiadne pokročilé funkcie, vytvorit sa dá len čisto textový dokument s jednym typom písma. Ovela viac funkcii ponúka napríklad Microsoft Word, tam sa dajú nastavovať rôzne typy písma, zarovnávat, používať odrážky, vkladať grafy a obrázaky, vytvárať tabulky(pokročilé funkcie ponúkaju tabulkové editory (Microsoft Excel)) a množstvo iných funkcii (alternatíva: Openoffice Writer). Textové editory obsahujú aj vývojové prostredia, tie umožnujú formátovať a zvýrazňovať zdrojový kód a ponúkajú rôzne nadštandardné funkcie pre programátorov. Napríklad: Pspad, Delphi C++Builder a podobne.

Grafický editor

Sluzi na tvorbu a úpravu grafických súborov. Rozdelujú sa na rastrové a vektorové. Obsahujú sadu nástrojov a funkcii na úpravu grafiky (pero, štetec, polygony, výber farby, zmena velkosti ...)

- Rastrové: Vytváraju a upravujú súbory v ktorých sú informácie o jednotlivých pixeloch v obrázku. Obrázok je rozdelený akoby podla mriežky. Príklady: Adobe Photoshop, Corel Photopaint, Paint.net, Gimp
- Vektorové: Obrázok je vo forme objektov, ako sú krivky, obdlžníky a podobne a každý objekt ma určité vlastnosti, napr. farba, velkosť, poloha, tieň. Adobe Ilustrator, Corel Draw
- 3D grafika: Sú to objekty a svetlá v priestore s rôznymi vlastnosťami. Programy: 3D Studio Max, Cinema 4D

Programovacie jazyky

Programovacie jazyky slúžia na vývoj spustitelných programov alebo scriptov.
Jazyk symbolických adries – ľudovo nazývaný assembler. Symbolické pomenovanie inštrukcií
Java - Moderný multiplatformový jazyk. Výhodou je jeho prenositelnosť na rôzne systémy, často využívaný na internete.
Delphi – Vizualny objektovo orientovaný Pascal. Vhodný pre začiatočníkov, hodi sa avšak aj na vývoj komerčných aplikácií.
PHP - Hypertext Preprocessor, Servre generujú stránky s php skriptov.
Pascal – zastaralý jazyk, ktorý pôvodne vznikol na výuku. Dnes už bežne nepoužívaný.
C# - Moderný vizuálny jazyk. Vyvinutý Microsoftom z C++ a Javy.

Výukové programy

Slúžia na vzdelávanie. Väčšinou pre deti. Môožu to byť rôzne encyklopédie, interaktívne kvízy, jednoduché hry na precvičenie určitých vedomosti a podobne, často postavené na nejakom príbehu.
Bežné su napríklad programy na vyučovanie cudzích jazykov.

Multimediálne programy

Tieto programy slúžia na prehrávanie multimediálneho obsahu. Patria medzi ne prehrávače hudby, videa, obrázkov. Obsahujú klasicvke funkcie ako prehrať, zastaviť, posunúť, zmena hlasitosti a podobne. Je možné v nich výtvárať zoznamy skladieb a videí. (Winamp, iTunes)
Taktiž existujú programy na tvorbu a úpravu multimedialneho obsahu. Strihanie videa, záznam zvuku a podobne. (MovieMaker)
K multimediálnym programom by sa dali zaradiť aj internetové prehliadače (Opera, Firefox).

Počítačové hry

Slúžia na odreágovanie a zábavu. Problém však je, že sú zabijákom času a môžu byť návykové.
Rozdelujú sa do rôznych kategórií, napr RPG, 1st person, stratégie, adventúry, simolatory, atd. Na hru niektorých hier sa používajú špecialne, vstupno výstupné zariadenia, ktore umocnujú pocit skutočnosti a

zvyšuju zážitok s hry (napr. volant, joystick posobne). Môžu sa hrať v single playerovom móde, kde hrač hrá sám, alebo multiplayer kde hrá viac hráčov, napríklad na jednom pc, alebo na viacerých pc cez internet, alebo lan sieť (Counter-strike, Need For Speed). Jednou s podkategórii hier su online hry – väčšinou sú vytvorene vo Flashi a užívateľ si ich nemusí inštalovať.

Komunikačné programy

Ide o programy používané na komunikáciu cez internet. Najčastejsšie sa jedná o textovú komunikáciu, ale môžu prenášať aj zvuk a obraz. Sú to IM (instant messging) klienti, e-mail klienti, Voip klienti a taktiež je možné používať internetové aplikacie slúžiace na komunikáciu cez prehliadače. (ICQ, MSN messenger, Skype, Thunderbird, Opera)

Antivírove programy

Chránia počítač pred vírusmi a nebezpečným softwarom. (nod32) Slúžia na detekciu a odstranovanie škodlivích vecí. Bežia rezidentne na pozadí hneď od štartu systému. Každý počítač pripojený k internetu by mal mať antivírovy program. Medzi tieto programi patrí tiež firewall – slúži na ochranu pred útokmi z internetu.(Kerio Firewall)

Diagnostické a špecialne programy

Slúžia na správu a diagnostiku systému, jedná sa o utility.
Scandisk – Zistuje poškodené súbory na disku a snaží sa ich opraviť.
Defragmentacia – preusporiadáva a zoskupuje časti súborov na disku.
RegEdit – slúži na správu systémových registrov.
Rôzne testy hardwaru a benchmarky.
Sledovanie sieťovej prevádzky.
Sledovanie procesov
S tymito programami sa bežný užívateľ dostáva do kontaktu pomerne zriedka.
OK, 5 kreditov

28. Scandisk, defragmentácia, Windows commander, Prieskumník, Plánovač úloh. Komprimačné programy, ich význam a použitie, porovnanie komprimačných programov, samorozbaľovací archív, viacmédiový archív. Zálohovanie.

Scandisk – utilita (nastroj) ktorý hľadá a opravuje súbory systémov a chybné „clustre“

poznáme viac tipov volania utility (nastroja) scandisk :

1. SD po zlom vypnutí počítača (v MS-DOS)
2. SD volaný užívateľom (vo Windows)

Defragmentácia - proces, pri ktorom sa znižuje alebo úplne odstraňuje *fragmentácia* súborového systému z počítačových diskových jednotiek alebo z jednotlivých diskových oddielov.

Deje sa fyzickým reorganizovaním častí súboru za sebou, čím sa zvýši rýchlosť spracovávania vďaka kontinuálnemu prístupu.

Fragmentácia môže byť: (rozlámanie, rozdelenie na úlomky/časti, zapisovanie časti súvisiacich dát do voľných miest na dátovom nosiči)

Realizuje sa prostredníctvom na to určeného počítačového programu, napríklad Defrag. Defragmentáciou je možné výrazne urýchliť prístup k disku a všetky diskové operácie, ako čítanie, zápis a prístupovú dobu.

Na zaistenie výkonnosti diskových operácií je potrebné defragmentáciu vykonávať pravidelne, predovšetkým po výraznom zásahu do obsahu disku, napríklad pri hromadnom mazaní a kopírovaní veľkého množstva dát.

Windows commander vs. Prieskumník (súčasť MS Windows)

- Program Windows Commander (program tretej strany, ktorý nie je voľne šíriteľný) je v podstate to isté ako program prieskumník.
- Windows Commander je dokonalejší, viac prehľadnejší a viac prepracovanejší ako samotný prieskumník od Windows.
- V oboch programoch je možné : kopírovať, prepisovať, hľadať a aj spúšťať rôzne programy.

Plánovač úloh - je to utilita určená na naplánovanie nejakých úloh, napr. sa tam dá naplánovať, že chceme aby sa spúšťal scandisk hneď po štarte počítača

- úlohy sa dajú naplánovať aj na niekoľko dní dopredu

Komprimačné programy – umožňujú iné programy resp. súbory „zhustiť“, to je komprimovať tak, že potom ich veľkosť je menšia, teda zaberajú menej siesta na disku. Opacný postup dekomprimacia – uvedie tieto programy resp. súbory do pôvodného stavu .

Jedným z prvých komprimačných programov bol ARJ (najčastejšie v MS-DOS) neskôr prišiel vylepšenejší komprimačný program WinZip a WinRar.

Jedným z najlepších komprimacných programov je WinAce lebo používa lepší komprimačný pomer, číže súbory zaberajú ešte menej ako z vyššie uvedených WinZip a WinRar.

Ešte poznáme aj program UHARC do ktorého sa vkladajú parametre iba cez príkazový riadok ale jeho hlavnou výhodou je to, že dokáze do archívu pridávať aj mp3 a zmenšiť ho cca. o 50%

Samorozbaľovací archív – je to skomprimovaný súbor s príponou .ehe, ktorý ma, okrem samotného ‘zabaleného‘ súboru, aj potrebnú informáciu na to aby sa aj bez použitia programu slúžiaceho na kompresiu/dekompresiu dokázal niekde rozbaliť

Obyčajné sa udáva len umiestnenie kde sa súbor ma rozbaliť

Viacmédiový archív - keď potrebujeme nejaký “zipnutý“ súbor o veľkosti 6MB preniesť na disketách, tak vieme udať podmienku aby ten súbor rozdelil na **n** rovnakých alebo rôznych s maximálnou veľkosťou nepresahujúcou 1.44MB + jeden súbor s príponou .bat, určený na spojenie “zlúčenie“ tých **n** súborov do jedného pôvodného soboru o veľkosti 6MB.

Zálohovanie dát (archivácia dát) je vytváranie záložnej kópie (kópií) dát. Môže byť vykonávané manuálne alebo automaticky. Automatické zálohovanie môže byť priebežné alebo sa vykonáva v určitých časových intervaloch. Niektoré operačné systémy počas zálohovania umožňujú súbežne vykonávať zmeny v zálohovaných dátach, iné umožňujú ich v tom čase prezerať, ale nemožno ich meniť a niektoré neumožňujú ani to.

{Medzi archiváciou a zálohovaním je menší rozdiel. Archívna kópia slúži na spätné zistenie, v akom stave boli dáta v určitom čase a podľa vnútorných predpisov organizácie, alebo zákonných nariadení sa musí archívna kópia archivovať predpísanú dobu. Záložná kópia slúži na obnovenie "živých" dát, ak došlo k ich poškodeniu alebo zničeniu. Priebežné zálohovanie asi nemožno nazývať archiváciou.}

Asi najviac chýbajú informácie v Zálohovaní, tu treba spomenúť zálohovanie systému a dát . Pri dátach – úplné, rozdielové

4 kredity

29. Pojem počítačový vírusu. Typy a prejavy počítačových vírusov. Spôsoby šírenia a spôsoby infiltrácií, ochrana proti vírusom. Práca s antivírovým programom

Počítačový vírus je program ktorý sa dokáže sám šíriť. To znamená, že za počítačový vírus sa považuje i taký program, ktorý nevykonáva žiadnu nebezpečnú akciu.

Základné typy vírusov podľa spôsobu šírenia:

- klasický vírus
- počítačový červ
- trójsky kôň

Klasický počítačový vírus je program, ktorý dokáže rozmnožovať sám seba pridávaním svojho kódu do iných programov. Pre svoje rozširovanie teda podobne ako biologický vírus potrebuje hostiteľa – iný program. Z toho vyplýva, že do počítača sa môže dostať jedine tak, že spustíme nainfikovaný program. Spolu so spustením nainfikovaného programu sa aktivuje vírus v operačnej pamäti, a potom napadne i ďalšie súbory v počítači. Bez vedomia používateľa modifikuje iné programy, dokumenty alebo systémové oblasti pevného disku. Niektoré vírusy slúžia na odchytávanie vzácnych údajov ako napr. Heslá, e-mailové adresy, loginy apod.
Špeciálnym druhom vírusov sú tzv. **Boot vírusy**, ktoré miesto programu napádajú miesto na nosiči dát, z ktorého sa dá zaviesť operačný systém (program, bez ktorého by počítač nefungoval napr. Windows). Do počítača sa môže dostať tak, že zabudneme disketu v mechanike pri zapínaní počítača. Našťastie tieto vírusy už nepredstavujú hrozbu pretože nefungujú v operačnom systéme Windows (dokážu fungovať iba v MS DOS).
Ďalším špeciálnym druhom vírusov sú **Makro vírusy**. Tieto sa dokážu síriť vďaka tomu, že Microsoft do svojho kancelárskeho balíka Office (Word, Excel, PowerPoint, Outlook...) pridal možnosť vytvárať „makrá“ – programy, ktoré mali pôvodne slúžiť na automatizáciu často vykonávaných krokov. Otvorením dokumentu, ktorý obsahuje makro vírus, sa telo vírusu najčastejšie skopíruje do šablóny „normal“, z ktorej sa vytvárajú všetky nové dokumenty. Každý nový dokument, ktorý potom vytvoríte, je napadnutý makro vírusom. Našťastie novšie balíky Office Vás na prítomnosť makra upozorňujú a ich dôveryhodnosť sa dá zabezpečiť digitálnym podpisom, pomocou ktorého sa dá zistiť, kto makro vytvoril.

Počítačovým červom na rozšírenie stačí niekoľko hodín. Kým vírus potrebuje našu pomoc, aby sa dostal z jedného počítača na druhý pomocou diskety, CD alebo iného nosiča, počítačový červ sa dokáže rozšíriť i sám pomocou počítačovej siete. Funguje tak, že sa skúša pripojiť na každý možný počítač v počítačovej sieti a na svoj prenos využije slabé miesto zle zabezpečeného počítača. Na tomto počítači sa červ aktivuje a znovu sa skúša šíriť do ďalších počítačov. Počet nakazených počítačov teda stúpa lavínovite. Niekedy si červy zo sebou nesú i zoznam zle zabezpečených počítačov, aby sa mohli čo najrýchlejšie rozšíriť. V prípade napadnutia jedného počítača červ napadne ďalšie dva a na oba pošle polovičný zoznam napadnuteľných počítačov. Takýmto spôsobom sa zabezpečí vyššia efektívnosť a teda i rýchlejšie rozšírenie.

Trójsky kôň je program, ktorý navodzuje dojem užitočnosti, ale svoju činnosť nemusí vykonávať a na pozadí vykonáva deštrukčnú činnosť (mazanie súborov, formát disku, narušuje súkromie užívateľa skrytou komunikáciou). Navonok sa od pôvodného programu okrem veľkosti neodlišuje, vzhľadom zapadá do bežného užívateľského prostredia

Ďalšími typmi vírusov sú tie, ktoré sa šíria technológiou **active X**- typy plug-inov v prehliadači Internet Exploler. Microsoft sa voči nim bráni zavedením digitálneho podpisu.

Tiež **JAVA applet**y sú vírusy vytvorené jazykom java vložené na stránku. Sú ním ohrozované všetky operačné systémy a dá sa voči nim brániť inštaláciou tzv. Java Runtime Enviromentom.
Droppery sú vírusy ktoré v pravidelných intervaloch do systému vypúšťajú rôzny malware alebo vírusy, červy či **spyware** (špionážny software).
Takto vypustený červ napáda sieť s vnútra, pričom je náročné odhaliť zdroj. Odhalenie trójskeho koňa sťažuje i technika nazývaná **rootkits** (voľne preložené ako nástroje správcu). Touto technikou trójsky kôň dokáže poprieť svoju existenciu v systéme. Túto techniku môže trójsky kôň najľahšie využiť v prípade, že ho otvoríme s oprávneniami správcu systému.
Ďalšou nebezpečnou akciou, ktorú môžu trójske kone vykonávať, je otvorenie zadných vrátok **backdoor**. Cez tieto „zadné vrátka" sa vie dostať potom dostať počítačový červ do systému.

Detekovanie vírusu
Zistiť či je v počítači vírus nie je jednoduché. V prípade, že vírus začne vykonávať svoju činnosť okamžite ako sa ním počítač infikuje, prejaví sa to nezvyčajným správaním počítačového systému. Niektoré vírusy nazývané **Bomby** sú však naprogramované tak, že svoju deštrukčnú činnosť vykonajú v určitý deň.
Detekcia vírusov nie je jednoduchá i vďaka metóde **Stealth**, ktorú vírusy používajú. Táto metóda spočíva v tom, že pri antivírovej kontrole vírus dočasne odstráni svoju kópiu z infikovaných súborov a po jej skončení opäť súbory napadne. Pri antivírovej kontrole preto musíme vždy zabezpečiť aby vírus ešte nebol aktivovaný v operačnej pamäti. To docielime zavedením systému z média (diskety, CD ...) o ktorom vieme, že nie je napadnuté. Takéto medium je najlepšie vytvoriť ihneď po inštalácií operačného systému.
Ďalšie vírusy sa bránia proti odhaleniu tak, že každá vytvorená kópia vírusu sa odlišuje od svojho predchodcu. Ide o tzv. **Polymorfné vírusy**. Pri liečení takto zmutovaného vírusu hrozí, že antivírový systém nesprávne určí verziu vírusu a systém po liečení nebude fungovať.

Počítačové systémy sú zraniteľné najmä kvôli týmto dôvodom:

- **Homogenita systémov** - Väčšina počítačov na svete je vybavená rovnakým operačným systémom, rovnakým prehliadačom Internetu a poštovým klientom. Toto umožňuje vírom rýchle šírenie.
- **Chybovosť** - programy obsahujú chyby, ktoré vírusy dokážu zneužiť.
- **Nepotvrdený kód** - keď vkladáte CD, USB disk alebo iné médiá, po vložení sa hneď aktivuje program autorun.exe, ktorý môže byť napadnutý vírusom.
- P**oužívateľ s nadmerným oprávnením** - Vírusy získajú také oprávnenie ako má ten kto ich aktivuje. Niektoré systémy dokonca umožňujú meniť všetkým používateľom všetky súčasti systému (napr. Windows 95, 98, ME).

Prevencia

- Používajte menej rozšírený operačný systém, prehliadač a poštového klienta. Väčšina vírusov pracuje pod operačným systémom MS Windows, prehliadačom Internet Explorer a poštovým klientom Outlook.
- Zabezpečte svoj počítač proti neoprávnenému vniknutiu. Tento krok môžete urobiť použitím tzv. Firewallu (v preklade ohnivá stena), ktorý vytvára ochrannú hrádzu medzi vaším počítačom a potenciálne škodlivým obsahom na Internete. Je to veľmi účinná zbraň proti počítačovým červom. Môže byť buď hardvérový (zariadenie) alebo softvérový (program). Takýto softvérový firewall je i súčasťou Windows XP, ak ho aktualizujete servisným balíkom číslo 2 (alebo novším). Firewall je tiež súčasťou niektorých antivírových systémov. Staršie systémy Windows (95,98, a ME) sú na pripojenie do siete internet absolútne nevhodné, pretože nemajú dostatočnú ochranu proti napadnutiu.
- Nenavštevujte nebezpečné stránky a nesťahujte programy na sťahovanie hudby, filmov a

programov. Snažte sa vyhnúť stránkam s pornografiou, stránkam s mp3 hudbou, filmami, licenčnými kľúčmi a podobne.

- Pred stiahnutím každého programu si pozorne prečítajte licenčnú zmluvu. Pozorne si prečítajte podmienky používania programu a vyhnite sa všetkým programom, ktoré podmieňujú svoju inštaláciu nainštalovaním komponentov od tretích strán a tiež tým, ktoré sa v zmluve zbavujú zodpovednosti, preto je dôležité pozorne sledovať v priebehu inštalácie dialógové okno.
- Nezverejňujte svoju emailovú adresu. Vaša e-mailová adresa je Váš osobný údaj. Veľmi dobre si rozmyslite, komu (do akého formulára ju vyplníte). Používajte radšej niekoľko adries (jednu pre priateľov a druhú na vyplňovanie do formulárov).
- Neotvárajte neznáme prílohy. Ak neotvoríte spustiteľnú prílohu e-mailu, nemôžete dostať e-mailový vírus. Preto radšej všetky cudzojazyčné emaily vymazávajte. Vírus však môžete dostať i od rodiny či priateľov bez toho, že by vedeli, že vám taký e-mail poslali. Pred otvorením prílohy si preto overte či vám ju dotyčný aj chcel poslať.
- Nespúšťajte neoverené makrá v dokumentoch. V súčasnosti sa i makrá podpisujú digitálnym podpisom. Preto si pred jeho spustením overte platnosť digitálneho podpisu a neaktivujte neznáme makro.
- Udržujte všetky súčasti systému aktuálne, používajte najnovšiu verziu prehliadača a poštového klienta. Aktualizovaním súčastí systému odstraňujete jeho nedostatky, ktoré by škodlivé programy mohli využiť.
- Chráňte svoj počítač aktuálnym antivírovým systémom. Antivírový systém Vás ochráni pred väčšinou starších hrozieb a niektorými z najnovších. Aby bola ochrana účinná, antivírové systémy sa aktualizujú i niekoľkokrát za deňChráňte svoj počítač proti špionážnym programom. Na ochranu proti takýmto programom sú najvhodnejšie programy Windows Defender priamo od spoločnosti Microsoft. Tieto programy sa aktualizujú rovnako ako antivírové systémy, preto ich treba udržiavať vždy aktuálne.
- Znížte svoje oprávnenie. Nikdy nepracujte s internetom a poštou s oprávneniami správcu počítača. Oprávnenie správcu počítača umožňuje trójskym koňom dokonalé maskovanie.
- Používajte bezpečné heslá a nikdy sa nezabudnite odhlásiť po skončení činnosti. Heslo by malo mať najmenej 6 znakov a malo by pozostávať z veľkých i malých písmen, číslic a ďalších symbolov a nemalo by byť slovom žiadneho jazyka. Heslo je tiež potrebné meniť.
- Zvážte, ktoré priečinky budete zdieľať v sieti. Nastavte na všetky zdieľané priečinky príslušné oprávnenia a chráňte k nim prístup heslom. V prípade, že použijete sprievodcu na vytvorenie zdieľania priečinkov v sieti, ten Vám sprístupní iba zjednodušené zdieľanie.
- Zálohujte všetky svoje údaje na disky chránené proti zápisu. Zálohovaním dát sa vyhnete i strate konzistencie dát následkom výpadku prúdu alebo tvrdého reštartu.

Termínom **počítačová kriminalita** sa označujú trestné činy zamerané proti počítačom ako aj trestné činy páchané pomocou počítača. Ide o nelegálne, nemorálne a neoprávnené konanie, ktoré zahŕňa zneužitie údajov získaných prostredníctvom výpočtovej techniky alebo ich zmenu. Počítače v podstate neumožňujú páchať nový typ trestnej činnosti, iba poskytujú novú technológiu a nové spôsoby na páchanie už známych trestných činov ako je sabotáž, krádež, zneužitie, neoprávnené užívanie cudzej veci, vydieranie alebo špionáž.

Najviac rozšírenou trestnou činnosťou je používanie, šírenie a vytváranie prostriedkov na odstránenie ochranných prvkov slúžiacich na chránenie autorských diel alebo používanie a šírenie takto upravených autorských diel, s ktorými je nakladané v rozpore s autorským právom. Takéto prostriedky sa v počítačovom slangu nazývajú **Warez.** Podľa obsahu ho môžeme deliť na

apps – aplikácie: vo všeobecnosti maloobchodné verzie softvérových balíkov
cracks – cracky: záplaty určené na pozmenenie skúšobnej verzie programov na plné alebo na obídenie protipirátskej ochrany
keys/key generators - kľúče: kľúče k inštalačným programom alebo ich generátory, ktoré umožňujú nelegálne nainštalovať program alebo odstránia obmedzenia skúšobnej verzie programu
games – počítačové hry: počítačové hry a hry pre hracie konzoly
movies – filmy: pirátske filmy

music – MP3-ky: pirátske albumy, single alebo iné hudobné formáty distribuované vo forme hudobného formátu MP3
E-Books/Tutorials – knihy: do tejto kategórie spadajú pirátske knihy v elektronickej podobe, naskenované knihy a návody k programom.

Ďalšou rozšírenou trestnou činnosťou je **neoprávnené vniknutie do systému**. Človek zaoberajúci sa touto činnosťou sa v počítačovom slangu nazýva **hacker**.

Bankové krádeže uskutočnené pomocou počítača sú zatiaľ u nás zriedkavé no vo svete sa začínajú čoraz viac vyskytovať. Známe sú nasledujúce tri typy krádeží:

- **Phishing**: Tieto správy sú väčšinou nevinným žartom, ale existujú i také, ktoré vyzývajú na zmenu kódu k Vášmu účtu v banke. V takomto emaile je umiestnený odkaz , na ktorom si heslo máte zmeniť. Odkaz však nesmeruje na stránku banky, ale na jej dokonalú napodobeninu
- Najzákernejší spôsob, ktorým Vás hacker môže pripraviť o vaše úspory, je **Pharming** (farmárčenie). Táto metóda spočíva v presmerovaní názvu www stránky na inú adresu. Každej mennej adrese napríklad ib.vub.sk prislúcha číselná adresa napríklad 215.5.214.144. Pomerne jednoduchým spôsobom sa dá toto nastavenie zmeniť. Ak zadáte mennú adresu do Vášho prehliadača, miesto stránky banky sa zobrazí jej dokonalá napodobenina. Vy teda ani nezbadáte, že ste na inej stránke. Po zadaní údajov, ich získa neoprávnená osoba, ktorá takúto falošnú stránku vytvorila. Proti takejto hrozbe sa môžete brániť rôznym spôsobom. Najjednoduchším spôsob je zistiť si číselný kód stránky internetbankingu. Ďalšou možnosťou je overovanie platnosti certifikátu a upozornenie pri prechode zo zabezpečenej stránky na nezabezpečenú. Tieto funkcie sa dajú nastaviť v internetovom prehliadači.
- **Spoofing**: Do tejto kategórie patria všetky metódy, ktoré používajú hackeri na zmenu totožnosti odosielaných správ. Jednou z týchto metód je i náhrada emailovej adresy pri phishingu, ktorá zabezpečí, aby správa vyzerala tak, že ju odoslala banka. Ďalšou metódou je podvrh IP adresy na stránky, ktoré takýmto spôsobom overujú totožnosť prihlasujúceho. Najviac nebezpečnou je však metóda nazývaná MITM (man-in-the-middle v preklade "muž v strede"). Táto metóda spočíva v narušení komunikácie medzi klientom a bankou, pri ktorej útočník naruší šifrovací systém verejného a súkromného kľúča, ktorý sa používa pri komunikácii. Použiť metódu MITM však nie je jednoduché, pretože na narušenie komunikácie je potrebné získanie kľúča (niekedy tiež označovaný ako certifikát) banky, ktorý sa často mení. Je preto dôležité nastaviť Váš internetový prehliadač tak, aby overoval, či je certifikát ešte platný.

OK, môže byť, 5 kreditov
Na delení sa ťažko dohodnúť, čo zdroj, to iná klasifikácia, ale chýba tu niečo, čo na maturitách treba vedieť.
Dnes už nie sú hlavným nebezpečenstvom vírusy, ale iné infiltrácie (maware)
Ten je delený na - spyware, adware, trojske kone, vírusy a budeme sa pýtať, čo je čo.

30. Autorské práva tvorcov softvéru. Licencie, multilicencie, legálny softvér. Typy softvéru z hľadiska právnej ochrany (freeware, shareware, demoverzia, komerčný sw.), vysvetliť použitie. Softverové pirátstvo. Ochrana pred softvérovým pirátstvom.
Autorské práva:
Autora programu chráni **autorské zákon.** Je to preto, lebo každý program je podľa našich zákonov dielo a naň sa vzťahuje autorský zákon. Podľa tohto autorského zákona má autor právo udeľovať súhlas na každé použitie diela, najmä však na:

- vyhotovenie kópie diela
- verejné rozširovanie originálu diela alebo jeho kópie predajom alebo inou formou prevodu vlastníckeho práva
- verejné rozširovanie originálu diela alebo jeho kópie nájmom alebo vypožičaním
- spracovanie, preklad a adaptáciu diela
- zaradenie diela do súborného diela
- verejné vystavenie diela

- verejné vykonanie diela
- verejný prenos diela

Ak si kúpime nejaký softvér, jeho súčasťou je **licenčná zmluva,** ktorá zvyčajne stanovuje, že zakúpený softvér nemôžeme používať súčasne na viacerých počítačoch. V prípade, že chceme používať softvér na viacerých počítačoch(napr. počítačové učebne,...), musíme si zakúpiť väčší počet programov alebo **multilicenciu**(licencia pre viac počítačov). Ďalšou skupinou licencií sú **školské alebo študentské licencie.**

LEGÁLNY SOFTWARE

- predáva sa v originálnych neporušených obaloch s ochrannými prvkami (napr. hologramy)
- k inštalačnému médiu je pripojená licenčná zmluva, certifikát autenticity a riadny daňový doklad obsahujúci presný popis softvéru

Rozdelenie software podľa spôsobu a stupňa ochrany autorských práv:

- **Shareware**: Tento software dal autor k dispozícii voľným šírením. Je možné ho získať nakopírovaním z ľubovoľného zdroja. V skúšobnej dobe je možné ho používať bezplatne, ale po skončení skúšobnej doby sa nesmie program bez zaregistrovania používať. Podmienkou legálneho používania a získania licencie na používanie je registrácia, pričom podmienky registrácie sa vypíšu po spustení programu.
- **Freeware:** Tento software je voľne šíriteľný a používateľný bez akýchkoľvek poplatkov a obmedzení s výnimkou zásahu do zdrojového kódu. Ide zvyčajne o reklamné programy, nenáročné hry a demoverzie pripravovaných programov.
- **Public domain:** Taktiež voľne šíriteľné - ako FREEWARE - ale aj s prístupom k zdrojovým kódom, ktoré sa môžu upravovať a ďalej šíriť. Je potrebné zachovať aj pôvodného autora.
- **Demo – verzia:** Je program, ktorý nás má nalákať na kúpu určitého software - umožňuje pracovať s programom ako by to bol ponúkaný komerčný software, ale s určitým obmedzením. Demo verzie sa distribuujú zvyčajne bezplatne a môžu sa voľne šíriť ďalej.
- **Beta – verzia**: Beta verziou sa nazýva rozpracovaná verzia software, ktorá sa poskytne externým spolupracovníkom, novinárom, významným zákazníkom a iným vybraným záujemcom na otestovanie a zhromaždenie pripomienok (keďže ide o nedokončený program, má zvyčajne mnoho chýb).

Softwarové pirátstvo:

Je neoprávnené zaobchádzanie s počítačovými programami, dokumentáciou a ďalšími súčasťami počítačových programov, ktoré sú chránené autorským právom. Na kopírovanie, rozširovanie a predaj počítačových programov má výhradné právo jedine autor, ktorý môže toto právo previesť na iné osoby jedine zmluvou.

Spôsoby ochrany pred nelegálnym kopírovaním:

(V zásade sa rozlišujú dva typy ochrany)

1. **Softwarová ochrana** spočíva v tom, že software sa ochráni prístupovým heslom alebo kódom. Vyššia forma tejto ochrany sú náhodne generované čísla, ku ktorým pomocou kľúča, známeho len výrobcovi daného softvéru, bude po zaplatení zaslaný kód, ktorým software „odomkneme".
2. **Hardwarová ochrana** je považovaná za účinnejšiu ako softwarová. K zakúpenému programu dostaneme aj hardvérový kľúč, ktorý spravidla zasunieme buď do paralelného portu počítača, alebo do portu USB. Tento kľúč „odomkne" program. Ak program skopírujeme a pokúsime sa ho inštalovať na inom počítači, nebude bez kľúča fungovať. Existujú dva druhy vyhotovení – Hardlock Hardkey

OK, 5 kreditov, verím, že otázka**, čo je OEM** nikoho nezaskočí, aj keď to tu nie je popísané

31. Rozbor problému, algoritmus, program, ladenie, logické chyby, chyby počas behu programu, softvérové inžinierstvo, životný cyklus programu.

Tvorbou programov sa zaoberá **softvérové inžinierstvo**. *Pozostáva zo štyroch etáp(rozbor problému, algoritmus, program a ladenie).*

Rozbor problému(RP): Programátori musia presne vedieť, čo má program urobiť, aké bude mať výstupy (čo má program vypísať alebo nakresliť) a vstupy (aké údaje musí načítať od používateľa, či je to celé číslo, desatinné číslo alebo text) a tiež je potrebné povedať, čo sa má stať, ak používateľ programu vstup zadá nesprávne, tento proces sa nazýva *rozbor problému.*

RP teda pozostáva z týchto častí:

- špecifikácia vstupných dát (štruktúra, formát, množina prípustných hodnôt)
- špecifikácia výstupných dát (štruktúra, formát, funkčná závislosť od vstupných dát)
- špecifikácia výnimočných situácií (napr. reakcia na chybné vstupné dáta)

Algoritmus: základný elementárny pojem informatiky, je prepis, návod, realizáciou, ktorého získame zo zadaných vstupných údajov požadované výsledky.

Je to presne definovaná, konečná postupnosť príkazov, ktorých vykonanie umožní po konečnom počte krokov získať pre prípustné vstupné údaje zodpovedajúce výstupné údaje.

Program: je algoritmus zrealizovaný pomocou, konkrétneho programovacieho jazyka. Program je založený na konkrétnej reprezentácii a štruktúre údajov. Programátor, zapíše program pomocou príkazov zo slovníka jazyka, a potom ho preloží do podoby, ktorej rozumie počítač.

Testovaním zisťujeme , či nie sú v danom programe logické chyby, testovanie robíme pomocou trasovacej tabuľky.

Ladenie:

Proces, počas, ktorého zistené chyby hľadáme a odstraňujeme sa nazýva **ladenie.** Okrem odstraňovania chýb však ladenie slúži aj na optimalizovanie kódu. To znamená, že sa pokúšame dosiahnuť, aby sa program vykonával rýchlejšie, aby spotreboval menej pamäte, alebo aby jeho chod bol plynulý. Vyladiť môžeme program pomocou:

- krokovaním – spúšťanie programu po riadkoch, príkazoch
- kontrolou obsahu premenných
- zastavením programu na zvolenom mieste

Logické chyby: zistené počas alebo po skončení behu programu:

- vedúce k predčasnému zastaveniu programu s chybovou správou (run time errors) napr.: delenie nulou, súbor na otvorenie nenájdený, výpočet väčšieho čísla, aké sme si zadefinovali a pod.
- vedúce k chybným výsledkom, zacykleniu programu (zastavenie vykonávania programu: Run - Program Reset) a pod.

Softvérové inžinierstvo: sa zaoberá špecifikovaním, návrhom, vývojom a údržbou softvéru s využitím poznatkov informatiky, projektového manažmentu a ďalších oblastí. Alebo inak softvérové inžinierstvo sa zaoberá teóriou, metódami a nástrojmi potrebnými pre vývoj softvéru.

Životný cyklus programu: sú etapy tvorby programu, jeho používanie a údržba.

- Formulácia problému: **5%**(Obecná formulácia problému)
- Analýza problému: **20%**(Podrobný popis budúceho systému .Jeho vstupov a výstupov. KRITICKÁ FÁZA)
- Hľadanie riešení: **25%**(Možné a čiastočné riešenie systémov.)
- Výber riešení: **20%**(Podrobný plán realizácií)
- Realizácia: **30%**(Testovanie, predaj, údržba)

32. Vymedzenie pojmu algoritmus, vlastnosti a zápisy algoritmov, príklady algoritmov, efektívnosť algoritmov – pojem časovej a pamäťovej zložitosti. Metóda overenia správnosti algoritmov (algoritmus čiastočne správny a konečný)

Algoritmus:

Jednoznačná postup krokov, pomocou ktorej môžeme vyriešiť zadaný problém. Každý algoritmus musí splňovať určité vlastnosti (viď nižšie)

Vlastnosti algoritmu:

1. Determinovanosť (jednoznačnosť) – v každom kroku jasné, čo sa má vykonať a ktorý krok nasleduje
2. Konečný (rezultatívnosť) – vždy vedie k výsledku
3. Obecný (hromadný) – použiteľný pre riešenie danej úlohy s použitím ľubovoľných prípustných dát
4. Opakovateľný – vedúci vždy k rovnakému výsledku, ak sú zadané rovnaké dáta

Vyjadrenie algoritmu:

1. slovne – v prirodzenom jazyku
2. graficky – vývojovým diagramom alebo kopenogramom (štrukturogramom)
3. matematicky – vzťahom medzi veličinami, sústavou rovníc, maticami
4. programovacím jazykom

Zápis algoritmu:

1. JAVYD – JAzyk VYvojových Diagramov (Používa značky podľa CSN)
2. algoritmický jazyk – slovenský Pascal

Prepísaním ľubovoľného zápisu algoritmu do ľubovoľného programovacieho jazyka dostaneme PROGRAM

Overenie algoritmu (verifikácia):

Testuje sa **každá** vetva, využíva sa trasovacia tabuľka

- trasovanie algoritmu

Algoritmus, ktorý vždy keď skončí, dá správne výsledky, nazývame **čiastočne správny.**
Algoritmus, ktorý skončí svoju realizáciu pre všetky údaje vyhovujúce vstupným podmienkam nazývame **konečný algoritmus**
Správny algoritmu je ten, ktorý je čiastočne správny a konečný.

Efektívnosť algoritmov je vyjadrená pomocou:
Časová výpočtová zložitosť O(n)

Závislosť vyjadrujúca počet operácií potrebných na realizáciu algoritmu ako funkcia závisiaca od vstupných údajov

Pamäťová výpočtová zložitosť P(n)

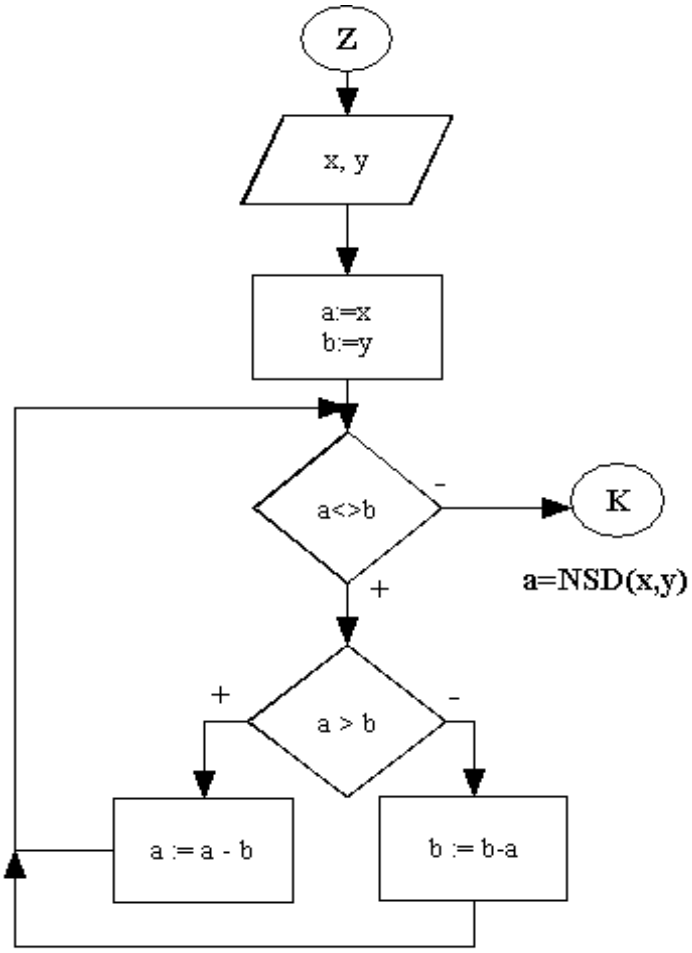
Závislosť vyjadrujúca počet údajov, ktoré si treba pamätať, ako funkcia závisiaca od vstupných údajov

Vývojový diagram
Algoritmus možno popísať aj slovne, ale prehľadnejší spôsob zadania algoritmu je bloková schéma alebo **vývojový diagram**.
Vývojový diagram je znázornenie algoritmu, postupu riešenia danej úlohy. Vývojový diagram môže obsahovať :

- príkaz na načítanie vstupov,
- príkaz na priradenie hodnôt premenným,
- rozhodovacie bloky so znakmi "+", "-" ("+" znamená, že podmienka je splnená a znamienko "-", že nie je splnená)

Overenie správnosti algoritmov – trasovanou tabuľkou
Overenie správnosti algoritmov možno realizovať tak, že do schémy algoritmu dosadíme konkrétne hodnoty a vykonáme jeden krok za druhým podľa vývojového diagramu. **Vývojový diagram je navrhnutý správne, keď pre každý vstup z definičného oboru dosiahneme požadovaný výstup.** Po overení správnosti vývojového diagramu možno pristúpiť k napísaniu programu v konkrétnom programovacom jazyku.

Úloha - Najväčší spoločný deliteľ čísiel x, y



Tento príklad vývojového diagramu rieší problém najväčšieho spoločného deliteľa čísiel x,y.

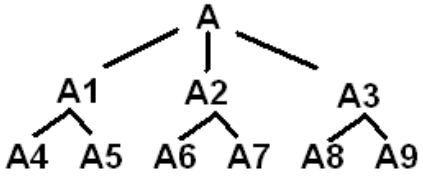
Na **vstupe** sú kladné prírodzené čísla **x,y** a na **výstupe a=NSD(x,y)** najväčší spoločný deliteľ čísiel x,y. Čísla a,b sú tzn pomocné premenné.

Test algoritmu : (Tu dosadíme konkrétne vstupy x=12, y=8 do našej schémy a vykonáme jeden krok za druhým. Tu môžeme dosadiť ľubovoľné iné hodnoty, kladné celé čísla, pre každý správny vstup x, y algoritmus nájde NSD(x,y))

33. Základné prvky štruktúrovanej tvorby algoritmu, metóda zhora nadol. Konjunktívny, disjunktívny, repetičný rozklad úlohy, zodpovedajúce algoritmické konštrukcie, dodržanie uvedených zásad vo zvolenom programovacom jazyku. Význam návrhu správnych štruktúr, procedúr a funkcií

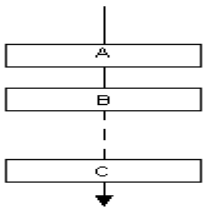
Zjednodušením stavby programu bolo zavedenie vývojových diagramov. Tie vychádzajú z predpokladu, že človek ľahšie vníma informáciu z obrazu, ako z textu. Preto bol vytvorený súbor dohodnutých tvarov a úsečiek, ktorými sa dá vyjadriť akákoľvek štruktúra v programe.
Nastáva však problém, a to pri tvorení stredne až veľmi rozsiahlych programov, a to, že vývojový diagram takéhoto programu by bol minimálne rovnako neprehľadný, ako samotný program. Nehovoriac o časovej náročnosti takéhoto programu.
Preto sa pristupuje k takzvanej štruktúrovanej tvorbe programov. Moderné programovacie jazyky (medzi ktoré patrí aj Pascal) majú takúto tvorbu programov danú implicitne – používame ju a ani si to neuvedomujeme, pretože jazyk nás sám núti písať program tak, ako to on „chce“. Pri programovaní používame určité zásady, ktoré sú základom tzv. štruktúrovaného programovania.
Dijkstrove zásady (princípy štruktúrovaného programovania):

1. Rozdelenie problému na podproblémy – tzv. vznik hierarchie podproblémov, ktoré majú medzi sebou isté súvislosti.
   Z podproblémov je nutné vedieť rekonštruovať návod na riešenie pôvodného problému
2. Riešenie problému od strechy. Rozbor problému (i+1) radu nezačneme skôr, kým nie je hotový i-ty rad (napr. nezačneme robiť podproblém A5 keď nemáme A3, viď obr)



V praxi je konštrukcia algoritmov nepriamočiara, preto pri ich tvorbe musíme postupovať tzv. metódou „zhora nadol“, t. j. celú úlohu postupne rozkladať na jednoduchšie zložky a tento postup opakovať tak dlho, kým to bude potrebné. V praxi to znamená, že najskôr určíme, aké budú vstupné a výstupné údaje a zostavíme hrubý vývojový diagram, podľa ktorého sa úloha ďalej rozkladá.
Pripomeňme si, že k tvorbe programu metódou zhora nadol a obecne štruktúrovaným programovaním nám taktiež slúžia podprogramy – procedúry a funkcie. Pomocou nich môžeme vytvoriť program príkladne rozložený na menšie problémy, kedy každý podprogram rieši určitú časť úlohy. Vďaka vnáraniu procedúr a funkcií do seba sa tak môže problém riešený podprogramom deliť na ešte menšie celky, takmer donekonečna. Výhoda procedúr a funkcií je aj v tom, že pri ich používaní nemusíme vedieť, ako pracujú, ale len to, aké vstupné parametre používajú. Podprogramy sú ideálne aj vtedy, ak program tvorí tím ľudí – každý člen tvorí jednu jeho časť.
Pri rozkladoch sa musíme snažiť používať len vhodné štruktúry, ktoré nazveme odporúčané štruktúry. Poznáme tri základné:

POSTUPNOSŤ

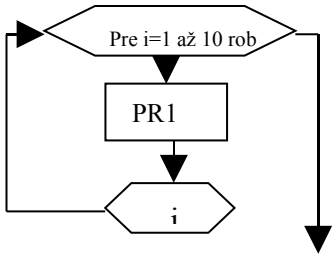


je určitý nešpecifikovaný počet príkazov
Je to základná štruktúra, od ktorej sa odvíjajú tie ostatné. Pod pojmom príkaz tu však môžeme rozumieť aj určitý súbor príkazov či použitie funkcie / procedúry.

VETVENIE

je miesto v programe, kde sa rozhoduje medzi viacerými možnosťami
V určitých miestach programu je nevyhnutné, aby sa program uberal viacerými možnými smermi, v závislosti od podmienok. Na to nám slúži vetvenie (teda konštrukcia if-else).

CYKLUS



je skupina príkazov, ktoré sa cyklicky vykonávajú daný počet krát
Niekedy v programe potrebujeme, aby sa daná činnosť vykonala alebo nami daný počet krát, alebo dovtedy, kým nenastane určitá situácia. Na to nám slúžia cykly s podmienkou na začiatku (while-do), na konci (repeat-until) a s daným počtom opakovaní (for-to/downto-do).

Vhodnou kombináciou týchto troch štruktúr je možné napísať akokoľvek zložitý program, Ktorý bude navyše prehľadný nielen pre kompilátor, ale aj pre jeho tvorcu alebo užívateľa.

Rozklad : **<u>disjunktívny</u>** ->problém sa rozloží na podproblémy a z nich sa realizuje len jeden
konjuktivny->problém sa rozloží na podproblémy a tie sa riešia všetky
<u>repetičný</u> -> problém sa rieši opakovaným riešením podproblému

Disjuktívny		**Konjuktívny**		**Repetičný**	
Podmienený	**Nepodmienený**	**Postupnostný**	**Množinový**	**Cyklický**	**Rekurzívny**
Napred známa jednoznačná podmienka	Musia sa preskúmať všetky možnosti	Poradie presne určené	Na poradí nezáleží	Opakovanie v cykle	Rekurzívne volanie
Počet dní do konca mesiaca	Kvadratická rovnica	Načítaj, rob, vypíš	Kalkulačka	NSD	Faktorial

OK, 5 kreditov

34. Číselné sústavy, ich použitie pri práci s počítačmi, dvojková sústava, operácie v dvojkovej sústave, prevody medzi sústavami. Hexadecimálna sústava.

Číselnou sústavou môžeme rozumieť spôsob zápisu danej číselnej hodnoty. Existuje mnoho sústav, najznámejšie sú tieto štyri:

- Binárna (dvojková) sústava (0,1)
- Oktálna (osmičková) sústava (0,1,2,3,4,5,6,7)
- Decimálna (desiatková) sústava (0,1,2,3,4,5,6,7,8,9)
- Hexadecimálna (šestnásťková) sústava (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

Vo výpočtovej technológii je binárna (dvojková) sústava nenahraditeľná. Počítače poznajú iba „jednotky a nuly“, všetky základné operácie a výpočty sú vykonávané za pomoci binárnych čísel. Všetky súbory sú len poradie núl a jednotiek. Digitálny obrázok je obrovské pole núl a jednotiek. Väčšinou to ale nevidíme, pretože tieto čísla sú prekladané do nami viditeľnej a pochopiteľnej podoby. Pri práci s počítačmi využívame rôzne číselné sústavy, väčšinou desiatkové pri výpočtoch, alebo programovaní, ale nech už robíme čokoľvek, vždy si to počítač prekladá do binárnej sústavy.

Binárna sústava
Pozná len dve číslice a to 0 a 1. Je nenahraditeľná u počítačov, je veľmi jednoduchá, založená na opakovaní číslic 0 a 1. Prvých niekoľko čísel: 0, 1, 10, 11, 100, 101, 110, 111, 1000 …

Operácie v binárnej sústave.
Sčitovanie funguje na rovnakom princípe ako v desiatkovej sústave. Spamäti to veľa ľudí asi nedokáže, preto je potrebné napísať čísla pod seba a postupovať podobne ako v desiatkovej sústave. T.j. sčítavaním čísel nachádzajúcich sa v stĺpci, keď dostaneme hodnotu väčšiu ako 1, zapíšeme jednotkovú číslicu na jednotkovej pozícii a číslicu z desiatkovej pozície prenesieme do nasledujúceho sčítania atd.
Príklady na pochopenie:

dec	0	1	2	3
bin	0	1	10	11

```
  1 1 1 1 1  (carried digits)
    0 1 1 0 1
+   1 0 1 1 1
-------------
= 1 0 0 1 0 0
```

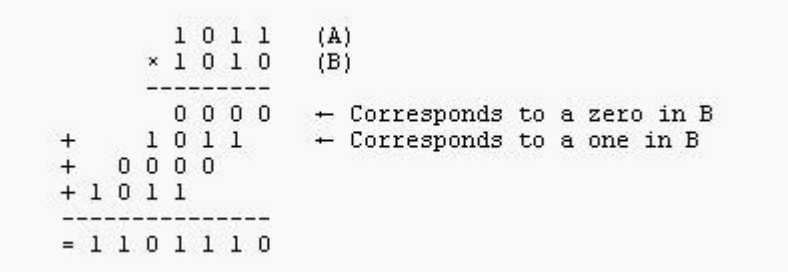
0 + 1 = 1
1 + 1 = 10
1 + 10 = 11
11 + 1 = 100
11 + 11 = 110

Odčítavanie je taktiež podobné ako v desiatkovej sústave pri zápise čísel pod sebou.

```
    *   * * *   (starred columns are borrowed from)
  1 1 0 1 1 1 0
-     1 0 1 1 1
---------------
= 1 0 1 0 1 1 1
```

0 – 0 = 0
1 – 0 = 1
1 – 1 = 0
11 – 1 = 10
110 – 11 = 11

Násobenie je veľmi jednoduché, taktiež porovnateľné s tým v desiatkovej sústave pri zápise čísel pod sebou.



1 x 1 = 1
1 x 0 = 0
10 x 1 = 10

Prevody medzi sústavami. Ako som už spomínal, je to len rôzny zápis číselnej hodnoty, to znamená že jednu hodnotu môžeme zapísať v ľubovoľnej číselnej sústave. Všeobecný zápis: $(x)_a = (y)_b$, kde x, y sú hodnoty, a, b značia číselné sústavy. Prevody ale nie sú až také jednoduché. Preto si ukážeme základné prevody.

- Z desiatkovej do dvojkovej:
  Delenie daného desiatkového čísla dvojkou, zapisovanie zvyškov, až dôjdeme k číslu 0, ktoré sa už ďalej nedá deliť. Príklad s číslom 235
  235 : 2 = 117 (zv. 1)
  117 : 2 = 58 (zv. 1)
  58 : 2 = 29 (zv. 0)
  29 : 2 = 14 (zv. 1)
  14 : 2 = 7 (zv. 0)
  7 : 2 = 3 (zv. 1)
  3 : 2 = 1 (zv. 1)
  1 : 2 = 0 (zv. 1)
  Zvyšky po delení zapíšeme od konca po začiatok, t.j. 11101011... Hotovo ☺
  Iný spôsob je o niečo zložitejší, zapíšeme si známe mocniny čísla 2:
  1 (2^0), 2 (2^1), 4 (2^2), 8 (2^3), 16 (atď), 32, 64, 128, 256, 512, 1024, …
  A teraz haluzné odčítavanie tých mocnín (začneme číslom, ktoré je menšie a najbližšie k danému číslu, zvolíme si napr. č.156 (128 < **156** < 256 … použijeme 128)
  156 – 128 = 28 (28 > 0 … dostaneme 1)
  28 – 64 = x (x < 0 … dostaneme 0) (x … pretože jeho hodnota nás nezaujíma)
  28 – 32 = x (x < 0 ... 0) (dostali sme 0, číslo 28 presúvame ďalej)
  28 – 16 = 12 (12 > 0 ... 1)
  12 – 8 = 4 (4 > 0 ... 1)
  4 – 4 = 0 (0 = 0 ... 1) (keď nám ostane 0 = 0, dostaneme 1)
  0 – 2 = x (x < 0 … 0)
  0 – 1 = x (x < 0 … 0) (musíme pokračovať do konca !!! (po 1)
  Čísla, ktoré sme dostali teraz zapíšeme od začiatku po koniec, t.j. 10011100. hotovo ☺
- Z dvojkovej do desiatkovej:
  Najjednoduchší a najčastejší spôsob je, že každú cifru daného binárneho čísla očíslujeme sprava do ľava mocninami čísla 2, teda:

Mocniny	64	32	16	8	4	2	1
Bin.č.	1	1	1	0	1	0	1

  Teraz čísla v stĺpcoch vynásobíme, výsledky posčitujeme, t.j.
  1*1 + 2*0 + 4*1 + 8*0 + 16*1 + 32*1 + 64*1 = 1 + 4 + 16 + 32 + 64 = 117 ... hotovo
- Z desiatkovej do šestnásťkovej je analogické prevádzaniu z desiatkovej do dvojkovej. Dané desiatkové číslo delíme číslom 16, pričom si zapisujeme zvyšky (ktoré následne prevádzame do šestnásťkovej sústavy. Delíme až kým výsledok sa nerovná 0, t.j. (zvolíme číslo 463)

463 : 16 = 28 (zvyšok 15 ... $(15)_{10} = (F)_{16}$)
28 : 16 = 1 (zvyšok 12 ... hex C)
1 : 16 = 0 (zvyšok 1 ... hex 1)

Analogicky zapíšeme od konca po začiatok, dostaneme číslo 1CF … hotovo ☺

- Zo šestnásťkovej do desiatkovej je analogické prevádzaniu z dvojkovej do desiatkovej. Používame ale mocniny čísla 16 (1, 16, 256, 4096, …)

Mocniny	4096	256	16	1
Hex. č.	4	B	2	D

1*D + 16*2 + 256*B + 4096*4 = 1*13 + 16*2 + 256*11 + 4096*4 = 19245 ☺

- Zo šestnásťkovej do dvojkovej sústavy sa dá ísť za pomoci desiatkovej (samozrejme aj bez nej, ale to by sme museli ovládať binárne vyjadrenie čísel A,B, ... F. Číslo napr. B3 si rozložím na 11 a 3 (desiatkové čísla), a tieto čísla prevediem do dvojkovej, pozor! Nie klasicky ale takou fintou: Predstavme si zase mocniny dvojky (... 32, 16, 8, 4, 2, 1) a číslo 11 môžem zapísať ako súčet dvojkových mocnín (iba štvoricu ale!!), teda 8 + 0 + 2 + 1 = 8*1 + 4*0 + 2*1 + 1*1 = 1011, to isté spravím s číslom 3, dostanem 3 = 0 + 0 +2 + 1 = 8*0 + 4*0 + 2*1 + 1*1 = 0011, tieto štvorice spojím ako idú za sebou, dostanem binárne číslo 10110011 … hotovo.
- Z dvojkovej do šestnástkovej je v podstate spätný chod predošlého postupu, binárne číslo si rozdelím na štvorice, tie premením do šestnástkovej sústavy a zapíšem ich za sebou ako idú a mám to.
- Všeobecne aby sme pochopili princíp, stačí nám vedieť ako sa píšu jednotlivé čísla v jednotlivých sústavách. Napríklad v sedmičkovej sústave je to 0, 1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15, 16, 20, 21, 22… ... ... 63, 64, 65, 66, 100, 101, 102 ... číslovanie pre x-ovú sústavu bude teda 0 ... x-1. V trojkovej sústave to bude teda logicky 0, 1, 2, 10, 11, 12, 20, 21, 22, 100, 101, 102, 110 … atď.

Hexadecimálna sústava:
Obsahuje 16 číslic. 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Keďže dvojciferné číslo ako napr. 11 nemôžem použiť ako číslicu, nahrádzam ho zrozumiteľným znakom. Podľa univerzálnej dohody prvé písmená abecedy.
OK, dobre napísané, 5 kreditov

35.Algoritmus, program. Abeceda, gramatika a syntax jazyka, vytvárajúce pravidlá. Lexikálne jednotky jazyka Pascal. Organizácia programu v jazyku Pascal. Syntaktické a sémantické chyby v programe.

Algoritmus - postup (návod) na riešenie problému
- postupnosť krokov (akcii), vykonaním ktorých prídeme k vyriešeniu problému
(napr. kuchárske recepty, návody na používanie ..)

Vlastnosti algoritmu: 1.Hromadnosť -> algoritmus rieši všeobecnú úlohu
2.Determinovanosť (jednoznačnosť) -> jednotlivé kroky algoritmu, ich postupnosť je presne určená
3.Rezultatívnosť -> po konečnom počte krokov vedie algoritmus k výsledku

Zápis algoritmov 1.Slovný
2.Grafický-Vývojový
3.Formou programu (v programovacom jazyku)

Krok algoritmu - jeden príkaz
- jedna elementárna akcia

Program: prepis algoritmov do ľubovoľného programovacieho jazyka

(Každý programovací jazyk je definovaný **abecedou** a **gramatikou**.)

Abeceda - základná množina navzájom odlíšiteľných objektov (znakov), s ktorými jazyk pracuje.
Z týchto znakov sa podľa určitých pravidiel vytvárajú prípustné tvary čiastkových konštrukcií jazyka, jeho príkazov a celého programu. Množinu týchto pravidel nazývame **syntax jazyka**.

Gramatiku programovacieho jazyka tvorí :
1. množina symbolov prípustných pre zápis programu
   (napr. v Pascale sú prípustné symboly: begin, * , := , A , B , C ,...)

2. množina pravidiel spájania týchto symbolov do jazykových konštrukcií, tzv. syntax programovacieho jazyka. (Napr. konštrukcia A:=begin je zložená z prípustných symbolov, ale nie je vytvorená podľa syntaktických pravidiel.)

Lexikálne jednotky jazyka Pascal

Abecedu Turbo Pascalu tvoria:
1.písmená latinskej abecedy => A..Z, a..z (bez mäkčeňov a dĺžňov!)
2.číslice desiatkovej sústavy => 0..9
3. špeciálne symboly => operátory, zátvorky ...

Organizácia programu v jazyku Pascal

Každý program pozostáva z 3 základných častí: 1.hlavička programu
2.časť deklarácií a definícií
3.príkazová časť

Hlavička programu obsahuje **meno** programu a **parametre**, pomocou ktorých identifikujeme vstupné a výstupné údaje programu.
V časti deklarácií a definícií, deklarujeme a definujeme procedury, funkcie, konštanty, typy a premenné, ktoré budeme používať v príkazovej časti programu. Ide o návestia, konštanty, typy, premenné, procedúry, funkcie,

Ad 1) Časť deklarácií a definícií tvorí spolu s príkazovou časťou jednu jazykovú konštrukciu, ktorú nazývame **BLOK**.
Blok je základnou konštrukciou nielen programu ale aj procedúr a funkcií. Program tak nadobudne hierarchickú blokovú štruktúru.
Ad 3) Príkazová časť je tvorená postupnosťou príkazov. Príkaz predstavuje akciu, ktorá sa má vykonať.
Príkazy v danej postupnosti sa pri realizácií programu vykonávajú v takom poradí ako sú napísané, ak nie je zadané iné poradie- skok.
Všetky objekty, ktoré používame v príkazovej časti bloku, musia byť deklarované alebo definované v jeho časti deklarácií a definícií.

Syntaktické a sémantické chyby Zistiť chybu v programe nie je jednoduché. Celý rad chýb nám však zistí samotný Turbo Pascal (TP). Ak zistí, že niektorá konštrukcia nevyhovuje gramatickým pravidlám daného jazyka, signalizuje chybu. Takejto chybe hovoríme syntaktická chyba. TP zvyčajne vypíše miesto v programe, kde sa daná

chyba vyskytla, a stručne udá aj jej charakter. Ako príklady syntaktických chýb možno uviesť:

a) If A> B *then* A:= A+1 *ele* B:=B+1; *ele* namiesto *else*
b) While X> Y repeat X:=X+1; *repeat* namiesto *do*
c) Wrietln; *Wrietln;* namiesto *Writeln;*

TP nájde obyčajne všetky syntaktické chyby. Spustiť môže iba syntakticky správny program.
Okrem syntaktickej kontroly vykonáva aj sémantickú kontrolu. Kontroluje napríklad, či premenné na pravej strane príkazu priradenia majú priradené hodnoty, či operandy výrazu sú požadovaného typu, či je premenná iba jedného typu a pod. takéto chyby nazývame sémantické chyby.

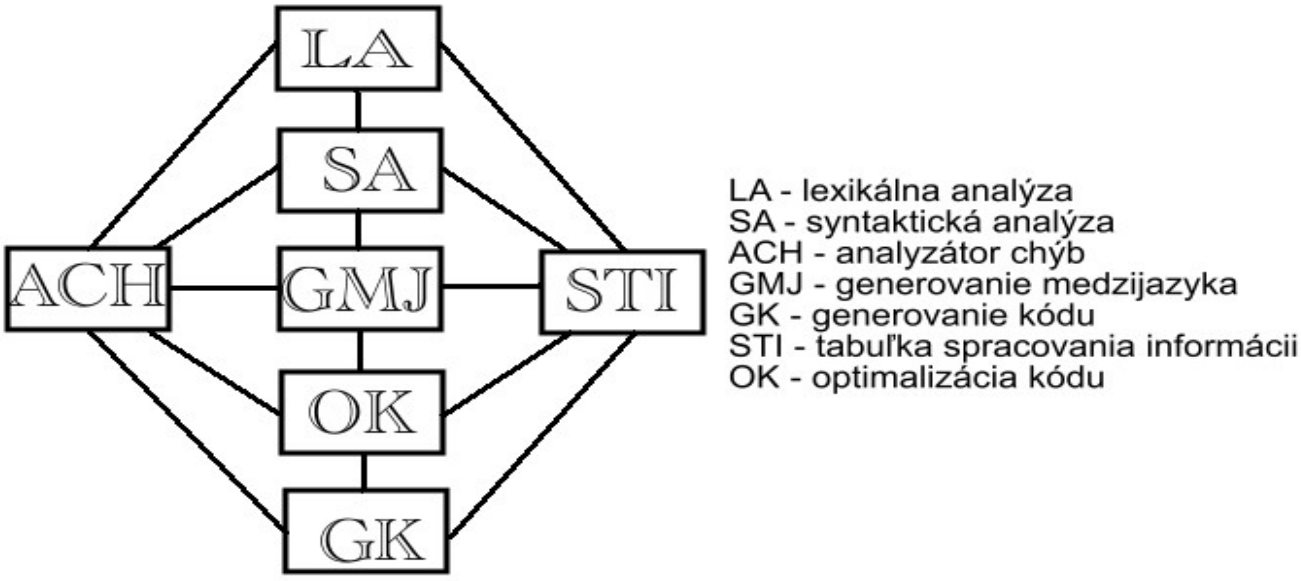
36. Kompilácia, spustenie programu, volanie knižníc, vyhľadávanie chýb v programe a ich oprava

Kompilácia, spustenie programu

Rozdiel medzi kompiláciou a spustením je v tom, že kompilácia preloží program do jazyka zrozumiteľného počítaču, ale nespustí ho, a spúšťanie program skompiluje a následne spustí. Každú kompiláciu kontroluje kompilátor, t.j. program, ktorý prekladá program tak, aby bol čo najefektívnejšie spustiteľný. Každá kompilácia kontroluje zdrojový text a ak je celý v poriadku, program preloží.

Prekladače:

-kompilátor: prekladá z vyššieho programovacieho jazyka (VPJ) do strojového kódu, alebo do jazyka symbolických adries (JSA)
-assembler: z JSA do strojového kódu
-predprocesor: z VPJ do strojového kódu
-interpreter: z VPJ do medzijazyka



Volanie knižníc

`uses` v programe uvádza zoznam knižníc (jednotiek), ktoré má prekladač prehľadať pri preklade programu, to znamená, že má zmysel uvádzať len tie jednotky, ktorých príkazy program využíva. Všetky pascalovské príkazy sú kvôli rýchlosti spracovania rozdelené do skupín - knižníc. Je rýchlejšie prehľadať

zopár jednotiek, ako zoznam všetkých príkazov. Navyše Pascal umožňuje vytvárať jednotky vlastné, v ktorých si možno zadefinovať vlastné príkazy (procedúry a funkcie) a vlastné premenné.

Ak program používa príkazy určené pre textovú obrazovku, stačí napísať len `uses crt;`. Ak program začne používať aj grafickú obrazovku, je potrebné volať grafickú knižnicu pomocou `uses crt, graph;`.

V programe sa môžu vyskytovať chyby lexikálne, syntaktické, logické. Prvé dve skupiny chýb sú odhalené pri kompilácii, tretia skupina nespôsobí nekompilovateľnosť, ale program dáva nekorektné výsledky. V tomto prípade pomôže krokovanie programu spolu s kontrolou hodnôt premenných, ktoré nadobúdajú počas vykonávania programu (CTRL+F7 = Watch okno)

Vyhľadávanie chýb v programe a ich oprava

Pri spúšťaní programu v prípade nájdenia chyby kurzor zastane na najbližšom mieste (hneď) za chybou v programe.
Najčastejšie chybové hlásenia, ktoré sa vyskytujú pri kompilácii:

Číslo	Text	Význam
3	Unknown identifier	nedefinovaný identifikátor
4	Duplicate identifier	identifikátor už bol použitý
8	String constant exceeds line	neukončený reťazec (chýba apostrof)
10	Unexpected end of file	nečakaný koniec súboru (neukončený cyklus, niekde chýba End)
11	Line too long	príliš dlhý riadok
26	Type mismatch	nekompatibilné typy premenných, nesúlad typov formálnych a skutočných parametrov
42	Error in expression	chyba vo výraze (zabudnutý operátor)
62	Division by zero	delenie nulou
85	“;“ expected	chýba bodkočiarka
89	“)“ expected	neuzavretá zátvorka
97	Invalid FOR control variables	premenná cyklu musí byť jednoduchého typu (napr. integer)
113	Error in statement	chybný príkaz (asi je na nesprávnom mieste)
121	Invalid qualifier	nedovolené indexovanie
106	Invalid numeric format	chyba v prečítanom čísle (desatinná čiarka namiesto desatinnej bodky)
200	Division by zero	delenie nulou počas vykonávania programu
207	Invalid floating point operation	reálnu hodnotu nemožno priradiť celočíselne premennej, nevhodný argument funkcie Sqrt alebo Ln

OK, 4 kredity.

- **Vyhodnocovanie výrazov, priorita pri vyhodnocovaní výrazov. Operandy, operátory vo výraze**

Pod výrazom rozumieme predpis na získanie hodnoty. Vytvára sa z operandov, operátorov a okrúhlych zátvoriek. Ako operandy možno používať konštanty, premenné a zápisy funkcií – /*zatiaľ štandardné funkcie*/. Ako operátory používame operátory definované pre zavedené údajové typy.
Zápis výrazov je lineárny, na jednej úrovni. Nemožno používať zlomky ani exponenty. Na vyjadre zlomku používame zavedené operároty delenia, umocňovanie robíme pomocou štandardných funkcií.

Pre operátory zavedených typov platí priorita uvedená od najvyššej po najnižšiu:

- `not`
- `*, /, div, mod, and`
- `+, -, or`
- `<. <=, =, <>, >=, >`

Vyhodnocovanie výrazov sa robí v zásade zľava doprava, pričom rešpektujeme prioritu operátorov a zátvorky. Pri vyhodnocovaní výrazu treba brať do úvahy typ operandov a zistený typ výsledku vykonanej operácie. Typ výslednej hodnoty určuje aj typ výrazu.

V Pascale je priorita operátorov iná ako v matematike. Rozdiel je v priorite logických operátorov. Ak používame ako operand logického výrazu reláciu, musíme ju dať do zátvoriek.
OK - 5 kreditov

39. Úplný a neúplný podmienený príkaz, význam podmieneného príkazu, vetvenie programu, syntax úplného a neúplného príkazu IF, typy podmienok, vnáranie príkazu IF. Syntax a realizácia príkazu CASE.

Podmienené príkazy umožňujú vybrať na realizáciu jeden z dvoch alebo viacerých príkazov na základe splnenia alebo nesplnenia určitej podmienky. Jazyk Pascal poskytuje tri podmienkové príkazy:

1. *Neúplný príkaz IF*

2. *Úplný príkaz IF*

3. *Príkaz CASE*

1. **Príkaz IF**

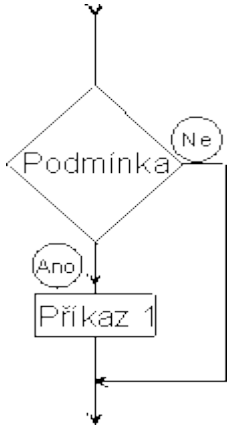
Úplný a neúplný príkaz IF slúžia na podmienené vetvenie programu.

a) **Neúplný príkaz IF**

má tvar: **IF** podmienka **THEN** príkaz **;**
alebo **IF** podmienka **THEN Begin** príkaz$_1$ **;**
príkaz$_2$ **;**
...
príkaz$_n$ **;**
End;

Príkaz sa vykoná, resp. príkaz$_1$...príkaz$_n$ sa vykonajú len v tom prípade, ak podmienka nadobúda hodnotu *true* (ak je podmienka splnená). Ak podmienka nadobúda hodnotu *false* (nie je splnená), nevykoná sa ani jeden s uvedených príkazov. Podmienka ja výraz typu *boolean* (ide o logický výraz, ktorý buď platí, alebo neplatí).Podmienka môže byť buď relenčný výraz vyjadrený ako *a>b* alebo logický výraz (a>10) and (a<=20). Príkazu **if** podmienka **then** pr zodpovedá konštrukcia **ak** podmienka **tak** príkaz 1.

Vetvenie programu: **JAVYD**



b) **Úplný príkaz IF**

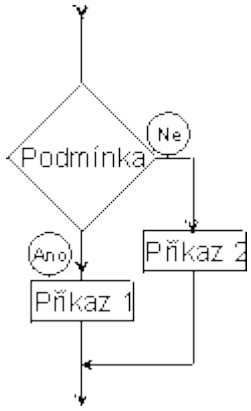
má tvar: **IF** podmienka **THEN** príkaz$_1$
ELSE príkaz$_2$ **;**
alebo **IF** podmienka **THEN Begin** príkaz$_1$ **;**
príkaz$_2$ **;**

...
príkaz$_n$ **;**
End
ELSE Begin príkaz$_{n+1}$ **;**
príkaz$_{n+2}$ **;**
...
príkaz$_k$ **;**
End ;

Príkaz$_1$ sa vykoná, resp. príkaz$_1$ až príkaz$_n$ sa vykonajú len v tom prípade, ak podmienka nadobúda hodnotu *true* (ak je podmienka splnená). Ak podmienka nadobúda hodnotu *false* (nie je splnená), vykoná sa príkaz2,

Príkazu **if** podmienka **then** príkaz1 **else** príkaz2 zodpovedá konštrukcia **ak** podmienka **tak** príkaz 1 **a ak podmienka nebola splnená tak,** príkaz 2.

Vetvenie programu: **JAVYD**



DOLEŽITÉ !
If b1 then if b2 then p1 else P2
Význam tohtopríkazu sa dá vysvetľovať dvoma sposobmi
a) if b1 then a1 -> kde a1 je priikaz if b2 then p1 else p2
b) if b1 then a2 else p2 -> kde a2 je príkaz if b2 then p1

If A > 0 then if A < 1 then B:= 0.1 else B:=1.0
Ak A=1.5, tak v intrpretacii:

Ad a) sa bude realizovat prikaz B:=1.0;
Ad b) je celý podmienený príkaz bez účinku

2. Príkaz CASE

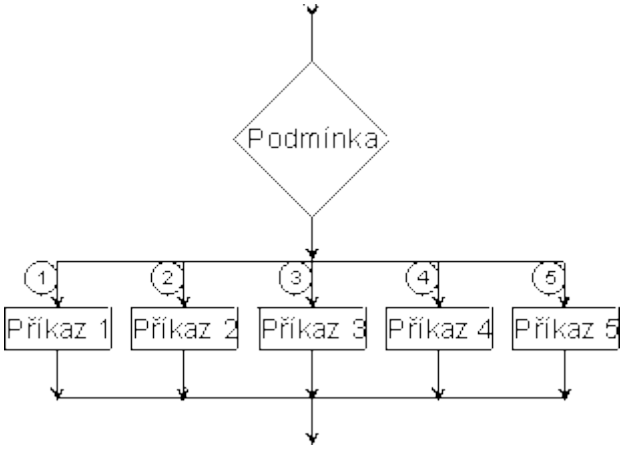
Príkaz CASE umožňuje viacnásobné podmienené vetvenie programu do niekoľkých alternatív v závislosti na hodnote premennej alebo výrazu.

má tvar: **CASE** premenná **OF**
hodnota$_1$ **:** príkaz$_1$ **;**
hodnota$_2$ **:** príkaz$_2$ **;**
...
hodnota$_n$ **:** príkaz$_n$
ELSE príkaz$_{n+1}$ **;**
End ;

Príkaz CASE sa dá opísať ako prikaz IF, ale mame viac možností ako len true alebo false.

Premenná musí byť ordinárneho typu napr. *integer* alebo *char*.

Príkazu **case** premenná **of** hodnota1:príkaz1 **else** príkaz1+n zodpovedá konštrukcia **ak** pemenná = hodnota1 **tak** príkaz1 **inak** príkaz1+n.
Vetvenie programu: **JAVYD**



OK, 5 kreditov.

40. Syntax a realizácia príkazu FOR, cyklu s daným počtom opakovaní, požiadavky na riadiacu premennú cyklu, inkrementácia a dekrementácia riadiacej premennej. Cyklus v cykle.

Príkaz FOR v programovacom jazyku Pascal má využitie ako cyklus. Pri vstupe do cyklu FOR je už vopred známy počet opakovaní daného cyklu. Existuje viac možností syntaxu príkazu FOR a tie sú nasledovné:

1. for i := a to b do príkaz; možnosti 1 a 2 sú totožné, to PRIKAZ môže byť zložený príkaz
2. for i := a to b do begin príkaz_1; príkaz_2; … príkaz_n; end;
3. for i := a downto b do príkaz; možnosti 3 a 4 sú totožné
4. for i := a downto b do begin príkaz_1; príkaz_2; ... príkaz_n; end;

i – riadiaca premenná cyklu, musí byť **ordinálneho** typu typu nesmie byť real
a – začiatok cyklu, **ordinálny** typ môže byť for I:= ‘A’ to ‘G’ do…
b – koniec cyklu, taktiež **ordinálny** typ
a to b – inkrementujúci cyklus, t.j. a<=b, hodnota riadiacej premennej v každom opakovaní cyklu narastá o 1, počínajúc hodnotou a, končiac hodnotou b, počet opakovaní cyklu je n = b-a+1
a downto b – dekrementujúci cyklus, t.j. a>=b, hodnota riadiacej premennej v každom opakovaní cyklu klesá o 1, počínajúc hodnotou a, končiac hodnotou b, počet opakovaní cyklu je n = a-b+1
do – využíva sa keď požadujeme v každom opakovaní cyklu vykonať práve jeden príkaz, resp. priradenie
do begin ... end; – využíva sa keď požadujeme v každom opakovaní cyklu vykonať aspoň 2 príkazy, resp. priradenia

Keď raz už cyklus FOR začal, nemôžeme zmeniť počet opakovaní cyklu a nemôžeme ani zasahovať do hodnoty riadiacej premennej, jej hodnotu využívame pri výpočtoch, výpisoch a pod. Pri vstupe do cyklu sa hodnota riadiacej premennej nastaví na hodnotu premennej a, nasledovne sa riadiaca premenná inkrementuje, alebo dekrementuje o 1. Posledné opakovanie cyklu prebieha pri hodnote riadiacej premennej rovnej b, potom nasleduje opustenie cyklu.

Využitie cyklu FOR je rozmanité. Pokiaľ potrebujeme vykonať množstvo rovnakých po sebe idúcich operácii (príkazov, priradení a pod.) a vieme aj požadovaný počet opakovaní týchto operácii, vtedy je najlepšie využiť cyklus FOR, ktorý nám umožní sprehľadniť a uľahčiť prácu.

Cyklus v cykle nastáva, keď sa v cykle nachádza ďalší cyklus. Syntax je jednoduchý, stačí keď za príkaz (pozri vyššie) dosadíme celý syntax nového cyklu, ktorý obsahuje ďalšie príkazy a v nich sa môže nachádzať ďalší cyklus atd.

Využitie cyklu v cykle je časté. Pokiaľ potrebujeme vykonať množstvo rovnakých po sebe idúcich operácii (pričom je medzi nimi aj cyklus) vtedy vytvárame cyklus v cykle.
Napr. keby sme chceli vypísať na obrazovku štvorec o rozmeroch 20x20 z hviezdičiek. Bolo by príliš kruté napísať pod seba 20 cyklov, kde každý by vypisoval hviezdičky. Lepšie je napísať jeden cyklus s 20 opakovaniami v ktorom sa bude nachádzať cyklus, ktorý vypíše 20 hviezdičiek. Ďalším príkladom sú klasické algoritmy triedenia (cez MAX)
OK, môže byť, ale opravil som v úvode vážnu chybu. Riadiaca premenná nemusí byť len celočíselná, musí byť ordinálna. Význam slova ordinálny je POVINNÁ ZNALOSŤ pri maturitách. 4 kredity

41.Príkazy cyklu riadené podmienkou v jazyku Pascal (WHILE, REPEAT), vzájomný vzťah príkazov WHILE a REPEAT, rozdiely medzi nimi. Vhodnosť použitia jednotlivých typov, cyklus v cykle.

V jazyku Pascal poznáme dva cykly, ktoré sú viditeľne riadené podmienkou a to REPEAT a WHILE.
V oboch prípadoch záleží vykonanie príkazov v tele daných cyklov od podmienky.

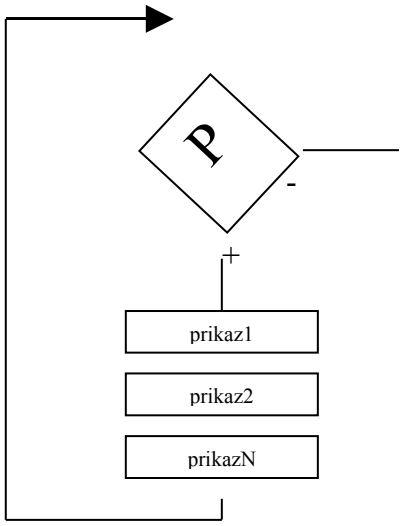
While

Kód

```
{pre jeden príkaz}
while (podmienka) do príkaz;

{pre viacero príkazov}
while (podmienka) do
  begin
    príkaz1;
    príkaz2;
    ...
    príkazN;
  end;
```

JAVYD



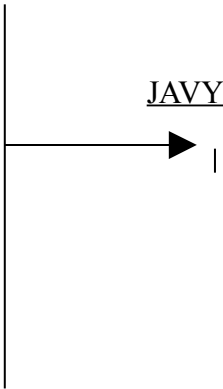
Cyklus WHILE má podmienku na začiatku, z toho vyplýva, že postupnosť príkazov v tele cyklu sa môže vykonať 0 až n krát.
Pokiaľ bola podmienka splnená, vykonajú sa príkazy a cyklus sa vráti na začiatok, kde sa znova skontroluje podmienka. Tento postup sa opakuje pokiaľ podmienka vyhovuje.
Podmienka, ktorá sa vyhodnocuje je vlastne logický výraz, ktorý nadobúda logickú hodnotu *true/false*.

Repeat

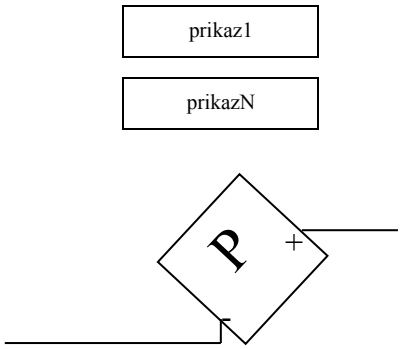
Kód

```
repeat
  príkaz1;
  príkazN;
```

JAVYD



```
until (podmienka);
```



Cyklus REPEAT na rozdiel od cyklu WHILE má podmienku na konci, a to nám zaručuje, že príkazy v tele cyklu sa môžu vykonať 1 až n krát.
Cyklus prebieha nasledovne: Vykonajú sa všetky príkazy v tele cyklu, následne sa skontroluje podmienka. Ak logický výraz v podmienke nadobudne hodnotu *true*, cyklus sa ukončí, naopak ak je hodnota logického výrazu *false* cyklus sa vráti na začiatok a všetky príkazy sa vykonávajú odznova.

Hlavný rozdiel a využitie týchto cyklov

Hlavným rozdielom medzi cyklom WHILE a REPEAT teda je počet vykonania príkazov.

WHILE.....0 – n
REPEAT.....1 – n

Ďalším rozdielom je aj opačné reagovanie na podmienku. Zatiaľ čo cyklus WHILE pokračuje pokiaľ je podmienka splnená, cyklus REPEAT naopak v takom prípade končí svoju činnosť.
Od týchto vlastností sa nám následne rozvíja aj využitie cyklov v praxi.

While

Svoje uplatnenie nájde v algoritmoch ako je napr. NSD , prevod sekúnd na hodiny – minúty - sekundy ale taktiež v algoritme na ciferný súčet.

```
{algoritmus pre NSD}
function nsd(c1,c2:integer):integer;
begin
  while c1<>c2 do
        begin
          if c1>c2 then c1:=c1-c2
          else c2:=c2-c1;
        end;
  nsd:=c1;
end;
```

Repeat

Svoje hlavné uplatnenie nájde v programoch, kde je potrebné opakovane vykonávať jednu činnosť, pričom niekedy stačí keď je táto činnosť vykonaná iba raz. Nádherným príkladom je vykresľovanie menu programu. Výpis menu sa môže opakovať 1 až N krát, pričom podmienka môže reagovať na stlačenie klávesy ESC. Keď užívateľ stlačí túto klávesu, dáva cyklu signál, že ukončil prácu s programom a ďalší výpis menu už nie je potrebný. Iný pekný, no zároveň primitívny spôsob využitia tohto cyklu je jednoduché ošetrenie vstupu od užívateľa:

```
{algoritmus pre jednoduche osetrenie vstupu}
repeat
  writeln('Zadaj cislo vecsie ako 20: ');
  readln(cislo);
until cislo>20;
```

OK, dobre, 5 kreditov

42. Spôsob deklarácie poľa v jazyku Pascal, indexované premenné a práca s nimi, typ pole - jednorozmerné, dvojrozmerné, príklady použitia jednorozmerných a dvojrozmerných polí.

Pole je štruktúrovaný údajový typ zložený z pevného počtu zložiek rovnakého typu, kde pozícia každej zložka je presne daná pomocou vlastného indexu. Využíva sa najmä vtedy, ak potrebujeme pomocou premenných uchovať veľa hodnôt rovnakého typu (mená všetkých ľudí v triede, známky žiakov apod.). Na sprístupnenie prvku poľa je potrebné poznať meno celej štruktúry a index prvku v poli. Indexom zvyčajne býva interval, alebo viac intervalov.
Deklarácia: ***meno_pola* =ARRAY[*index*] OF *typ_položky.***
Viacrozmerné pole možno deklarovať rovnakým spôsobom, pričom n- indexov je potrebné oddeliť čiarkou, teda: ***meno_pola* =ARRAY[$index_1$, $index_2$, $index_n$] OF *typ_položky.***
Premennú tohto typu je možno deklarovať v programe takto: *a,b:meno_poľa*

Príklad:

```
var i: byte;
n: integer;
a: array [1..10] of integer;
begin
for i:=1 to 10 do
begin
writeln('zapis',i,'. cislo:')
readln(n);
a[i]:=n;
end;
for i:=1 to 10 do
writeln(i,'. cislo je:'a[i]);
end.
```

Program načítava čísla z klávesnice a ukladá ich do poľa. Na záver ich vypíše.
Hm, tak tohoto je veľmi málo, v Indexe sa môže spomenúť ordinalita, veľa riadkov sa dá popísať o využití jedno a dvojrozmerného poľa (doporučujem modrý pascal), kto si toto vylosuje, musí mať možnosť sa o niečo oprieť,
3 kredity

43. Pojem a význam vyhľadávania, lineárne a binárne vyhľadávanie, porovnanie jednotlivých metód z hľadiska časovej a pamäťovej zložitosti. Výhody a nevýhody jednotlivých spôsobov

Vyhľadávanie je proces prezerania zložiek poľa s cieľom zistiť, či pole obsahuje zložku s danými vlastnosťami. Rozoznávame dva druhy vyhľadávania: lineárne a binárne. (lineárne, lineárne do zarážok (so zarážkami), binárne)

Lineárne (sekvenčné) vyhľadávanie :

- používa sa väčšinou v neutriedených poliach
- je nutné porovnať každú zložku poľa
- je pomalšie ako binárne

Ak chceme len zistiť či sa hľadaná zložka v poli nachádza alebo nie:
zadáme VZOR- hľadanú zložku
i :=1
ak VZOR=a[i], potom sme našli hľadanú zložku inak i:=i+1

1.2. Lineárne vyhľadávanie môžeme uskutočniť pomocou cyklu so známym počtom opakovaní, ak potrebujeme zistiť počet opakujúcej sa hľadanej zložky (napr. koľkokrát sa v poli vyskytuje číslo.2):

```
var j,i,n:integer;
a:array [1..10] of integer;
```

```
begin
        clrscr;
        for i:=1 to 10 do
        a[i]:=random(20);
readln(n);  {hodnota n je daná z klávesnice}
j:=0;
for i:=1 to 10 do
if a[i]=n then j:=j+1;
if j>0 then writeln (´Číslo', n,' sa v poli vyskytuje ´,j,´ krát)
else writeln (Číslo n sa v poli nevyskytuje);
readln;
end.
```

1.3. Cyklus s podmienkou na konci alebo na začiatku je výhodné použiť, ak nás zaujíma iba miesto-index neopakujúcej sa zložky:

```
var i,n:integer;
a:array [1..20] of integer;
begin
randomize;
for i:=1 to 20 do a[i]:=random(50);
readln(n);
i:=0;
repeat
i:=i+1;
until (a[i]=n) or (i=20);
if a[i]=n then writeln (Číslo n sa v poli nachádza na ´,i,´. mieste.)
else writeln (Číslo n sa v poli nenachádza.);
readln;
end.
```

Výpočtová zložitosť je 3N+1

Lineárne vyhľadávanie so zarážkou:

- je to jedna z modifikácií lineárneho vyhľadávania
- je rýchlejšia
- princíp lineárneho vyhľadávania s zarážkou spočíva: máme n- prvkové pole, hľadaný prvok x priradíme na koniec poľa na (n+1)miesto. Vyhľadávanie prebieha v poli [1.. n+1]. Ak sa hľadaný prvok x nájde až na (n+1)mieste, znamená to, že prvok x dané pole [1.. n] neobsahuje.

```
var i,n:integer;
a:array [1..20] of integer;
begin
randomize;
for i:=1 to 20 do a[i]:=random(50);
readln(n);
a[i+1]:=n;
i:=0;
repeat
i:=i+1;
until (a[i]=n) or (i=20+1);
if a[i]=n and (i<20+1) then writeln (Číslo n sa v poli nachádza na ´,i,´. mieste.)
else writeln (Číslo n sa v poli nenachádza.);
readln;
end.
```

Výpočtová zložitosť je 2N+1

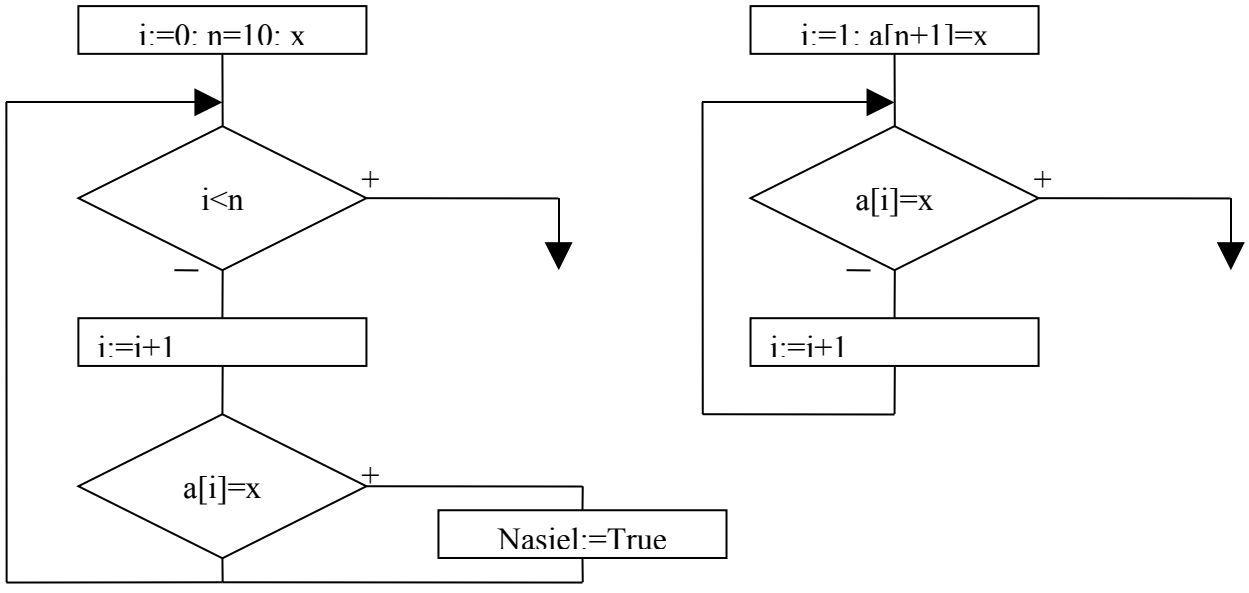
Binárne vyhľadávanie :

-používa sa v utriedených poliach
-je rýchlejšie
-zložitosť ln(logaritmus)
-zisťuje, v ktorej polovici poľa sa hľadaná zložka môže vyskytovať a potom zmenšuje interval hľadania posunutím dolnej alebo hornej hranice poľa. Využíva sa tu cyklus s podmienkou na konci alebo na začiatku.

príklad s podmienkou na konci :

```
var j,i,n,k,z:integer;
a:array [1..10] of integer;
begin
randomize;
readln(n); { n vyjadruje počet prvkov poľa }
for i:=1 to n do a[i]:=random(20);

i:=1; { i je dolná hranica poľa }
j:=n; { j je horná hranica }
repeat
k:=(i+j) div 2;
if a[k]<>z then
if a[k]
else j:=k-1;
until (a[k]=z) or (i>j); { z je hľadané číslo zadané z klávesnice }
if z=a[k] then writeln (Číslo sa v poli nachádza na ´,k,´. mieste)
else writeln (Číslo sa v poli nenachádza.);
```



44. Algorimy triedenia. Popísať triedenie cez minimum (maximum), bublinkové, vsúvaním, zlučovaním, lexikografické, quicksort. Porovnanie z hľadiska časovej a pamäťovej zložitosti. Príklady použitia, posúdenie výhodnosti zvoleného algoritmu.

Triediaci alogoritmus

Je algoritmus, ktorý usporadúva zoznam prvkov podľa určitých pravidiel. Najpoužívanejšie usporiadania sú numerické a lexikografické.

Triedenie cez minimum (maximum)

Známe pod názvom **Selection Sort**

Selection sort (triedenie výberom) je porovnávací triediaci algoritmus, ktorý funguje nasledovným princípom:
1. Nájdenie minimálnej / maximálnej hodnoty
2. Výmena tejto hodnoty s prvou hodnotou v zozname (poli)
3. Utriedenie zvyšku poľa

Je to asi najjednoduchší algoritmus na navrhnutie. Tento algoritmus sa používa prevažne na vysvetlenie fungovania porovnávacích algoritmov. Pre praktické aplikácie je nevhodný kvôli konštante dlhej dobe behu $O(n^2)$.

Obojsmerná varianta (v jednom cykle hľadá aj minimálnu aj maximálnu hodnotu) sa nazýva shaker sort.

Časová zložitosť – priemerná	Časová zložitosť – najhoršia	Pamäťová zložitosť
-	$O(n^2)$	O(1)

Bublinkové triedenie

Alebo **Bubble Sort**

Bubble sort funguje porovnávaním dvoch susedných prvkov a ich následnou výmenou. Pole sa prechádza dovtedy, kým nie je celé usporiadané, teda nedôjde k žiadnej výmene.

V najlepšom prípade (napríklad v prípade už usporiadaného poľa) je časová zložitosť O(n) v najhoršom však až $O(n^2)$. Preto sa bubble sort neodporúča používať na väčších poliach. Vo väčšine prípadov je dokonca pomalší ako hore uvedený selection sort.

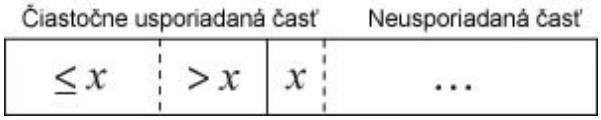
Časová zložitosť – priemerná	Časová zložitosť – najhoršia	Pamäťová zložitosť
$O(n^2)$	$O(n^2)$	O(1)

Triedenie vsúvaním

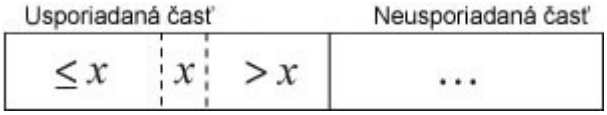
Alebo **Insertion Sort**

Jednoduchý algoritmus, ktorý je relatívne efektívne na čiastočne utriedených poliach a je často využívaný ako súčasť komplexnejších algoritmov.
Funguje vyňatím jedného prvku z poľa a jeho umiestnením na správne miesto v poli. Vkladanie je však pomalá operácia, vyžadujúca posunutie všetkých väčších resp. menších prvkov a jednu pozíciu vyššie/nižšie.
Väčšina implementácií prechádzam poľom od prvého prvku, ktorý je na začiatku považovaný za usporiadanú časť poľa a do tejto časti pridáva prvky z neusporiadanej časti.

Z poľa v tvare:



dostaneme



a pokračujeme pre ďalší prvok z neusporiadanej časti až kým sa neusporiada celé pole.

Časová zložitosť – priemerná	Časová zložitosť – najhoršia	Pamäťová zložitosť
$O(n^2)$	$O(n^2)$	$O(1)$

Triedenie zlučovaním

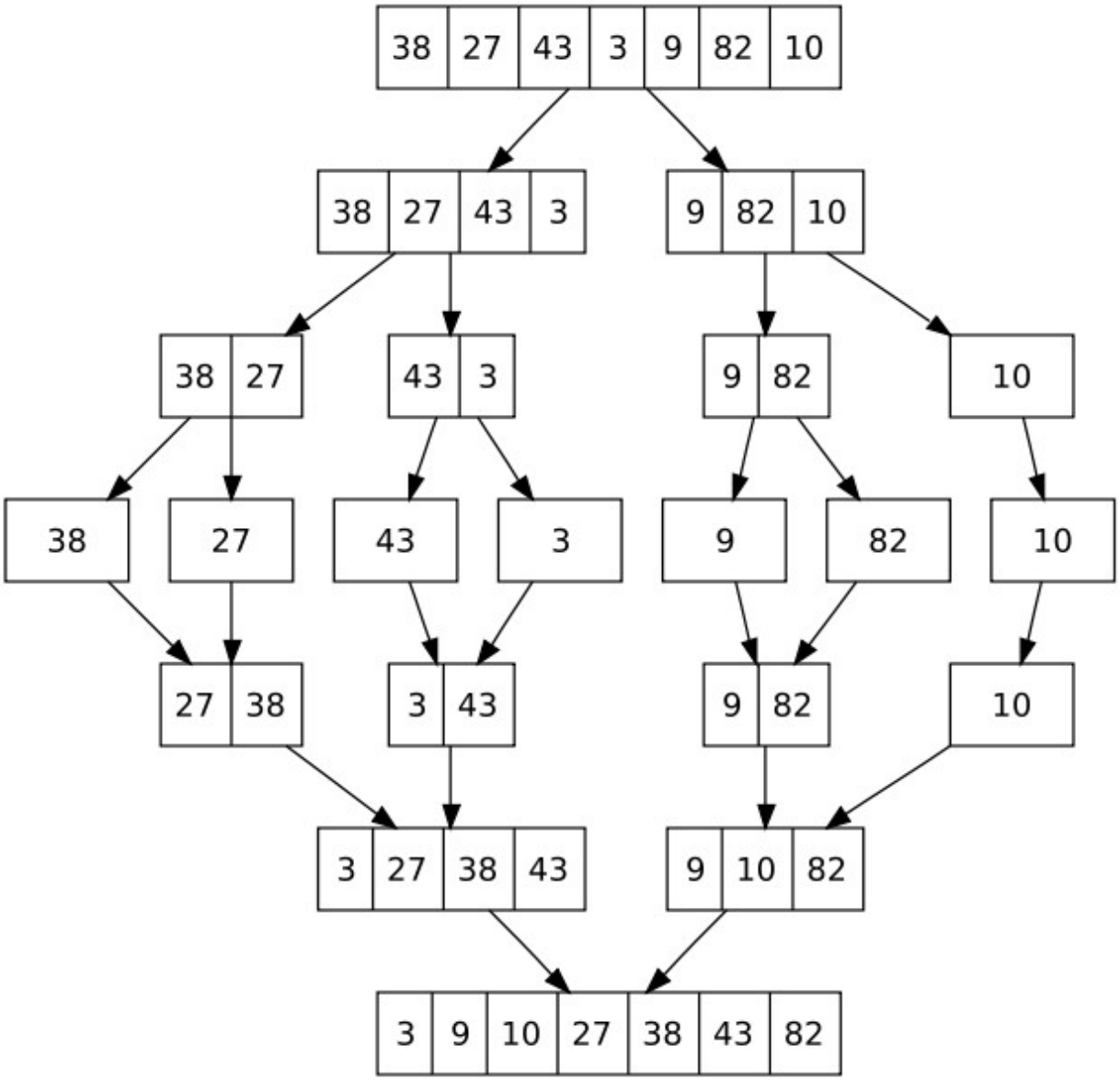
Alebo **Merge sort**

Merger sort v princípe funguje nasledovne:

1. Rozdeliť neusporiadané pole na dve rovnaké časti
2. Znovu rozdeliť každé polovičné pole rekurzívne až na polia o veľkosti jedného prvku v prípade ktorých sa vráti jednoprvkové pole samotné
3. Spojiť dve polia do jedného usporiadaného

Dva hlavné princípy, ktoré využíva merger sort:

- Usporiadať menšie polé trvá kratšie ako usporiadať veľké pole (divide et impera)
- Na vytvorenie usporiadaného poľa z dvoch usporiadaných polí je potrebných menej krokov ako na vytvorenie rovnakého poľa z dvoch neusporiadaných polí.



Algoritmus dobre funguje aj na veľmi veľkých poliach a dosahuje konštantne dobré výsledky.

Časová zložitosť – priemerná	Časová zložitosť – najhoršia	Pamäťová zložitosť
O(n log n)	O(n log n)	O(n)

Quicksort

Quicksort je asi najznámejší z rýchlych triediacich algoritmov. Všeobecne dosahuje quicksort najlepšie výsledky v porovnaní s inými algoritmami v jeho kategórii. Aj keď sa jeho časová zložitosť pohybuje od O(n log n) po $O(n^2)$ v najhoršom prípade, vďaka jeho vnútornému cyklu je možné implementovať ho tak dobre, že pravdepodobnosť kvadratického času výpočtu sa zníži na minimum.

Quicksort je tiež typickým príkladom algoritmu divide et impera.
V prvom kroku sa vyberie pivot – medián (prvok, ktorý približne rozdelí pole na dve rovnaké časti), od výberu závisí či bude quicksort bežať v kvadratickom čase alebo nie.

V ďalšom kroku dôjde k preusporiadaniu prvkov tak, že všetky prvky menšie ako pivot sa dostanú pred a väčšie za pivot. Po tejto operácii sa pivot dostane na svoje miesto vo výslednom poli, ideálne zvolený pivot sa dostane do stredu poľa.

Tieto dva kroky sa rekurzívne opakujú pre každé pod-pole.

Časová zložitosť – priemerná	Časová zložitosť – najhoršia	Pamäťová zložitosť
O(n log n)	$O(n^2)$	O(log n)

OK, veľmi dobre napísané, keby tu ešte bolo lexikografické, bolo by to super.
Maturujúci by mal vedieť na konkrétnych piatich číslach dokumentovať na tabuli spôsob zvoleného triedenia, 5 kreditov

45. Údajové typy - implicitné a užívateľsky definované, množiny povolených operácií nad údajovým typom. Význam návrhu správnych dátových štruktúr v programe. Deklarácia vlastných typov, typy string, char. Typ pole, množina, záznam, súbor, príklady použitia, dynamické dátové štruktúry.

Údajové typy

<table>
<tr><td colspan="2">Jednoduché:</td><td rowspan="11">Statické</td></tr>
<tr><td>integer</td><td rowspan="5">Ordinálne</td></tr>
<tr><td>boolean</td></tr>
<tr><td>vymenovaný</td></tr>
<tr><td>interval</td></tr>
<tr><td>char</td></tr>
<tr><td>real</td><td>Neordinálne</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Štruktúrované:</td></tr>
<tr><td>pole</td><td></td></tr>
<tr><td>množina</td><td></td></tr>
<tr><td>záznam</td><td></td></tr>
<tr><td>súbor</td><td></td></tr>
<tr><td colspan="2">Typ ukazateľ:</td><td>Dynamické</td></tr>
</table>

Každý údajový typ je definovaný:

1. Identifikátorom
2. Množinou hodnôt
3. Množinou prípustných operácií
4. Množinou štandardných funkcií

Identifikátor	Množina hodnôt	Operácie	Štandardné funkcie
Boolean	false, true	and, or, not, <,<=,...	succ, pred. ord
Integer	<-32768,32767>	+,-,*,div,mod,<,<=...	abs, sqr, trunc, round, odd
Real	-2.1 az 2.1 mld	nie je div a mod	aj sqrt, goniom, log, exp
Char	množ. znakov	<, <=,>,>=	ord, chr, pred, succ

Vlastné údajové typy

Najčastejšie sa vlastný údajový typ používa práve v situáciách, keď chceme používať nejakú vlastnú údajovú štruktúru.
Tu treba doplniť vymenovaný a interval (to sú užívateľom definované typy)
type dlzka = integer;

Záznam (record)

je ideálnym nástrojom na vytvorenie vlastnej „na mieru šitej“ údajovej štruktúry. Tvoríme ich pomocou už existujúcich typov, využiť môžeme aj polia. Môže obsahovať niekoľko prvkov (napr. meno, priezvisko, dátum narodenia, a pod.).

```
type osoba = record
  meno: string;
  priezvisko: string;
  vek: integer;
end;
..
ziak: osoba;
...
write('Zadaj meno: ');
readln(ziak.meno);
write(‘Zadaj priezvisko:‘);
readln(ziak.priezvisko);
write(‘Zadaj vek‘);
readln(‘ziak.vek‘);
```

Tiež môžeme použiť príkaz With:

```
with ziak do begin
write('Zadaj meno: ');
readln(meno);
write(‘Zadaj priezvisko:‘);
readln(priezvisko);
write(‘Zadaj vek‘);
readln(‘vek‘);

end;
```

Množina (set)

Vymenovaním sa stanoví bázový typ (popis množiny).

V rámci údajového typu množina sú definované množinové a relačné operácie:

+ zjednotenie
* prienik
- rozdiel
= rovnosť
<> nerovnosť
<= je podmnožinou
>= obsahuje (nadmnožina)

```
mnz1: set of  char;
...
mnz1:=[“1“..“9“];
```

Práca s údajovým typom množina poskytuje niekoľko výhod, ale aj niektoré obmedzenia.

Veľmi silná je operácia IN, používaná v tvare *prvok in mnozina.*

```
if znak in mnz1 then write(‘Prvok patri do mnoziny mnz1’)
```

tu treba nutne doplniť deklarácie pre súbor a dynamickú premennú (stačí v krátkosti), pri príprave to čítajúci iste ocenia

OK, zatiaľ 4 kredity

46. Základné pojmy, údajový typ súbor, práca s databázou údajov. Štandardné procedúry a funkcie pre prácu so súbormi. Typové a netypové súbory, textové súbory. Prehľadávanie súboru, štandardné procedúry pre prácu so súbormi assign, rewrite, reset, close, eof...

Súbor:
Sekvencia dát uložená na

- **typový**
- **netypový**
- **textový**

Typový súbor - sekvencia položiek rovnakého typu, typ je daný deklaráciou (FILE OF Typ_Zložiek_Súboru). Typ zložiek môže byť ľubovoľný. V súbore je uložená VNÚTORNÁ reprezentácia položiek - tak ako sa položky ukladajú v operačnej pamäti.
Netypový súbor - sekvencia položiek ľubovoľného typu okrem typu SÚBOR. Ukladá sa VNÚTORNÁ reprezentácia položiek

Textový súbor - súbor pozostáva zo ZNAKOV ORGANIZOVANÝCH DO RIADKOV (sekvencia znakov #13#10 označuje koniec riadku). V textovom súbore sa ukladá ZNAKOVÁ reprezentácia hodnôt.
ASSIGN - Priradenie premennej predstavujúcu súbor:
Priradí meno externého súboru Cesta súborovej premennej. Súbor NESMIE BYŤ OTVORENÝ. Po tomto priradení sa so súborom manipuluje iba prostredníctvom súborovej premennej. Cesta je výraz typu String, obsahujúci meno externého súboru (ak treba, vrátane cesty).

```
Assign(subor, 'C:\MOJE.TXT');
```

REWRITE
Vytvorí a otvorí nový súbor. Ak súbor existuje, zruší ho a vytvorí prázdny.
Do súboru je možné zapisovať, z typových a netypových súborov je možné i čítať.

```
Rewrite(subor)
```

RESET - Overenie existencie súboru:
Otvára EXISTUJÚCI súbor, ukazovateľ aktuálnej pozície nastaví na začiatok súboru. Ak súbor neexistuje - vstupno / výstupná chyba (ošetrenie pomocou inštrukcie {$I-} a funkcie IOResult)
Zo súboru je možné čítať, do typových a netypových súborov je možné i zapisovať.

```
{$I-}
reset(subor);
{$I+}
if IOResult<>0 then begin
                        rewrite(f);
                        .....
                        end;
```

CLOSE - Zavretie otvoreného súboru

```
Close(subor);
```

APPEND
Otvára existujúci TEXTOVÝ súbor pre pridávanie - ukazovateľ aktuálnej pozície je na konci.
Do súboru možno iba zapisovať

```
Append(subor) ;
```

OK, môže byť, 5 kreditov

47. Pojem procedúry, procedúra s parametrami a bez parametrov, Vymenovať spôsoby nahradzovania parametrov - volanie hodnotou a odkazom, schéma konformného poľa. Volanie procedúry, formálne a skutočné parametre procedúry. Porovnať uvedené spôsoby nahradzovania parametrov, lokálne a globálne objekty v programe_

Procedúry tvoria spolu s funkciami podprogramy. Procedúry tvoria postupnosti inštrukcií, ktoré potrebujeme v programe opakovať viackrát a na rôznych miestach. Procedúry musia byť deklarované ešte pred prvým príkazov begin v programe Po deklarácii môže byť procedúra použitá kdekoľvek v nasledujúcom bloku programu. Rozdiel medzi procedúrou a funkciou je v tom, že funkcia vracia hodnotu a môže sa použiť priamo vo výrazoch. Procedúra sa vyvolá príkazom volania procedúry a vykoná jednu alebo viac inštrukcií. Príkazy, ktoré bežne používame, sú procedúry a funkcie, ale už sú dopredu deklarované. Procedúry môžu byť bez parametrov, s parametrami volanými hodnotou a s parametrami volanými odkazom.

Deklarácia: PROCEDURE meno_procedúry(definícia vstupných premenných);

```
CONST definovanie_konštánt_procedúry;
TYPE definovanie_nových_typov_premenných;
VAR definovanie_lokálnych_premenných_pre_procedúru;

BEGIN     {začiatok procedúry}
príkaz;
 ..
END;     {koniec procedúry}
```

Globálne premenné sú definované v hlavnom tele programu a sú viditeľné v celom programe aj vrátane procedúr, pričom lokálne deklarované pre danú procedúru môžu byť použité len v nej.

Procedúry bez parametrov operujú s objektmi deklarovanými mimo danej procedúry (t.j. deklarovanými v celom programe)

Procedúry s lokálnymi objektmi obsahujú pomocné premenné zadefinované v tele procedúry

PARAMETRE:

formálne- parameter uvedený v procedúre: *procedure NSD(var a,b:integer*)

a,b sú formálne parametre procedúry NSD

skutočné- parametre uvedené pri volaní procedúry v hlavnom programe

begin
....
NSD(x,y);
x,y sú skutočné parametre procedúry NSD

DELENIE PARAMETROV

Parametre volané hodnotou- predstavujú v tele lokálnu procedúru platnú len v tele procedúry. Tie hodnoty privážajú, majú charakter vstupných premenných, takže hodnota skutočného parametra sa nemení.

Parametre volané odkazom- formálny parameter sa na začiatku procedúry stotožní so skutočným. Zmena formálneho parametra predstavuje zmenu skutočného parametra. Takéto parametre z procedúry hodnoty vyvážajú

Schéma konformného poľa - parametrom procedúry je pole, ak je volané odkazom, je možné do procedúry poslať 'prázdne pole' a z procedúry sa vráti pole naplnené hodnotami.

OK, poprosím doplniť príklady parametrov volaných hodnotou a odkazom tak, ako sú uvedené príklady parametrov formálnych a skutočných, 4 kredity

48: Štandardné procedúry jazyka Pascal, rozbor počtu parametrov v procedúrach.

V jazyku PASCAL poznáme množstvo funkcií a procedúr, ktoré slúžia na určité úlohy.
Medzi procedúrou a funkciou je rozdiel v tom, že funkcia má jedinú návratovú hodnotu.
Spoločné však majú to, že niektoré z nich majú parametre, s ktorými sú volané.
Štandardné procedúry a funkcie sú uložené v „unitoch".

Základný unit, ktorý sa nedeklaruje v uses, je System. Medzi ďalšie často používané patria CRT a graph.

Procedúry môžu mať vstupné ale aj výstupné parametre, tie výstupné (ktoré daná procedúra naplní datami) deklarujeme s klúčovým slovom „var“.

Vybrané procedúry funkcie unitu system:

Exit
Procedúra spôsobí návrat z podprogramu, nemá parametre;
Insert(source,s,index)
Procedúra vkladá podreťazec udaný parametrom.
(source: reťazec ktorý ma vložiť, s: reťazec do ktorého vkladá, index: pozícia)

Val(znaková_prem,čís_prem,code);
Procedúra konvertuje hodnotu typu reťazec na zodpovedajúcu numerickú hodnotu.

Readln([var F: Text;] V1 [, V2, ...,Vn]);
Priradí prečítanú hodnou do parametrov.

Write([var F: Text;] P1 [,P2,...,Pn]);
Vytlačí hodnoty premenných.

Vybrané procedúry unitu GRAPH:

Bar3D(x1,y1,x2,y2:integer;hlbka:word;vrchol:boolean);
Procedúra nakreslí kváder.

Circle(x,y : integer; r : word)
Procedúra nakreslí kruh.
(x,y: súradnice stredu)

DetectGraph(var gd, dm : integer)
Procedúra zistí, aký grafický driver a aký grafický režim potrebujeme pre činnosť grafického programu.

Ellipse(x,y:integer; zu,ku:word; vo,zo:word)
Procedúra nakreslí eliptický oblúk so stredom x,y, začiatočným uhlom zu, koncovým uhlom ku, veľkosťou vodorovnej polosi vo a veľkosťou zvislej polosi zo.

FloodFill(x,y:integer; farba_hranice:word)
Procedúra zaplní ohraničenú oblasť aktuálnym typom výplne, je potrebné zadať bod x,y ležiaci vo vnútri oblasti a farbu hranice uzatvorenej oblasti.

InitGraph(var gd:integer;var gm:integer;cesta:string)
Procedúra inicializuje grafický systém a nastavuje grafický režim (gm) na grafickom adaptéri (gd), pozri detectgraph.
Line(x1, y2, x2, y2 : integer)
Procedúra nakreslí čiaru ako spojnicu dvoch bodov [x1,y1]a [x2,y2].

PutPixel(x,y : integer; farba : word)
Procedúra nakreslí na pozícii [x,y] bod zadanej farby.

SetColor(farba : word)
Procedúra nastaví farbu pre kreslenie čiar a iných obrazcov.

Vybrané procedúry unitu GRT:

TextColor(farba:byte);
Nastavuje farbu textu. Farbu možno voliť od 0 do 15.

TextBackGround(farba:byte);
Nastavuje farbu pozadia. Farbu možno voliť od 0 do 7.

GotoXY(x,y:inter);
Presunie kurzor na pozíciu so súradnicami x,y (stĺpec, riadok) Page 2.

Procedúra ClrScr;
Zmaže obsah aktuálneho okna.

Procedúra DelLine;
Zmaže riadok na pozícii kurzora.

Procedúra InsLine;
Vloží prázdny riadok na pozícii kurzora.

Procedúra Delay(m:word);
Zastaví beh programu na dobu uvedenú v zátvorke v milisekundách.
OK, môže byť, 5 kreditov

49. Procedúry a funkcie pre prácu s typom STRING a CHAR - length, str, val, copy, delete, insert, pos, concat. Funkcie ORD, PRED, SUCC.

Typ CHAR je definovaný ako množina všetkých znakov v danej tabuľke znakov, najčastejšie ASCII, ktorá umožňuje kódovať až 256 znakov (0..255).
Typ STRING je definovaný ako reťazec (poradie znakov) o maximálnej dĺžke 255 znakov.
Na oboch typoch fungujú štandardné relačné operácia porovnávania:

```
<, <=, =, <>, >=, >
```

a	b	a+b (zreťazenie)	a <,<=,=,<>,>=,> b
char	char	string	boolean
char	string	string	boolean
string	char	string	boolean
string	string	string	boolean

Funkcie ORD, PRED, SUCC

Dané funkcie sú definované pre všetky ordinálne typy v jazyku pascal. Tieto typy sú: *celočíselné typy, char, boolean, interval, vymenovaný typ*. Všetky tieto typy sú definované ako lineárne usporiadané množiny hodnôt a pre ľubovoľné dva prvky z množiny platí práve jedna z relácií:

```
a < b, a = b, a > b
```

Na základe týchto vlastností môžeme definovať funkcie ORD, PRED a SUCC takto:

```
succ(ti) = ti+1  pre i=1,2,...,n-1
pred(ti) = ti-1  pre i=2,3,...,n
ord(ti)  = i-1   pre i=1,2,...,n
```

ORD

Funkcia, ktorá priradí každej hodnote z množiny poradové číslo pričom platí:

```
x < y <=> ord(x) < ord(y)
```

typ	x	ord(x)	pred(x)	succ(x)
integer	9	9	8	10
	-11	-11	-12	-10
char	"D"	68	"C"	"E"
	"5"	53	"4"	"6"

boolean	FALSE	0	-	TRUE
	TRUE	1	FALSE	-

Pozn.: Je treba si všimnúť, že *ord(C) != C*, kde C je číslica z intervalu 0..9. Pokiaľ potrebujeme zistiť hodnotu číslice C môžeme využiť vzťah:

```
C = ord(C) - ord('0') => 3 = ord('3') - ord('0')
```

Základné operácie s typom STRING

deklarácia typu:

```
s: string[c];  {kde 1 <= c <= 255}
s: string;  {ekvivalentné s string[255]}
```

Príklad:

```
s:string[20];     {v pamäti zaberá 21B - jeden byte naviac, kvôli dĺžke
                  string-u}
s:='ahoj';        {s[0]=#4, s[1]='a' ... s[4]='j'}
s:=s+' Jurko';    {s='ahoj Jurko'}
s:='Alena';
s2:='Anna';       {s2 > s}
```

Length

```
function length(s:string):integer;
```

Funkcia, ktorá dynamicky vráti dĺžku premennej typu STRING. Tiež platí nasledovné:

```
length(s) = ord(s[0])
```

Str

```
procedure str(x[:width[:decimals]]; var s:string);
```

Konvertuje numerickú hodnotu na STRING.

Príklad:

```
str(-4.9,s);      {s=-4.9000000000E+00}
str(-4.9:10:2);   {s=-4.90}
str(-4.9:10:0);   {s=-5 => funkcia zaokrúľuje}
```

Val

```
procedure val(s:string; var v; var code:integer);
```

Konvertuje STRING na jeho numerickú reprezentáciu. STRING môže obsahovať len číslice a znamienko. Inak konvertovanie zlyhá a do premennej *code* sa uloží, pozícia na ktorej konvertovanie zlyhalo.

Príklad:

```
val('-4.9',v,code);      {v=-4.9000000000E+00}
val('-49',v,code);       {v=-49}
val('-49a',v,code);      {v=0.0000000000E+00, code=4}
```

Copy

```
function copy(s:string; index:integer;
                  count:integer):string;
```

Vracia sub-string z pôvodného string-u, pričom *index* určuje začiatočnú pozíciu kopírovania a *count* určuje aký dlhý string sa má zkopírovať.

Príklad: *{v komentároch je zobrazený výstup na monitore pomocou funkcie writeln}*

```
copy('abcdef',2,4);      {bcde}
copy('abcdef',-3,6);     {abcdef}
copy('abcdef',2,-1);     {}
```

Delete

```
procedure delete(var s:string; index:integer;
                 count:integer);
```

Maže sub-string z pôvodného string-u, pričom *index* určuje začiatočnú pozíciu mazania a *count* určuje aký dlhý string sa má zmazať.

Príklad: *{v komentároch je zobrazený stav vstupného string-u po mazaní}*

```
s:='abcdef';
delete(s,2,4);          {af}
delete(s,-3,6);         {abcdef}
delete(s,2,-1);         {abcdef}
delete(s,1,6);          {}
```

Insert `procedure insert(source:string; var s:string; index:integer;);`

Vkladá *source* sub-string do výstupného string-u na pozíciu, ktorú určuje *index*. Sub-string môže mať akúkoľvek dĺžku, ak má po prevedení procedúry výsledný string dĺžku väčšiu ako 255 je zrezaný.
Príklad: *{v komentároch je zobrazený stav výstupného string-u po vkladaní}*

```
s:='abcdef';
insert('gh',s,6);       {abcdeghf}
insert('gh',s,-1);      {ghabcdef}
insert('gh',s,10);      {abcdefgh}
```

Pos `function pos(substr:string; s:string;):byte;`

Hľadá sub-string v zadanom string-u a vracia začiatočnú pozíciu tohto sub-string-u v string-u. Pokiaľ sa zadaný sub-string v string-u nenachádza vracia nulu.
Príklad: *{v komentároch je zobrazený výstup na monitore pomocou funkcie writeln}*

```
pos('a','abcdef');      {1}
pos('x','abcdef');      {0}
pos('cd','abcdef');     {3}
```

Concat `function concat(s1 [,s2,...,sn]):string;`

Spája zadané string-y do jedného výsledného string-u. Ak dĺžka výsledného string-u presahuje 255 znakov, je string zrezaný. Funkcia CONCAT má rovnaký efekt ako spájací operátor +.
Príklad: *{v komentároch je zobrazený výstup na monitore pomocou funkcie writeln}*

```
concat('abcdef');       {abcdef}
concat('abcdef','gh');  {abcdefgh}
s:='a'+'b'; <=> concat('a','b');
```

OK, dobre, 5 kreditov

50. Rozdiely medzi funkciou a procedúrou, deklarácia, volanie funkcie, spôsoby nahrádzania formálnych parametrov skutočnými. použitie parametrov vo funkcii. Rekurzia. Ordinálne údajové typy.

Procedúry a funkcie tvoria postupnosti inštrukcií, ktoré potrebuje v programe používať na rôznych súboroch dát alebo na rôznych miestach programu. Procedúra alebo funkcia môže byť po deklarácii použitá kdekoľvek v nasledujúcom texte bloku programu. **Rozdiel** medzi procedúrou a funkciou je v tom, že funkcia vracia **jedinú** hodnotu a môže sa použiť priamo vo výrazoch. Procedúra sa vyvolá príkazom volania procedúry na splnenie jednej alebo viacerých operácií.

Užitočnosť týchto pojmov sa prejaví, ak si chceme zapamätať určitý súbor príkazov. Najprv si ho zadefinujeme a potom ho môžeme vykonávať pomocou názvu procedúry alebo funkcie. Pri správnom použití sa **zlepší prehľadnosť kódu a zmenší veľkosť programu.**

Deklarácia procedúry bez parametrov

```
program nazov_programu;
 {deklaracia procedury}
procedure moja_procedura;                    {hlavička procedúry}
begin
 prikazy_procedury;                          {deklaračná časť}
end;
```

```
 {samotny program}

begin
 moja_procedura;                                  {príkazová časť}
end.
```

Deklaračná časť procedúry sa riadi rovnakými pravidlami ako hlavný program. Deklarácia procedúry priradí identifikátor k bloku kódu programu. Pri vyvolaní procedúry sa vykonajú príkazy, ktoré sú uvedené v tele procedúry. Ak sa v tele procedúry vyskytne identifikátor vyvolanej procedúry, vykoná sa procedúra rekurzívne.

Deklarácia funkcie

```
program mocniny;
 {deklaracia funkcie}
function mocnina (zaklad, exponent: integer): integer; {hlavička funkcie}
   var c, vysledok: integer;
begin                                            {deklaračná časť}
 vysledok := 1;
 for c := 1 to exponent do
  vysledok := vysledok * zaklad;
 Result := vysledok;
end;
  {samotny program}

begin
 writeln(mocnina(10,2));                         {príkazová časť}
 readln;
end.
```

Deklarácie funkcií majú podobnú štruktúru ako deklarácie procedúr, s tým rozdielom, že sa končia dátovým typom, ktorý funkcia vracia:

function *<meno funkcie> (formálne parametre):dátový typ;*

Funkcia sa aktivuje uvedením identifikátoru funkcie pri vyhodnotení výrazu, funkcia sa objaví ako operand výrazu. Keď sa výraz vyhodnotí, funkcia sa vykoná a hodnota operandu sa stane hodnotou, ktorú funkcia vracia. V tele funkcie sú podobne ako u procedúry definované príkazy, ktoré sa majú vykonať po aktivácii funkcie. **Telo funkcie musí obsahovať príkaz priradenia, ktorý priraďuje identifikátoru funkcie hodnotu.** Výsledkom funkcie je potom posledná priradená hodnota. Ak v tele funkcie príkaz priradenia neexistuje, funkcia vracia nedefinovanú hodnotu.

Parametre procedúr a funkcií

V hlavičke deklarácie procedúry alebo funkcie môžeme definovať zoznam formálnych parametrov. Každý parameter deklarovaný v zozname je lokálny v deklarovanej procedúre alebo funkcii. Vo vnútri bloku procedúry alebo funkcie je možne sa na neho odvolávať jeho identifikátorom.

Typy parametrov:

Formálne - objavujú sa v deklarácii procedúry alebo funkcie, sú dva druhy:

- Parametre volané odkazom (premenné)

  procedure moja_procedura(zoznam parametrov s udaním typu);
- Parametre volané hodnotou (hodnotové)

  procedure moja_procedura(var parameter: typ parametra);

Skutočné - sú to parametre, ktoré sa uvádzajú pri volaní procedúry alebo funkcie. Musia zodpovedať svojím dátovým typom parametrov, definovaným v podprograme a ich počet musí byť rovný počtu, ktorý sme v podprograme definovali.

Hodnotové parametre sa v deklarácii uvádzajú svojim identifikátorom, nasledovaným definíciou typu. Parametre, ktoré majú rovnaký charakter a typ, sa môžu uvádzať v zozname, oddelené čiarkou. Ak sú v deklarácii procedúry alebo funkcie parametre rôznych typov a charakterov, jednotlivé typy sa musia oddeliť bodkočiarkou. Formálne hodnotové parametre pôsobia ako lokálne premenné v procedúre alebo funkcii, s tým rozdielom, že dostávajú počiatočnú hodnotu pri aktivácii procedúry alebo funkcie. Zmena formálneho hodnotového parametru v tele procedúry alebo funkcie neovplyvní hodnotu skutočného parametru pri aktivácii procedúry alebo funkcie. Skutočný parameter, ktorý zodpovedá hodnotovému parametru v príkaze procedúry alebo volania funkcie, musí byť výraz a jeho hodnota nesmie byť typu súbor ani štruktúrovaný typ, ktorý obsahuje typ súbor. Skutočný parameter musí byť kompatibilný s formálnym parametrom.

Premenný parameter sa používa tam, kde sa musí predať hodnota z procedúry alebo funkcie späť volajúcej procedúre alebo funkcie. Zodpovedajúci skutočný parameter v príkaze procedúry alebo vo volaní funkcie musí byť odkazom na premennú. Formálny premenný parameter reprezentuje v priebehu aktivácie procedúry alebo funkcie skutočnú premennú. Akákoľvek zmena hodnoty premenného formálneho parametru sa prenesie na skutočný parameter volajúcej procedúry. Typ skutočného parametru musí byť identický s typom formálneho premenného parametru. Ako premenné parametre je možné používať typ súbor.

Rekurzia

- *priama rekurzia* - keď procedúra alebo funkcia v tele volá samú seba
- *nepriama rekurzia* - keď procedúra A volá procedúru B a tá zas procedúru A

Príklad
Rekurentná definícia výpočtu n- faktoriálu pre n≥0 hovorí:

1. 0! = 1 {výpočet 0! -triviálny problém}
2. n! = n * (n-1)! pre n > 0 {výpočet n! vedie k analogickému problému, výpočtu (n-1)!}
V zmysle rekurentnej definície možno napr. 5! vypočítať ako 5*4! = 5*(4*3!) =
5*(4*(3*2!)) =
5*(4*(3*(2*1!))) = 5*(4*(3*(2*(1*0!)))) = 5*(4*(3*(2*(1*1)))) = 120 (zátvorky naznačujú, ako prebehne výpočet)

Každá rekurzia obsahuje rekurzívnu vetvu, v ktorej je vlastne schovaný cyklus – opakované volanie procedúry, a nerekurzívnu vetvu, ktorá zabezpečuje ukončenie „cyklu“. Aktuálne hodnoty premenných pred vnorením do ďalšej procedúry sa odkladajú do zásobníka.

Údajový typ priraďujeme premenným v deklaračnej časti *(var)* a vyberáme ho podľa toho, akého druhu je premenná (celé číslo, reálne číslo, znak, text,...) a aký povolený rozsah hodnôt premenná nadobúda.
ORDINÁLNE údajové typy - existuje určiteľná predchádzajúca i nasledujúca hodnota
- patria medzi jednoduché údajové typy

- *Celočíselné*

Typ	Rozsah
byte	0 .. 255
shortint	-128 .. 127
integer	-32 768 .. 32 767
word	0 .. 65 535
longint	-2 147 483 648 .. 2 147 483 647

- *Ostatné*

Typ	Rozsah
boolean	true, false
char	tabuľka ASCII znakov; hodnotu zadávame v apostrofoch (napr. 'p')

string; string[počet znakov]	reťazec znakov; hodnotu zadávame v apostrofoch (napr. 'cela veta'); počet znakov (nepovinné) max. 255

OK, doplnil som viacero vecí, (def. Procedúr s parametrami) , 4 kredity

51. Vymenovať a popísať niektoré štandardné funkcie jazyka Pascal, vysvetliť vzťah parametrov funkcie a výsledku funkcie.

Procedúry a funkcie rozdeľujeme na štandardné a uživateľské. Štandardná funkcia vyzerá napr. y: = round(a,b) a uživateľská funkcia, ktorá je vytvorená uživateľom, vyzerá y: = NSD(a,b). Vstup pre každú funkciu musí byť správneho typu a počet vstupných parametrov sa tiež musí zhodovať.

succ( ) – *nasledovník* – *funkcia vracia nasledovníka argumentu, ktorý je ordinárneho typu , výsledok je rovnakého typu ako argument*
deklaracia: succ(x: ordinal): ordinal

pred( )- *predchodca* - *funkcia vracia predchodcu argumentu, ktorý je ordinárneho typu, výsledok je rovnakého typu ako argument*
deklaracia: pred(x: ordinal): ordinal

ord ( )- *poradové číslo*
- *funkcia vracia ordinárne číslo špecifikovanej hodnoty ordinárneho typu, konvertuje ordinármy typ na typ integer*
deklaracia: ord(x): longint

chr( ) – funkcia vracia znak ktorý je reprezentovaný číselnou hodnotou, konvertuje celočíselný typ na typ char
deklaracia:chr(x:byte): byte

odd ( ) - *vracia hodnotu true, ak je číslo párne, inak vracia false*
deklaracia: odd s(x:longintl):boolean
odd(2) = true
odd(3) = false

trunc( ) - *výsledkom je celočíselná časť reálnej hodnoty*
deklaracia: trunc(x: real): longint
trunc(9,9) = 9

abs ( ) - *vracia absolútnu hodnotu vstupu*
deklaracia:abs(x: y): y
abs(-6) = 6

sqr( ) - *vracia druhú mocninu hodnoty vstupu*
deklaracia: sgr(x: real): real
sqr(5) = 25

sqrt( ) - *vracia druhú odmocninu hodnoty vstupu*
deklaracia: sqrt(x: real): real
sqrt(25) = 5

round( ) – funkcia zaokruhĺuje réalne čilo na celočiselnú hodnotu, konvertuje typ real na typ celočíselný
deklaracia: round(x : real): longint
round(8,5) = 9
round(8,4) = 8

sin ( ) – *sínus, funkcia vracia sínus argumentu (argument je v radiánoch)*
deklaracia: sin(x: real): real

cos ( ) – *kosínus, funkcia vracia cosínus argumentu (argument je v radiánoch)*
deklaracia: cos(x: real): real

arctan( ) – *funkcia vracia arkustangens (výsledok je v radiánoch)*
deklaracia: arctan(x: real): real

ln( ) – *funkcia vracia prirodzený logaritmus argumentu (logaritmus pri základe*

e=2.718281828) deklaracia: ln(x: real): real

exp( ) – *funkcia vracia hodnotu* e^x
deklaracia: exp(x: real): real

OK, na úplný začiatok treba doplniť nejaký úvod, nemožno začať odpovedať 'zhurta' štandardnými procedúrami, ale najskor aspoň v skratke vysvetliť rozdiel medzi štandardnými a užívateľskými procedúrami. 4 kredity

52. Procedúry a funkcie na generovanie náhodných čísel, ošetrenie vstupu pre nepovolené hodnoty, testovanie programu

Pascal poskytuje funkciu a procedúru **random** ktorá generuje náhodné čísla.
* Ako procedúra
random;

Výsledkom je desatinne číslo typu **real** v rozmedzí <0, 1>.

*Ako funkcia
random(x: integer);

Vráti číslo typu **integer** v rozsahu 0, 1, 2 ... x - 1.

Náhodné čísla v pascale sú len "pseudonáhodné", tj. tieto čísla sa vyrábajú istým vzorcom, ktorý je dostatočné zložitý na to aby jeho výsledok vypadal ako skutočné náhodné čísla. Obyčajný pascalový random používa vzorec, ktorý sa opiera o iste počiatočné číslo. Na to aby sme zabezpečili aby sa toto počiatočné číslo neopakovalo môžme použiť funkciu **randomize**, ktorú keď zavoláme ako procedúru (tj. bez parametra) tak toto počiatočné číslo odvodí od aktuálneho systémového času a dátumu.
Môžme ju ale zavolať aj ako funkciu (napr. **randomize(5664)**), čím môžme priamo určiť toto počiatočné číslo. V tomto prípade nám program vygeneruje zakaždým spustením rovnakú postupnosť čísel.

Ošetrenie vstupu

Pod ošetrením vstupu si predstavujeme skontrolovanie vstupných údajov na nepovolené hodnoty. Jeden z príkladov môže byt program, ktorý požaduje na vstupe dve čísla a vypisuje ich podiel. Keďže sa v menovateli nemôže nachádzať nula (vznikla by chyba *delenie nulou*) tak je vhodne nejakým spôsobom zamedziť, alebo ošetriť prípad kedy sa na vstupe nula objaví.

Jedným zo spôsobov môže byt napríklad použitie príkazu **if** na zistenie či sa menovateľ nerovná nule, a ak sa rovná tak tuto skutočnosť používateľovi oznámiť. Ďalej to môže byt napríklad umiestnenie príkazu **readln** do cyklu **repeat..until** a určiť podmienku o tom že sa číslo nesmie rovnať nule. Tato verzia ma výhodu v tom, že používateľa núti opraviť chybný vstup ihneď, a tento sa ďalej nešíri do programu.

Pr.
```
repeat
  writeln('zadajte cislo rozne od nuly: ');
  readln(x);
until not ( x = 0 );
```
.....ok, tu už x určíte nulu neobsahuje

Testovanie programu

Testujeme pomocou rôznych vstupov. Je potrebné vyskúšať čo najviac variácii vstupov ako napríklad záporné hodnoty, nuly, veľké čísla (tzv. pretečenie zásobníka) a pod.
Otestovať sa musia všetky vetvy programu. Napr. naprogramovanú kvadratickú rovnicu nestačí raz spustiť. Tento program má 3 vetvy a programátor musí otestovať správanie sa každej z nich. Preto musí program trikrát spustiť a vložiť n-ticu údajov, takých, ktoré preveria každú z vetiev.
OK, 5 kreditov

53. Typy výpočtovej zložitosti - kvadratická, lineárna, odmocninová, logaritmická, odvodenie vzťahov pre jej výpočet, Pamäťová výpočtová zložitosť.

Pod pojmom výpočtová zložitosť rozumieme

- Časová výpočtová zložitosť
- Pamäťová výpočtová zložitosť

Časovou zložitosťou algoritmu rozumieme závislosť jeho časových nárokov na veľkosť konkrétneho riešeného algoritmu. Nemôžeme ju ale merať časom. Pretože by sme nedostali porovnateľné výsledky. Časovú zložitosť určujeme počtom operácii pri daných vstupných údajoch, resp. koľko krokov vykoná algoritmus, kým dôjde k výsledku. Keďže časovú výpočtovú zložitosť môžeme skúšať na rôznych vstupoch, je potrebné ju určovať pri najhoršom prípade. Značíme O(n), pričom n je počet daných krokov. Typy časovej výpočtovej zložitosti:

- Exponenciálna: $O(n^k)$
- Kubická: $O(n^3)$
- Kvadratická: $O(n^2)$
- Lineárna: $O(n)$
- Odmocninová: $O(\sqrt{n})$
- Logaritmická: $O(n*\log(n))$

Kvadratická výpočtová zložitosť, zvyčajne vyskytujúca sa pri jednoduchých algoritmoch triedenia polí. Máme n operácii, pričom musíme každú z nich vykonať najviac n-krát.

Lineárna výpočtová zložitosť, ktorá je typická pre jednoduché vyhľadávanie (jedného prvku medzi n prvkami). Máme n operácii, pričom musíme každú z nich vykonať najviac raz.

Odmocninová výpočtová zložitosť, využívaná napr. pre vyriešenie AxB, postupným pričítaním o B-viac. Napr. súčet A+A+A+...+A pretransformujeme do súčtu prírastkov nasledovne:
(A) + (A+A) + (A+A+A) + (A+A+A+A) + ...+(A+A+...+A)
tj. urobíme n(n+1) sčítaní B/2 krát, potom sa cyklus vykoná $K=1+2+3+...+n = n(n+1)/2$,kde K>=B na konci cyklu, čiže $n^2+n-2B>=0$ kvadratická nerovnica s riešením $n=3/2*\sqrt{B}$ čo približne predstavuje $n=\sqrt{2}*B$ (odmocnina rastie pomalšie ako lineárna funkcia).

Logaritmická výpočtová zložitosť, napr. veľmi známy algoritmus „Quick-sort" využívaný na triedenie, ktorý má práve túto zložitosť.
Príklad na log. zložitosť (súčin AxB): Pripočítavaním dvojnásobku predchádzajúceho prírastku dosiahneme počet prechodov $K=1 + 2 + 4 + ...+ 2^{n-1}$ a cyklus skončí vtedy, keď bude $2^{n-1}>=B$. Z toho $n=\log^2(B + 1)$. Ak presiahneme potrebné B, je nutná korekcia postupným odčítaním A, čo je postup s lineárnou výpočtovou zložitosťou, ale zvýšením nárokov na pamäť je možné urýchliť aj korekčný cyklus.

Pamäťovú zložitosť algoritmu definujeme ako závislosť pamäťových nárokov algoritmu na veľkosť riešeného programu. Meriame ju v bežných jednotkách pamäti, teda v bitoch, či v bytoch. Teda pamäťovú zložitosť určujeme počtom pamäťových miest potrebných na vykonanie algoritmu, pri veľkosti vstupných dát n.
HM, tak v prípade, že si túto otázku potiahne Chrenko a iný zdroj na učenie nemá, nastali by značné problémy.
Prosím, treba ten algoritmus násobenia AxB pomocou vylepšovaného sčitovania sem dostať. Je to celé na webe (VYUKA). Zatiaľ len 3 kredity so zažmúrenými očami.

54. Dynamické premenné a údajový typ ukazateľ, spojovacie štruktúry. Vytvorenie zoznamu, pridávanie, odoberanie prvkov zo zoznamov, spojenie zoznamov. Dynamické typy pamäte - halda a zásobník.

Dynamické premenné reprezentujeme pomocou dátového typu **ukazateľ** (pointer).

Pointer je dátový typ, ktorý priamo odkazuje na inú hodnotu v pamäti PC. Pointer teda obsahuje adresu pamäti, na ktorú je možné pristupovať. Prístup k hodnote uloženej na adrese pointeru sa nazýva **dereferencia**.

V Pascale existujú dva druhy pointrov:

- Typové
- Beztypové

Pri inicializácii typových pointrov nie je potrebné udávať veľkosť, ktorú budeme potrebovať.

Príklad použitia typového pointra:

```
var
  p: ^integer;

begin
  new(p);          { alokácia miesta pre integer }
  p^ := 3;           { nastavenie hodnoty, cez dereferenciu }
  writeln(p^);
  dispose(p);  { uvoľnenie pamäte }
end.
```

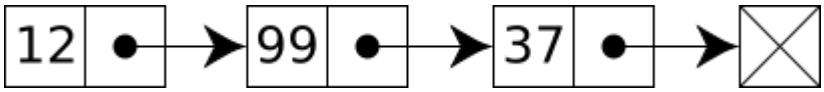
Dátové štruktúry využívajúce pointre

Spájaný zoznam

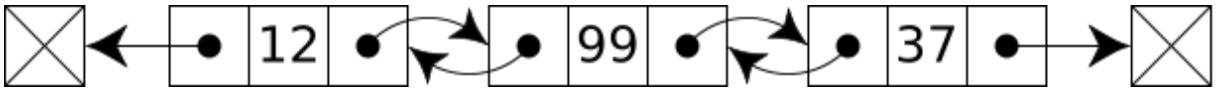
Najzákldanejšia dátová štruktúra pozostávajúca z x uzlov (napr. record), ktorý každý ma definovaný niekoľko ľubovoľných premenných a jeden alebo dva pointre ukazujúce na predchádzajúci resp. ďalší prvok alebo oboje. Spájaný zoznam dovoľuje pridávanie resp. odoberanie uzlov na ktoromkoľvek mieste ale na rozdiel od poľa nedovoľuje náhodný prístup (je potrebné prechádzať celým zoznamom).

Niekoľko typov zoznamov:

Jednoduchý spájaný zoznam – iba dve premenné, hodnota a pointer na ďalší uzol.



Dvojito spájaný zoznam – každý uzol obsahuje aj pointer na predchádzajúci prvok.



Pri každom zozname je potrebná ešte jedna premenná a to vstupný pointer, ukazujúci na celý zoznam resp. na jeho prvý prvok.

Deklarácia zoznamu:

```
type ukazatel=^osoba;
osoba = record
  meno:integer;
  dalsi: ukazatel;
end;

var vstup, aktualny, temp:ukazatel;
```

Vytvorenie:

Vytvoríme počiatočný prvok + k nemu pridáme ďalšie dva prkvy

```
new(vstup);
vstup^.meno := ‘Risko’;

aktualny := vstup;
new(aktualny^.dalsi);
aktualny := aktualny^.dalsi;
akutalny^.meno := ‘Nepodstatne’;
```

Prvok “akutalny” tak pridáme za prvok vstup. Vstup teda odkazuje na prvý prvok, aktuálny na druhý prvok v zozname. Keď skončíme s vytváraním zoznamu, poslednému prvku nastavíme nil (prázdny pointer)

```
akutalny^.dalsi := nil;
```

Pridávanie prvkov:
Prvok na začiatok pridáme tak, že po vytvorení nového prvku mu ako ďalší prvok nastavíme ‘vstup’, teda doteraz prvý prvok a ako vstup nastavíme samotný nový prvok.
Pre pridanie prvku na koniec alebo do iného miesta zoznamu, bežíme cez zoznam v cykle až kým nie je splnená určitá podmienka.
Napríklad nastavené nil – posledný prvok:

```
while z^.dalsi<>nil do
        z:=z^.dalsi;
```

Mazanie prvkov:
Po nájdení prvku, ktorý má byť vymazaný (s tým, že máme pointer na aktuálny aj predchádzajúci prvok) zmením pointer ‘dalsi’ predchádzajúceho prvku na pointer aktuálneho prvku. Tým preskočíme prvok, ktorý má byť zmazaný a ten sa stane nedostupný.

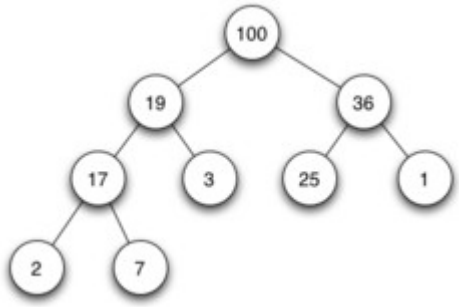
Spájanie zoznamov:
Ak potrebujeme spojiť zoznam **a** so zoznamom **b**, stačí sa dostať na koniec zoznamu **a** ako ‘dalsi’ miesto nil nastaviť začiatok zoznamu **b**.

Zásobník

Dátová štruktúra typu Last In First Out (LIFO). Prístupný vždy len posledný prvok. Len ten je možné zo zásobníku vybrať alebo pridať ďalší pred tento prvok.

Halda

Špecifická dátová štruktúra – strom, ktorý spĺňa podmienku haldy: *ak B je dcérsky vrchol vrcholu A, potom platí hodnota(A) >= hodnota(B).* Z toho vyplýva, že vrchol s najvyššou hodnotou bude vždy koreňový vrchol.



Hm, je to OK, ale odbornosť tohto textu je pravdepodobne nad ‘čo najjednoduchšie’ vysvetľovanou úrovňou tematiky dynamickej premennej z vyučovacích hodín.
Pre spolužiakov, ktorí toto nestrávia, odporúčam všetky odkazy zo stránky

http://www.gymzv.sk/~vyuka/inf_pro/dyn_prem/index.html
5 kreditov

55. Črty a požiadavky multiprogramových OS, HW a SW zabezpečenie. Správa pamäti, typy správy pamäte, úlohy správy pamäte. Správa procesora, jej význam, základné pojmy, stratégie prideľovania procesora, pamäte a zariadení

Teória OS

Úlohy OS: Transformácia PC do stavu pohodlnejšieho pre užívateľa a snaha využívať maximálne HW prostriedky

Požiadavky na OS: efektívnosť, ochrana, predikcia, pohodlné užívanie, spoľahlivosť

HW zabezpečenie: ochrana pamäte, privilegovaný užívateľský režim, systém prerušení

Správa pamäte(P):

funkcie / úlohy:

1. sledovanie stavu pamäte
2. stratégia prideľovania pamäte
3. samotné pridelenie pamäte
4. odobratie pamäte

typy prideľovania pamäte:

1. jednoduché súvislé
2. priehradkové
    a. statické (rovnaké priehradky)
    b. dynamické (dynamická veľkosť)
3. priehradkové s relokáciou (za pomoci relokačného registra)
4. stránkovanie, s. s vyžiadaním prenosu
    a. náhodný výber
    b. Round Robin (cyklický smerník)
    c. FIFO (prvý dnu, prvý von)
    d. najmenej časté používanie
    e. najdávnejšie používaná stránka
    f. (!) tzv. Beladyho alg. – nahraďte tú, ktorú budete používať najneskôr (len teoretická myšlienka, ktorá je nenaprogramovateľná)

Správa procesora(C):

úlohy C:

1. minimalizovať priechodnosť
2. krátky reakčný čas
5. efektívne využitie celého systému

funkcie:

1. sledovať stav C
2. stratégia prideľovania času
3. samotné pridelenie času – vykonanie operácie v aktívnej oblasti
4. odobratie prostriedkov = uvoľnenie C pre ďalšiu operáciu

stratégie prideľovania:

1. na úrovni úloh
    a. FIFO
    b. LIFO
    c. najkratší výpočet

   d. prioritné – statické (napr. beh úlohy pod rootom)
2. na úrovni procesora
   a. Round Robin
   b. Round Robin s 2 frontami
   c. prioritné (dynamické)
   d. najmenší prevedený čas
   e. najmenší zostávajúci čas

hm, tomuto textu síce nemožno nič vyčítať, čo sa týka presnosti a stručnosti, ale neposkytuje žiadnu možnosť doštudovania, príp. pochopenia doteraz nepochopeného (čiže je didakticky ako text na prípravu ku skúške slabo použiteľný)
Sú 2 možnosti. Buď sa aspoň doplní o obrázky (jednotlivé druhy správy pamäte + obr. čo bol na písomke),
alebo sa rozšíri text aj o vysvetlenia okrem vymenovačiek (pamäťové učenie, ktorému ja neverím) ☺
Zatiaľ 3 kredity.

56. Pojem matica. Úpravy matíc, eliminácia, riešiteľnosť sústav rovníc, násobenie matíc. Priamy a spätný chod Gaussovej eliminačnej metódy.

Matica je v podstate obdĺžniková tabuľa obsahujúca čísla, resp. rôzne matematické objekty. Najčastejšie sa stretneme s dvojrozmernou maticou (m riadkov x n stĺpcov), hovorí sa o matici typu m krát n.
V prípade rovnosti m=n hovoríme o štvorcovej matici. Pri výpočtoch s maticami, sa dané matematické operácie týkajú všetkých čísel matice. Čísla matice môžeme považovať za koeficienty premenných rovníc matice, alebo môžeme povedať, že matica typu m x n je sústava m rovníc o n-1 neznámych. Teda napr.

$$\begin{bmatrix} 0 & 1 & 2 \\ 4 & -7 & 8 \\ 3 & 9 & -8 \end{bmatrix} = \begin{bmatrix} 0x + 1y & = & 2 \\ 4x - 7y & = & 8 \\ 3x + 9y & = & -8 \end{bmatrix}$$

Úpravy matíc:
-Ktorúkoľvek rovnicu v matici môžeme prenásobiť konštantou alebo premennou.
-Poradie rovníc môžeme ľubovoľne prehadzovať.
-Tak isto aj pripočítať n-násobok ľubovoľnej rovnice k inej rovnici.
Pri sčítavaní alebo násobení matíc je to kúsok iné. Súčtom, resp. rozdielom matíc A, B je matica C, pričom každý jej prvok zodpovedá súčtu alebo rozdielu daných prvkov matíc A a B.

$$\begin{bmatrix} 1 & 3 & 2 \\ 1 & 0 & 0 \\ 1 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \\ 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 & 2+5 \\ 1+7 & 0+5 & 0+0 \\ 1+2 & 2+1 & 2+1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 7 \\ 8 & 5 & 0 \\ 3 & 3 & 3 \end{bmatrix}$$

Súčin matice s konštantou je jednoduchý, každý prvok matice prenásobíme danou konštantou.

$$2\begin{bmatrix} 1 & 8 & -3 \\ 4 & -2 & 5 \end{bmatrix} = \begin{bmatrix} 2\times 1 & 2\times 8 & 2\times -3 \\ 2\times 4 & 2\times -2 & 2\times 5 \end{bmatrix} = \begin{bmatrix} 2 & 16 & -6 \\ 8 & -4 & 10 \end{bmatrix}$$

Súčinom matíc A(M,N) x B(N,P) rozumieme maticu C(M,P), kde C[i,j] prvok je súčet súčinov I-teho riadku matice A a J-teho stĺpca matice B.

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} (1\times 3+0\times 2+2\times 1) & (1\times 1+0\times 1+2\times 0) \\ (-1\times 3+3\times 2+1\times 1) & (-1\times 1+3\times 1+1\times 0) \end{bmatrix}$$

Elimináciou rozumieme odstránenie matice jej vyriešením (resp. vyriešenie sústavy rovníc), je mnoho spôsobov ako vykonať túto operáciu. Pričom výsledok môže byť rôzny, t.j. jedno riešenie, viacej riešení, nekonečno riešení, alebo riešenie neexistuje.

Gaussova eliminačná metóda:

1. získanie trojuholníkového tvaru (priamy chod Gaussovej metódy):
    a. ideálne je mať v prvom riadku prvé číslo 1, preto najprv celý prvý riadok prenásobíme vhodnou konštantou tak, aby sme dostali požadované číslo.
    b. vhodný násobok danej rovnice pričítame k ostatným rovniciam pod ňou tak aby sme vo všetkých dostali všade pod naším číslom 1 v rovniciach číslo 0
    c. posuniem sa v diagonále na nasledujúcu rovnicu, nastavím sa na číslo nasledujúce za nulou. A teraz pre toto číslo vykonám operácie "a" a "b". Takto pokračujem až prídem na koniec, kde sa už ďalej nedá takto upravovať, v tomto momente by som sa mal mať tvar rovnice:
        i. jedna premenná a hodnota na pravej strane (1 riešenie)
        ii. viacero premenných a hodnota na pravej strane (viacero riešení)
        iii. rovnosť (nekonečne veľa riešení)
        iv. neplatná rovnosť (neexistuje riešenie)
2. pokiaľ máme i, môžeme pokračovať v riešení matice, tzv. spätný chod Gaussovej eliminačnej metódy. Dosadíme získanú hodnotu premennej do rovnice hore. V tej hornej rovnici tým pádom opäť získam premennú a hodnotu. Dostávam ďalší koreň a spätný chod opakujem, až kým nedôjdem ku poslednej (rovnica úplne hore) rovnici. V tomto momente by sme mali mať maticu vyriešenú.

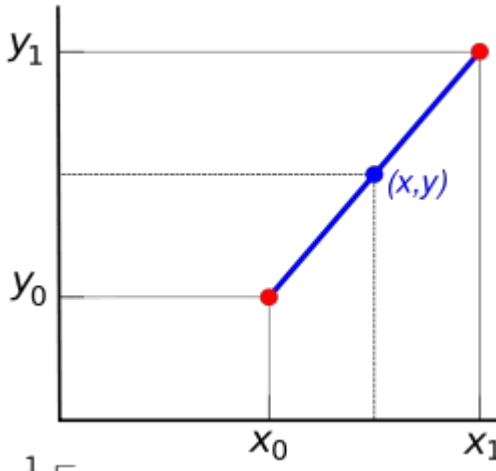
OK, super, 5 kreditov

57. Lineárna interpolácia. Metóda intervalov, regula falsi, Newtonova iterácia. Hornerova schéma.

Lineárna interpolácia

Najjednoduchšia metóda interpolácie (výpočtu nového bodu medzi dvomi existujúcimi bodmi) je lineárna interpolácia.

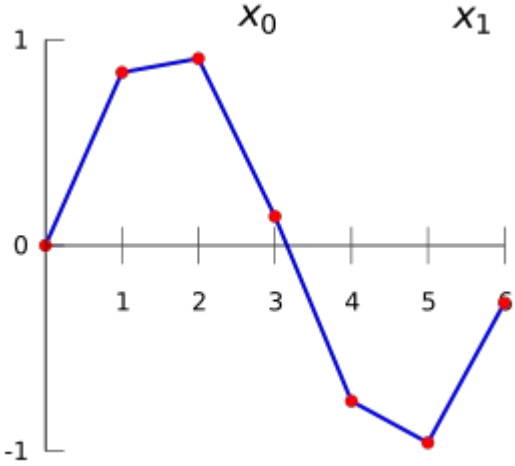
Pokiaľ sú dané dva body $[x_0,y_0]$ a $[x_1,y_1]$ lineárny interpolant je spojnica týchto dvoch bodov.



Na obrázku vpravo je lineárny interpolant modrá úsečka medzi bodmi $[x_0,y_0]$, $[x_1,y_1]$ a hodnotu súradnice y bodu je možné vypočítať lineárnov interpoláciou dosadením súradnice x.
Vzorec lineárnej interpolácie:

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}$$

[x,y]



Lineárnu interpoláciu je rovnako možné využiť pri množine bodov, kde jej výsledkom je viacero úsečiek spájajúcich dvojice bodov.

Metóda intervalov

Využíva sa pri riešení rovníc s absolútnou hodnotou. Postupujeme rozdelením rovnice na čiastkové lineárne rovnice bez absolútnej hodnoty, ktoré sú platné na určitých intervaloch vyhranených nulovými bodmi.

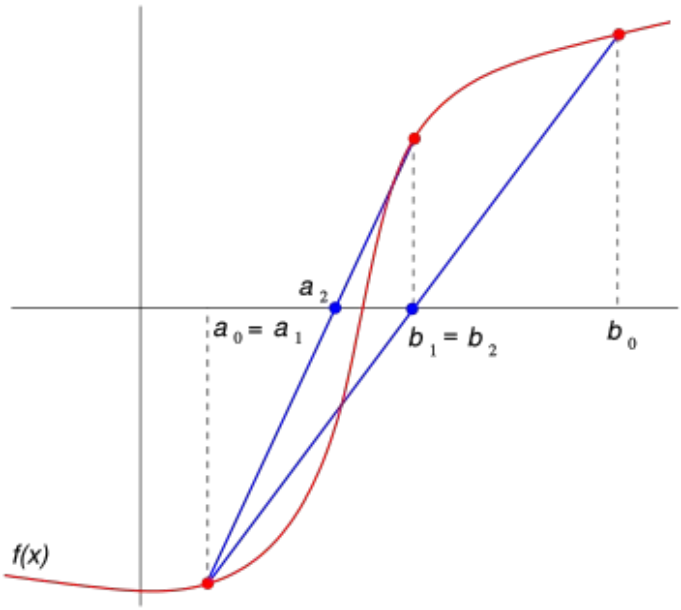
Príklad: |x-2| + |2x+7| = 6

Nulové body:

1. x – 2 = 0 {2}
2. 2x + 7 = 0 {-7/2}

Z toho dostávame intervaly (-∞, –7/2), <-7/2, 2>, (2, ∞) a riešime rovnicu zvlášť pre každý interval s prislúchajúcou zmenou znamienok.

Regula Falsi



Metóda hladanie koreňa nelinéarnych rovníc využívajúca poznatok, že pokiaľ vezmeme interval (a_0,b_0) a pre a_0 b_0 platí, že $f(a_0)$ má opačné znamienko ako $f(b_0)$ tak podľa teorému strednej hodnoty vyplýva z tvrdenia, že koreň hľadanej nerovnice sa nachádza v intervale (a_0,b_0).

Zmenšovaním tohto intervalu podľa určitých pravidiel (na obrázku intervaly (a_1,b_1) a (a_2,b_2)) prichádzame ku konečnej aproximácii koreňa rovnice.

Newtonova iterácia

Používaná pri hľadaní koreňa nelineárnych rovníc
Niekedy sa tejto metóde hovorí aj metóda dotyčníc, pretože uvedená metóda využíva na nájdenie koreňa nelineárnej rovnice dotyčnicu.

Princíp:

1. nájdenie dotyčnice k funkcii f v bode T
2. nájdenie koreňa tejto dotyčnice - f´(x) = 0
3. nájdenie nového bodu T (dosadením x z f´(x) = 0 do pôvodnej f)
4. opakovanie 1-3 pokiaľ rozdiel medzi starým a novým T nedosiahne požadovanú presnosť
5. T je koreňom rovnice

Hornerova schéma

Používaná pri výpočte polynómov x-tého stupňa. Pomocou tejto metódy je možné výpočet maximálne zjednodušiť bez nutnosti opakovane umocňovať premennú.

Princíp: vynímaním čo najviac rozložiť polynóm
Príklad:
Daný polynóm $p(x) = 6x^4 + x^3 + 2x^2 + x + 5$
Postupným vynímaním dostaneme:

1. $(6x^3 + x^2 + 2x + 1)x + 5$
2. $((6x^2 + x + 2)x + 1)x + 5$
3. $(((6x + 1)x+2)x+1)x+ 5$
4. $((((6)x+1)x+2)x+1)x+ 5$

Dostávame p(x) = ((((6)x+1)x+2)x+1)x+ 5

Z toho koeficienty polynómu p(x): $a_4 = 6$, $a_3 = 1$, $a_2 = 2$, $a_1 = 1$, $a_0 = 5$

Algoritmus:
Najprv koeficientmi naplníme pole aby index v poli bol rovný stupňu čiže [4] = 6 a [0] = 5. Následne v cykle prechádzame poľom od najvyššieho prvku po 0 a podľa vzorca
N = x * N + a[i]. Počiatočné N = 0. x je hľadaná hodnota a a[i] je i-ty prvok poľa a naplneného koeficientov.

Výpočet pre x = 2 bude vyzerať nasledovne:
N = 0; i = <4;0>

- N = x * N + a[4] > N = 2 * 0 + 6 > N = 6
- N = x * N + a[3] > N = 2 * 6 + 1 > N = 13
- N = x * N + a[2] > N = 2 * 13 + 2 > N = 28
- N = x * N + a[1] > N = 2 * 28 + 1 > N = 57
- N = x * N + a[0] > N = 2 * 57 + 5 > N = 119

Výsledok: **119**
Geniálne a zároveň sa hnevám ☺ 4 kredity
Hnevám sa, lebo metóda, ktorej sme venovali najviac času a predpokladám jej prezentáciu na maturitách je rozpracovaná najmenej. Predpokladám, že spolužiaci nie sú plne schopní odôvodniť iné (tu popísané) algoritmy na základe tých pár riadkov, čo si prečítajú. Poprosím doplniť taký krásny obrázok a vysvetlenie aj pre Newtonovu metódu ☺

58. Architektúra procesora I8086, zbernice, reprezentácia dát v pamäti. Systém prerušení, spracovanie prerušenia, spracovanie resetu PC.
Procesor Intel 8086 je prvým procesorom x86 architektúry, ktorú Intel uviedol na trh 1978. Všetky registre, interné a externé dátové zbernice tohto procesora sú 16 bitové, ustanovujúc tak 16 bitový mikroprocesor.

Základné časti

1. Register inštrukcií
2. Aritmetická a logická jednotka
3. Registre
4. Dekódovanie inštrukcií a riadenie procesora
5. Obsluha zberníc

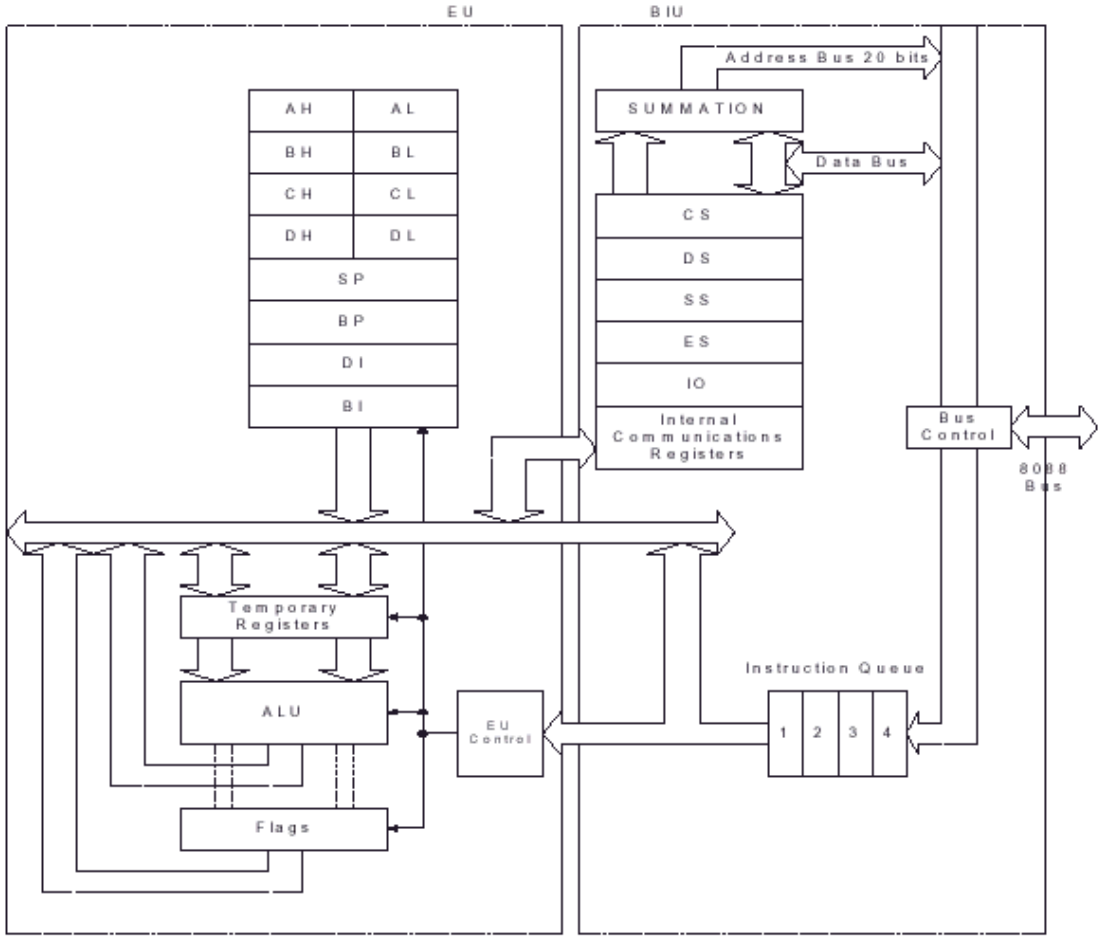
Zbernice:

- Dátová zbernica
- Adresová zbernica
- Zbernica inštrukcií

Procesor je celkovo rozdelený na dve časti:

- **EU (Execution Unit)**
- **BIU (Bus Interface Unit)**

BIU sprostredkúvava viacero činností – správa pamäte, správa zberníc.
EU sa stará o samostatné spracovanie inštrukcií a dát, ktoré dostáva od BIU. Spracováva tieto dáta a ukladá ich do všeobecných registrov. Vrátením dát späť do BUI môžu byť dáta uložené znova do pamäti alebo odoslané na výstup. EU nemá žiadne spojenie s vonkajšími zbernicami, je prepojená len s BIU.



Registre

Univerzálne registre

- ***AX*** - hlavný register (akumulátor) používaný pri aritmetických a logických operáciách na uloženie jedného z operandov, poprípade aj výsledku.
- ***BX*** - (base) register sa používa ako ukazovateľ prístupu k rôznym blokom údajov a tabuľkám v pamäti.
- ***CX*** - (count) register sa používa ako počítadlo prechodov cez slučku, resp. počítadlo opakovaní prenosu dát, alebo ako počítadlo počtu iterácii.
- ***DX*** - (data) register používaný pri ukladaní dát a tiež špeciálne, napríklad pri delení sa do neho ukladá zvyšok delenia operandu uloženého v AX, alebo sa používa tiež na uloženie adresy I/O zariadenia.

Segmentové registre

CS, DS, ES, SS, pomocou ktorých možno v rôznych oblastiach pamäte lokalizovať oblasť pamäte pre dáta, zásobník, vlastný kód programu a pod.

- ***CS*** - (Code) segment, ktorý určuje oblasť pamäti, kde sú uložené inštrukcie alebo kód programu, ktorý sa má aktuálne vykonať.
- ***DS, ES -*** (Data , Extra) segmenty, pomocou ktorých sa adresujú dáta (premenné) používané v programe. ES nemá presné implicitné určenie.
- ***SS*** - (Stack) zásobník, fungujúci ako pamäť typu LIFO. Registre SS:SP udávajú dohromady adresu, na ktorej sa v pamäti nachádza zásobník.

Reprezentácia dát v pamäti

Mikroprocesor 8086 pracuje s 20-bitovou adresou v hlavnej pamäti. Hlavná pamäť je slabiková t.j. CPU adresuje až 1 MB, čo sa rovná 2^{20}.

Architektúra 8086 vyžaduje rozdelenie hlavnej pamäti na dva segmenty. *Segment* časť pamäti do 64 kB a jeho začiatočná adresa musí byť násobkom 16, t.j. segmentov môže byť ľubovoľný počet. Môžu susediť, prekrývať sa, byť oddelené a z toho vyplýva, že segmentácia je iba *logické delenie pamäti.*

Prerušenia

V princípe sú dve základné úrovne prerušenia:

a) **na úrovni OS** – napr. požiadavka na tlač
b) **na úrovni aplikácie** – delenie nulou v Pascale

OS sa dokáže s niektorými chybami vysporiadať. Existuje 256 chybových hlásení a následne 256 vektorov prerušenia, ktoré odkazujú na adresu programu, ktorý danú chybu ďalej rieši. Pri chybovom hlásení sa uloží aktuálna adresa, kde zastal program, v tvare CS:IP (PWD - program status work).

Spracovanie resetu

Mäkký reset – signál spôsobí, že sa 8086 dostane do predefinovaného stavu – sú vymazané registre a efektívna programová adresa CS:IP je nastavená na F000:0FFF

Tvrdý reset – odstavenie prúdu, dôjde k vymazaniu obsahu ram. PC naštartuje z BIOS-u z adresy FFFF:0000 (CS:IP).

Hm, prečo komplikovať spolužiakom život? Neboli jednoduché obrázky na hodinách jasnejšie ako komplexný obrázok z prvej strany? Nepredpokladám, že nájdu bez dlhšieho študovania zhodu medzi učivom z hodín a týmto materiálom. 4 kredity

- **Registre a inštrukčný súbor I8086, zbernice, inštrukcie presunov, skokov, aritmetické inštrukcie, inštrukčný cyklus, popis jeho fáz, druhy adresácie. Rozdelenie inštrukcií do skupín, popis ich významu a činnosti.**

Procesor I8086 poskytuje 14 registrov. Sú rozdelené do troch hlavných skupín.

Datové: 16 bitové, pričom je možné adresovať samostatne aj vrchný byte(AH) alebo dolný byte(AL)
- AX – akumulátor – vhodný na aritmetické operácie
- BX – base index – v niektorých adresných módoch slúži na výpočet adresy
- CX – counter – čítač pri intrukciách cyklu (loop)
- DX – nemá špeciálne určenie

Zásobníkové:
- SP – stack pointer – ukazuje na vrch zásobníku
- BP – base pointer – adresuje data v hlavnej pamäti
- SI, DI – source index, destination index – používajú sa pri presunoch dát

Segmentové:
- CS – code – obsahuje číslo segmentu práve vykonávaného kódu
- DS – data – obsahuje segment s dátami
- SS – stack – obsahuje segment zásobníka, adresa zásobníka je SS:SP
- ES – extra – využíva sa pri presune dát, napr. Inštrukcia MOVS, ES:SI -> ES:DI

Zvyšné 2 registre sú

IP alebo tiež PC – instruction pointer (program counter) – obsahuje adresu práve vykonávanej inštrukcie, jeho obsah sa dá meniť inštrukciami vetvenia

F – flags – obsahuje 9 príznakov ktoré určujú stav procesora
CF – carry – 1 ak pri aritmetickej alebo logickej operácii dôjde k prenosu z najvyššieho bitu

PF – parity - nastavený, ak je parita výsledku aritmetickej operácie párna (párny počet núl alebo l)

AF - auxiliary carry - nastavený, ak pri aritmetickej operácii dojde k prenosu medzi nižšími a vyššími 4 bitmi

ZF – zero – nastavený, akje výsledok aritmetickej operácie O

SF – signum - znamienkový bit, je nastavený zhodne s najvyšším bitom výsledku aritmetickej operácie

OF – overflow - indikátor pretečenia pri aritmetických operáciách

DF - direction - určuje smer posunu ukazateľov pri presunoch dát, napr. MOVS ak je O vzostupne, akje l zostupne

IF – interrupt - maska prerušenia, ak je l, prerušenia sú povolené

TF - trace - indikátor trasovania, ak je nastavený, po inštrukcii sa vykoná prerušenie

Inštrukčný cyklus

Každá strojová inštrukcia je predstavovaná postupnosťou bytov, dĺžka inštrukcie býva od 1 do 6 byte. Vykonávanie inštrukcií prebieha nasledovne:

1/ mikroprocesor si prečíta ďalšiu inštrukciu programu z pamäte, jej adresa sa nachádza v CS:PC.

2/ zistí typ operácie a podľa toho odvodí jej d žku a tvar operandov

3/ na základe uvedených zistení vyberie z pamäte alebo registrov potrebné údaje a vykoná s nimi požadovanú operáciu

Tejto opakovane vykonávanej činnosti hovoríme inštrukčný cyklus.

Adresovanie v JSA - Adresovacie módy

Väčšina inštrukcií môže pracovať rovnako s údajmi v pamäti alebo v registroch ako aj s priamymi hodnotami. Aké sú možné spôsoby a ako sa k dátam dostať?

a/ priama hodnota sa zapisuje ako číslo alebo konštanta, môže byť použitá 8 alebo 16 bitová hodnota. Register sa uvádza svojim názvom. Je možné použiť ľubovoľný 8 i 16 bitový register okrem IP a SW (Status Word).

b/ ostatné adresové módy sa týkajú práce s pamäťou, okrem dlhej priamej adresy, špecifikujú všetky iba offset adresy dát, segmentová časť adresy sa berie z príslušného segmentového registra. Aby sa adresa odlíšila od priamej hodnoty, píše sa celá, alebo jej časť do hranatej zátvorky. Pri výpočte adresy je možné použiť konštantu a registre BX, BP, SI, DI.

Najobecnejšie možno adresu napísať takto:

c[Bx+xI], kde c je konštanta,

Bx je jeden z registrov BX, BP,

xI je jeden z registrov SI, DI.

Táto adresa je súčtom konštanty a obsahu dvoch registrov (ľubovoľnú časť je možné vynechať).

Prehľad adresovacích módov pre dáta v pamäti:

Názov	**Príklad**	**Použité registre**
Priama adresa	[113]	žiadne
Nepriama adresa	[BX]	BX, BP, SI, DI
Nepriama adresa s posunom	22[BX]	BX, BP, SI, DI
Nepriama adresa s bázou	[BX+SI]	BX+SI, BX+DI
a indexom		BP+SI, BP+DI
Nepriama adresa s bázou,	13[BX+DI]	BX+SI, BX+DI
indexom a posunom		BP+SI, BP+DI

Inštrukcie presunu.

MOV - silná skupina inštrukcií s rozsiahlymi možnosťami. Obmedzením je, že nie možné presúvať dáta v rámci pamäte z miesta na miesto, čo rieši osobitná inštrukcia MOVS.

MOV umožňuje prenos dát medzi segmentovými registrami navzájom. MOVS D,S - prenáša dáta z ES:DI -> na DS:SI. Tieto je potrebné naplniť ešte pred inštrukciou.

LODS - slúži na prenos z pamäte do akumulátora (registra AX)

STOS - prenáša dáta z akumulátora do pamäte, operand len formálne určuje šírku prenášaných údajov.

K práci so zásobníkom sú určené nasledovné jednooperandové inštrukcie:
PUSH - spôsobí zápis na zásobník
POP - čítanie zo zásobníka
PUSHF, POPF - na zápis a čítanie obsahu stavového slova na zásobníku
LDS, LES - na presun 32 bitovej adresy (segment:offset) z RAM do registra
XCHG - určená k výmene dát medzi registrami alebo pamäťou a registrom

Inštrukcie skoku.

Sú to vlastne špeciálne prípady inštrukcií presunu. Určitá hodnota sa presunie do PC, prípadne i do CS. Nepodmieneným skokom je len JMP.
Podmienené skoky sa vykonávajú podľa nastavenia indikátorov v stavovom slove. V procesore I8086 sú možné "skoky na krátku vzdialenosť", čo je 127 byte vpred alebo 127 byte vzad. Rôznych podmienených skokov je 16, je možnosť zápisu 30 rôznymi mnemotechnickými zápismi.

Skoky pri aritmetickom porovnávaní bez znamienka sú:

JA - Jump Above A > B
JNA - Jump Not Above A <= B
JB - Jump Below A < B
JNB - Jump Not Below A >= B
JAE - Jump Above or Equal A >= B
JNAE - Jump Not Above or Equal A < B
JNB - Jump Not Below A >= B
JNBE - Jump Not Below or Equal A > B so znamienkom:
JG - Jump Greater A > B
JNG - Jump Not Greater A <= B
JGE - Jump Greater or Equal A >=B
JNGE - Jump Not Greater or Equal A < B
JL - Jump Less A < B
JNL - Jump Not Less A >= B
JLE - Jump Less or Equal A <= B
JNLE - Jump Not Less or Equal A > B

ďalšie podmienené skoky:

JNO - Jump Not Overflow ak indikátor F=0 -> nedošlo k pretečeniu
JNP - Jump Not Parity ak je nepárna parita
JNS - Jump Not Sign ak sledovaná hodnota je >=0
JNZ - Jump Not Zero ak A <> B
JO - Jump Overflow ak došlo k aritmetickému pretečeniu
JP, JS, JZ s negovaným významom
JCXZ - Jump if CX Zero ak je CX = 0
LOOP - Loop if CX Not Zero ak CX<>0 tak CX:=CX-1

Aritmetické inštrukcie.

ADD - sčítanie dvoch operandov
INC - pričítanie hodnoty 1
SUB - inštrukcia odčitovania
DEC - odčítanie hodnoty 1
NEG - zmena znamienka
CMP - porovnanie (vedie k nastaveniu indikátorov)
MUL - násobenie bez znamienka
IMUL - násobenie so znamienkom
DIV - delenie bez znamienka
IDIV - delenie so znamienkom

Najčastejšie používane inštrukcie by sa dali rozdeliť na inštrukcie:

presunu: MOV - presun
XCHG - vymena
PU SH - uloženie na zásobník
POP - vyber zo zásobníku

aritmetické: ADD - pričítaj
SUB - odčítaj
MUL - násobenie
DIV - delenie
INC - pričítaj jedna
OEC - odčítaj jedna
logicke: AND - logicky súčin
OR - logicky súčet
XOR - logicka non-ekvivalencia
NOT - negácia
porovnanie: CMP - porovnanie
TEST - bitove porovnanie
posuvy: SHR - log. posun vpravo
SHL - log. posun vľavo
rotacie: RCR - rotácia cez príznak prenosu (carry) vpravo
RCL - rotácia cez príznak prenosu (carry) vlavo
skokov: JMP - nepodmieneny skok
LOOP - skoč kým nie je (E)CX nula
JZ - skok ak je nastavený ZF (zero flag)
JC - skok ak je nastavený CF (carry flag)
JNZ - skok ak nie je nastaveny priznak nuly ZF JNC - skok ak nie je nastavený príznak prenosu CF
podprogramov: CALL - zavolaj podprogram
RET - návrat z podprogramu
INT - vyvolaj prerušenie

OK, 5 kreditov

- **Konečné a nedeterministické automaty, vysvetlenie vzťahu medzi jazykmi a automatmi. Kontextové a bezkontextové gramatiky, Turingov stroj.**

Abeceda je ľubovoľná neprázdna konečná množina, ktorej prvky nazývame **písmená**. Konečné postupnosti písmen nazývame slová. Prázdna postupnosť písmen sa nazýva prázdne slovo a označujeme ju ε. Dĺžka slova w=a1....an je počet písmen v slove w=n. Ak u= a1....an a v= b1....bm tak u.v= a1....anb1....bm je zreťazenie slov.
Množinu všetkých slov v abecede Σ označujeme Σ^*. Jazykom v abecede Σ rozumieme ľubovoľnú množinu slov v abecede Σ t.j. ľubovoľnú podmnožinu množiny Σ^*.

Jazyk definovaný gramatikou G je množina L(G) (t.j. jazyk nad gramatikou G) všetkých vetných foriem v terminálovej abecede.
Základné operácie s jazykmi sú: zjednotenie, prienik, rozdiel, doplnok, zreťazenie.
Jednou z možností, ako opísať jazyk, je udať spôsob, ktorým sa dajú vytvoriť, vygenerovať, všetky slová tohto jazyka , t.j. udať gramatiku.

2. Gramatiky

DEF: Gramatika je štvorica G=(N, T, P, δ), kde N, T - sú abecedy neterminálnych, terminálnych symbolov, P je konečná množina pravidiel a δ je počiatočný symbol.

Gramatika je

regulárna, ak sú všetky jej pravidlá tvaru $A \rightarrow aB$ alebo $A \rightarrow b$, kde $A,B \in N$, $a,b \in T$.

bezkontextová, ak sú všetky jej pravidlá tvaru $A \rightarrow b$, kde $A \in N$, $b \in T$.

kontextová, ak pre každé pravidlo $u \rightarrow$ platí |u| <= |v| , t.j. dĺžka slova u je menšia alebo rovná v

neobmedzená, ak na pravidlá nie sú kladené žiadne požiadavky

Potom poznáme triedy jazykov regulárnych, bezkontextových, kontextových, neobmedzených. Druhým spôsobom na "opísanie" vlastností nového jazyka je navrhnúť (udať) zariadenie, ktoré by bolo schopné rozhodovať o slovách, či patria (nepatria) do daného jazyka. V ďalšej časti sa budeme zaoberať takýmito zariadeniami.

3. Konečné automaty - KA

DEF: Deterministický konečný automat je pätica A= (K, Σ, δ, q0, F), kde K je konečná množina stavov, Σ je vstupná abeceda, δ je prechodová funkcia $\delta: K \times \Sigma \Rightarrow K$, q0 je počiatočný stav, $F \subseteq K$ je množina koncových stavov.

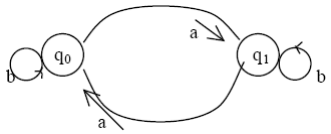
Konečný automat si môžeme predstavovať ako zariadenie s konečnostavovou riadiacou jednotkou a vstupnou páskou, na ktorej je jedna čítacia hlava a tá sa môže posúvať len zľava doprava. Zariadenie pracuje v taktoch, pričom sa v každom takte (kroku) na základe stavu riadiacej jednotky a čítaného symbolu zmení stav riadiacej jednotky a pohne čítacou hlavou doprava na ďalšie písmeno vstupného slova.

DEF: Konfigurácia konečného automatu A je dvojica (q, w), kde $q \in K$, $w \in \Sigma$. Skutočnosť, že automat je v konfigurácii (q, w) znamená, že automat je v stave q a w je neprečítaná časť slova, pričom hlava je na prvom písmene zostávajúceho slova.

DEF: Krok KA je relácia $(q, aw) \Rightarrow (p, w)$, ak d $(q, a)=p$, kde $a \in \Sigma$, $w \in \Sigma^*$.
DEF: Jazyk rozpoznávaný KA je množina $L(A)= \{ w \mid (q0, w) \Rightarrow (q, \varepsilon)\ q \in F\}$.
Postupnosť konfigurácií nazývame výpočet automatu na slove w.
Prechodové funkcie je možné zadať ešte pomocou:
1/ Prechodovej tabuľky

δ	a	b
q_0	q_1	q_0
q_1	q_0	q_1

2/ Prechodového diagramu



Výpočet automatu pre slovo aababb je: (q0 , aababb) ⇒ (q1 , ababb) ⇒ (q0 ,babb) ⇒ (q0 ,abb) ⇒ (q1 , bb) ⇒ (q1 , b) ⇒ (q1 , e) a keďže q1 je povolený koncový stav, slovo aababb je akceptované.

Nedeterministické konečné automaty
Definícia je skoro rovnaká, rozdiel je v prechodovej funkcii. Táto neurčuje nasledujúci stav jednoznačne, ale udá množinu stavov, do ktorých sa môže automat dostať. Zmena je aj v tom (vzhľadom na argument ε-prázdne slovo), že automat môže zmeniť stav aj bez posunu hlavy (ak prečítal ε). Navyše sa podľa tejto definície môže dostať automat do stavu, keď nemôže vykonávať žiadnu činnosť (akoby zmrzol).
Čiže výpočet končí vtedy**,** keď existuje aspoň jeden výpočet na slove w, zabezpečujúci koncový stav.
Na to, aby v nedeterministickom KA slovo **w nepatrilo do L(A),** nesmie existovať výpočet na **w**, alebo všetky výpočty musia končiť v stavoch rôznych od koncových!!!
DEF: Nedeterministický zásobníkový automat je sedmica A= (K, Σ, Γ, δ, q0, Z0, F), kde K je konečná množina stavov, Σ je vstupná abeceda, Γ je zásobníková abeceda, δ je prechodová funkcia $\delta: K \times (\Sigma \cup \varepsilon) \times \Gamma \Rightarrow 2^K$, q0 je počiatočný stav, $Z0 \in \Gamma$ je začiatočný symbol v zásobníku , $F \subseteq K$ je množina koncových stavov.
Zásobníkový automat si môžeme predstaviť ako konečný automat, ktorý má k dispozícii pomocnú pamäť vo forme zásobníka (t.j. vždy vie, čo si naposledy uchoval). Pozor, hovoríme o nedeterministickom automate.
DEF: Konfigurácia zásobníkového automatu je trojica (q, w, γ), kde $q \in K$, $w \in \Sigma^*$, $g \in \Gamma^*$. Konfigurácia reprezentuje skutočnosť, že automat je v stave q, zostatok vstupu je w a obsah zásobníka je γ. Čítacia hlava je na prvom písmenku slova w a dostupný symbol v zásobníku je posledný symbol γ.

4/ Turingove stroje a lineárne ohraničené automaty.
Na konečné a zásobníkové automaty sa môžeme pozerať ako na modely reálnych počítačov. Sú však napriek výhodám pre bežné použitie dosť obmedzujúce. Nasledujúci typ automatu, ktorý podľa všeobecne uznávanej hypotézy (Turingova teória) dokáže to, čo skutočný počítač sa nazýva Turingov stroj. Môžeme si ho predstaviť ako konečný automat, ktorý môže svojou hlavicou nielen čítať, ale aj zapisovať a vie ňou pohybovať v dvoch smeroch. Okrem toho môže vstupnú pásku podľa potreby ľubovoľne predlžovať smerom doprava.

Def: Turingov stroj je šestica A= (K, Σ, Γ, δ, q0, F), kde K je konečná množina stavov, Σ je vstupná abeceda, Γ je abeceda páskových symbolov, δ je prechodová funkcia $\delta: K \times \Gamma \Rightarrow K \times (\Gamma - \{B\}) \times \{-1,1\}$, kde B je prázdny symbol, q0 je počiatočný stav, $F \subseteq K$ je množina koncových stavov.

Konfigurácia Turingovho stroja je trojica (q, w, i), kde **q** ∈ K, **w** ∈ x (Γ-{B}) a **i**>=1 je prirodzené číslo. Konfigurácia reprezentuje skutočnosť, že Turingov stroj je v stave q, neprázdna časť pásky je w a hlavica je práve na i-tom symbole slova w. Pri svojej práci bude Turingov stroj prechádzať z konfigurácie do konfigurácie na základe prechodovej funkcie.
Pre každý algoritmus existuje Turingov stroj!!!

Skúsme sa zamyslieť nad postupom práce Turingovho stroja.
Označí si najľavejší výskyt písmena **a** symbolom **X**. Potom presunie hlavicu doprava cez všetky znaky **a** a **Y** a označí si prvý výskyt znaku **b** symbolom **Y**. V ďalšej fáze posunie hlavicu doľava nad najľavejšie **a** a činnosť opakuje. V prípade, že už nezostali **a**, skontroluje písmena **b** a akceptuje slovo.
Pravdepodobne si viete predstaviť predchádzajúce príklady rozkreslené ako grafy (stromy), kde sa skúša, ktorá vetva (konár) nás privedie k predpokladanému cieľu (akceptácia slova). Toto skúšanie má tiež zabehnuté spôsoby a jeden z nich je skúmanie od najľavejších vetiev.
OK, 5 kreditov