Manuál k počítači Didaktik M

D I D A K T I K M

Užívateľská príručka

(c) 1990
v. d. DIDAKTIK SKALICA
Nálepkova 22, 909 01 SKALICA

Tato uživatelská príručka byla upravena z originálu Tomáše Holčapka (MIRACLESOFT) během října až listopadu 2008. neručí za správnost přepisu a schémat.

Obsah

Prílohy

Úvod

Užívateľská príručka mikropočítača Didaktik M je rozdelená na dve časti.

V prvej časti sú popísané základné časti počítačovej zostavy a hlavný dôraz je položený na popis jazyka BASIC. Táto časť je určená hlavne pre začiatočníkov.

V druhej časti sú vysvetlené niektoré pojmy (číselné sústavy) a ďalšie podrobnosti, týkajúce sa práce operačného systému (mapa pamäti, systémové premenné, ...).

Ak nebudete pri čítaní príručky niektorému pojmu rozumieť, pozrite sa do prílohy "Vysvetlivky niektorých pojmov", kde sú vysvetlené základné pojmy z oblasti výpočtovej techniky. Všetky prílohy sú umiestené na konci príručky.

D I D A K T I K M

Užívateľská príručka

1. Prvá časť

1.1. Didaktik M sa predstavuje

Didaktik M je osobný osembitový mikropočítač, konštrukčne odvodený od mikropočítača Sinclair ZX Spectrum.

Je určený predovšetkých pre tých, ktorí sa chcú s počítačom hrať. Ďalej pre tých, ktorí sa chcú učiť základy programovania v jazyku BASIC, základy programovania v iných programovacích jazykoch a strojový kód mikroprocesora Z80. Teda všetkých tým, ktorí cítia, že by mohli v počítači nájsť nielen zábavného spoločníka a súpera v neprebernom množstve hier, ale i pomocníka pri vzdelávaní a pri prenikaní do tajov výpočtovej techniky.

Iste si teraz položíte otázku: "To je všetko pekné, ale aké programy sa pre môj počítač hodia?". Odpoveď je jednoduchá. Všetky, ktoré boli a budú vytvorené pre počítač Sinclair ZX Spectrum, pre počítač, ktorý patrí u nás medzi najrozšírenejšie.

Paleta programov je pomerne široká. Veľmi náročné hry, ktoré od hráča vyžadujú nielen dobrú pamäť, ale napríklad I trojrozmernú predstavivosť a logické myslenie. Letové a automobilové simulátory umožňujúce získať cvik a návyky pri radenie týchto prostriedkov. Šachové programy s rôznymi stupňami obťažnosti, často ťažko poraziteľné. Textové editory, slúžiace na písanie a archiváciu písomností. Výukové programy, napr. pre angličtinu, nemčinu, astronómiu, zemepis, hudobnú výchovu atď.. Rôzne programovacie jazyky, programy pre rádioamatérov a začínajúcich elektrotechnikov.

A prichádza na rad druhá otázka. "Kde vziať tieto programy?". Prístup je možný nasledovnými spôsobmi:

Prvý je možnosť zakúpenia programov v sieti maloobchodných predajní, ktorých ponuka postupne rastie.

Druhý je ponuka niektorých zväzarmovských organizácií a súkromných podnikateľov, ktorí takéto programy vytvárajú a predávajú.

Tretí je prostredníctvom klubov združujúcich užívateľov mikropočítačov ZX Spectrum (a s nim kompatibilných typov. V týchto kluboch získate aj rady od skúsenejších kolegov.

Štvrtý spôsob je získanie programov od kamarátov a známych, ktorí vlastnia kompatibilný počítač.

Piaty priamo od výrobného družstva Didaktik Skalica, ktoré dodáva programy formou zásielkovej služby.

Šiesty spôsob je najťažší, ale pre intelektuálny rozvoj Vašej osobnosti najhodnotnejší: urobiť si program sám.

1.1.1. Technický popis počítača

Tretia otázka by mohla znieť: "Aké sú technické parametre počítača?".

Ako už bolo povedané, DIDAKTIK M je osobný osembitový počítač odvodený od počítača ZX Spectrum, s ktorým je kompatibilný (zlučiteľný) ako po technickej, tak po programovej stránke.

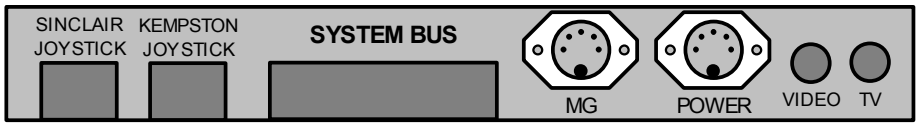
Všetky informácie o práci počítača musia byť nejakým spôsobom zobrazené. Na to slúži TV prijímač, popr. monitor. Farebné zobrazenie je v ôsmich farbách systému PAL a v dvoch jasových úrovniach. V prípade použitia čiernobieleho TV prijímača počítač zobrazuje namiesto ôsmich farieb osem odtieňov šedej farby. Farebné zobrazenie nie je možné docieliť na TV prijímači pracujúcom iba v norme SECAM.

Ďalej je potrebné isté informácie počítaču zadávať. Na to slúži 45 klávesová kontaktná klávesnica s vysokou životnosťou.

Prirodzenou požiadavkou užívateľa je, aby si uchoval program, popr. dáta zadané do počítača cez klávesnicu na ďalšie použitie. Didaktik M používa na úschovu dát obyčajný, najlepšie monofónny **magnetofón**.

K práci počítača je nutný **napájací zdroj**. Ten je umiestnený v samostatnej škatuľke a môže byť pripojený len na elektrickú sieť 220 V / 50 Hz.

Počítač je vybavený vlastným **interpreterom** jazyka BASIC umiestneným v pevnej pamäti ROM (pamäť len pre čítanie s veľkosťou 16 kB. Pamäť RAM (pamäť, do ktorej sa dá zapisovať i čítať má veľkosť 48 kB.



Na obrázku je znázornený zadný panel mikropočítača Didaktik M, ku ktorému sa pripájajú všetky prídavné zariadenia. Vysvetlíme si význam jednotlivých konektorov sprava doľava (pri pohľade zozadu):

TV – konektor určený pre pripojenie anténneho vstupu TV prijímača (príslušný kábel je súčasťou dodávky).

VIDEO – konektor určený pre pripojenie monitora alebo tzv. video vstupu TV prijímača (vstupu pre pripojenie videomagnetofónu). Keďže tento konektor je možno pripojiť k rôznym zobrazovacím zariadeniam s rôznymi typmi použitých konektorov, nie je možné dodávať univerzálny kábel. Záujemci, ktorí nevyžadujú farebné zobrazenie, si môžu objednať monochromatický monitor vo v. d. Didaktik Skalica.

POWER – sedemkolíkový kruhový konektor určený pre pripojenie napájacieho zdroja.

MG – päťkolíkový kruhový konektor určený pre pripojenie magnetofónu (magnetofónový kábel je súčasťou dodávky).

SYSTEM BUS – na tento tzv. systémový konektor je vyvedená väčšina signálov použitých v mikropočítači a je určený pre pripojenie periférnych zariadení. Konektor je odvodený od obdobného konektora na mikropočítači Sinclair ZX Spectrum a umožňuje pripojenie väčšiny zariadení určených pre spoluprácu s počítačom ZX Spectrum. Najbežnejším zariadením určeným na pripojenie k tomuto konektoru je tzv. interface, ktorý umožní pripojení tlačiarne, zapisovača, myši a ďalších periférnych zariadení. Interface s paralelným výstupom si môžete objednať vo v. d. Didaktik Skalica.

KEMPSON JOYSTICK – konektor určený pre pripojenie joysticku označeného DIDAKTIK M (výrobcom je Kovodružstvo Náchod). Pre použitie joysticku pripojeného na tento konektor je potrebné zvoliť v programe (resp. hre) mód využívajúci KEMPSTON JOYSTICK.

SINCLAIR JOYSTICK - platí to isté ako v predchádzajúcom iba s tým rozdielom, že v programe je potrebné zvoliť mód SINCLAIR JOYSTICK. Môžete používať ten istý joystick buď ako KEMPSTON alebo ako SINCLAIR; záleží iba na tom, ku ktorému konektoru ho pripojíte.

Je samozrejme možné pripojiť dva joysticky súčasne (ku každému konektoru jeden) a potom využívať hry pre dvoch hráčov tak, že jeden hráč má joystick KEMPSTON a druhý SINCLAIR.

To sú informácie o základnej zostave počítača. Ďalšie podrobnosti sú uvedené v druhej časti príručky v kapitole popisujúcej hardware (technické vybavenie počítača).

Ďalej sa budeme venovať tomu, ako po vybalení uviesť počítač do činnosti.

1.1.2. Uvedenia počítača do činnosti

Teraz pristúpime k vlastnému zapojeniu počítača a jeho uvedeniu do činnosti. Počítač je zabalený v obale, ktorého obsah je nasledujúci:

- vlastný počítač
- napájací zdroj
- úvodná magnetofónová kazeta
- kábel pre pripojenie k TV prijímaču
- kábel pre pripojenie k magnetofónu
- príručka
- záručný list

Po vybalení nechajte počítač asi 15 až 20 minút zahriať na izbovú teplotu. Najskôr prepojte **počítač s TV prijímačov** pomocou šnúry s obdĺžnikovými koncovkami s guľatou zdierkou takto: jednu koncovku zasuňte do konektora TV prijímača, v ktorej mávate anténu pre druhý program (teda anténu pre UHF pásmo, 21 – 60 kanál). Druhú koncovku zasuňte v počítači do konektora označeného "TV". Tým je toto prepojenie hotové.

Ďalej je treba **prepojiť počítač s magnetofónom**. Toto prepojenie však nie je nutné pre základné funkcie počítača. Magnetofón doporučujeme použiť monofónny; u stereofónnych magnetofónov

a walkmanov môže byť spolupráca počítača s týmito magnetofónmi horšia než s magnetofónem monofónnym. Súčasťou dodávky je kábel na pripojenie počítača k magnetofónu. Zasúva sa do konektora na počítači označeného MG. Druhý koniec kábla sa pripája k magnetofónu do konektora pre vstup a výstup. Magnetofón pripájajte vtedy, keď je počítač vypnutý!

Zapamätajte si, že pripájať k počítaču akékoľvek zariadenie je dovolené len pri odpojení napájacieho zdroja!!!

Nakoniec je treba **prepojiť zdroj s počítačom**. Zasuňte kábel vychádzajúci z jednej strany napájacieho zdroja (zo sedemkolíkovým konektorom) do konektora v počítači, nad ktorým je napísané "POWER". Druhý koniec vychádzajúci zo zdroja (je zakončený vidlicou) zasuňte do zásuvky na 220 V. Po zasunutí sa musí na počítači rozsvietiť dióda nad nápisom "POWER". Pokiaľ sa tak nestane, mohla sa prepáliť poistka v zdroji. Technicky zdatnejším doporučujeme v tomto prípade po odpojení od napájacej siete otvoriť napájací zdroj (4 skrutky v rohoch) a príslušnú poistku vymeniť. Menej zdatným doporučujeme zaslať počítač na naše servisné oddelenie.

Poslednou činnosťou, ktorú je treba urobiť, je **naladenie TV prijímača** na kanál, na ktorom pracuje počítač. Najskôr si zvoľte na svojom TV prijímači číslo predvolby a prepnite na rozsah kanálov 21 až 60. Teraz pomaly prelaďte celé pásmo. Približne na 48. kanále by sa Vám mal na obrazovke objaviť nejaký obrazec.

Teraz stlačte zároveň klávesy CAPS SHIFT a RESET. Po ich stlačení sa obrazovka zafarbí čierno, potom bielo a v spodnej časti sa objaví nápis "**© 1990 DIDAKTIK M BASIC**". Skúste stlačiť kláves označený nápisom ENTER. Po jeho stlačení nápis zmizne a v ľavom spodnom rohu sa objaví blikajúce písmeno K.

Súčasné stlačenie klávesov CAPS SHIFT a RESET budeme označovať ako **"reset počítača"** a musí byť vykonané vždy po zapnutí počítača do siete. Vykonaním resetu sa totiž počítač nastaví do základného pracovného stavu.

S týmto postupom uvedenie počítača do chodu iste vznikne otázka "Je to závada počítača, keď počítač nezačne pracovať ihneď po zapnutí do siete, ale je treba previesť reset počítača a až potom počítač začne pracovať?". Nie, nie je to závada, naopak, ako sme už uviedli, je to bežný postup pre uvedenie počítača do činnosti.

Teraz už máte počítač pripravený prijať Vaše prvé príkazy. Doporučujeme Vám najprv sa zoznámiť s úvodnou kazetou, ktorá je súčasťou dodávky počítača. Obsahuje programy, ktoré Vám ukážu možnosti Vášho počítača a uľahčia Vám prvé kroky pri práci s ním.

Pri **nahrávaní programu z kazety** postupujte nasledovne:

- vsuňte kazetu do magnetofónu
- pretočte ju na začiatok strany 1 (pokiaľ nie je).
- na počítači stlačte kláves J, v spodnej časti obrazovky sa vypíše slovo LOAD. Stlačte kláves SYMBOL SHIFT, držte ho stlačený a stlačte kláves P; na obrazovke sa vypíšu úvodzovky.

Tomuto postupu, keď ste najprv stlačili kláves SYMBOL SHIFT, držali ho stlačený a stlačili ste kláves P, budeme hovoriť, že ste stlačili SYMBOL SHIFT a P (SYMBOL SHIFT + P). Podobne výraz "stlačte klávesy CAPS SHIFT a 2" znamená, že najskôr stlačíte kláves CAPS SHIFT, podržíte ho stlačený a stlačíte kláves označený číslicou 2.

Opäť stlačte klávesy SYMBOL SHIFT a P, na obrazovke sa vypíšu druhé úvodzovky. Teraz stlačte kláves ENTER. Po jeho stlačení zmizne z obrazovky napísaný riadok.

Teraz spustíte magnetofón. Okraj obrazovky začne modro a červeno preblikávať. Asi po 10 až 20 sekundách sa ozve z počítača zvuk a na okraji obrazovky sa objavia červené a modré pruhy. Potom sa na obrazovke vypíše hlásenie "Program:uvod" a začnú sa po okrajoch striedať tenké modré a žlté pruhy. Po nahraní sa program sám odštartuje.

Pri nahrávaní môže dôjsť k niekoľkým chybám. Pred ich odstránením vykonajte reset počítača:

1) po spustení magnetofónu okraj obrazovky zostane biely a nepreblikáva červeno a modro. Vtedy je prepojovacia šnúra zasunutá v magnetofóne v zlej zdierke, alebo magnetofón nie je vôbec pripojený. Zasuňte šnúru do správnej zdierky a opakujte postup nahrávania.

2) obrazovka po spustení magnetofónu síce preblikáva, ale hlásenie "Program:uvod" sa na ňu nevypíše. V tomto prípade ide zrejme o znečistenú alebo zle nastavenú hlavu magnetofónu. Hlavu omyte čistým liehom a opakujte postup nahrávania. Pokiaľ ani po tomto zákroku nápis na obrazovke neobjaví, ide zrejme o zle nastavenú kolmosť hlavy magnetofónu. Čo to znamená?

   Na hlave magnetofónu je úzka štrbina, pomocou ktorej sa sníma záznam z kazety. Táto štrbina musí byť kolmá na pásku. Ak nie je, dochádza ku skresleniu záznamu a počítač ho nenahrá. Kolmosť hlavy sa nastavuje skrutkou, ktorá je umiestnená na jednom z koncov hlavy. Pri správne nastavenej hlave je zvuk záznamu vysoký, nezašumený s malým množstvom basov. Pre záznamy, vykonané na rôznych magnetofónoch, môže byť kolmosť hlavy vždy iná. Po nastavení kolmosti hlavy opakujte postup nahrávania. Pokiaľ sa Vám ani po týchto opatreniach nepodarí úvodný program nahrať, obráťte sa prosím na naše servisné oddelenie.

3) po zapnutí magnetofónu sa na obrazovke vypíše "Program:uvod" a program sa začne nahrávať. Počas nahrávania, alebo po jeho skončení sa však vypíše v spodnej časti obrazovky hlásenie:

   R Tape loading error, 0:1

   Toto hlásenie znamená, že došlo k chybe pri nahrávaní programu. Vtedy skúste niekoľkokrát zopakovať postup nahrávania.

Pokiaľ sa Vám ani po niekoľkých pokusoch nepodarí program do počítača nahrať, závada je rovnaká ako v bode 2, popr. je program na kazete zle nahraný. Pri jej odstraňovaní postupujte rovnako ako v bode 2.

Nahrávanie programu je náročná operácia, ktorá vyžaduje perfektný technický stav magnetofónu, počítača a záznamu na kazete. V prípade, že nemôžete nahrať do počítača záznam z kazety, doporučujeme zaslať počítač, magnetofón i kazetu s popisom závady na naše servisné oddelenie.

Po nahraní programu sa venujte nejaký čas tomu, čo Vám ponúka. Po jeho skončení sa začneme učiť komunikovať s Vaším počítačom.

1.1.3. Popis klávesnice

V tejto kapitole sa zoznámite s klávesnicou Vášho počítača a spôsobom, ako s ňou pracovať.

Už pri vybalení počítača ste si iste všimli, že sa klávesnica podstatne líši od klávesnice písacieho stroja, pretože je na nej okrem písmen ešte rad ďalších skratiek a symbolov.

Ďalej obsahuje v pravej spodnej časti štyri šípky (ich význam si vysvetlíme neskôr) a tlačidlo RESET, ktoré slúži na resetovanie počítača.

Zresetujte teraz počítač a stlačte kláves označený **ENTER**. Text v spodnej časti obrazovky zmizne a v ľavom spodnom rohu sa objaví blikajúce písmeno "**K**". Tým ste sa zoznámili s asi najdôležitejším klávesom, a tým je práve kláves ENTER. V preklade sa dal označiť ako kláves VSTUP a pre počítač znamená, že ste ukončili zadávanie príkazov na riadku a očakávate od neho spracovanie zadaného riadku.

V tomto prípade ste stlačením klávesu ENTER počítaču zadali prázdny riadok (riadok, na ktorom nie je nič zadané).

Blikajúce písmeno "**K**" budeme nazývať **kurzor** a označuje miesto v riadku, na ktoré práve zapisujete. Písmeno "**K**" označuje, že počítač očakáva vstup **kľúčového slova** jazyka BASIC, ktoré je napísané čiernou farbou na klávese vpravo dole. Kľúčové slovo je slovo s pevne definovaným významom v jazyku BASIC. Na klávese P je to napríklad slovo PRINT. Skúste tento kláves stlačiť. Na obrazovke sa vypíše PRINT a za ním blikajúce písmeno "**L**". Teraz tento riadok odošlite na spracovanie počítaču klávesom ENTER. Po jeho stlačení sa v spodnej časti obrazovky vypíše správa

0 OK, 0:1

Čísla v ňom nás zatiaľ nezaujímajú, dôležité je OK. Znamená to, že počítač Váš príkaz spracoval, nenašiel v ňom žiadnu chybu a vykonal ho. Kurzor "**K**" teda znamená, že počítač od Vás očakáva zadanie kľúčového slova. Zároveň Vám však nedovolí zadať nič iné, než kľúčové slovo (Vy ste stlačili kláves P, ale počítač vypísal PRINT). Kľúčové slovo od Vás počítač očakáva vždy na začiatku riadku.

Jediná výnimka je stlačenie klávesu, označeného číslicou. Skúste stlačiť kláves, označený číslicou 1. Na obrazovke sa vypíše číslica "1", ale za ňou opäť je možné za touto číslicou zadať len kľúčové slovo, alebo ďalšiu číslicu.

Odošlite riadok počítaču. Vypíše sa hlásenie

```
0 OK, 0:1
```

Stlačte kláves ENTER. Objaví sa blikajúci kurzor "**K**".

Iste ste si všimli, že po vypísaní kľúčového slova PRINT (stlačte kláves P) sa za ním vypísal kurzor "**L**". Tým Vám dáva počítač najavo, že teraz už neočakáva zadanie kľúčového slova, ale písmena, číslice, popr. špeciálneho symbolu (napr. +, !, = atď.). Blikajúce "**L**" je odvodené od anglického Letter – písmeno. Teraz môžete po kľúčovom slove PRINT zadať napr. znak "a" stlačením klávesu A. Skúste to. Na obrazovke máte teraz napísané:

```
PRINT a
```

Za písmenom "a" bliká kurzor **L**, teda môžete zadať ďalšie znaky, napr. "0". Zadajte teda nulu. Na obrazovke je napísané:

```
PRINT a0
```

Odošli tento riadok počítaču. V spodnej časti sa vypíše hlásenie

```
2 Variable not found, 0:1
```

Je to hlásenie o chybe. Hláseniami o chybe sa nebudeme v tejto kapitole zaoberať, zatiaľ si ich nevšímajte.

Zadajte znova PRINT. Za ním sa objaví kurzor **L**. V predchádzajúcom pokuse ste zadali znak "a" stlačením klávesu A. Vypísalo sa malé písmeno "a". Teraz však chcete napísať veľké písmeno "A". Cesta je dvojaká. Prvý spôsob, ako napísať, je stlačiť klávesy CAPS SHIFT a A. Za slovom PRINT sa vypísalo "A". Teraz stlačte kláves A. Opäť sa vypíše malé "a". Takto môžete písať veľké a malé písmená. Kláves CAPS SHIFT slúži ako prepínač na písanie veľkých písmen. Podobnú funkciu u písacieho stroja plní kláves vľavo dole, po stlačení ktorého sú na stroji písané veľké písmena.

U písacieho stroja však možno tento kláves natrvalo stlačiť tak, aby stroj písal len veľké písmená. Toto je možné aj u počítača. Najskôr odošlite riadok. Vypíše sa už spomínané hlásenie o chybe:

```
2 Variable not found, 0:1
```

To si nevšímajte. Stlačte opäť kláves P a vypíše sa PRINT. Kurzor **L** označuje, že počítač očakáva vstup malých písmen. Stlačte teraz klávesy CAPS SHIFT a 2. Kurzor **L** sa zmení na kurzor **C**. Tento kurzor označuje, že počítač bude všetky písmená písať ako veľké. Stlačte kláves A. Za slovom PRINT sa vypíše "A". Skúste stlačiť kláves S, vypíše sa "S". Všetky písmená sú teda vypísané veľké.

Spätný prechod k písaniu malých písmen je možný opätovným súčasným stlačením klávesov CAPS SHIFT a 2. Kurzor sa zmení z **C** na **L**. Stlačte kláves D a vypíše sa opäť malé "d". Na obrazovke teda máte napísané

```
PRINT Asd
```

Odošlite riadok a nevšímajte si hlásenie o chybe.

Iste ste zistili, že na klávese 2 je napísané CAPS LOCK. Tento nápis označuje, že po stlačení klávesov CAPS SHIFT a 2 sa dostanete do režimu písania veľkých písmen a v ďalšom texte nim budeme označovať stlačenie klávesov CAPS SHIFT a 2.

Tak a už poznáte tri módy (režimy), v ktorých je možné zadávať slová, číslice a písmená počítaču. Sú to:

- mód **K** – kľúčové slová
- mód **L** – číslice a malé písmená
- mód **C** – číslice a veľké písmená

Ostávajú nám ešte dva režimy zadávania z klávesnice. Prvý je rozšírený mód (označovať ho budeme ako **E**-mód). Umožňuje vypísať čierne kľúčové slová, umiestnené na klávese vpravo hore. Zresetujte počítač a stlačte kláves ENTER. Objaví sa kurzor **K**. Teraz stlačte klávesy CAPS SHIFT a SYMBOL SHIFT. Namiesto kurzora K sa vypíše kurzor **E**, označujúci rozšírený (extended) mód. Stlačte kláves V. V spodnom riadku sa vypíše kľúčové slovo LLIST. Odošlite riadok a vypíše sa hlásenie.

```
0 OK, 0:1
```

Stlačte ešte raz klávesy CAPS SHIFT a SYMBOL SHIFT (teda rozšírený mód). Kurzor **K** je nahradený kurzorom **E**. Spätný prechod ku kurzoru **K** je možný opätovným stlačením klávesov rozšíreného módu. Po ich stlačení dostanete opäť kurzor **K**.

Posledný mód je grafický. Umožňuje písať grafické symboly umiestnené na klávesoch 1 až 8. Po stlačení klávesu A až U sa vypíšu znaky užívateľsky definovanej grafiky (udg), ktoré si môžete definovať (podrobnosti sa dozviete neskôr). Do tohto módu prejdete súčasným stlačením klávesov CAPS SHIFT a 9 (všimnite si, že nad číslicou 9 je napísané GRAPHICS, teda grafický mód). Namiesto kurzora **K** teraz máte na obrazovke kurzor **G**. Stlačte napr. kláves 7. Na obrazovku sa vykreslí štvorček, ktorému chýba ľavý spodný roh. Kurzor zostane nezmenený (je to stále **G**). Pre spätný prechod musíte opäť stlačiť klávesy CAPS SHIFT a 9. Kurzor sa zmení na **L**, Resetujte počítač.

Týmto ste si vyskúšali všetky módy, ktorými disponuje Didaktik M. Na ľubovoľnom klávese však zostali dve slová, popr. symboly, ktoré sa Vám zatiaľ nepodarilo vypísať. Sú to červené kľúčové slová a niektoré špeciálne znaky.

Prístup k nim je jednoduchý. Je možný z akéhokoľvek módu (s výnimkou módu **G**). Napíšte kľúčové slovo PRINT a stlačte zároveň klávesy SYMBOL SHIFT a O. Na obrazovku sa vypíše

```
PRINT;
```

Teda červený symbol, alebo kľúčové slovo umiestnené medzi čiernymi kľúčovými slovami, dostanete po súčasnom stlačení klávesov SYMBOL SHIFT a príslušného klávesu. Odošlite riadok.

Červené príkazy a znaky umiestnené medzi čiernymi kľúčovými slovami, dostanete po súčasnom stlačení klávesov SYMBOL SHIFT a príslušného klávesu. Odošlite riadok.

Červené príkazy a znaky umiestnené na klávese celkom dole vypíšete po prechode do rozšíreného módu a súčasnom stlačení klávesu SYMBOL SHIFT a príslušného klávesu. Skúste napísať kľúčové slovo PRINT. Teraz prejdite do rozšíreného módu (kurzor sa zmení na E) a stlačte zároveň SYMBOL SHIFT a X. Vypíše sa PRINT INK

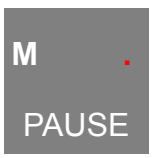
a kurzor sa zmení na **L**. Stlačte kláves 0 a na poslednom riadku máte napísané

```
PRINT INK 0
```

Odošlite riadok, vypíše sa hlásenie

```
0 OK, 0:1
```

Ostáva nám už len opakovanie. Na jednom klávese (napr. M) si zopakujeme postup, ako vypísať všetky kľúčové slová a symboly, ktoré sú na ňom umiestnené:



m stlačením klávesu po kurzore **L** (M po **C**)

Po prechode do **E** módu stlačením klávesu súčasným stlačením klávesu a SYMBOL SHIFT Stlačením klávesu po kurzore **K**

INVERSE Po prechodě do **E** módu súčasným stlačením SYMBOL SHIFT a klávesu

Okrem písmen, rôznych znakov a kľúčových slov sú na klávesnici počítača ešte ďalšie slová, pomocou ktorých môžete zadávať 6počítaču niektoré špeciálne príkazy a znaky. Tieto príkazy sú umiestnené na klávesách 0 až 9 a jeden znak je umiestnený na klávese označenom SPACE.

Znak, umiestnený na klávese SPACE, je znak "medzere". Dovoľuje Vám umiestniť v programe, popr. v texte jednu alebo viac medzier. Jeho použitie je potrebné predovšetkým v rôznych výpisoch textov, kde chcete od seba oddeliť jednotlivé slová. Zresetujte počítač a stlačte kláves SPACE. Kurzor K sa presunie

o jedno miesto doprava. Kláves stlačte znova a nechajte ho stlačený. Kurzor K sa posunie opäť o jedno miesto doprava a tento posun sa začne po chvíli opakovať. Je to tým, že kláves držíte stále stlačený. Posun trvá tak dlho, dokiaľ kláves zasa neuvoľníte. V prípade, že ste kláves držali stlačený dostatočne dlho, kurzor K sa mohol presunúť na ďalší riadok, potom zasa na ďalší riadok atď.

Okrem klávesu SPACE ste sa takto zoznámili i s funkciou "autorepeat", ktorú má každý kláves. Táto funkcia (dala by sa preložiť ako "automatické opakovanie výpisu klávesu"), je vyvolaná u každého klávesu v prípade, že daný kláves držíte dlhšie stlačený. V režime autorepeat sa výpis postupne spomaľuje. Stlačte kláves ENTER. Teraz k špeciálnym funkciám, dosiahnuteľným cez klávesy 0 až 9.

Stlačte kláves P. Na obrazovke sa Vám vypíše

```
PRINT
```

Stlačte kláves 1, potom 2. V poslednom riadku je vypísané

```
PRINT 12
```

Vy však nechcete napísať 12, ale 13. Číslicu "2" je treba zmazať. Stlačte klávesy CAPS SHIFT a O. Kurzor sa presunie o jedno miesto doľava, číslica "2" sa vymaže a na jej miesto môžete napísať napr. "3". Táto funkcia teda vymaže z riadku, v ktorom píšete, znak alebo kľúčové slovo umiestnené vľavo od kurzora. V ďalšom texte budeme označovať stlačenie klávesov CAPS SHIFT a 0 ako stlačenie klávesu DEL.

Pokiaľ stlačíte znova kláves DEL, vymaže sa opäť posledná číslica ("3"). Ďalším stlačením je vymazaná číslica "1", a ďalším stlačením celé kľúčové slovo PRINT.

Stlačenie klávesov CAPS SHIFT a 1 budeme označovať ako stlačenie klávesu EDIT. S jeho použitím sa zoznámite neskôr.

Význam stlačenia klávesov CAPS SHIFT a 2 sme už vysvetlili skôr (je to prechod medzi veľkými a malými písmenami) a budeme ho označovať ako stlačenie klávesu CAPS LOCK.

Napíšte opäť PRINT (kláves P). Ďalej napíšte "1" a stlačte klávesy CAPS SHIFT a 4. Teraz napíšte "2". Na obrazovke sa Vám vypíše

```
PRINT 12
```

ale číslica "2" nie je písaná čiernou farbou na bielom podklade, ale inverzne, teda bielou farbou na čiernom podklade. Napíšte číslicu "3". Opäť je písaná bielo na čiernom podklade. Stlačenie klávesov CAPS SHIFT a 4 teda umožňuje písať kľúčové slová a znaky inverzne a ich stlačenie budeme označovať ako INV VIDEO.

Stlačte klávesy CAPS SHIFT a 3 a napíšte číslicu "4". Táto číslica je už napísaná čierno na bielom podklade, vrátili sme sa teda k pôvodnému zobrazovaniu. Stlačenie klávesov CAPS SHIFT a 3 budeme označovať ako TRUE VIDEO a spôsobí prechod od inverzného zobrazovania k pôvodnému. Zresetujte počítač.

Nakoniec popíšeme použitie klávesov so šípkami, ktoré sú umiestnené vpravo dolu. Napíšte:

```
PRINT 123456789
```

Stlačte kláves ŠÍPKA VĽAVO. Blikajúci kurzor sa presunie o jeden znak doľava, tzn. za číslicou "8". Stlačte opäť kláves ŠÍPKA VĽAVO a držte ho stlačený. Kurzor sa presunie o jeden znak doprava, v tomto prípade za slovo PRINT. Stlačte ešte dvakrát kláves ŠÍPKA VPRAVO. Kurzor sa presunie o jeden znak doprava, v tomto prípade za slovo PRINT. Stlačte ešte dvakrát kláves ŠÍPKA VPRAVO a kurzor sa presunie za číslicu "2". Teraz stlačte kláves DEL. Číslica "2" sa vymaže. Stlačte kláves 0. Na riadku zostane

```
PRINT 103456789
```

Pomocou klávesov ŠÍPKA VĽAVO A ŠÍPKA VPRAVO môžete pohybovať kurzorom po celom riadku od jeho ľavého okraja až po pravý okraj. Z riadku môžete po nastavení kurzora na správne miesto vymazávať zle napísané znaky a nové pridávať.

Stlačením klávesov ŠÍPKA DOLE a ŠÍPKA HORE sa môžete pohybovať po riadkoch programu smerom k vyšším číslam (ŠÍPKA DOLE) a k nižším číslam (ŠÍPKA HORE) riadkov programu. Podrobnejšie vysvetlenie podáme v časti venovanej programov.

Ten istý význam, ako klávesy ŠÍPKA VĽAVO, DOLE, HORE a VPRAVO má aj súčasné stlačenie klávesu SHIFT a klávesov 5, 6, 7, 8. Pohodlnejšie je však používať samostatné klávesy pre šípky.

Nakoniec si uvedieme prehľadnú tabuľku činnosti klávesnice v rôznych módoch. V tejto tabuľke je uvedené, čo vypíše počítač po stlačení klávesu v danom režime, v zátvorke je konkrétny príklad pre kláves M. Pre horný rad klávesov bude podrobná tabuľka významu klávesov v jednotlivých módoch uvedená neskôr.

Mód	Nastaví	bez SHIFT	+CAPS	+SYMBOL
K	počítač	čierny príkaz v strede (PAUSE)	čierny príkaz v strede (PAUSE)	červený príkaz v strede (.)
L	počítač	malé písmeno a číslica (m)	veľké písmeno (M)	červený príkaz v strede (.)
C	CAPS LOCK	veľké písmeno a číslica (M)	veľké písmeno (M)	červený príkaz v strede (.)
E	CAPS+SYMBOL	čierny príkaz hore (PI)	červený príkaz dole (INVERSE)	červený príkaz dole (INVERSE)
G	GRAPHICS	znaky udg a grafika (M)	znaky udg a grafika (M)	znaky udg a grafika (M)

Po základnom zoznámení sa s klávesnicou už môžete začať vnikať do tajov programovacieho jazyka označovaného ako BASIC. Popis základných príkazov je v nasledujúcich kapitolách.

1.2. Popis jazyka BASIC

V tejto časti bude podrobne popísaný jazyka BASIC pre počítač Didaktik M. Tento popis môžete však využiť i pre všetky počítače kompatibilné s počítačom Sinclair ZX Spectrum.

1.2.1. Prvé pokusy

Skôr než začnete pracovať, povedzme si niečo o zobrazovanie na obrazovke. Časť obrazovky, na ktorú je počítač schopný vypísať text, sa skladá z 24 riadkov, každý riadok má 32 znakov. Prvých 22 riadkov zhora sa používa pre výpis programu a pre štandardný výstup. Spodné 2 riadky sa používajú pre zadávanie príkazov, programových riadkov, editovanie (úprava) programových riadkov, vstup dát a pre výpis hlásení. Táto spodná časť sa nazýva aj dialógový riadok. V prípade potreby sa dialógový riadok zväčšuje smerom hore. Po resetovaní počítača sa Vami zadávané údaje zobrazujú do dialógového riadku.

Čo musíte urobiť, aby ste napísali na hornú časť obrazovky vaše meno? Počítaču musíte nejakým spôsobom oznámiť, čo od neho požadujete.

Pre výpis na obrazovku použijete kľúčové slovo **PRINT**. Ak zadáte cez klávesnicu

```
PRINT "JAN NOVAK"
```

a stlačíte kláves ENTER, na hornú časť obrazovky sa vypíše text

```
JAN NOVAK
```

Na dolnú časť obrazovky sa vypíše hlásenie

```
0 OK, 0:1
```

čo znamená, že všetko je v najlepšom poriadku.

Ak zadáte počítaču

```
PRINT "NALEPKOVA 10"
```

a stlačíte kláves ENTER, počítač vypíše pod prvý výpis text

```
NALEPKOVA 10
```

Ďalej zadajte príkaz

```
PRINT "SKALICA"
```

a stlačte kláves ENTER. Počítač pripíše k predchádzajúcim výpisom ďalší výpis

```
SKALICA
```

Výpis v hornej časti obrazovky má tvar:

```
JAN NOVAK
NALEPKOVA 10
SKALICA
```

Na týchto príkladoch sme ukázali, ako sa dá najjednoduchšie komunikovať s počítačom. Predchádzajúcu činnosť si podrobnejšie analyzujme.

Ak chcete na obrazovku vypísať ľubovoľný text, musíte pred text napísať slovo PRINT. Toto slovo znamená v angličtine vytlač, píš. Patrí do skupiny slov jazyka BASIC, ktoré sa nazývajú **príkazy**. Každý príkaz jazyka BASIC je tvorený anglickým názvom (alebo skratkou), ktorý určuje, akú činnosť ma počítač vykonať. Príkaz PRINT je príkaz pre výpis na obrazovku. Text, ktorý chcete pomocou neho vypísať, napíšete za príkazové slovo PRINT a uzavriete ho úvodzovkami. Text sa vypíše na obrazovku od ľavého okraja časti obrazovky pre výpis. Všimnite si, že na obrazovke sa zobrazí iba text uzavretý v úvodzovkách, úvodzovky ani slovo PRINT sa nezobrazia.

Ak použijete ďalší príkaz PRINT, výpis sa zobrazí pod predchádzajúci výpis.

Doteraz ste počítaču zadávali iba jeden príkaz na spracovanie. Zadávaný riadok sa nazýva **príkazový riadok**. Po stlačení klávesu ENTER sa vykoná príkaz, ktorý príkazový riadok obsahuje. Príkaz sa však nevykoná, ak sa pri zápise príkazu dopustíte chyby. Ak zadáte počítaču

```
PRINT "PRACUJ
```

a stlačíte kláves ENTER, za písmenom J začne blikať otáznik **?**, ktorý Vás upozorňuje na miesto v príkazovom riadku, kde počítač predpokladá chybu (chýbajú koncové úvodzovky). Po stlačení klávesu ENTER počítač vykoná analýzu príkazového riadku a v prípade chyby Vás "nepustí" ďalej. Chybu musíte opraviť a až keď je príkazový riadok napísaný bez chyby, je počítač ochotný vykonať príkaz v príkazovom riadku.

Príkazový riadok môže obsahovať aj viac príkazov. Príkazy sa v takom prípade oddeľujú od seba dvojbodkou. Hovoríme, že dvojbodka má funkciu **oddeľovača príkazov**.

Vytvorte príkazový riadok, ktorého výsledkom bude výpis mena a adresy. Príkazový riadok bude mať tvar:

```
PRINT "JAN NOVAK":PRINT "NALEPKOVA 10":PRINT "SKALICA"
```

Po odoslaní príkazového riadku sa na obrazovke vypíše text

```
JAN NOVAK
NALEPKOVA 10
SKALICA
```

Vidíte, že výsledok je rovnaký, ako keď ste každý príkaz odoslali osobitne. Aby sme mohli vysvetliť ďalšie vlastnosti príkazu PRINT, vysvetlíme si, ako pracuje pri výpise na obrazovku počítač. Ten musí v každom okamžiku "vedieť", na ktoré miesto obrazovky sa bude vypisovať ďalšie písmeno. Na to je určený takzvaný ukazovateľ výpisu pre príkaz PRINT. Pred prvým použitím príkazu PRINT je ukazovateľ nastavený na ľavý okraj prvého riadku obrazovky a nasledujúci výpis sa vykoná z tohto miesta. Miesto výpisu môžeme ovládať nastavením ukazovateľa výpisu. V predchádzajúcom príklade v príkazovom riadku nasledovala za textom dvojbodka. Pre počítač to značí, že na riadku obrazovky, na ktorom sa vypisoval text, nechcete už nič iné vypísať. Ukazovateľ sa nastaví na ľavý okraj nasledujúceho riadku obrazovky.

Vytvorme nasledujúci príkazový riadok:

```
PRINT "JANO MA ";:PRINT "OSEM ";: PRINT "ROKOV"
```

Počítač vypíše na obrazovku

```
JANO MA OSEM ROKOV
```

Ak je za textom pre výpis bodkočiarka, ukazovateľ pre výpis sa nastaví za predchádzajúci text a nasledujúci text sa začne vypisovať z tohto miesta. Rovnaký výsledok dostanete aj v prípade, ak zadáte nasledujúci príkazový riadok:

```
PRINT "JANO MA ";"OSEM ";"ROKOV"
```

Vidíte, že v jednom príkaze PRINT môže byť viacej "textov", ktoré sú od seba oddelené bodkočiarkou. Hovoríme, že bodkočiarka je oddeľovač pre príkaz PRINT. Jej význam je ten, že nasledujúci text sa vypíše hneď za predchádzajúci text.

Rovnaký výsledok dostaneme samozrejme aj príkazom

```
PRINT "JANO MA OSEM ROKOV"
```

Zložitejší spôsob výpisu sme uviedli preto, aby sme ukázali význam bodkočiarky ako oddeľovača v príkaze PRINT.

Kvôli lepšej prehľadnosti je niekedy vhodné začínať výpis na čistú obrazovku. Na zmazanie obrazovky použijete príkaz **CLS**. Po vykonaní tohto príkazu sa obrazovka zmaže a nasledujúci výpis začína na prvom riadku obrazovky. Ak máte na obrazovke nejaký výpis a chcete na prvý riadok obrazovky vypísať text TABULKA, použijete príkazový riadok

```
CLS:PRINT "TABULKA"
```

Ako ďalší oddeľovač môžete použiť čiarku. Vytvorte nasledujúci príkazový riadok:

```
PRINT "JANO","EMIL","FERO","EVA"
```

Po vykonaní vidíte, že medzi menami sú väčšie medzery. Obrazovka je rozdelená na tzv. zóny, každá zóna má 16 znakov (čiže v jednom riadku sú 2 zóny). Ak použijete čiarku ako oddeľovač, ukazovateľ sa nastaví na začiatok nasledujúcej voľnej zóny a nasledujúci text sa začne vypisovať z tohto miesta.

V prípade potreby sa začne výpis na nasledujúcom riadku, napr.:

```
PRINT "JANO, EMIL A FERO","EVA"
```

Prvý text sa začne vypisovať od začiatku riadku (prvá zóna), pretože však presahuje do druhej zóny, nasledujúci text sa vypíše do nasledujúcej voľnej zóny, tj. do nasledujúceho riadku.

Príkazom PRINT môžete zobraziť aj čísla, ktoré nemusia byť uzavreté v úvodzovkách. Príkazom

```
PRINT "VYSLEDOK JE ";5;" METROV"
```

sa na obrazovke zobrazí

```
VYSLEDOK JE 5 METROV
```

V jednom príkaze PRINT môžete použiť viac druhov oddeľovačov, pričom môžu byť oddeľovače aj vedľa seba, napr.:

```
PRINT "JANO";8,,"JOZO"
```

Prvá čiarka za číslom 8 nastaví ukazovateľ na druhú zónu v riadku, ďalšia čiarka na ďalšiu zónu, t.j. na nasledujúci riadok.

Odošlite počítaču nasledujúci príkazový riadok:

```
PRINT "VYNECHAJ":PRINT:PRINT "RIADOK"
```

Počítač vypíše

```
VYNECHAJ


RIADOK
```

tj. medzi VYNECHAJ a RIADOK je jeden riadok voľný. Čo sa stalo? Prvý príkaz PRINT vypíše VYNECHAJ a nastaví ukazovateľ na nasledujúci riadok.

Druhý príkaz PRINT nevypíše nič a pretože nie je ukončený čiarkou ani bodkočiarkou, nastaví ukazovateľ na nasledujúci riadok, čím vlastne jeden riadok vynechá.

Tretí príkaz PRINT vypíše RIADOK a nastaví ukazovateľ na nasledujúci riadok.

Ak chcete vynechať väčší počet riadkov, príkaz PRINT zadáte viackrát za sebou. V takom prípade je vhodnejšie využiť tvar príkazu PRINT s apostrofom. Apostrof v príkaze PRINT nastaví ukazovateľ na ďalší riadok. Príkaz

```
PRIKAZ ````
```

vynechá 4 riadky.

To je zatiaľ všetko o príkaze PRINT. Ďalšie možnosti ovládanie výpisu na obrazovku sa dozviete v ďalších kapitolách.

1.2.2. Prvý program

Doteraz ste sa naučili, že po odoslaní príkazového riadku klávesom ENTER sa príkazy tohto riadku ihneď vykonali. Takýto režim práce sa označuje ako priamy režim práce. Výhodou tohto režimu je, že výsledky vidíte okamžite na obrazovke. Nevýhodou je, že počítač si nepamätá všetko, čo vykonal predtým.

Teraz sa naučíte vkladať údaje uloží do pamäti a v prípade potreby vie uložené údaje použiť. Tým sa dostávame k programovému režimu práce, naučíte sa, čo je to program a ako sa zostavuje.

Zhruba môžeme povedať, že **program** v jazyku BASIC je postupnosť príkazov, ktoré sú usporiadané podľa určitých pravidiel (syntax jazyka). Tieto pravidlá sa musíme naučiť, lebo počítač ich pozná a riadi sa presne podľa nich. Každú odchýlku od syntaxe počítač spozná a obyčajne ohlási chybu výpisom správy do spodného riadku.

Program sa skladá z riadkov programu (programových riadkov). **Programový riadok** je taký príkazový riadok, ktorý začína číslicou.

Vložte tieto 2 riadku programu do počítača:

```
20 PRINT "tabulku"
10 PRINT "Hlavicka ";
```

a obrazovka vyzerá takto:

```
10>PRINT "Hlavicka ";
20 PRINT "tabulku"
```

Tieto riadky začali číslicami a preto neboli hneď vykonané, ale uchované ako riadky programu. Všimnite si, že čísla riadkov v programe sú zoradené. Je to dôležité pre jeho beh a tiež sa to prejaví vo výpise riadkov, ktoré teraz môžete vidieť na obrazovke. Teraz odošlite riadok

```
15 PRINT "pre ";
```

Ak by boli riadky číslované bez medzier, nebolo by možné medzi dva napísané riadky už vložiť ďalší riadok. Preto je výhodné číslovať riadky namiesto 1, 2, … radšej 10, 20, … avšak maximálne 9999. Dodatočné vkladanie riadkov je dôvod, prečo číslovať riadky s krokom 10.

Teraz chceme zmeniť riadok 20 na:

```
20 PRINT "tabulku vysledkov"
```

Môžete to urobiť tak, že riadok 20 napíšete znova. Ak odošlete nový riadok s číslom 20, nový riadok prepíše starý.

Výhodnejšie je však využiť možnosť režimu EDIT (CAPS SHIFT + 1). Šípka > pri riadku 15 sa nazýva **programový kurzor** a riadok, na ktorý ukazuje, sa nazýva **bežný riadok**. Pri vkladaní programu je to posledný riadok, ktorý bol vložený, ale na nastavenie programového kurzora môžete použiť šípku dole alebo hore. Skúste si pohyb programového kurzora a nechajte ho nastavený na riadok 20. Keď stlačíte EDIT, objaví sa kópia bežného riadku v spodnej časti obrazovky – vo Vašom prípade je to kópia riadku 20. Pomocou šípky doprava presuňte kurzor L na koniec riadku pred úvodzovku a potom vložte text

```
vysledkov
```

Riadok by mal teraz mať tvar:

```
20 PRINT "tabulku vysledkov"
```

Po stlačení ENTER sa nahradí starý riadok 20 a obrazovka vyzerá takto:

```
10 PRINT "Hlavicka ";
15 PRINT "pre ";
20>PRINT "tabulku vysledkov"
```

Program máte uložený v pamäti. Na odštartovanie programu slúži príkaz **RUN** (kláves R). Takže zadajte RUN a stlačte ENTER.

Na obrazovku sa vypíše text:

```
Hlavicka pre tabulku vysledkov
```

Príkaz RUN zmaže obrazovku a odštartuje program od začiatku. Ale program môžete odštartovať od ľubovoľného riadku programu príkazom

RUN číslo riadku

Tak po odoslaní príkazu

```
RUN 20
```

dostanete na obrazovku výpis

```
tabulku vysledkov
```

Ešte sa vrátime k režimu EDIT. Môžete ho použiť na vymazanie spodnej časti obrazovku. Niečo napíšte (bez ENTER) a potom sa rozhodnete, že to nechcete. Jedna z možností, ako to vykonať, je držať kláves DELETE, dokiaľ nie je riadok zrušený. Existuje i nasledujúca možnosť. Pokiaľ stlačíte EDIT, bude text v spodnej časti obrazovky nahradený kópiou bežného riadku. Pokiaľ teraz stlačíte ENTER, bežný riadok sa uloží do programu bez zmeny a spodná časť obrazovky ostane voľná. Pokiaľ odošlete riadok s chybou, napr.

```
12 PRINT "zly riadok ";
```

uloží sa tento riadok do programu a až potom si uvedomíte chybu. Pre vynechanie tohto riadku odošlite:

```
12
```

Teraz si s údivom všimnete, že programový kurzor zmizol. Môžete si predstaviť, že je niekde medzi 10. a 15. riadkom. Pokiaľ stlačíte šípku hore, bude sa nachádzať na riadku 10, ak stlačíte šípku dole, bude na riadku 15. Odošlite

```
12
```

Programový kurzor bude opäť medzi riadkami 10 a 15. Teraz stlačte EDIT a dole bude kópia riadku 15. Keď je programový kurzor medzi riadkami, potom EDIT vyberie riadok s vyšším číslom. Napíšte ENTER na zmazanie spodnej časti obrazovky. Potom odošlite

```
30
```

Teraz je programový kurzor za koncom programu. Pokiaľ stlačíte EDIT, bude riadok 20 vypísaný dole.

Doteraz vždy po stlačení ENTER sa vypísal program na obrazovku. Tento výpis nazvime automatický listing (listing je výpis programu). Existuje však špeciálny príkaz pre výpis programu na obrazovku a to príkaz

LIST

ktorý vypíše program od prvého riadku.

Ak chcete vypísať program od určitého riadku, zadáte číslo požadovaného riadku za slovo LIST v tvare

LIST číslo riadku

Zadajte napr. príkaz:

```
LIST 15
```

a na obrazovke vidíte:

```
15>PRINT "pre ";
20 PRINT "tabulku vysledkov"
```

Riadok 10 nie je zobrazený, ale je stále vo Vašom programe. Môžete si to overiť stlačením ENTER. Výsledkom príkazu LIST 15 je, že sa vykoná výpis programu, začínajúci riadkom 15 a ďalej je na tomto riadku nastavený programový kurzor. Pokiaľ máte veľmi dlhý program, potom príkazom LIST nastavíte programový kurzor oveľa rýchlejšie ako šípkami.

Vložte do programu nasledujúci riadok:

```
5 REM hlavicka tabulky
```

a odštartujte program príkazom RUN.

Určite ste si všimli, že ste dostali ten istý výsledok, ako keď v programe tento riadok nebol. Príkaz **REM** je totiž nevýkonný. Ak sa dostane vykonávanie programu na príkaz REM, text za príkazom REM sa preskočí a pokračuje sa na ďalšom riadku. Preto môže vyť za REM akýkoľvek text. Príkaz REM s používa ako poznámka v programe, ktorou môžete označiť časti programu pre Vašu informáciu, na vykonávanie programu nemá žiadny vplyv.

Program môžete odštartovať aj príkazom **GO TO**, ktorý má tvar

```
GO TO číslo riadku
```

kde číslo riadku má ten istý význam ako v príkaze RUN. Tým odštartujte program od zadaného čísla riadku. Napr.:

```
GO TO 20
```

Zdá sa Vám, že medzi príkazmi RUN a GO TO nie je žiadny rozdiel. V uvedenom programe je pravda. Všeobecne to však neplatí. Príkaz GO TO vykonáva program od zadaného miesta, pričom berie do úvahy stav, v ktorom sa počítač nachádza. Príkaz RUN je nebezpečnejší, lebo pri jeho odoslaní sa vždy nastaví časť pamäti do počiatočného stavu, čím stratíte výsledky, ktoré má počítač uložené v pamäti, zmaže sa obrazovka atď. (podrobne sa to dozviete neskôr).

Ak v príkaze RUN a GO TO zadané číslo riadku v programe neexistuje, pokračuje vykonávanie programu na najbližšom vyššom riadku.

To, že príkaz GO TO ponecháva doterajšie výsledky v pamäti, je veľmi výhodné. Príkaz GO TO môžete totiž použiť priamo v programe, čím sa dostávame k pojmu programový cyklus. Je to časť programu, ktorá sa vykonáva opakovanie. Doplňme program o nasledujúci riadok

```
30 GO TO 10
```

a program odštartujte. Na celú obrazovku sa vypíše opakovane text

```
hlavicka pre tabulku vysledkov
```

a v dolnom riadku sa objaví výpis

```
scroll?
```

Čo sa vlastne stalo? Po prvom výpise riadku sa dostaneme na riadok 30, ktorý nás opäť vráti na riadok 10, nasleduje výpis a zase riadok 30, ktorý nás opäť vráti na riadok 10, nasleduje výpis a zase riadok 30 atď. Keď je na obrazovke zobrazených 22 riadkov, výpis sa zastaví a v dolnom riadku sa vypíše scroll?.

Výpis sa zastavil preto, aby ste si mohli prezrieť výpis na obrazovke a výpisom scroll? sa počítač pýta, či má pokračovať ďalej vo výpise, alebo má prerušiť program.

Ak stlačíte kláves N, kláves SPACE alebo kláves BREAK (CAPS SHIFT + SPACE), vykonávanie programu sa preruší a v dolnom riadku sa vypíše správa

```
D BREAK - CONT repeats, 10:1
```

Ak stlačíte iný kláves, na obrazovku sa vypisujú ďalšie riadky tým spôsobom, že predchádzajúce riadky sa posúvajú smerom nahor a nasledujúci riadok sa vypisuje vždy na uvoľnené miesto dole.

Správa pri prerušení oznamuje, že program bol prerušený na riadku 10 v príkaze 1 (10:1) a v prerušenom programe je možné pokračovať príkazom

```
CONTINUE
```

Zaujímavé je, že počas prerušenia programu môžete s počítačom vykonávať inú činnosť, napr. odoslať iné príkazy (PRINT), prezerať program príkazom LIST, ba dokonca je možné opravovať program v režime EDIT. Počítač si "pamätá", na ktorom mieste bol program prerušený a v programe môžete pokračovať príkazom CONTINUE. Skúste to!

V programe urobíte niekedy chybu a program beží v cykle bez výpisu na obrazovku, napr. program obsahuje tento riadok:

```
200 GO TO 200
```

Ak sa dostane vykonávanie programu na tento riadok (môžete ho odštartovať príkazom RUN 200), dostali ste sa do programového cyklu, z ktorého zdanlivo niet úniku. Na ukončenie činnosti stačí stlačiť kláves BREAK (CAPS SHIFT a SPACE). Vykonávanie programu sa zastaví a v dolnom riadku sa zobrazí

```
L BREAK into program, 200:1
```

čo znamená prerušenie programu klávesom BREAK. Pokračovať môžete opäť príkazom CONTINUE od toho miesta programu, kde bol prerušený. Kláves BREAK možno použiť kedykoľvek počas vykonávania programu. Jeho nevýhodou (a zároveň výhodou) je však to, že prerušenie môže nastať na ľubovoľnom mieste programu. Pri testovaní programu je vhodné zastaviť program na presne určenom mieste programu. Na to slúži príkaz

```
STOP
```

Doplňte predchádzajúci program o riadok

```
25 STOP
```

a odštartujte program. Po výpise textu sa objaví hlásenie

```
9 STOP statement, 25:1
```

čo znamená, že program bol prerušený príkazom STOP. Ďalšia činnosť je podobná ako pri BREAK.

Ak ukončíte prácu s vytvoreným programom a chcete vytvoriť iný, potrebujete starý program zmazať. Mohli by ste to robiť rušením riadkov, ako ste sa to už naučili. V tomto prípade by to ešte išlo, ale u dlhých programov by ste sa "narobili". Preto je k dispozícii príkaz

```
NEW
```

ktorý zmaže starý program. Používajte ho však veľmi opatrne, lebo po jeho odoslaní je program beznádejne stratený!

Videli ste použitie príkazov PRINT, RUN, LIST, GO TO, CONTINUE, NEW a REM. Všetky mohli byť použité v programe alebo v priamom režime. To platí takmer o všetkých príkazoch jazyka BASIC. Príkazy RUN, LIST, CONTINUE a NEW nie sú často používané v programoch, ale môžu byť použité.

1.2.3. Zavedenie nových pojmov

Zatiaľ ste využili počítač len na výpis textu na obrazovku. Počítač však dokáže pracovať s číslami, dokáže ich sčítať, násobiť, deliť a to veľmi rýchlo. Aby ste mohli tieto výhody využiť, musíte poznať niektoré pojmy, ako číslo, reťazec, premenná, výraz atď.

1.2.3.1. Číselné konštanty

Číselné konštanty sú dáta vyjadrené číslami. Môžu to byť čísla kladné, záporné, celé, desatinné. Vyjadrené môžu byť v bežnom alebo v takzvanom semilogaritmickom tvare. Dôležité je, že u desatinných čísiel musíme písať namiesto desatinnej čiarky desatinnú bodku. Ak zapisujete číselnú konštantu, ktorá nemá celú časť, nulu pred desatinnou bodkou nemusíme písať. Takisto nemusíte písať pred kladnou konštantou znamienko +.

Každé číslo je v počítači uložené na ohraničenej časti pamäti, preto nemôžete používať ľubovoľne veľké alebo malé čísla. Číslo môžete zadať na maximálne 8 platných číslic. Ak zadáte číslo, ktoré má viac ako 8 platných číslic, počítač toto číslo zaokrúhli na 8 platných číslic. Každé číslo, ktorého absolútna hodnota je v rozsahu od 0.00001 až 99999999 vypíše počítač v takzvanom pevnom formáte bez exponentu.

Všetky ostatné čísla sa vypisujú v **semilogaritmickom formáte**, ktorý má tvar:

```
SM.MMMMMMMESXX
```

kde jednotlivé písmená majú nasledovný význam:

S – znamienko mantisy alebo exponenta (vypisuje sa iba -)
M – číslica mantisy (0 – 9)
E – označenie exponenta písmenom e, E
X – číslica exponenta

Tento zápis znamená číslo

SM.MMMMMMM krát 10 na SXX, t.j. mantisa krát 10 na exponent.

Uveďme si na ozrejmenie tohto pojmu niekoľko príkladov:

```
31 = 3.1x10 = 3.1E+1
123400000 = 1.234E+8
-0.0000089 = -8.9E-6
```

Rozsah čísiel, ktoré je možné spracovať, je asi

$1.7 \times 10^{+38}$ až 1.7×10^{-38}

Nasledujúce príklady Vám ukážu rozdiel medzi správne a nesprávne zadaným číslom:

Správny zápis čísla	Nesprávny zápis čísla
5.3	5,3
-3.78	-3.7.8
1.1.	E 02
36E02	5.31E53

V nasledujúcej časti si vysvetlíme pojem premenná. Počítač má k dispozícii určitý počet pamäťových miest, na ktoré môže uložiť hodnotu konštánt. Každé takéto miesto si môžete predstaviť ako nejakú schránku, do ktorej je možné vhodiť, alebo z nej možno vybrať lístok s ľubovoľnou konštantou. S obsahom týchto schránok môžete manipulovať za týchto podmienok:

a) každá schránka môže obsahovať iba jeden lístok
b) ak do schránky vložíme nový lístok, starý lístok tým zničíme (pokiaľ sme ho nepremiestnili do novej schránky)
c) obsah schránky môžeme prečítať bez toho, že by sme tento obsah zničili

Každú schránku môžeme označiť môžeme označiť, dať jej meno. Podobne sa dá symbolicky označiť každé pamäťové miesto v počítači, do ktorého chcete nejaký obsah uložiť, ktorého obsah chcete zmeniť, resp. ktorého obsah chcete prečítať. S týmto označením môžete potom ďalej pracovať a budeme ho nazývať premenná.

Premenné je možné rozdeliť na skupiny podľa rôznych kritérií.

Podľa obsahu ich delíme na:

a) číselné (numerické)
b) reťazcové (alfanumerické)

Podľa štruktúry ich delíme na:

a) jednoduché
b) indexované (polia)

1.2.3.2. Číselné premenné

Sú to premenné, ktorých hodnotou je číslo. Meno číselnej premennej môže byť zložené z písmen a číslic, prvé musí byť písmeno. V názve premennej nie je rozdiel v použití malých a veľkých písmen. Dokonca kvôli prehľadnosti môžete použiť i medzery, ktoré sa však vnútorne nepoužívajú. Uveďme

niekoľko prípustných mien:

```
x
a318
meno s medzerou
male a VELKE
maleaVELKE (rovnaké ako predchádzajúce)
```

Neprípustné sú napríklad nasledujúce mená:

```
100     (začína číslicou)
3 sudy  (začína číslicou)
a*b88   (* nie je písmeno ani číslica)
```

Keď viete, ako sa označujú premenné, musíte sa naučiť priradiť premennej hodnotu. Na to je určený príkaz **LET**.

Ak chcete, aby číselná premenná c mala hodnotu 5, priradíte jej túto hodnotu príkazom

```
LET c=5
```

Za slovom LET je označenie premennej, ktorej chcete priradiť hodnotu, nasleduje znak = a za týmto znakom ja číselná konštanta, ktorá sa priradí premennej. Znak = v tomto prípade nie je symbolom označujúcim rovnosť, ale je **symbolom priradenia**. Preto je nesprávna interpretácia príkazu

```
LET c=5
```

ako c rovná sa 5 (aj keď sa často používa), ale premennej c sa priradí hodnota 5. Podobne premennej ad priradíte hodnotu 32.45 príkazom

```
LET ad=32.45
```

Pomocou známych príkazov môžete vytvoriť krátky program:

```
10 LET aa=82
20 LET b=2.73
30 LET c4=10E2
40 PRINT aa: PRINT b: PRINT c4
```

Po odštartovaní programu príkazom RUN vypíše počítač na obrazovku

```
82
2.73
1000
```

Činnosť programu si pozrime bližšie.

Na riadku 10 sa premennej aa priradí hodnota 82.

Na riadkoch 20 a 30 sa premennej b priradí hodnota 2.73 a premennej c4 hodnota 10E2=1000.

Na riadku 40 je použitý príkaz PRINT v tvare, v akom ho ešte nepoznáte. Ak v príkaze PRINT použijete označenie premennej, počítač vypíše hodnotu premennej, nie jej meno. To znamená, že počítač sa "pozrie" do pamäti na miesto, ktoré je označené ako aa, z tohto miesta zoberie číslo, ktoré je tam uložené (my sme tam uložili na riadku 10 hodnotu 82) a vypíše toto číslo. Takisto je to s premennými b, c4.

Všimnite si, že hodnotu premennej c4 nevypísal počítač v tom tvare, ako sme ju zadali, ale v pevnom formáte.

Zadajte teraz príkaz

```
GO TO 40
```

Počítač opäť vypíše na obrazovku hodnoty premenných.

Pre zmenu odštartujte program príkazom

```
RUN 40
```

Počítač vypíše správu

```
2 Variable not found, 40:1
```

lebo na riadku 40 hľadá počítač v pamäti premennú aa, ale nenašiel ju. Je to preto, lebo príkaz RUN najprv zruší všetky premenné a ak chcete potom s nejakou premennou pracovať, musíte jej najprv priradiť hodnotu (tým sa vytvorí v pamäti pre ňu zároveň miesto).

To isté sa stane, ak zrušíte riadok 10 a program odštartujeme príkazom RUN.

Preto si zapamätajte, že každá premenná musí byť najprv **definovaná** (nemusíme to byť len príkazom LET) a až potom s ňou môžete pracovať! Toto platí pre všetky premenné, nielen pre číselné.

Ak chcete počas behu programu zrušiť všetky premenné, použijete príkaz **CLEAR**. Týmto príkazom sa zároveň zmaže obrazovka.

Hodnotu premennej môžete priradiť aj priamo z klávesnice, ak v programe použijete príkaz INPUT. Úplný popis tohto príkazu uvedieme neskôr, zatiaľ si ukážeme jeho použitie na príkladu:

```
10 INPUT "Zadaj cislo ";b
20 PRINT b
30 GO TO 10
```

Príkaz INPUT vypíše do dialógového riadku správu

```
Zadaj cislo
```

a čaká, až zadáte číslo cez klávesnicu a odošlete ho stlačením klávesu ENTER. Počítač zadané číslo priradí premennej b (tým ju zároveň definuje). Na riadku 20 sa vypíše hodnota b a celý postup sa opakuje. Uvedený program môžete využiť pri zisťovaní, v akom tvare sa vypisujú čísla na obrazovku. Možný je aj tvar bez správy

```
INPUT b
```

Analogicky sa dá priradiť hodnota reťazcovej premennej.

1.2.3.3. Reťazcové konštanty

Počítač si dokáže zapamätať nielen číslo, ale i iné postupnosťí znakov, ktoré nemusia byť čísla. To ste videli pri vysvetľovaní činnosti príkazu PRINT, kde sme vypisovali text uzavretý v úvodzovkách. To nás privádza k pojmu **reťazcová konštanta**.

Je to ľubovoľná postupnosť znakov, ktorá sa chápe ako jeden celok. Táto postupnosť znakov je uzavretá v úvodzovkách.

Maximálna dĺžka reťazcovej konštanty (počet jej znakov) je 255 znakov.

Uveďme si príklady:

```
"DIDAKTIK M"
" computer"
"56p : 45.s "
"5.78"
```

Počítač berie do úvahy všetky znaky medzi úvodzovkami, aj medzery. Poslednú konštantu môže chápať niekto ako číslo. Nie však počítač. Neskôr uvidíte, že počítač vie pracovať s číslami aj s reťazcami, ale s číslami vie robiť úplne iné operácie ako s reťazcami a naopak.

Podobný význam ako číslo 0 pre čísla má pre reťazce prázdny reťazec, ktorý sa označuje ako "" (t. j. dve úvodzovky vedľa seba).

Pretože reťazcové konštanty sa ohraničujú do úvodzoviek, je problém v označení konštanty, ktorá má úvodzovky obsahovať. Preto platí dohoda, že v takom prípade sa zadajú dve úvodzovky vedľa seba. Teda

```
"a""b"
```

je konštanta obsahujúca znaky a"b.

Analogicky

```
""""
```

je konštanta obsahujúca jednu úvodzovku.

1.2.3.4. Reťazcové premenné

Pre **reťazcové premenné** platí to isté, čo sme uviedli o číselných premenných - označujú nejaké miesto v pamäti počítača. V prípade reťazcovej premennej je na tomto mieste uložená reťazcová konštanta (t. j. konkrétna postupnosť znakov). Meno reťazcovej premennej môže byť len jedno písmeno, za ktorým je znak $ (dolár). Tento znak rozlišuje meno reťazcovej premennej od mena číselnej premennej, musí byť vždy na konci mena reťazcovej premennej.

Správne mená reťazcových premenných sú:

```
a$, e$, k$
```

Názvy uvedených reťazcových premenných sa čítajú ako a-string, e-string, k-string.

Nesprávne mená reťazcových premenných sú

```
aa$, h3$, abeceda, x
```

Pri označovaní premenných môžete použiť pre číselné aj reťazcové premenné rovnaké meno (okrem znaku $ pre reťazcovú premennú), napr.

```
a, a$, c, c$,
```

počítač ich chápe ako rozdielne premenné.

Hodnotu reťazcovej premennej priradíte príkazom LET

```
LET a$="ulica"
LET b$="meno a priezvisko"
```

Na ľavej strane znaku = je meno reťazcovej premennej, na pravej strane je reťazcová konštanta, ktorej hodnota sa premennej priradí. Ak chceme vypísať hodnotu reťazcovej premennej, v príkaze PRINT zadáte meno tejto premennej

```
10 LET a=5:LET a$="Vysledna hodnota je "
20 PRINT a$;a
```

Po odštartovaní programu počítač vypíše

```
Vysledna hodnota je 5
```

Pod pojmom **reťazec** budeme zatiaľ rozumieť reťazcovú konštantu alebo reťazcovú premennú, neskoršie tento pojem rozšírime.

Pokiaľ je daný reťazec, **podreťazec** je jeho súvislá časť. Tak sú "auto","mobil" podreťazce z "automobil", ale "util" nie je podreťazec z "automobil" (nie je to súvislá časť).

Ak chcete označiť podreťazec, použite nasledovný tvar:

```
reťazec (začiatok TO koniec),
```

kde začiatok a koniec označujú pozíciu znaku v reťazci.

Tak napr.

```
"abcdef"(2 TO 5) = "bcde"
```

Keď vynecháte začiatok, očakáva sa, že je 1, keď vynecháte koniec, potom je reťazec ponechaný do konca. Preto

```
"abcdef"(TO 5) = "abcdef"(1 TO 5) = "abcde"
"abcdef"(2 TO) = "abcdef"(2 TO 6) = "bcdef"
"abcdef"(TO) = "abcdef"(1 TO 6) = "abcdef"
```

Posledný príklad môžete písať tiež ako "abcdef"().

Podobne platí

```
"abcdef"(3) = "abcdef"(3 TO 3) = "c"
```

V uvedených príkladoch znak = značí rovnosť reťazcov.

Musí platiť, že začiatok aj koniec musí zodpovedať existujúcej časti reťazca. Výnimkou je, ak je začiatok väčší než koniec, výsledkom je prázdny reťazec. Napr.

```
"abcdef"(5 TO 7)
```

dáva chybu

```
3 Subscript wrong
```

ale

```
"abcdef"(8 TO 7)=""
"abcdef"(1 TO 0)=""
```

Začiatok a koniec nesmú byť záporné čísla, inak dostaneme toto hlásenie o chybe:

```
B Integer out of range.
```

Výber podreťazcov si vyskúšajte na príkladoch nasledujúceho typu

```
10 LET a$="abcdefgh"
20 PRINT a$(2 TO 4):PRINT a$(3)
```

Pre reťazcové premenné nemusíte len vyberať podreťazce, ale tiež ich meniť. Napr. zadajte

```
LET a$ = "V praci nespi!"
LET a$(3 TO 7) = "skolenie"
PRINT a$
```

Všimnite si, že podreťazec a$(3 TO 7) je dlhý len 5 znakov, preto bolo použitých len prvých 5 znakov z reťazca "skolenie". To je charakteristické pre zmenu podreťazcov, podreťazec musí mať presné rovnakú dĺžku ako mal skôr. Z dlhšieho reťazca sa zoberie len potrebný počet znakov, kratší sa doplní sprava medzerami.

Presvedčíte sa, že je to tak!

Z reťazcov môžeme nielen vyberať podreťazce, ale reťazce môžeme aj **spájať** pomocou znaku +. Platí:

```
"abc" + "def" = "abcdef"
```

Na komplikované reťazcové vyjadrenie budete potrebovať zátvorky, pretože z nich môže byť ďalej vyberané. Platí:

```
"abc"+"def"(1 TO 2) = "abcde"
("abc"+"def")(1 TO 2) = "ab"
```

1.2.3.5. Indexované premenné (polia)

Dôvodom pre zavedenie **indexovaných premenných** je snaha uľahčiť prácu s dátami. Vysvetlíme si, čo to vlastne je.

Chceme označiť každý deň v roku. Môžeme to urobiť takto:

```
d(1), d(2), d(3), …, d(365)
```

Použili sme označenie písmenom d, za ktorým je v okrúhlych zátvorkách uvedené poradové číslo dňa v roku. Číslu v zátvorkách budeme hovoriť **index**. Jednotlivé značenia dni sa líšia od seba len hodnotou indexov, písmeno d je spoločné.

Podobne by sme mohli označiť dni v roku tak, že v zátvorke budú dve čísla oddelené čiarkou. Prvé číslo označuje mesiac, druhé deň v mesiaci.

Napr. d(5,20) označuje 20-ty deň v piatom mesiaci.

Hovoríme, že indexované premenné, ktoré majú to isté označenie, tvoria **pole**. Jednotlivé indexované premenné z tohto poľa nazývame **prvkami** tohto poľa. Každý prvok poľa je určený menom poľa a svojou pozíciou v poli, pozícia je určená hodnotou indexov prvku.

Meno poľa môže byť len jedno písmeno, za meno sa do okrúhlych zátvoriek udávajú rozmery jednotlivých dimenzií (t. j. maximálne hodnoty indexov, ktoré môžu byť pri označení prvkov tohto poľa použité). Indexy môžu byť len celé čísla väčšie ako 0 (1,2,3,..), oddeľujú sa od seba čiarkou.

Správne označené indexované premenné sú:

```
b(3,3),y(10)
```

Nesprávne označené indexované premenné sú:

```
q(4;2), c(1,0), d(-1,-1).
```

Ak chcete použiť pole dát, musíte počítaču oznámiť, ako sa bude pole volať a aké bude mať rozmery. Na to sa používa príkaz **DIM**. Tento príkaz musíte použiť pred prvým použitím prvku tohto poľa (nikdy nepracujeme s poľom ako s celkom, ale vždy iba s prvkom poľa). Príkazom DIM sa rezervuje v pamäti počítača miesto pre jednotlivé prvky:

```
10 DIM a(10)
20 DIM b(4,5)
```

Príkaz DIM na riadku 10 rezervuje v pamäti miesto pre číselné pole, ktoré má 10 prvkov a(1), a(2), ..., a(10) a hodnoty prvkov sú nastavené na 0. Zároveň zruší akékoľvek pole a, ktoré bolo definované skorej.

Príkaz DIM na riadku 20 rezervuje v pamäti miesto pre číselné pole, ktoré má 4x5 prvkov:

```
b(1,1), b(1,2), …, b(1,5)
b(2,1), b(2,2), …, b(2,5)
…
b(4,1), b(4,2), …, b(4,5)
```

Počet indexov ani veľkosť jednotlivých indexov nie je obmedzená. Obmedzenie je však v tom, že pole sa musí do voľnej pamäti zmestiť. Po zadaní príkazu

```
DIM c(100,100)
```

ohlási počítač chybu

```
4 Out of memory
```

čo značí, že pre pole c nie je v pamäti dosť miesta.

Upozornenie: V jednom príkaze DIM môže byť uvedené len jedno pole.

Existujú tiež **reťazcové polia**. Reťazec a reťazcové pole sa od seba líšia tým, že pole má konštantnú dĺžku a priradenie je vykonané, že zvyšok dĺžky, ktorý nie je obsadený, je nahradený medzerami. Môžeme mať tiež pole jednotlivých znakov (ale s jedným ďalším rozmerom). Meno reťazcového poľa je jedno písmeno a znak $. Reťazec reťazcové pole nesmú mať navzájom zhodné meno (na rozdiel od polí číselných). Nemôžeme použiť preto súčasne premenné

```
a$, a$(1,1)
```

Predpokladajte, že chcete pole a$ s piatimi reťazcami. Musíte sa rozhodnúť, ako dlhé budú tieto reťazce. Predpokladajte, že 10 znakov postačí. Pole definujte príkazom

```
DIM a$ (5,10)
```

Tak je definované pole 5x10 znakov, ale môžete tiež uvažovať o rade znakov jednotlivých:

```
a$(1)=a$(1,1)a$(1,2) … a$(1,10)
a$(2)=a$(2,1)a$(2,2) … a$(2,10)
…
…
a$(5)=a$(5,1)a$(1,2) … a$(5,10)
```

Teda, keď zadáte 2 indexy, potom dostanete jeden znak, ale keď nenapíšete druhý index, dostanete reťazec definovanej dĺžky, kde dĺžka reťazca je daná druhou položkou v príkaze DIM. Tak napr. A$(2,7) je znak reťazca A$(2). Mohli by sme vlastne tiež písať A$(2)(7). Skúste:

```
LET a$(2)="1234567890"
PRINT a$(2),a$(2,7)
```

a dostanete

```
1234567890   7
```

Pre posledný index (ten, ktorý môžeme vynechať) možno použiť techniku pre výber časti reťazca:

```
a$(2,4 TO 8)=a$(2)(4 TO 8)="45678"
```

Pamätajte:

V reťazcovom poli majú reťazce rovnakú dĺžku. Posledný index príkazu DIM určuje dĺžku každého reťazca.

Po zadaní

```
DIM a$(10)
```

uvidíte, že a$ sa chová ako reťazcová premenná s výnimkou, že má dĺžku 10 a po priradení hodnoty je zvyšok doplnený medzerami.

1.2.4. Použitie pre výpočtoch

Dostávame sa konečne k tomu, prečo sa počítač volá počítačom. Dokáže totiž veľmi rýchlo počítať, presnejšie povedané dokáže vykonávať **matematické operácie**. Uveďme si príklad:

```
10 LET a=3: LET b=4
20 LET c=a+b
30 PRINT c
```

Čo bude výsledkom programu?

Na riadku 10 sme priradili hodnotu premenným a, b.

Na riadku 20 počítač sčíta hodnoty premenných a, b a výslednú hodnotu priradí premennej c (t.j. c=3+4=7).

Na riadku 30 počítač vypíše hodnotu premennej c t.j. číslo 7.

Uveďme si ďalší príklad:

```
10 LET x=4
20 LET x=x+1
30 PRINT x
```

Niekomu sa môže zdať, že na riadku 20 je napísaný nezmysel, lebo nemôže platiť rovnosť x= x+1. To je pravda. Ide o to, že znamienko= nevyjadruje rovnosť, ale priradenie hodnoty premennej.

Počítač pracuje takto:

Na riadku 10sa premennej x priradí hodnota 4.

Vypočíta, čomu sa rovná x+1. Pretože x bolo 4, bude x+1=5.

Výslednú hodnotu 5 priradí počítač premennej x.

Vypíše sa hodnota premennej x, t.j. číslo 5.

Pri počítaní môžete čísla nielen sčítať, ale i odčítať, násobiť, deliť a umocňovať.

Ak chcete sčítať dve čísla, dáte medzi ne znamienko +, ak ich chcete odčítať, dáte medzi ne znamienko -.

V matematike ste boli zvyknutý dávať pri násobení medzi čísla x alebo boku, pri delení dvojbodku.

Pri práci s počítačom je to inak. Odteraz budete pri počítaní používať tieto znaky:

+ pre sčítanie
- pre odčítanie
***** pre násobenie
/ pre delenie
^ pre umocňovanie

Značky +, -, *, /, ^ budeme nazývať **aritmetické operátory**. Operátor sa musí písať vždy, nemožno ho vynechať. Keby sme napr. namiesto b=a*8 napísali b=a8, pre počítač to znamená, že premennej b má priradiť hodnotu premennej a8 (vynechaním * sme dostali meno premennej).

Spomenuli sme operáciu **umocňovanie**. Pre tých, ktorí nevedia, čo to je, nasleduje malé vysvetlenie.

Umocňovať číslo na mocninu znamená násobiť číslo samo sebou toľkokrát, koľko určuje mocnina. V matematike sa mocnina píše nad prvé číslo; pre počítač je to obťažné a tak používame symbol ^ :

```
2 ^ 1=2
2 ^ 2 = 2*2 = 4
2 ^ 2 = 2*2*2 = 8
2 ^ 4 = 2*2*2*2 = 16
```

Teda x ^ y znamená násobiť výraz x y-krát. Obvykle to má zmysel, keď y je celé kladné číslo. Nájdeme však takú definíciu, že to pracuje pre ďalšie hodnoty b. Uvažujme pravidlo:

a ^ (b+c) = (a ^ b) * (a ^ c)

Nemusíme uvažovať, že b a c sú celé kladné čísla, nasledujúce rovnosti platia obecne:

`a ^ 0 = 1`
`a ^ (-b) = 1/(a ^ b)`
`a ^ (1/b) =` b-ta odmocnina z a, je to číslo, ktoré musíme násobiť b-krát, aby vzniklo a.
`a ^ (b * c) = (a ^ b) ^ c`

Pokiaľ ste nič z tohto skôr nevideli, nemusíte si to pamätať. Stačí, ak budete zatiaľ vedieť:

a ^ (-1) = 1/a

V počítačovej terminológii je základným pojmom pojem **aritmetický výraz**. Čo to je?

Aritmetický výraz je kombinácia konštánt, premenných a funkcií spojených prípustnými aritmetickými operátormi (s funkciami sa zoznámime v ďalšej časti príručky).

Aritmetický výraz môže obsahovať viac operátorov, ktoré spájajú viac konštánt, premenných a funkcií (**operandov**).

Príklady aritmetických výrazov:

```
a+b-c
(a+b)*d
(e+f)^4-e^2
```

Ak obsahuje aritmetický výraz viac operátorov aj operandov, hodnota výrazu sa vyhodnocuje postupne podľa **priority** (prednosti) jednotlivých operátorov.

Priorita aritmetických operátorov je táto:

1. **^** umocnenie
2. ***, /*** násobenie a delenie
3. **+, -** sčítanie a odčítanie

Najvyššiu prioritu má umocňovanie, najnižšiu sčítanie a odčítanie. Pri rovnakej priorite sa operácie vykonávajú zľava doprava. Ak chcete poradie vykonávania zmeniť, použijete v aritmetickom výraze okrúhle zátvorky. Zátvorkové páry môžete vkladať do seba. Vtedy sa vyhodnotí najprv výraz v najvnútornejších zátvorkách, potom v druhých najvnútornejších atď. Zátvorky majú najvyššiu prioritu.

Ukážme si, ako sa vyhodnotí výraz

```
aa+(b-c)*d^2
```

Počítač spracuje výraz v týchto krokoch:

1. b-c
2. d^2
3. výsledok z 1. je vynásobený výsledkom z 2.
4. výsledok z 3 je pripočítaný k aa

Ak neviete presne, v akom poradí vyhodnotí počítač aritmetický výraz, použite zátvorky! Tým sa stane výraz aj prehľadnejší. Uveďme si príklady aritmetických výrazov a ich zápis v jazyku BASIC:

Aritmetický výraz	Zápis v jazyku BASIC
$\frac{a+b}{c}-c$	`(a+b)/c-d`

Aritmetický výraz	Zápis v jazyku BASIC
$\frac{a+b}{c-d}$	`(a+b)/(c-d)`
$a+\frac{b}{c \cdot d}$	`a+b/(c*d), a+b/c/d`
$\frac{a \cdot c}{b}+d$	`a/b*c+d, a*c/b+d, (a*c)/b+d`
$a+b \cdot c^d$	`a+b*c^d`

1.2.5. Práca s funkciami

Uvažujte stroj na výrobu klobás. Na jednom konci vložíte kus mäsa, pohnete pákou a na druhom konci vypadne klobása. Z bravčového sú bravčové, z rýb rybie, z hovädzieho hovädzie. Od stroja na klobásy sa počítač líši tým, že pracuje s číslami a reťazcami namiesto mäsa. Vložíte jednu hodnotu (nazvanú **argument**), urobíte nejakú operáciu a dostanete ďalšiu hodnotu (výsledok):

MÄSO.....STROJ NA KLOBÁSY.....KLOBÁSY

Podobne pracujú funkcie:

ARGUMENTY.....FUNKCIE.....VÝSLEDOK

Rôzne argumenty dávajú rôzne výsledky a pokiaľ je argument úplne nevhodný, funkcia sa nevykoná a počítač hlási chybu.

Rovnako ako existujú rôzne stroje, jeden na klobásy, druhý na oblečenie, tretí na konzervy, budú rôzne funkcie rôzne počítať. Každá bude mať vlastné meno, ktoré ju odlíši od ostatných. Funkciu použijete tak, že napíšete meno funkcie, za ktorým nasleduje argument funkcie.

Pri riešení matematických problémov sa používajú často všeobecne známe matematické funkcie. Počítač vie s týmito funkciami tiež pracovať, vie vypočítať ich hodnoty.

V tejto kapitole sa budeme zaoberať **štandardnými** (vstavanými) funkciami, ktoré môžete použiť v jazyku BASIC, aj keď niektoré funkcie ešte nepoznáte. Každá štandardná funia je označená menom, za ktorým sa uvádza argument, ktorý môže byť uzavretý v okrúhlych zátvorkách (ale nemusí).

Dohodneme sa teraz, že ak to nebude v konkrétnom prípade povedané inak, na mieste, kde v popise funkcie alebo príkazu môže byť číslo, môže byť tiež aritmetický výraz. Ak sa požaduje celé číslo, z hodnoty výrazu sa automaticky berie celá časť.

Analogicky na mieste reťazca môže byť výraz, ktorého hodnota je reťazec.

1.2.5.1. Základné funkcie

Ako prvú uvedieme funkciu **LEN**, ktorá dĺžku reťazca. Jej argumentom je reťazec, ktorého dĺžku chcete zistiť.

Napr. zadajte:

```
PRINT LEN "Skalica"
```

a počítač napíše 7, čo je počet písmen reťazca (na získanie LEN, ako u väčšiny funkčných mien, musíte použiť rozšírený mód klávesnice a potom stlačte kláves K). Pokiaľ spojíte funkcie pomocou operátorov do jedného výrazu, potom budú funkcie spracované pred operáciami. Toto pravidlo môžete samozrejme meniť použitím zátvoriek. Napr. sú tu dva výrazy, ktoré sa rozlišujú len v zátvorkách a operácie sa v každom prípade vykonávajú v inom poradí (hoci sú výsledky rovnaké):

```
LEN "Jano" + LEN "Sevcik"     LEN ("Jano"+"Sevcik")
4+LEN "Sevcik"                LEN ("JanoSevcik")
4+6                           LEN "JanoSevcik"
1.                            10
```

Ďalšou funkciou je funkcia **STR$**, ktorá konvertuje (prevádza) čísla na reťazce; argumentom je číslo a výsledkom je reťazec.

Skúste napísať:

```
LET a$= STR$ 1e2
```

čo má ten istý význam ako

```
LET a$="100"
```

Ak zadáte

```
PRINT LEN STR$ 100.0000
```

a dostanete odpoveď 3, lebo STR$ 100.0000 ="100"

Funkcia **VAL** má opačný význam ako STR$, mení reťazec na číslo.

Napr.

```
VAL "3.5"=3.5
```

Funkcia VAL je obrátená k funkcií STR$, pretože ak máte nejaké číslo, aplikujete naň STR$ a potom ešte VAL, výsledkom je pôvodné číslo. Pokiaľ sa však vezme reťazec a v ňom sa aplikuje VAL a potom STR$, nie vždy dostaneme pôvodný reťazec. VAL je veľmi mocná funkcia, pretože reťazec, ktorý je argumentom, nie je obmedzený len na číslo, ale môže byť i aritmetický výraz.

Napr.:

```
VAL "2*3"=6
```
alebo dokonca
```
VAL ("2"+"*3")=6
```

Tu sú spojené dve dôležité veci.

Po prvé argument VAL je spracovaný ako reťazec. Reťazcové vyjadrenie "2" + "*3" je najprv spojené do reťazca "2*3". Potom je z reťazca zostavený výraz, ako keby nebolo úvodzoviek, je vynásobené 2*3 s výsledkom 6.

Teraz sa Vás pokúsime zmiasť:

```
PRINT VAL "VAL""VAL""""2""""""""
```

(Pamätajte, že vnútorné úvodzovky v reťazci musia byť písané dvojito). Čo bude výsledkom tohto príkazu?

Existuje podobná funkcia ako VAL, ale asi menej používaná, nazýva sa **VAL$**. Jej argumentom je stále reťazec, ale výsledok je takisto reťazec. Táto funkcia funguje ako VAL v dvoch krokoch: najprv je argument chápaný ako reťazec, potom sú odstránené úvodzovky a všetko je chápané ako číslo. VAL$ funguje rovnako, ale výsledok je považovaný opäť za reťazec. Preto:

```
VAL$ """dva""" = "dva"
```

Teraz zadajte:

```
LET a$ = "99"
```

a vypíšte hodnoty nasledovných výrazov:

```
VAL a$, VAL "a$", VAL """a$""",
VAL$ a$, VAL$"a$", VAL$"""a$"""
```

Niektoré z týchto príkladov budú pracovať, niektoré nie. Skúste nájsť odpoveď, prečo.

Funkcia **INKEY$** (nemá argument) číta klávesnicu. Pokiaľ stlačíte 1 kláves (alebo SHIFT a jeden kláves), potom výsledok je znak, ktorý dostanete stlačením tohto klávesu v L-móde. Ak nie je stlačený žiadny kláves, výsledkom je prázdny reťazec.

Nasledujúci program pracuje ako písací stroj.

```
10 IF INKEY$ <> "" THEN GO TO 10
20 IF INKEY$ = "" THEN GO TO 20
30 PRINT INKEY$ ;
40 GO TO 10
```

Na riadku 10 sa čaká, až dáte prst preč z klávesnice a na riadku 20 sa čaká, až stlačíte nový kláves. Pamätajte, že na rozdiel od príkazu INPUT funkcia **INKEY$** nevyžaduje potvrdenie klávesom ENTER.

Porozmýšľajte čo sa stane, ak vypustíte riadok 10 v poslednom programe!

Ďalší spôsob použitia INKEY$ je využitie s PAUSE:

```
10 PAUSE 0
20 PRINT INKEY$ ;
30 GO TO 10
```

A do tretice:

```
10 IF INKEY$ ="" THEN GO TO 10
20 PRINT AT 11,14;"jaj !"
30 IF INKEY$ <> "" THEN GO TO 30
40 PRINT AT 11,14;"     "
50 GO TO 10
```

Písmená, číslice a ďalšie symboly, ktoré sme používali v reťazcoch, sa nazývajú znaky a tvoria abecedu alebo **systém znakov**, ktoré používa počítač. Väčšina týchto znakov sú jednoduché symboly, ale tiež symboly, ktoré reprezentujú kľúčové slová jazyka BASIC (PRINT, STOP, ...). Existuje celkom 256 znakov, ktoré majú **kód** od 0 do 255 (kód je číslo, pod ktorým je znak vnútorne uložený v počítači). Ich úplný zoznam je uvedený v prílohe. Prevod medzi znakmi a ich kódmi umožňujú funkcie CODE a CHR$. Majú tvar

CODE reťazec
CHR$ číslo

Funkcia **CODE** je aplikovaná na reťazec, jej výsledkom je kód prvého znaku v reťazci alebo 0, pokiaľ ide o prázdny reťazec. Funkcia **CHR$** je aplikovaná na číslo a pre čísla 32 až 255 je jej výsledkom znak, ktorého kód je daný argumentom funkcie:

```
CODE "ABC"=65    , lebo kód "A" je 65
CODE "B"=66      , lebo kód "B" je 66
CHR$ 65="A"
CHR$(3*22)="B"
```

Nasledujúci program tlačí na obrazovku všetky znaky s kódom 32 až 255:

```
10 FOR a=32 TO 255: PRINT CHR$ a;: NEXT a
```

Výpis možno rozdeliť na dve časti. Prvá časť sú znaky s kódom po hodnotu 127. Všetky tieto znaky (okrem dvoch) sú z kódu známeho ako **ASCII** (tento kód sa používa u väčšiny mikropočítačov).

Zvyšok znakov nie je časťou kódu ASCII a je použitý v počítačoch typu ZX Spectrum. Prvý medzi nimi je medzera a 15 čiernobielych vzorov. Sú to grafické symboly a používajú sa na kreslenie obrázkov. Môžete ich vkladať z klávesnice použitím grafického módu. Keď stlačíte GRAPHICS (CAPS SHIFT + 9), kurzor sa zmení na **G**. Teraz stlačte číslice 1 - 8 a budú sa objavovať grafické symboly, ktoré sú zobrazené na klávesoch, so stlačením SHIFT dávajú invertovaný symbol, tj. čierne sa stáva bielym a naopak. V grafickom móde, bez ohľadu na iné klávesy, kláves 9 nastavuje späť L-mód a kláves 0 funguje ako DELETE.

Po grafických symboloch potom uvidíte kópie abecedy od A do U. Tieto znaky môžete modifikovať sami. Keď je počítač prvý raz zapnutý, sú k dispozícii len znaky základné. Sú to takzvané znaky **užívateľsky definovanej grafiky** (udg), ktoré sa vkladajú z klávesnice v grafickom móde **G** použitím písmen od A do U. Postup definovania užívateľských znakov bude uvedený neskôr.

Po užívateľských znakoch nasledujú kľúčové slová jazyka BASIC.

Mohli ste si všimnúť, že sme netlačili prvých 32 znakov s kódmi 0 až 31. Sú to **riadiace znaky**. Nevytvárajú nič, čo sa dá tlačiť, ale majú vplyv pri výpise na obrazovku alebo sú využité na riadenie iných vecí. Obrazovka používa 3 znaky s kódmi 6, 8 a 13.

CHR$ 6 nastavuje ukazovateľ pre výpis začiatok ďalšej zóny, má ten istý význam ako čiarka vo význame oddeľovača v príkaze PRINT.

Porovnajte výsledok týchto dvoch príkazov:

```
PRINT 1; CHR$ 6;2
PRINT 1,2
```

Výhodné je využitie, ak tento znak vložíte priamo do reťazca:

```
LET a$="1"+CHR$ 6+"2"
PRINT a$
```

CHR$ 8 posúva tlačovú pozíciu o 1 späť. Skúste napísať:

```
PRINT "1234"; CHR$ 8;"5"
```

Výsledkom je výpis 1235, lebo najprv sa vypíše 1234, potom sa ukazovateľ pre výpis nastaví späť a znak 4 sa prepíše znakom 5.

CHR$ 13 posúva ukazovateľ výpisu na začiatok ďalšieho riadku.

Využitie ďalších riadiacich znakov bude uvedené neskôr.

Použitím kódov a funkcie CHR$ môžete vytvoriť reťazce s ľubovoľnými znakmi, nielen s písmenami. Počet všetkých znakov je 256.

Tu sa dostávame k pojmom **porovnanie reťazcov, usporiadanie reťazcov**. Nasledujúce reťazce sú usporiadané vzostupne:

```
CHR$ 3 + "ZOOLOGICKA ZAHRADA"
CHR$ 8 + "ABECEDA"
" AAAARGH!"
"(okruhle zatvorky)"
"100"
"129.95"
"AAAARGH!"
"PRINT"
"Zoo"
"aaaargh!"
"zoo"
"zoologicka"
```

Všimnite si, že veľké písmená sú pred malými.

Reťazce sú usporiadané podľa nasledujúcich pravidiel:

Znaky sa berú postupne od začiatku a porovnávajú sa ich kódy.

Najprv sa porovnajú prvé znaky. Pokiaľ sú rozdielne, potom jeden má kód menší než druhý a reťazec, ktorého znak má nižší kód je menší. Keď sú prvé znaky rovnaké, porovnávajú sa rovnakým spôsobom znaky ďalšie a reťazce sú zhodné, keď sú si všetky znaky rovné.

Relácie =, <, >, <=, >=, <> sú použiteľné pre reťazce i pre čísla. Pre reťazce < znamená "je pred" a > znamená "je za".

Nasledujúce výrazy sú pravdivé:

```
"AA man" < "AAAA"
" AAAA" < "AA man"
```

< = a > = pracuje podobne ako s číslami

```
"Rovnaky retazec" <= "Rovnaky retazec"     je pravdivé
"Rovnaky retazec" < "Rovnaky retazec"      je nepravdivé
```

Odštartujte tento program:

```
10 INPUT a
20 PRINT CHR$ a;
30 GO TO 10
```

Pokiaľ budete experimentovať, zistíte, že ak nie je zadané číslo v rozsahu 0 až 255, program sa zastaví s hlásením

```
B Integer out of range
```

Niekomu je čudné, prečo je rozsah práve 0 až 255. Tieto hodnoty sú dané technickými prostriedkami počítača. Podrobnejší popis bude uvedený v časti pre náročných. Zatiaľ si povieme to, že pamäť počítača sa skladá z "buniek", táto bunka sa nazýva **byte** (čítaj bajt) a zmestí sa do nej celé číslo od 0 do 255. Každý znak je uložený v jednom byte, preto kód znaku je od 0 do 255.

Každý byte má svoju adresu, čo môže byť celé číslo od 0 do 65535. Každý byte sa skladá z 8 menších častí, táto časť sa nazýva **bit** (bit je skratka anglického binary digit a znamená dvojková číslica). Jeden bit môže mať hodnotu 0 alebo 1.

Kto sa vyzná v dvojkovej sústave, využije možnosť zadávať čísla v tejto sústave pomocou funkcie **BIN**, ktorá má tvar:

BIN dvojk_číslo

kde dčíslo je dvojkové číslo (teda číslo v dvojkovej sústave) o maximálnej dĺžke 16 bitov, takže hodnota tejto funkcie bude od 0 do 65535. Presvedčíte sa, že

```
BIN 101 = 5
BIN 11111111 = 255
BIN 11111111 11111111 = 65535
```

Z toho by malo byť jasné, že adresa je daná hodnotou 16 bitov.

Keďže vieme adresovať každý byte pamäti a vieme, čo môže obsahovať, popíšeme si teraz, ako môžete s pamäťou pracovať.

Hodnotu jedného bytu v pamäti je možné nastaviť príkazom **POKE**, ktorý má tvar:

POKE adresa,hodnota

kde adresa a hodnota majú známy význam.

Pre načítanie hodnoty bytu z pamäti sa používa funkcia **PEEK** v tvare:

PEEK adresa

ktorá udáva hodnotu bytu so zadanou adresou.

Príkazom

```
POKE 30000,5
```

alebo

```
POKE 30000,BIN 101
```

nastavíme hodnotu bytu s adresou 30000 na hodnotu 5. Čí je to skutočne pravda zistíme príkazom

PRINT PEEK 30000

Príkaz POKE umožňuje meniť takzvané systémové premenné, čo podstatne ovplyvňuje "chovanie" počítača. Podrobnejšie sa o tom dozviete v kapitole o systémových premenných.

1.2.5.2. Matematické funkcie

Táto kapitola sa zaoberá ďalšími štandardnými funkciami.

Je celkom možné, že niektoré funkcie hneď nepoužijete, lebo ste sa s nimi doteraz nestretli. Zahŕňa funkcie SGN, ABS, INT, SQR, EXP, LN, trigonometrické funkcie SIN, COS, TAN a ich inverzné funkcie ASN, ACS, ATN.

Funkcia **SGN x** je funkcia signum (znamienková funkcia). Jej argument i výsledok sú čísla. Výsledok je +1, keď je argument kladný, 0 keď je argument 0 a -1, keď je argument záporný.

Funkcia **ABS x** je ďalšia funkcia, ktorej argument i výsledok je číslo. Mení argument na kladné číslo.

Tak napr.:

```
ABS-3.2 = ABS 3.2 = 3.2
ABS 1000 = 1000
```

Funkcia **INT x** znamená "celá časť". Celá časť z čísla x je najväčšie celé číslo, ktoré je menšie alebo rovné ako zadané číslo. Platí:

INT(x) <= x <= INT(x)+1

Uveďme si konkrétny príklad:

```
INT 3.9 = 3
```

Pozor na záporné čísla:

```
INT - 3.9 = -4
```

Funkcia **SQR x** vypočítava druhú odmocninu čísla x.

```
SQR 4 = 2            , pretože 2×2=4
SQR 0.25 = 0.5       , pretože 0.5×0.5=0.25
SQR 2 = 1.414211136
```

Pokiaľ umocňujete číslo (i záporné), výsledok je vždy kladný. To znamená, že záporné čísla nie je možné odmocňovať a pokiaľ tak urobíte, dostanete správu

```
A Invalid argument
```

Exponenciálna funkcia **EXP x** vypočíta hodnotu e^x (e na x), kde e je základ prirodzených logaritmov (e = 2.7182818...).

EXP x = e ^ x

Ľahko zistíte hodnotu čísla e. Zadajte

```
PRINT EXP 1
```

a dostanete čísla e, pretože EXP 1 = e ^ 1 = e. Je to len približné vyjadrenie, hodnota e sa nedá nikdy presne vyčísliť.

Funkcia **LN x** je prirodzený logaritmus x. Obecne platí, že logaritmus čísla x so základom z je výraz, na ktorý musíme umocniť základ z, aby sme získali číslo x a píše sa $\log_z x$. Preto platí:

z ^ ($\log_z x$) = x $\log_z (z^x)=x$

V škole ste možno poznali logaritmy so základom 10, to sú bežné logaritmy. Počítač má funkciu LN, ktorá počíta logaritmy so základom e. Na výpočet logaritmu s iným základom použite vzťah

$\log_z x$ = LN x / LN z .

Argument x funkcie LN musí byť kladné číslo.

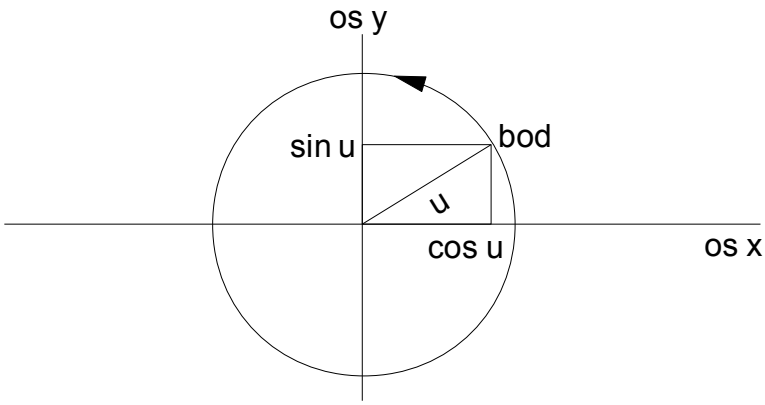
Keď je daná kružnica, môžete spočítať jej obvod násobením priemeru číslo **PI**. Číslo PI má v počítači hodnotu 3.1415927. Získate ho v rozšírenom móde stlačením M. PI funguje ako každá iná číselná premenná a má dôležité použitie hlavne s funkciami **SIN**, **COS**, **TAN**, **ASN**, **ACS**, **ATN**.

Pre tých, ktorí nevedia o týchto funkciách nič, sa pokúsime teraz vysvetliť ich význam. Nakreslite si dve osi kolmé na seba (jednu vodorovnú, jednu zvislú), ich spoločný bod nech je stredom kružnice, ktorá má polomer l. Po kružnici sa začne pohybovať z pravej strany vodorovnej osi v smere proti pohybu hodinových ručičiek. Súradnice bodu x, y v tejto súradnicovej sústave (poloha bodu vzhľadom na os x a y) sa nazývajú kosínus a sínus uhla u, uhol určuje otočenie bodu od začiatočného bodu (napravo od stredu na osi x). Uhol u budeme zadávať v **radiánoch** (360 stupňov = 2*PI radiánov). Pre prevod medzi stupňami a radiánmi platia vzorce:

```
rad = stupne * PI/180
stupne = rad * 180/PI
```

Napr. platí:

```
SIN 0 = 0
COS 0 = 1
SIN PI/2 = 1
COS PI = -1
```



Všimnite, si, že keď sa bod dostáva vľavo od osi y, kosínus je záporný a keď sa bod dostáva pod os x, stáva sa i sínus záporným. Ďalšia vlastnosť je, že keď sa b od dostane späť do východzej pozície, majú sínus a kosínus opäť rovnaké hodnoty ako na začiatku pohybu. Preto platí:

```
SIN (a+2*PI) = SIN a
COS (a+2*PI) = COS a
```

Funkcia tangens je definovaná ako podiel sínus/kosínus. Zodpovedajúca funkcia na počítači je TAN.

Občas potrebujeme opačný postup, kedy poznáme hodnotu jednej z týchto funkcií a potrebujeme určiť odpovedajúci uhol. Vtedy použijeme funkcie arcsin (na počítači ASN), arccos (ACS) a arctg (ATN).

1.2.6. Náhodné čísla

Táto kapitola pojednáva o funkcií **RND** a kľúčovom slove **RANDOMIZE**, pomocou ktorých pracujeme s náhodnými číslami. Obidve na rovnakom klávese (T). RANDOMIZE je označené skratkou RAND. Funkcia RND nemá žiaden argument. Vždy, keď ju použijete, je jej výsledkom nové náhodné číslo medzi 0 a 1. Niekedy môže dať hodnotu 0, ale nikdy nie 1. Skúste:

```
10 PRINT RND
20 GO TO 10
```

a vidíte, ako sa menia hodnoty funkcie. V skutočnosti RND nie je úplne náhodné číslo, pretože ide vždy o číslo z postupností 65536 čísiel (táto postupnosť je pevne daná). Preto hovoríme, že RND je pseudonáhodné číslo. Ako sme už uviedli, dáva RND náhodné čísla medzi 0 a 1, ale môžete ľahko získať čísla v inom rozsahu. Napr. 5 * RND je číslo medzi 0 a 5, 1.3+0.7*RND je medzi 1.3 a 2. Na získanie celých čísiel použite funkciu INT (pamätajte, že INT zaokrúhľuje vždy dole), ako 1+INT(RND*6), čo simuluje hádzanie kockou. RND*6 je v rozsahu 0 - 6, ale v skutočnosti nikdy nedostanete 6, preto +1.

Príkaz **RANDOMIZE** sa používa na to, aby odštartoval RND v určitom mieste postupnosti čísiel, ako ukazuje nasledujúci program:

```
10 RANDOMIZE 1
20 PRINT RND
30 GO TO 20
```

Po každom vykonaní RANDOMIZE 1 začne postupnosť RND číslom 0.0022735596. Môžete samozrejme použiť i iné čísla od 1 - 65535 v príkaze RANDOMIZE, potom je začiatok postupnosti náhodných čísiel v iných miestach. Samotné RANDOMIZE (RANDOMIZE 0 má účinok) je rozdielne, pretože skutočne tvorí RND, ako ukazuje ďalší program:

```
10 RANDOMIZE
20 PRINT RND : GO TO 10
```

Postupnosť, ktorú dostanete, nie je náhodná, pretože RANDOMIZE využíva čas od zapnutia počítača. Od tej doby je každé RANDOMIZE vykonané rovnako a preto i RND sú rovnaké. Na získanie náhodných čísiel by bolo lepšie nahradiť príkaz GO TO 10 príkazom GO TO 20.

Poznámka: väčšina verzií jazyka BASIC používa RND a RANDOMIZE na tvorbu náhodných čísiel, ale nie všetky ich získavajú rovnakým spôsobom.

1.2.7. Rozhodovanie v programe

Všetky programy, ktoré ste doteraz videli, mali jednoduchú štruktúru. Program sa vykonával postupne od začiatku do konca, prípadne sa vracal na určenú časť.

V praxi je výhodné, ak počítač sám môže rozhodovať na základe nejakej informácie o ďalšom postupe v činnosti. Na to sa používa príkaz **IF - THEN**, ktorý sa nazýva tiež **príkaz podmieneného skoku** (podmieneného preto, lebo ďalšie vykonávanie programu je závislé od splnenia určitej podmienky). Príkaz má tvar

```
IF podmienka THEN prikazy
```

Ak nie je splnená podmienka uvedená medzi slovami IF a THEN, vykonávanie programu pokračuje na ďalšom riadku.

Ak je podmienka splnená, vykonávanie programu pokračuje vykonávaním príkazov za slovom THEN.

Napr. použitím NEW vymažte predošlý program, vložte a odštartujte nasledujúci program (je to hra na hádanie celého čísla od 0 do 9, toto číslo si "vymyslí" počítač).

```
10 REM hadanie cisla
20 PRINT "Hadaj cislo od 0 do 9"
30 LET x = INT(10* RND)
40 INPUT "hadaj cislo", y
50 IF x=y THEN GO TO 80
60 PRINT "neuhadol si":PRINT
70 GO TO 40
80 PRINT "uhadol si":PRINT "KONIEC"
```

Program vytvorí na riadku 30 náhodné číslo od 0 do 9 a priradí ho premennej x. Po zadaní čísla z klávesnice a jeho uložení do premennej y sa dostávame na riadok 50. Ak je splnená podmienka x=y (číslo sme uhádli), vykonávanie programu pokračuje za časťou THEN, teda vykoná odskok na riadok 80.

Ak nie je splnená podmienka, pokračuje sa na riadku 60. Po výpise "neuhadol si" sa z riadku 70 vracia program na riadok 40.

Program môžeme skrátiť zmenou riadku 50 na

```
50 IF x=y THEN PRINT "uhadol si":PRINT "koniec":STOP
```

a riadok 80 môžeme zmazať.

Podmienka x=y v prechádzajúcom príklade sa nazýva jednoduchá podmienka. Jednoduché podmienky porovnávajú dve čísla alebo dva reťazce a môžu testovať, či sú si čísla rovné, či je jedno väčšie než druhé a môžu testovať, či dva reťazce si sú rovné alebo sa alfanumericky predchádzajú. Používajú sa značky:

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

ktoré sa nazývajú relačné operátory. Ich význam je nasledovný:

= znamená rovná sa; i keď symbol je rovnaký ako v príkaze LET, má tu celkom iný význam.

< (SYMBOL SHIFT + R) znamená je menší než
1 < 2, -2 <-1, -3 < 1 sú pravdivé podmienky,
1 < 0, 0 < -2 sú nepravdivé podmienky

> (SYMBOL SHIFT + T) znamená je väčší než, t. j. opačné ako <

<= (SYMBOL SHIFT + Q) znamená menší alebo rovný
1 <= 2, 2 <= 2 sú pravdivé podmienky,
3 <= 2, 0 <= -1 sú nepravdivé podmienky

>= (SYMBOL SHIFT + E) znamená je väčší alebo rovný

<> (SYMBOL SHIFT + W) znamená nerovnosť a stojí ako protiklad k operátoru =

Upozorňujeme, že symboly <=, >=, <> nemôžeme zložiť z dvoch znakov, ale musíme ich napísať na jeden krát.

V matematike sa píšu výrazy tiež ako

2<3<4,

čo znamená

2 < 3 a zároveň 3 < 4,

ale v jazyku BASIC to nie je možné.

V programe môžete použiť podmieňovací príkaz viackrát, čo umožňuje vetviť program na viac ako dve možnosti.

To si ukážeme na príklade hádanie celého čísla od 0 do 99, pričom nám počítač trochu pomôže.

```
10 REM hadanie cisla od 0 do 99
20 PRINT "Hadaj cislo od 0 do 99"
30 LET x = INT(100* RND)
40 INPUT "hadaj cislo", y
50 IF x=y THEN GO TO 100
60 IF y>x THEN GO TO 90
70 IF y<x THEN GO TO 80
80 PRINT "tvoje cislo je mensie":PRINT:GO TO 40
90 PRINT "uhadol si":PRINT "koniec"
```

Ak číslo neuhádneme, vypíše počítač informáciu o tom, či je hádané číslo väčšie alebo menšie a program pokračuje na riadku 30. V programe sme použili trikrát podmieňovací príkaz.

Riadok 70 môžeme vynechať. Ak sa dostane riešenie programu až na tento riadok, podmienky na riadkoch 50 a 60 nie sú splnené, takže musí platiť podmienka na riadku 70.

Je zrejmé, že program sa dá opäť skrátiť. Skúste to!

Teraz si ukážeme jeden praktický príklad použitie podmieneného príkazu. Nasledujúci program usporiada 3 čísla podľa veľkosti.

```
10 REM usporiadanie cisiel
20 INPUT "zadaj tri cisla",a,b,c
30 IF a<b THEN GO TO 50
40 LET x=a:LET a=b:LET b=x
50 IF a<c THEN GO TO 70
60 LET x=a:LET a=c>:LET c=x
70 IF b<c THEN GO TO 90
80 LET x=b:LET b=c:LET c=x
90 PRINT "usporiadane cisla";a,b,c
```

Program pracuje tak, že hodnoty premenných a, b ,c sa vymieňajú navzájom tak, aby na konci programu boli premenné a, b, c usporiadané od najmenšieho po najväčšie. Porovnávajú sa vždy dve hodnoty. Zoberme si napr. riadok 30.

Ak a<b, čísla sú usporiadané a pokračujeme na riadku 50 ďalšou dvojicou.

Ak neplatí, že a<b, hodnoty premenných a, b navzájom vymeníme na riadku 40. Hodnotu a uložíme do pomocnej premennej x príkazom LET x=a, lebo nasledujúcim príkazom LET a=b by sme hodnotu premennej a stratili. Až potom môžeme príkazom LET b=x priradiť premennej b pôvodnú hodnotu premennej a.

Analogicky sa robí porovnanie pre ďalšiu dve dvojice čísiel. V podmienke môžeme porovnávať aj reťazce.

Analogicky ako je príklad pre usporiadanie troch čísiel, môžeme nahradením premenných a, b, c, x premennými a$, b$, c$, x$ dostať program pre usporiadanie troch reťazcov.

Vyskúšajte to!

1.2.7.1. Logické operátory

Pri rozhodovanie môžeme z jednoduchých podmienok vytvoriť podmienku zloženú, ktorá sa nazýva **logický výraz**. Podmienky spájame pomocou **logických operátorov**. V jazyku BASIC používame tieto logické operátory:

Symbol	Príklad	Význam
NOT	**NOT A**	negácia (opak); ak A je pravdivá podmienka, potom NOT A je nepravdivá (a naopak)
AND	**A AND B**	logický súčin; A AND B je pravdivá podmienka, ak sú pravdivé podmienky A a B súčasne, inak je A AND B nepravdivá
OR	**A OR B**	logický súčet; A OR B je pravdivá podmienka, ak je pravdivá aspoň jedna z podmienok A a B, inak je nepravdivá

Význam logických operátorov si objasníme na konkrétnych príkladoch. Zoberme do úvahy celé čísla s nasledujúcimi podmienkami:

A: x > 3 t.j. 4, 5, 6, 7, ...
B: x < 7 t.j. 6, 5, 4, 3, ...

Potom

`NOT A`	`x <= 3`	t. j. 3, 2, 1, ...
`A AND B`	`x>3 AND x<7`	t. j. celé čísla väčšie ako 3 súčasne menšie ako 7, čiže 4, 5, 6.
`A OR B`	`x>3 OR x<7`	t. j. celé čísla väčšie ako 3 alebo menšie ako 7, čiže všetky celé čísla

Použitie v programe ukazuje táto časť programu:

```
10 INPUT "Zadaj cislo od 10 do 20",a
20 IF a<10 OR a>20 THEN PRINT "Este teraz":GO TO 10
30 …
```

Po chybnom zadaní čísla nás počítač nepustí ďalej, až dokiaľ nezadáme číslo z uvedeného rozsahu.

Logické výrazy môžu byť zapísané s AND, OR, NOT rovnako ako aritmetické výrazy môžu byť zapísané s AND, OR, NOT rovnako ako aritmetické výrazy vyjadrujeme pomocou +, -, atď. Pokiaľ je to potrebné, použite zátvorky.

Priorita logických operátorov je nasledovná:

1) NOT
2) AND
3) OR

V prípade rovnocenných operátorov sa postupuje zľava doprava. NOT je operátor s jedným argumentom, jeho priorita je najvyššia z logických operátorov. Preto jeho argument nepotrebuje zátvorky, aj keď podmienka obsahuje AND alebo OR (alebo obidve).

Teraz si uvedieme niekoľko vlastností logických operátorov, platnosť ktorých si skúste preveriť na konkrétnych príkladoch (ako pomoc použite uvedený príklad s číslami).

NOT a=b znamená to isté ako NOT(a=b) a to isté ako a<>b. <> je negácia =, znamená to, že je pravdivá len vtedy, keď rovnosť je nepravdivá. Taktiež a<>b je to isté ako NOT a=b.

Ďalej NOT a<>b je to isté, ako a=b. Presvedčíte sa, že >= a <= sú negácie <, > a tak NOT môžete niekedy vynechať zmenením relácií. Ďalej platí:

NOT(výraz1 AND výraz2) je to isté ako NOT výraz1 OR NOT výraz2
NOT(výraz1 OR výraz2) je to isté ako NOT výraz1 AND NOT výraz2

Pri použití NOT v programe nie je nutné používať medzi vzťahmi zátvorky.

Ďalšia časť výkladu je trochu komplikovaná a môžete ju zatiaľ preskočiť (pri hlbšom štúdiu si ju však preštudujte).

Skúste zadať:

```
PRINT 1=2,1<>2
```

Asi očakávate syntaktickú chybu, ale všetko je poriadku.
Počítač totiž prevádza logické hodnoty na číslo, s ktorými vie pracovať podľa nasledujúcich pravidiel:

a) relácie =, <, >, <=, >=, <> dávajú číselné výsledky:

   1 pre pravdivú podmienku
   0 pre nepravdivú podmienku;

   preto vyššie uvedený príkaz PRINT tlačí hodnotu 0 pre podmienku 1=2, pretože ide o nepravdivú podmienku, podmienka 1<>2 dáva hodnotu 1, lebo je pravdivá.

b) v príkaze IF podmienka THEN

   môže byť podmienka akýkoľvek aritmetický výraz. Pokiaľ je jeho hodnota 0, potom sa chová ako keby tam bola nepravdivá podmienka. Všetky ostatné hodnoty (vrátane 1) sa berú ako pravdivá podmienka. Preto

   IF podmienka

   v skutočnosti znamená

   IF podmienka <> 0 THEN ...

c) AND, OR, NOT sú operácie tvoriace číselné hodnoty

x AND y	má hodnotu x, pokiaľ y je pravdivé (nie 0); 0, pokiaľ je y nepravdivé
x OR y	má hodnotu 1, keď je y pravdivé (nie 0); x, keď je y nepravdivé
NOT x	má hodnotu 0, keď x je pravdivé (nie 0); 1, keď x je nepravdivé

Nakoľko sme si vysvetlili doteraz všetky potrebné pojmy, uvedieme si prioritu vyhodnocovanie výrazov:

1) vyhodnotenie zátvoriek
2) vyhodnotenie funkcie
3) umocňovanie
4) unárne mínus
5) násobenie a delenie
6) sčítanie a odčítanie
7) relačné operátory
8) negácia
9) logický súčin
10) logický súčet

V prípade rovnakej priority sa postupuje zľava doprava. Ak chcete zmeniť poradie vyhodnocovania, použite zátvorky (majú najvyššiu prioritu).

Nakoniec si ukážme, ako sa dajú použiť zložité logické výrazy v programe:

```
10 INPUT a
20 INPUT b
30 PRINT (a AND a>=b)+(b AND a<b)
40 GO TO 10
```

Výsledkom je výpis väčšieho z dvoch čísiel a, b.

Analogicky sa používajú reťazcové hodnoty v podmienkových výrazoch, ale len pri použití AND.

x$ AND y má hodnotu	x$, keď y nie je 0 "", keď y je 0,

Prezrite si nasledujúci program, ktorý vypíše dva vstupujúce reťazce v abecednom poradí.

```
10 INPUT "vlozte dva retazce",a$, b$
20 IF a$>b$ THEN LET c$= a$: LET a$= b$: LET b$= c$
30 PRINT a$;" ";("<" AND a$<b$)+("=" AND a$=b$);" ";b$
40 GO TO 10
```

1.2.8. Použitie cyklu v programe

Pri programovaní sa veľmi často opakujú tie isté výpočty. Pomocou príkazu **GO TO** môžeme vracať riešenie programu na žiadané opakované výpočty.

Máme za úlohu sčítať celé čísla od 1 do 30.

```
10 LET s=0:LET c=1
20 LET s=s+c
30 LET c=c+1
40 IF c<=30 THEN GO TO 20
50 PRINT "sucet = ";s
```

Na riadku 10 nastavíme začiatočné hodnoty s a c.

Premenná c prestavuje číslo, ktoré máme do celkového súčtu pričítať. Premenná s predstavuje súčet čísiel. Okamžitá hodnota s predstavuje medzisúčet čísiel od 1 do c.

Na riadku 20 pričítame do medzisúčtu číslo c, kde je nasledujúce číslo z požadovaných čísiel.

Na riadku 40 testujeme, či je takto vytvorené číslo <= 30. Ak je, program pokračuje na riadku 20.

Ak je číslo väčšie ako 30, v medzisúčte sú už všetky čísla od 1 do 30 sčítané, podmienka v príkaze IF - THEN nie je splnená a počítač vypíše celkový súčet (riadok 50).

Opakované využívanie časti programu nazývame **programový cyklus**.

Pri použití cyklu platia nasledujúce zásady:

a) pred vstupom do cyklu nastavíme začiatočné hodnoty premenných, ktoré v cykle využívame (v našom prípade s, c)
b) v cykle sa hodnota niektorých premenných mení o určenú hodnotu (c sa zvyšuje o 1)
c) v cykle sa testuje, či je splnená podmienka ukončenia cyklu; ak nie je splnená, skáče sa na začiatok cyklu

Programový cyklus, ktorý má tieto vlastnosti, môžeme jednoduchšie vytvoriť pomocou príkazu **FOR-TO-STEP-NEXT**.

Začiatok cyklu má tvar:

```
FOR premenná = od TO po STEP krok
```

Koniec cyklu sa označuje

```
NEXT premenná
```

Premenná je takzvaná **riadiaca premenná (parameter cyklu)**, ktorej meno môže byť len jedno písmeno.

Výrazy od, po, krok sú ľubovoľné aritmetické výrazy.

Časť programu medzi začiatkom cyklu a koncom cyklu sa volá telo cyklu.

Výraz od určuje začiatočnú hodnotu riadiacej premennej.

Výraz po určuje hornú hranicu riadiacej premennej.

Výraz krok určuje veľkosť kroku, o ktorý sa hodnota riadiacej premennej zmení pri každom prechode cyklom.

Telo cyklu sa vykoná opakovane takto:

Riadiacej premennej sa priradí hodnota výrazu od a vykoná sa telo cyklu. Na konci cyklu sa k okamžitej hodnote riadiacej premennej pričíta hodnota výrazu krok. Testuje sa, či je hodnota riadiacej premennej väčšia (pri kladnom kroku), alebo menšia (pri zápornom kroku), ako hodnota výrazu po (koncová hodnota). Ak nie je koncová hodnota prekročená, telo cyklu sa vykoná s novou hodnotou riadiacej premennej. V opačnom prípade je cyklus ukončený a vykonávanie programu pokračuje ďalším príkazom za príkazom NEXT.

Ukážme si, ako bude vyzerať program pre súčet čísiel od 1 do 30 s využitím tohto príkazu:

```
10 LET s=0
20 FOR c=1
30 LET s=s+c
40 NEXT c
50 PRINT "sucet = ";c
```

Na riadku 10 je nastavený medzisúčet s na hodnotu 0.

Na riadku 20 sú nastavené vlastnosti cyklu; riadiaca premenná bude nadobúdať hodnoty od 1 do 30 s krokom 1, riadok 30 tvorí vlastné telo cyklu.

Riadok 40 je ukončenie cyklu, za NEXT je uvedený názov riadiacej premennej cyklu.

Ak má krok hodnotu +1, časť príkazu STEP výraz 3 sa nemusí v príkaze uvádzať. Riadok 20 sme mohli preto napísať takto:

```
20 FOR c=1 TO 30
```

Ak chcete zmeniť program na súčet iných čísiel, stačí zmeniť jediný riadok programu, riadok 20.

Ak chcete sčítať čísla od 10 do 100 s krokom 0.5, zmeníme riadok na tvar:

```
20 FOR c=10 TO 100 STEP 0.5
```

Hodnota kroku môže byť i záporná. Riadok 20 môžete upraviť na:

```
20 FOR c=100 TO 10 STEP -0.5
```

Tu sme však museli zmeniť aj začiatočnú a koncovú hodnotu riadiacej premennej.

Ak je pri kladnom kroku začiatočná hodnota riadiacej premennej väčšia ako koncová hranica a pri zápornom kroku menšia ako koncová hranica, telo cyklu sa nevykoná ani raz a program pokračuje ďalším príkazom za príkazom NEXT.

Hodnota riadiacej premennej môže byť v tele cyklu znamená. Na konci cyklu sa testuje vždy jej okamžitá hodnota. Hranice cyklu sa nedajú meniť vo vnútri cyklu.

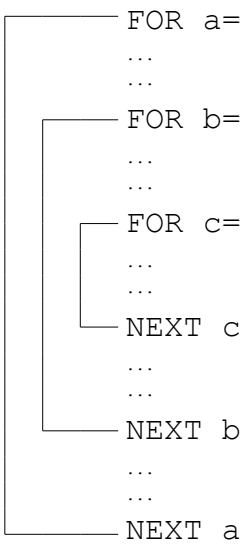
Do cyklu môžeme vstúpiť iba cez počiatok cyklu (cez FOR-NEXT).

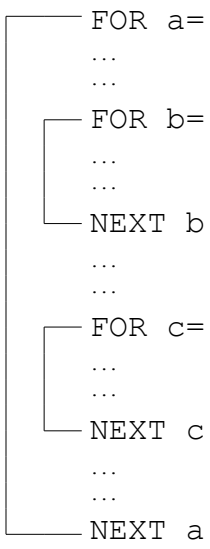
Skok zvonku dovnútra cyklu nie je dovolený. Je ale možné vyskočiť z cyklu pred jeho ukončením.

V jednom programe môžeme použiť niekoľko cyklov. Ak sú vzájomne nezávislé, nemali by sme urobiť chybu. Ak sa použijú cykly vložené do seba, musí sa vždy ako prvý ukončiť najvnútornejší cyklus.

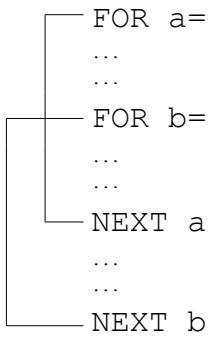
Na niekoľkých príkladoch si ukážeme rôzne umiestnenie cyklov v programe.

Správne vložené cykly:

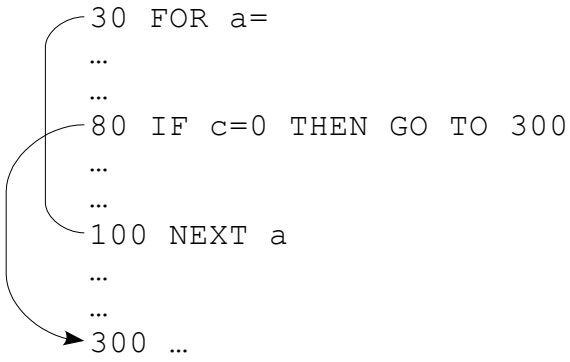




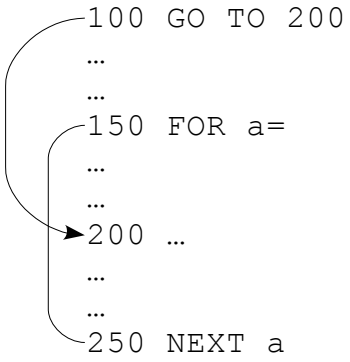
Nesprávne vložené cykly:



Povolený skok z cyklu



Nesprávny skok do cyklu



V nasledujúcom príklade uvidíte, aký užitočný je príkaz cyklu.

Úlohou je vytvoriť program pre načítanie mien a ich usporiadanie podľa abecedy. Meno je dlhé maximálne 10 znakov.

```
 10 PRINT "Usporiadanie mien podla abecedy"
 20 PRINT:INPUT "Zadaj pocet mien",k
 30 DIM a$(k,10)
 40 FOR i=1 TO k:PRINT "Zadaj meno cislo ";i
 50 INPUT a$(i)
 60 NEXT i
 70 FOR i=1 TO k-1
 80 FOR j=i+1 TO k
 90 IF a$(i)<a$(j) THEN GO TO !!)
100 LET x$=a$(i):LET a$(i)=a$(j):LET a$(j)=x$
110 NEXT j
120 NEXT i
130 PRINT: PRINT "Usporiadane mena":PRINT
140 FOR i=1 TO k:PRINT a$(i):NEXT i
```

Na riadku 10, 20 je úvodný text.

Na riadku 30 sa zadá počet mien a rezervuje sa miesto v pamäti pre tieto premenné.

Na riadkoch 40-60 sa načítajú mená do reťazcových premenných a$(1)...a$(k).

Vlastné usporiadanie sa vykoná na riadkoch 70-120. Princip usporiadania je ten, že sa zoberie prvé meno (i=1) a porovná sa so všetkými nasledujúcimi (to sú tie, ktorých indexy sú väčšie ako i, teda pre j=i+1 až k na riadku 80). Vlastné porovnanie je na riadku 90. Ak je splnená podmienka, nechá sa reťazec na svojom mieste. Ak nie je, vymení sa s tým, ktorý je "menší" (riadok 100). Po ukončení vnútorného cyklu bude v a$(1) uložená "najmenších" meno. Hodnota i sa zvýši a postup sa opakuje. Tak sa nájde druhé "najmenšie" meno a uloží sa do a$(2), atď.

Na riadku 70 je horná hranica k-1, lebo je nájdených k-1 "najmenších" mien a to ostávajúce nemôže byť "menšie".

Na riadkoch 130, 140 sa vykoná výpis usporiadaných mien.

Na tomto príklade si môžete uvedomiť, aký význam majú indexované premenné. Keby sme ich nemali, problém usporiadania by sa riešil veľmi ťažko.

Ďalšiu výhodu vidíte na riadku 140, kde sa jedným krátkym príkazovým riadkom vypíše ľubovoľný počet premenných.

Príkaz cyklu spolu s možnosťou použitia indexovaných premenných tvoria veľmi silný programovací prostriedok.

Ak v uvedenom programe nahradíte reťazcové premenné číselnými premennými, program usporiada a vypíše zadaný počet čísiel.

1.2.9. Zadávanie dát počítaču

Naučili ste sa, ako priradíte premenným hodnotu príkazom LET a INPUT. Tento spôsob nie je niekedy výhodný, hlavne vtedy, ak je zadávaných údajov viac. V takom prípade sa používa príkaz READ, s ktorým úzko súvisia príkazy DATA a RESTORE.

Príkaz **READ** má tvar

```
READ zoznam premenných
```

kde zoznam premenných je zoznam číselných a reťazcových premenných, oddelených od seba čiarkou.

Príkaz **DATA** má tvar

```
DATA zoznam výrazov
```

kde zoznam výrazov je zoznam aritmetických výrazov alebo reťazcov, ktoré sú oddelené od seba čiarkou.

V programe môžete použiť viac príkazov READ a DATA, pričom príkaz DATA môže byť na ľubovolnom mieste programu. Priradenie hodnôt premenným sa vykoná takto:

Podobne, ako má počítač ukazovateľ pre výpis na obrazovku, má pre príkaz DATA ukazovateľ dát, ktorý ukazuje na výraz v príkaze DATA, ktorého hodnota sa bude priraďovať ako nasledujúca.

Na začiatku programu sa nastaví ukazovateľ na prvý výraz v prvom príkaze DATA. Príkazom READ sa priradí premennej hodnotou výrazu, na ktorý je nastavený ukazovateľ. Po priradení nastaví na ďalší výraz v príkaze DATA. Ak sa položky zo zoznamu DATA vyčerpali, ukazovateľ sa nastaví na prvú položku v nasledujúcom príkaze DATA.

V príkaze READ môže byť v zoznamu vedľa seba číselné aj reťazcové premenné. Číselnej premennej sa môže priradiť len číselná hodnota, t. j. odpovedajúci výraz v príkaze DATA musí byť aritmetický výraz. Reťazcovej premennej sa môže priradiť len reťazec.

```
10 DATA 2*3,.5,"den"
20 READ a,b,c$
30 PRINT a,b,c$
```

Po vykonaní programu budú mať premenné tieto hodnoty:

a=6, b=0.5, c$="den"

Ak je položiek v príkazoch DATA menej ako premenných v príkazoch READ, počítač ohlási chybu. Zmeňte riadok 10 na

```
10 DATA 1,2
```

Po odštartovaní programu vypíše počítač hlásenie

```
E Out of DATA
```

Niekedy je treba použiť rovnaké hodnoty pre rôzne premenné. Aby sa nemusel zoznam dát vypisovať viackrát za sebou v programe, využijeme príkaz **RESTORE**, ktorý má tvar:

```
RESTORE číslo riadku
```

Tento príkaz nastaví ukazovateľ dát na prvú položku v príkaze DATA na riadku so zadaným číslom. Ak nie je číslo riadku zadané, ukazovateľ dát sa nastaví na prvú položku v prvom príkaze DATA v programe.

Vyskúšajte to na príklade:

```
10 DATA 1,2,3
20 READ a,b,c,pocet
30 RESTORE
40 READ a1,a2
50 DATA 10,20
60 RESTORE 50
70 READ i,j
```

Premenným a,b,c,pocet,a1,a2,i,j sa príkazom READ priradia hodnoty 1,2,3,10,1,2,10,20.

Často sa používa príkaz READ vy cykle:

```
10 DIM a(8)
20 DATA 3,9,5,6,4,1,7,8,78,55
30 FOR i=1 TO 8
40 READ a(i)
50 NEXT i
```

Všimli ste si, že v príkaze DATA je viac ako 8 položiek, hoci indexovaných premenných je len 8. To však nevadí, položky v príkaze DATA nemusia byť nikdy prečítané.

Ak použijete v programe príkaz CLEAR na zrušenie premenných, počítajte s tým, že sa vykoná zároveň RESTORE.

1.2.10. Doplnenie príkazu PRINT a INPUT

Príkaz PRINT sme použili v doterajších príkladoch dosť často, takže iste už viete ako pracuje a ako sa používa. Výrazy, ktorých hodnoty sú tlačené, sa nazývajú položky príkazu PRINT a sú oddelené čiarkami alebo bodkočiarkami, ktoré sa nazývajú oddeľovače príkazu PRINT. V príkaze PRINT môžeme použiť špeciálne položky, ktoré nám uľahčujú prácu pri výpisoch.

Často chceme výpis na obrazovku umiestniť na presne stanovené miesto. Na to používame položku **AT**, ktorá ma tvar:

```
AT riadok,stĺpec
```

Týmto sa nastaví ukazovateľ pre výpis na špecifikované miesto obrazovky (riadok, stĺpec). Riadky sa číslujú od 0 (horný) po 21, stĺpce od 0 (vľavo) po 31. Príkazom

```
PRINT AT 12,16; "*"
```

vypíšeme znak "*" do stredu obrazovky. Pretože položky AT je chápaná ako položka v príkaze PRINT, musí byť od ostatných položiek oddelená povoleným oddeľovačom (v našom prípade bodkočiarka). Ukazovateľ pre výpis sa nastaví analogicky, ako sme uvádzali už na začiatku príručky.

Funkcia **SCREEN$** je opačná k činnosti položky AT. Zisťuje, aký znak je v určitej pozícii na obrazovke zobrazený. Používa číslo riadku a stĺpca rovnako ako položka AT, ale sú uzatvorené v zátvorkách:

```
SCREEN$ (riadok,stĺpca)
```

Príkaze

```
PRINT SCREEN$ (12,16)
```

znovu vypíše hviezdičku, ktorá bola tlačená v predchádzajúcom príklade. Znaky sa berú normálne ako znaky, medzery sa vracajú ako medzery. Ak sú na zadanom mieste obrazovky užívateľské znaky, grafické symboly a grafy, funkcia má hodnotu nulového (prázdneho) reťazca.

Ďalšou položkou je **TAB** v tvare:

```
TAB (stĺpec)
```

Táto položka tlačí medzery na presun do špecifikovaného stĺpca riadku, riadok sa nemení. Pamätajte, že počítač redukuje číslo stĺpca modulo 32 (delí 32 a berie zvyšok), tak napr. TAB 33 znamená to isté ako TAB 1.

Skúste odštartovať program:

```
10 FOR n=0 TO 20
20 PRINT TAB 8*n;n;
30 NEXT n
```

Program ukazuje, že čísla pre TAB sú redukované modulo 32 (do úvahy sa berú zvyšky po delení číslom 32). Aby bol príklad elegantnejší, zmeňte 8 v riadku 20 na 6.

Niektoré dôležité poznámky:

1) Nastavenie pomocou AT je najlepšie ukončovať bodkočiarkou. Ak nie je na konci bodkočiarka, pozícia sa znovu zmení a potom je ju treba opäť nastaviť.
2) Nemôžete tlačiť na spodné dva riadky (22 a 23), pretože sú rezervované pre príkazy, vstupné dáta, správy atď. Odkazy na spodný riadok obvykle znamenajú riadok 21.
3) AT môžete použiť na to, aby PRINT bol vykonaný tam, kde je už niečo vypísané, starý výpis bude prepísaný.

Niekedy je vhodné, aby sa počítač nezastavil po výpise obrazovky a nečakal na stlačenie klávesu. Dá sa to urobiť pomocou príkazu

```
POKE 23692,255
```

Po tomto príkaze sa bude scrolling vykonávať 255x bez zastavenia s otázkou scroll?

Skúste tento príklad:

```
10 FOR n=0 TO 10000
20 PRINT n: POKE 23692,255
30 NEXT n
```

a pozorujte, čo sa bude diať na obrazovke.

Zoznam v príkaze PRINT môže obsahovať rôzne kombinácie čiarok, bodkočiarok, AT, TAB. Ich spracovanie prebieha zľava doprava a pre každú položku platia pravidlá, ktoré sme doteraz uviedli.

Ďalším príkazom, ktorý sa týka vstupu a výstupu dát, je príkaz INPUT, ktorý dokáže viac, než sme doteraz uviedli. Zatiaľ sme tento príkaz používali asi následovne:

```
INPUT "Kolko mas rokov"; vek
```

v ktorom tlačí počítač uvedené otázku v dolnej časti obrazovky a Vy potom zadáte svoj vek.

V skutočnosti je príkaz INPUT zostavený z položiek rovnako ako PRINT a tak text "Kolko mas rokov" a premenná vek sú obe položky príkazu INPUT. Všeobecne sú položky v príkazoch INPUT a PRINT rovnaké, ale sú tu veľmi dôležité rozdiely.

Obvyklý vstup INPUT je premenná, ktorej hodnota musí byť vložená, v našom príklade vek. Ak položka v INPUT začína znakom, musí to byť premenná, ktorej hodnotu potom vložíme. Ak sa očakáva reťazec, kurzor vo vstupnom riadku je zobrazený v úvodzovkách.

Ak chcete tlačiť hodnoty premenných ako časť záhlavia, urobíte to tak, že okolo premennej dáte zátvorky. Dokonca môžete do zátvoriek uzavrieť ľubovoľný výraz, ktorého hodnotu chcete tlačiť v príkaze INPUT.

Tu je názorný príklad ilustrujúci tento postup:

```
LET moj vek=INT(RND*100):INPUT("Ja mam ";moj vek;" rokov,"); " ty kolko mas
   rokov ?",tvoj vek
```

Premenná moj vek je v zátvorke preto, lebo sa jej hodnota tlačí, tvoj vek nie je v zátvorke, preto musíte vložiť jej hodnotu. Všetko, čo je v príkaze INPUT, sa píše v dolnej časti obrazovky a chová sa nezávisle na hornej časti obrazovky.

Vo všetkých prípadoch sú tieto riadky vztiahnuté k hornému riadku spodnej časti, i keď vkladáte veľa údajov v príkaze INPUT.

Aby ste poznali, ako pracuje AT v príkaze INPUT, skúste zadať:

```
10 INPUT "To je riadok 1.", a$; AT 0,0;"To je riadok 0.",a$;AT 2,0; "To je
   riadok 2.",a$; AT 1,0;"To je stale riadok 1.",a$
```

Teraz skúste:

```
10 FOR n=0 TO 19: PRINT AT n,0;n:NEXT n
20 INPUT AT 0,0;a$; AT 1,0;a$;AT 2,0;a$; AT 3,0;a$; AT 4,0;a$; AT 5,0;a$
```

Dolná časť obrazovky sa posúva smerom nahor, horná časť je neporušená. Keď má byť písané na ten istý riadok ako v PRINT, začne sa horná časť obrazovky rolovať.

Často sa stáva, že chcete prerušiť program počas zadávanie dát v príkaze INPUT. Stlačenie klávesu BREAK tu nepomôže. Ak sa očakáva vstup číselnej premennej, zadáte príkaz STOP (SYMBOL SHIFT + A) a stlačíte ENTER. Ak sa očakáva reťazec, presuniete kurzor pred úvodzovky, zadáte STOP a stlačíte ENTER. Ešte doplňme, že v prípade zadávanie reťazca sa môže zadať namiesto reťazcovej konštanty uzavretej v úvodzovkách i zložitejší reťazcový výraz (úvodzovky môžeme prípadne zmazať).

Ďalšou možnosťou v príkaze INPUT, ktorú ste doteraz nepoznali, je typ **LINE** vstup, kde ide o iný spôsob vkladanie hodnôt reťazcovým premenným. Pokiaľ napíšete LINE pred meno reťazcovej premennej (pri vstupe) napr.

```
INPUT LINE a$
```

potom počítač nevypíše reťazcové úvodzovky, ako je to pri vstupe reťazca obvyklé, ale očakáva vstup reťazca, ako keby tam úvodzovky boli. Tak keď napíšete "ano" ano vstup dát, dostane hodnotu ano. Pretože reťazcové úvodzovky nie je vypísané, nemôžete ich zrušiť a vkladať tak iný druh reťazcového vyjadrenia pre vstup dát. Pamätajte, že nemôžete použiť LINE vstup pre číselné premenné.

Riadiace znaky CHR$ 22 a CHR$ 23 majú účinok ako AT a TAB. Pretože ide o riadiace znaky, musia byť nasledované dvoma ďalšími znakmi, s ktorými sa počíta (presnejšie s ich kódmi ako s číslami na špecifikáciou stĺpca a riadku (pre AT) a pozície (pre TAB). Takmer vždy je ľahšie použiť AT a TAB v ich obvyklej forme, než použiť riadiace znaky, ale i tie sú niekedy užitočné. V riadiacom znaku po CHR$ 22 (pre AT) prvý znak špecifikuje číslo riadku a druhý číslo stĺpca. Preto

```
PRINT CHR$ 22+CHR$ 1+CHR$ c;
```

má presne rovnaký účinok ako

```
PRINT AT 1,c;
```

Riadiaci znak pre TAB je CHR$ 23 a dva znaky za tým určujú číslo od 0 do 65535, špecifikujúce číslo v položke TAB.

```
PRINT CHR$ 23+CHR$ a+CHR$ b;
```

má rovnaký účinok ako

```
PRINT TAB a+256*b;
```

Vyskúšajte si nasledujúci program pre deti, ktorý testuje, ako vedia násobiť:

```
 10 LET m$=""
 20 LET a=INT (RND*12)+1:LET b=INT (RND*12)+1
 30 INPUT (m$) "kolko je ";(a);"*";(b);"?";c
100 IF c=a*b THEN LET m$="Spravne.":GO TO 20
110 LET m$="Zle. Zadaj znova.":GO TO 30
```

Pokiaľ sú deti vnímavé, mohli by zistiť, že vlastne nemusia vôbec nič počítať. Napr. keď sa ich počítač spýta, koľko je 2*3, môžu namiesto správnej odpovede napísať 2*3. Cesta, ako tomu zabrániť, je čítať vstupný reťazec namiesto čísla. Nahraďte c v riadku 30 c$, v riadku VAL c$ a vložte riadok

```
40 IF c$ <> STR$ VAL c$ THEN LET m$="piste spravne ako cislo.":GO TO 30
```

To ich bude hnevať. Za niekoľko dní môžu objaviť, že možno vkladať reťazec STR$ (2*3). Aby sa tomu zabránilo, je treba nahradiť c$ v riadku 30 príkazom LINE c$.

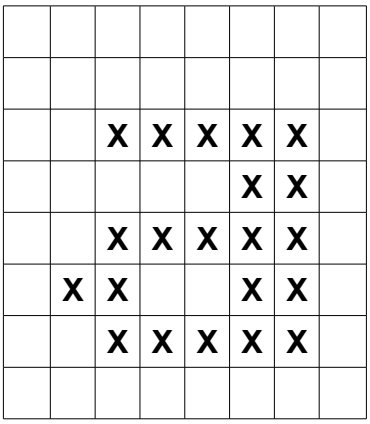
1.2.11. Práca s farbami

Počítač je schopný zobraziť na farebnom televízore (alebo monitore) 8 farieb. Pokiaľ je Váš televízor čiernobiely, uvidíte len odtiene šedej farby. Jednotlivé farby sú označené číslom:

0. čierna
1. modrá
2. červená
3. purpurová (fialová)
4. zelená
5. svetlo modrá
6. žltá
7. biela

Aby ste používali farby správne, musíte pochopiť, ako je obraz zostavený. Obraz je rozdelený na 768 pozícií (24 riadkov a 32 stĺpcov), kde môžu byť znaky vytlačené.

Každý znak je tlačený ako 8x8 bodov obrazovky, ako ukazuje obrázok malého a.



Každý bod z 8x8 bodov štvorca pre výpis jedného znaku môže mať jednu z dvoch farieb: farba znaku (farba tmavých bodov v našom štvorci) a farba pozadia (je použitá pre biele body).

Na začiatku práce s počítačom majú všetky zobrazené body čierne farbu a pozadie (podklad) farbu bielu a tak sa znaky javia ako čiernobiele. Každý znak má tiež jas (normálny alebo zvýšený) a môže blikať (blikanie je vlastne výmena farby pozadia a zobrazených bodov). Všetko je kódované v číslach a počítač potrebuje pri výpise jedného znaku o znakovej pozícii vedieť nasledujúce informácie:

1. 8×8 bodov, kde 0 znamená body podkladu a 1 znamená zobrazené body, definujúce tvar znaku
2. farby zobrazených bodov a podkladu a 1 znamená zobrazené body, definujúce tvar znaku
3. jas - 0 normálny, 1 zvýšený
4. blikanie - 0 bez blikanie, 1 bliká

Z toho vyplýva, že jedna znaková pozícia (8x8 bodov) môže mať najviac dve farby, všetky body majú normálny alebo zvýšený jas a všetky body blikajú alebo neblikajú.

Hodnoty dvoch farieb, jasu a blikania sa nazývajú **atribúty**. Atribúty sa dajú programovo meniť a príkazy na ich zmenu si teraz popíšeme.

Programovo môžeme ovládať nielen farbu vypisovaných znakov, ale tiež okraj obrazovky, t. j. časť, do ktorej sa nedá nič vypísať príkazom PRINT.

Farba okraja obrazovky sa mení príkazom **BORDER** v tvare:

```
BORDER farba
```

kde farba je celé číslo od 0 do 7. Ihneď po zadaní tohto príkazu sa okraj obrazovky zafarbí zadanou farbou.

Farbu podkladu pracovnej častí obrazovky meníme príkazom **PAPER** v tvare:

```
PAPER farba
```

kde farba je celé číslo od 0 do 9. V prípade, že je v rozmedzi od 0 do 7, určuje farbu podkladu. Ak má hodnotu 8, farba podkladu sa nemení, t. j. ostáva nastavená tak, ako to bolo pred týmto príkazom. Ak má hodnotu 9, farba podkladu bude kontrastná oproti farbe výpisu znakov, t. j. bude.

- biela ako kontrast pre tmavú (čiernu, modrú, červenú, purpurovú)
- čierna ako kontrast pre svetlú (zelená, svetlo modrá, žltá, biela)

Farbu výpisu znakov zadáme príkazom **INK** v tvare:

```
INK farba
```

kde pre farbu platí to isté, ako v príkaze PAPER.

Pre určenie jasu použijeme príkaz **BRIGHT** v tvare:

```
BRIGHT číslo
```

kde číslo môže mať tieto hodnoty:

0 - jas je normálny
1 - jas je zvýšený
8 - jas sa nemení, ostáva nastavený podľa predchádzajúceho príkazu BRIGHT

Blikanie zadáme príkazom **FLASH** v tvare:

```
FLASH číslo
```

kde číslo môže mať tieto hodnoty

0 - bez blikania
1 - blikanie
8 - bez zmeny podľa predošlého nastavenia

Použitie uvedených príkazov môžete odskúšať na nasledujúcom príklade:

```
 10 FOR m=0 TO 1:BRIGHT m
 20 FOR n=1 TO 10
 30 FOR c=0 TO 7
 40 PAPER c: PRINT "    ";:REM 4 farebne medzery
 50 NEXT c: NEXT n: NEXT m
 60 FOR m=0 TO 1: BRIGHT m:PAPER 7
 70 FOR c=0 TO 3
 80 INK c:PRINT c;"    "
 90 NEXT c: PAPER 0
100 FOR c=4 TO 7
110 INK c: PRINT c;"    ";
120 NEXT c: NEXT m
130 PAPER 8: INK 7: BRIGHT 0
```

Program zobrazuje 8 farieb (vrátane čiernej a bielej) a dve úrovne jasu, ako sa zobrazujú na farebnom televízore.

Tlačí sa vo farbách podľa zadaných príkazov (medzera je znak, kde farba INK a PAPER je zhodná).

Kontrastnú farbu vyskúšajte nasledovným príkazovým riadkom:

```
INK 9: FOR c=0 TO 7: PAPER c: PRINT c: NEXT c
```

Ďalšie dva príkazy **INVERSE** a **OVER** neriadia atribúty, ale spôsob akým sú znaky vypisované na obrazovke. Tieto príkazy majú tvar:

```
INVERSE číslo
OVER číslo
```

kde číslo môže mať len hodnotu 0 a 1.

Ak zadáte INVERSE 1, potom výpis znaku bude invertovaný vzhľadom k normálnemu výpisu (farby určené príkazmi BORDER a INK sa navzájom vymenia). Pokiaľ máte čierny INK a biely BORDER (ako po prvom zapnutí), tak po inverzii budú písmená biele na čiernom podklade.

Zadaním INVERSE 0 sa prejde do pôvodného stavu.

Príkaz OVER 1 zapína zvláštny druh výpisu - prepisovanie. Normálne, keď je niečo napísané do znakovej pozície, je celkom jedno, čo tam bolo skôr, ale po OVER 1 nový znak nezmaže starý znak, ale znaky sa prelínajú cez seba. Môže to byť zvlášť užitočné pre písanie kompozičných znakov, podobne ako znakov, na ktoré je kladený dôraz, rovnako ako sa v nemčine píše prehlasované o.

```
10 OVER 1
20 FOR n=1 TO 32
30 PRINT "o";CHR$ 8;""""";
40 NEXT n
```

Veľkou výhodou príkazov pre farbu je to, že tieto príkazy môžu byť položkami v príkaze PRINT a ich účinok platí v tom prípade iba v tom príkaze PRINT, v ktorom sú uvedené.

Napr. po odoslaní

```
PRINT PAPER 6;"x";:PRINT "y"
```

bude iba x na žltom pozadí.

Tieto príkazy nemajú účinok na farby v dolnej časti obrazovky, kde sú vkladané príkazy INPUT. Dolná časť obrazovky používa farbu pre podklad podľa príkazu BORDER a kód 9 (ako kontrast) pre farbu INK, bez blikania s normálnym jasom.

Keď vkladáte údaje pomocou INPUT, farbu môžete zmeniť použitím INK a PAPER ako v príkaze PRINT. Účinok trvá do konca príkazu alebo do ďalšieho príkazu INPUT. Skúste:

```
INPUT FLASH 1; INK 1; "Zadaj svoje meno";n$
```

Ďalší spôsob zmeny farieb je použitím riadiacich znakov, kde

```
CHR$ 16     zodpovedá     INK
CHR$ 17      ------||------     PAPER
CHR$ 18      ------||------     FLASH
CHR$ 19      ------||------     BRIGHT
CHR$ 20      ------||------     INVERSE
CHR$ 21      ------||------     OVER
```

Za nimi nasleduje znak, ktorého kód udáva farbu, napr.:

`PRINT CHR$ 16+CHR$ 9; ...` má rovnaký účinok ako
`PRINT INK 9; ...`

Tieto riadiace znaky nemusíte používať, používajte radšej príkazy. Niekedy je však užitočné pracovať s nimi pri vkladani programu. Výpis programu bude mať v rôznych častiach rôznu farbu, aby sa časti odlíšili od seba, alebo pekne vyzerali. Aby ste ich dostali do programu, musíte ich zadať priamo z klávesnice použitím rozšíreného módu. Zmenu farby zadáme vždy za číslom riadku. Keď ste v **E** móde a stlačíte číslo 6 (pre žltú farbu), potom sa vložia znaky; najprv CHR$ 17 ako PAPER a CHR$ 6 ako kód pre žltú farbu. Keď stlačíte CAPS SHIFT+6, dostávate kód CHR$ 16 ako INK a CHR$ 6 pre farbu. Pretože sa vložili 2 znaky, vymazanie musíte vykonať tak, že dvakrát stlačíte DELETE. Po prvom stlačení sa objaví ?. Nebojte sa a stlačte DELETE znovu.

Taktiež ostatné klávesy horného radu klávesnice majú v rozšírenom móde **E** nejaký význam:

8 dáva CHR$ 19 a CHR$ 0 pre normálny jas
9 dáva CHR$ 19 a CHR$ 1 pre zvýšený jas
CAPS SHIFT a 8 dáva CHR$ 18 a CHR$ 0 pre neblikanie
CAPS SHIFT a 9 dáva CHR$ 18 a CHT$ 1 pre blikanie

V obyčajnom **L** móde platí:

CAPS SHIFT a 3 dáva CHR$ 20 a CHR$ 0 pre normálne znaky
CAPS SHIFT a 4 dáva CHR$ 20 a CHT$ 1 pre inverzné znaky

V nasledujúcej tabuľke je podrobný popis významu horného radu klávesnice v rôznych módoch:

Mód	SHIFT	Význam									
E	SYMBOL	DEF FN	FN	LINE	OPEN	CLOSE	MOVE	ERASE	POINT	CAT	FORMAT
	CAPS	INK modrá	INK červená	INK purpur	INK zelená	INK sv.modrá	INK žltá	INK biela	FLASH nie	FLASH áno	INK čierna
	ŽIADEN	PAPER modrá	PAPER červená	PAPER purpur	PAPER zelená	PAPER sv.modrá	PAPER žltá	PAPER biela	BRIGHT norm.	BRIGHT zvýšený	PAPER čierna
	OBA									koniec GRAPHICS	DELETE
G	ŽIADEN									koniec GRAPHICS	DELETE
K	CAPS	EDIT	CAPS LOCK	TRUE VIDEO	INVERS VIDEO					GRAPHICS MOD	DELETE
L	SYMBOL	!	@	#	$	%	&	'	(	)	_
C	ŽIADEN	1	2	3	4	5	6	7	8	9	0

Zadávať príkazy pre farebné zobrazenie znaku sme sa už naučili. Niekedy je vhodné vedieť, ako je zobrazený znak na určitej pozícii na obrazovke, presnejšie, aké atribúty má určitá pozícia na obrazovke. Na to použijeme funkciu **ATTR**, ktorá má tvar:

`ATTR (riadok,stĺpec)`

kde riadok a stĺpec majú ten istý význam, ako v príkaze AT. Výsledkom je číslo udávajúce atribúty zodpovedajúceho znaku na obrazovke. Toto môžete využiť samostatne alebo ako časť inej funkcie. Číslo, ktoré je výsledkom tejto funkcie, je súčet 4 ďalších čísiel:

1) 128 pokiaľ pozícia bliká, 0 keď nebliká
2) 64 ak je na pozícii zvýšený jas, 0 keď je normálny
3) 8*n, kde n je kód farby podkladu pre PAPER
4) kód farby pre INK

Napr. nech pozícia bliká s normálnym jasom, žltým podkladom a farba znaku je modrá. Potom čísla, ktoré sa sčítajú, sú 128, 0, 8*6=48 a 1, výsledkom je 177.

Skúste príkazový riadok

```
PRINT AT 0, 0; FLASH 1; PAPER 6; INK 1;" "; ATTR (0, 0)
```

Teraz skúste:

```
PRINT "B"; CHR$ 8; OVER 1;"/";
```

Dostanete znak B prepísaný znakom /.

Je zaujímavé, že pokiaľ vypíšete to isté 2x, dostanete späť to, čo tam bolo. Skúste teraz napísať:

```
PRINT CHR$ 8; OVER 1;"/" (dostanete B - prečo?)
```

Zadajte

```
PAPER 0: INK 0: PRINT "NIC"
```

Čo ste dostali? Nie je dobré, že to neplatí v dolnej časti obrazovky?

Nakoniec nechajte bežať tento program

```
10 POKE 22527+RND*704,RND*127
20 GO TO 10
```

Program pracujte tak, že mení farbu pozícií pre výpis znakov, funkcia RND zaisťuje náhodnosť. Diagonálne prvky, ktoré momentálne vidíte, sú dokladom toho, že RND dáva pseudonáhodné číslo.

1.2.12. Počítač vie kresliť

V tejto kapitole uvidíte, ako sa dajú na počítači kresliť obrázky. Časť obrazovky, ktorú môžete použiť, má 22 riadkov a 32 stĺpcov, tj. 22*32=704 znakov. Ako si pamätáte z predchádzajúcej kapitoly, každý z týchto znakov je vytvorený z 8x8 bodov. Každý bod je špecifikovaný dvoma číslami - **súradnicami**. Prvá súradnica je x a druhá je y. Tieto súradnice sa píšu ako pár v zátvorke. (0,0), (0,175), (255,0), (255,175) sú rohové body kresliaceho poľa. To zároveň znamená, že pri kreslení môže byť súradnica x len celé číslo od 0 do 255 a súradnica y od 0 do 175. Ak nie je hodnota x a y celé číslo, berie sa do úvahy len celá časť.

V tomto poli máme k dispozícii nasledujúce grafické príkazy:

PLOT x, y

nakreslí bod daný súradnicami x, y. Súradnice x, y sa chápu ako absolútne súradnice (t. j. vzhľadom na ľavý dolný roh).

Nasledujúci program:

```
10 PLOT INT(RND*256), INT (RND*176): INPUT a$: GO TO 10
```

nakreslí bod s náhodnými súradnicami po každom stlačení ENTER.

Máme tu však oveľa zaujímavejší program. Kreslí grafy funkcie SIN (sínusovka) pre hodnoty 0 až 2*PI.

```
10 FOR n=0 TO 255
20 PLOT n,88+80*SIN (n/128*PI)
30 NEXT n
```

Ďalší program kreslí graf časti paraboly (funkcia SQR) medzi hodnotami 0 a 4:

```
10 FOR n=0 TO 255
20 PLOT n,80*SQR (n/64)
30 NEXT n
```

Všimnite si, že súradnice bodu pre kreslenie sa líšia od súradnice v AT pre nastavenie ukazovateľa výpisu znaku.

Počítač však dokáže vykresliť nielen bod, ale i úsečku, dokonca aj kružnicu alebo jej časť. Úsečku nakreslíme pomocou príkazu

DRAW x, y

Začiatočným miestom úsečky je bod, v ktorom skončil predchádzajúci grafický príkaz svoju činnosť (po zapnutí počítač a po príkazoch RUN, CLEAR, CLS a NEW je to spodný ľavý roh (0,0)) a končí bodom s udanými súradnicami, kde súradnice koncového bodu sa berú vzhľadom na začiatočný bod úsečky (relatívne súradnice, na rozdiel od príkazu PLOT). Preto môžu byť hodnoty x a y v príkaze DRAW aj záporné.

Skúšajte príkazy PLOT a DRAW napr.:

```
PLOT 0,100: DRAW 80,-35
PLOT 90,150: DRAW 80,-35
```

Môžete samozrejme kresliť vo farbách, pamätajte však, že farby platia pre celú pozíciu znaku (8x8 bodov) a nemôžu byť špecifikované pre jednotlivé body. Bod je tlačený farbou INK a farba všetkých bodov v jednej znakovej pozícii je dané tou istou farbou, ako ukazuje tento program:

```
10 BORDER 0: PAPER 0: INK 7: CLS
20 LET x1=0: LET y1=0: REM zaciatok priamky
30 LET c=1: REM farba pre INK - zacina sa modrou
40 LET x2=INT (RND*256):LET y2=INT (RND*176): REM dokoncenie
50 DRAW INK c; x2-x1, y2-y1
60 LET x1=x2: LET y1=y2: REM dalsia ciara zacina tam kde minula koncila
70 LET c=c+1: IF c=8 THEN LET c=1: REM nova farba
80 GO TO 40
```

Ako program postupuje, čiary sa zdajú širšie a to preto, že menia farbu. Pamätajte, že môžete použiť PAPER, INK, FLASH, BRIGHT, INVERSE, OVER v príkaze PLOT, DRAW, rovnako ako v PRINT a INPUT. Ďalšou možnosťou využitia DRAW je možnosť kresliť časť kružnice namiesto priamok pridaním ďalšieho čísla k príkazu, ktorý špecifikuje uhol pre kružnicový oblúk:

DRAW x, y, u

x a y udávajú polohu koncového bodu (relatívne) a u je uhol v radiánoch udávajúci otáčanie; pokiaľ je kladné, otáča sa doľava, pokiaľ je záporné, tak doprava. Na u sa môžeme pozerať tak, že je to časť úplnej kružnice. Celá kružnica je 2*PI radiánov a potom pre u=PI bude nakreslená polkružnica, pokiaľ u = 0,5 × PI bude to štvrťkruh atď. Predpokladajme, že u=PI.

Potom nasledujúci program

```
10 PLOT 100,100: DRAW 50,0,PI
```

nakreslí kruhový oblúk (polkružnicu).

Odštartujte program niekoľkokrát a PI vždy nahraďte iným číslom:

-PI, PI/2, 3*PI/2, PI/4, 1, 0

Posledný príkaz tejto kapitoly je príkaz **CIRCLE**, ktorý kreslí celé kružnice. Má tvar:

```
CIRCLE x, y, r
```

kde x a y sú súradnice stredu kružnice v absolútnych súradniciach a r je polomer kružnice.

Podobne ako v PLOT a v DRAW môžete použiť rôzne farby.

Funkcia POINT udáva, či má bod farbu podľa INK alebo PAPER.

Má tvar

```
POINT (x,y)
```

kde x, y sú súradnice bodu (musia byť v zátvorkách). Výsledok je 0, keď bod má farbu podľa PAPER a je 1, ak má farbu podľa INK.

Skúste, ako pracuje INVERSE, OVER, PLOT. Majú vplyv len na vlastný bod a na zvyšok pozície znaku. Ak nie sú OVER a INVERSE zadané, je to isté ako použitie s parametrom 0. Zmeňte ju na 1 a pozrime sa, čo sa stane:

```
PLOT INVERSE 1  ; - kreslí bod, vo farbe papiera
PLOT OVER 1     ; - zmení bod z čohokoľvek, vymení sa farba INK a PAPER
PLOT INVERSE 1  ; OVER 1; necháva bod ako bol pôvodne
```

Zadajte

```
PLOT 0,0:DRAW OVER 1;250,175
```

a skúste zmazať nakreslenú úsečku pomocou

```
DRAW OVER 1;-250,-175
```

Tento spôsob nepracuje presne, pretože body tvoriace druhú úsečku nie sú rovnaké ako body tvoriace prvú úsečku. Čiaru musíte vymazať v tom smere, v akom ste ju kreslili.

Doporučujeme precvičiť položky príkazov PAPER, INK, BRIGHT v príkaze PLOT. Tie platia pre celú znakovú pozíciu, obsahujúcu bod. Používajú sa v tvare:

```
PLOT PAPER a; FLASH b; BRIGHT c;x,y
```

Pokúste sa nakresliť kruhy s použitím funkcií SIN, COS a tiež pomocou príkazu CIRCLE:

```
10 FOR n=0 TO I*PI STEP PI/180
20 PLOT 100+80*COS n,87+80*SIN n
30 NEXT n
40 CIRCLE 150,87,80
```

Ako vidíte, príkaz CIRCLE je oveľa rýchlejší.

Zadajte:

```
CIRCLE 100,87,80: DRAW 50,50
```

Všimnite si, že po príkaze CIRCLE je nastavený kurzor pre grafické príkazy uprostred pravého okraja kružnice. Preto je vhodné na ďalšie kreslenie použiť PLOT na definovanie počiatočnej pozície.

Nakoniec ukážeme program, ktorý kreslí grafy takmer akejkoľvek funkcie. Na riadku 30 sa zadáva rozsah argumentu a funkcia pre graf. Funkcia sa vkladá ako reťazec, mal by to byť výraz, používajúci x ako argument funkcie.

```
 10 PLOT 0,87: DRAW 255,0
 20 PLOT 127,0: DRAW 0,175
 30 INPUT n,e$
 35 LET t=0
 40 FOR f=0 TO 255
 50 LET x=(f-128)*n/128: LET y=VAL e$
 60 IF ABS y>87 THEN LET t=0: GO TO 100
 70 IF NOT t THEN PLOT f, y+88: LET t=1: GO TO !))
 80 DRAW 1,y-stare y
100 LET stare y=INT (y+.5)
110 NEXT f
```

Nechajte program bežať a ako príklad vložte číslo 10 pre n a 10*TAN x pre funkciu. Výsledkom bude graf funkcie tangens v rozsahu -10 až +10.

1.2.13. Funkcie definované užívateľom

Pri programovaní budeme potrebovať často vypočítať hodnotu z neštandardných funkcií, ktoré sa v programe opakujú. Aby sme nemuseli vždy tieto zložité výrazy celé vypisovať, definujeme si vlastnú funkciu. Ak potrebujeme vypočítať hodnotu tejto funkcie, vypíšeme len meno funkcie (hovoríme, že voláme funkciu) s príslušným argumentom. Ak je funkcia raz definovaná, pracujeme s ňou ako so štandardnou funkciou. Pred volaním funkcie ju musíme definovať príkazom **DEF**, ktorý má tvar:

DEF FN názov (zoznam formálnych argumentov) = výraz

kde názov je názov nami definovanej funkcie. Názov funkcie je jedno písmeno (pokiaľ je výsledok číslo) alebo jedno písmeno a znak $ (pokiaľ je výsledkom reťazec). **Zoznam formálnych argumentov** je zoznam premenných oddelených čiarkou, kde mená sú jednopísmenové. Zoznam argumentov je nutné písať do uzatvorených zátvoriek. Funkciu definujete tým, že napíšete príkaz DEF niekde do programu. Definícia funkcie môže byť len na jednom riadku.

Napr. definujme funkciu, ktorej výsledkom je druhý mocnina argumentu.

```
10 DEF FN s(x)= x * x: REM druha mocnina x
```

DEF získame v rozšírenom móde použitím SYMBOL SHIFT + 1. Pri vkladaní počítač pridá FN automaticky (pretože v príkaze DEF vždy ihneď nasleduje FN). Rovnítko potom znamená skutočnú definíciu funkcie. Funkcia môže byť definovaná akýmkoľvek výrazom a môžete sa na ňu odkazovať ako na obyčajnú premennú.

Volanie funkcie sa vykoná takto:

FN názov (zoznam skutočných argumentov)

kde názov je názov definovanej funkcie, v zozname sú argumenty oddelené čiarkou. Skutočným argumentom môže byť ľubovolný odpovedajúci výraz (aritmetický alebo reťazec podľa definície funkcie). Pri volaní funkcie sa formálny argument v definícii nahradí skutočným argumentom (jeho hodnotu), ostatné premenné použité v definícii sa nahradia svojimi hodnotami v okamihu volania funkcie.

Príklad:

Našou úlohou je vypočítať obsah plochy medzikružia, polomer menšej kružnice je 10 cm, väčšej 15 cm. Obsah plochy medzikružia dostaneme tak, že od obsahu väčšieho kruhu odčítame plochu menšieho kruhu. Pretože obsah kruhu budeme počítať dvakrát, použijeme na jeho výpočet funkciu definovanú užívateľom.

```
10 DEF FN o(r)=PI*r*r
20 LET p1=FN o(10)
30 LET p2=FN o(15)
40 LET p=p2-p1
50 PRINT "Obsah medzikruzia je ";p
```

Na riadku 10 definujeme funkciu o(r), ktorá počíta obsah kruhu s polomerom r.

Na riadku 20 voláme funkciu o so skutočným parametrom 10. Do výrazu na riadku 10 sa dosadí za argument r skutočný argument 10 a vypočíta sa hodnota výrazu PI*10*10. Hodnota funkcie sa priradí

premennej p1.

Na riadku 30 analogicky voláme funkciu o so skutočným argumentom 15, vypočíta sa hodnota PI*15*15.

Na riadku 40 sa vypočíta rozdiel obsahov a výsledok sa vypíše (riadok 50).

Teraz si ukážeme definíciu užitočnej funkcie pre zaokrúhlenie. Funkcia INT vždy "zaokrúhľuje" dole. Aby zaokrúhlenie bolo "normálne" (do .5 dole, nad .5 hore), je treba použiť konštantu 0.5. Funkcia bude mať tvar:

```
20 DEF FN r(x)=INT (x+0.5): REM zaokruhlenie
```

Potom dostanete:

```
FN r(2.9) = 3      FN r(2.4)=2
FN r(-2.9) = -3    FN r(-2.4)=-2
```

Porovnajte výsledky s výsledkami, ktoré dostanete pri použití INT namiesto FN r.

Máme tu ďalší program pre použitie funkcií:

```
10 LET x = 0: LET y = 0: LET a = 10
20 DEF FN p(x,y)=a+x*y
30 DEF FN q()=a+x*y
40 PRINT FN p(2,3),FN q()
```

V tomto programe je viac zaujímavých bodov.

Po prvé, funkcia nie je obmedzená na 1 argument, môže mať viac alebo dokonca žiadny argument, ale stále musíte písať zátvorky. Po druhé, nezáleží na umiestnení DEF v programe, definícia funkcie môže byť na ľubovoľnom mieste programu. Funkcia nemôže byť však definovaná v priamom režime. V našom príklade sa po vykonaní riadku 10 sa preskočí na riadok 40, kde sa vytlačia hodnoty funkcií.

Po tretie, x a y sú mená premenných v programe a zároveň mená formálnych argumentov funkcie (čo nemá na hodnotu premenných žiadny vplyv). V definícii funkcie nie je však žiaden argument nazvaný a, ale iba premenná a. Tak pri vykonaní FN p(2,3) má premenná a stále hodnotu 10, pretože ide o premennú, argument x má hodnotu 2, pretože je to prvý argument, argument y má hodnotu 3, pretože je to druhý argument. Výsledkom je 10+2*3=16. Keď sa vykoná FN q() (funkcia nemá žiadne argumenty), a, x, y sú premenné a majú hodnoty 10, 0, 0. Odpoveď je v tomto prípade 10+0*0=10. Teraz zmeňte riadok 20 na:

```
20 DEF FN p(x,y)=FN q()
```

teraz FN p(2,3) bude mať hodnotu 10, pretože FN q sa bude vracať k hodnotám x a y a argument FN p nie je použitý.

Niektoré verzie jazyka BASIC majú funkcie LEFT$, RIGHT$, MID$ a TL$, kde

`LEFT$ (a$,n)`	dáva podreťazec z a$ majúci prvých n znakov
`RIGHT$(a$,n)`	dáva podreťazec z a$ majúci posledných n znakov
`MID$(a$,n1,n2)`	dáva podreťazec z a$, ktorý má n2 znakov a začína na pozícii n1.
`TL$(a$)`	dáva podreťazec z a$, ktorý sa skladá zo všetkých znakov mimo prvého

Kto je na tieto funkcie zvyknutý, môže napísať užívateľsky definované funkcie, ktoré robia to isté:

```
10 DEF FN t$(a$) = a$(2 TO): REM TL$
20 DEF FN l$(a$,n) = a$(TO n): REM LEFT$
```

Skontrolujte, či tieto funkcie pracujú s reťazcami s dĺžkou 0 alebo 1. Všimnite si, že naša funkcia l$ má dva argumenty, jeden je číslo a druhý je reťazec.

Funkcia môže mať až 26 číselných argumentov (prečo 26?) a zároveň až 26 reťazcových argumentov.

Na pravej strane definície funkcie môžeme použiť tiež ľubovoľnú funkciu (štandardnú i užívateľskú). Preto môžeme zadať napr.:

```
DEF FN s(x) = SIN x
```

1.2.14. Čo je to podprogram

Pri programovaní potrebujeme často vykonávať nejakú činnosť viackrát. Ak je treba túto činnosť vykonávať nepravidelne, nemôžeme použiť príkaz cyklu. Aj s takýmto problémom si vieme poradiť, ak použijeme príkaz pre volanie **podprogramu**. Podprogram voláme príkazom **GO SUB**, ktorý má tvar:

```
GO SUB číslo riadku
```

Príkaz pre návrat z podprogramu má tvar:

```
RETURN
```

Podprogram je relatívne samostatná časť programu.

Pri odštartovaní programu s podprogramom pracuje počítač takto:

Hlavný program sa vykonáva postupne podľa pravidiel doteraz uvedených až po príkaz GO SUB. Na tomto mieste pokračuje vykonávanie programu na riadku uvedenom v príkaze GO SUB. Začne sa vykonávať podprogram (hovoríme, že voláme podprogram). Ak vo vykonávaní programu sa vráti do hlavného programu za príkaz GO SUB, ktorým sme podprogram volali. Po ukončení podprogramu pokračuje vykonávanie programu teda prvým príkazom za príkazom GO SUB. Počítač si vždy "pamätá", z ktorého miesta programu je podprogram volaný a po ukončení podprogramu sa na toto miesto vráti:

```
 10 CLS
 20 PRINT:PRINT"Ucast v zaujmovych kruzkoch":PRINT
 30 GO SUB 200
 40 PRINT"Modelarky";TAB(20);12
 50 GO SUB 200
 60 PRINT"Tanecny";TAB(20);21
 70 GO SUB 200
 80 PRINT"Vytvarny";TAB(20);15
 90 GO SUB 200
100 STOP
200 PRINT:PRINT"-------------------------"
210 PRINT
220 RETURN
```

Tento program vypíše na obrazovku účasť v záujmových krúžkoch v škole. Činnosť programu je nasledovná:

Riadok 10-20 - mazanie obrazovky a výpis textu

Riadok 30 - volanie podprogramu na riadku 200; začne sa vykonávať program od riadku 200 až po príkaz RETURN na riadku 220 (výpis čiary "---...---" na oddelenie textu).

Riadok 40 - výpis textu

Riadok 50 - volanie podprogramu, po jeho ukončení návrat za príkaz GO SUB

Riadok 60 - výpis textu

Riadok 70 - volanie podprogramu

Riadok 80 - výpis textu

Riadok 90 - volanie podprogramu

Riadok 100- koniec hlavného programu

Riadok 200-220 - podprogram, jeho činnosť už bola popísaná

Podprogram sme využili celkom 4 krát na oddelení výpisu

Podprogram sme využili celkom 4 krát na oddelenie výpisu jednotlivých krúžkov od seba. Keby sme nemohli použiť volanie podprogramu, museli by sme namiesto príkazu GO SUB 200 vždy uvádzať celú postupnosť príkazov na riadkoch 200-210.

Na riadku 100 je použitý príkaz STOP na zastavenie programu. Keby sme riadok 100 vynechali, hlavný program vy pokračoval vykonávaním podprogramu bez použitia príkazu GO SUB. Ak by sa mal potom vykonávať príkaz RETURN, počítač ohlási chybu:

```
7 RETURN without GO SUB, 220:1
```

lebo počítač sa snaží vrátiť na miesto volania podprogramu, ale nevie kam, lebo nebolo použité volanie podprogramu príkazom GO SUB.

Ďalšie vlastnosti podprogramu vysvetlíme po nasledujúcom príklade:

```
 10 LET x=6:LET y=4:LET s=0
 20 GO SUB 100
 30 LET x=7:LET y=3
 40 GO SUB 100
 50 LET x=12:LET y=3
 60 GO SUB 100
 70 GO TO 140
100 LET n=x*y
110 PRINT x;"*";y;"=";n
120 LET s=s+1
130 RETURN
140 PRINT "Podprogram bol volany ";s;" krat."
```

Na riadkoch 10, 30, 50 sa priradia hodnoty premenným x, y. Podprogram na riadkoch 100-130 vypočíta súčin zadaných hodnôt a vypíše výsledok.

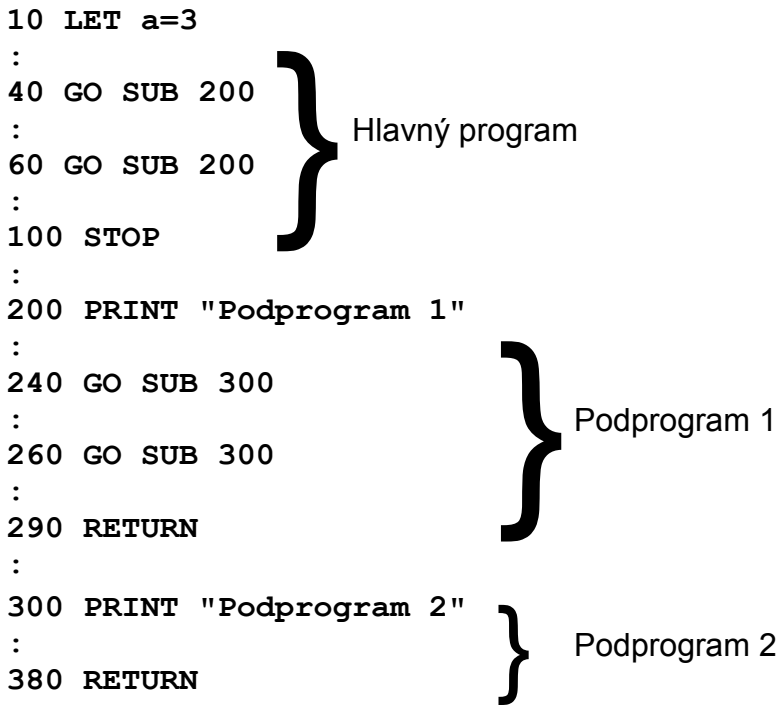
Podprogram môže pracovať s tými premennými, s ktorými pracuje hlavný program, dokonca môže tieto hodnoty zmeniť (to je niekedy nevýhoda, ale nedá sa s tým nič robiť). V našom príklade sa mení v podprograme hodnota premennej s. Ak sa skončí podprogram, hlavný program môže pracovať s touto premennou, pričom hodnota premennej zostala nastavená z podprogramu. Na toto musíme dávať pozor. U nás premenná s predstavuje počítadlo, koľkokrát bol podprogram volaný.

Na riadku 70 treba použiť príkaz GO TO, aby sa nezačal vykonávať podprogram bez príkazu GO SUB.

Na uvedených príkladoch je vidieť, že podprogram môžeme volať viackrát z rôznych miest programu.

Každý podprogram by sa mal končiť príkazom RETURN. Ak má podprogram viac logických ukončení (vetvenie), doporučuje sa každú vetvu podprogramu zakončiť skokom na spoločný riadok s príkazom RETURN (kvôli prehľadnosti). Vo vnútri podprogramu môžeme použiť ďalší príkaz GO SUB pre volanie ďalšieho podprogramu. Takýto postup voláme vloženie podprogramov do seba. Vo vloženom podprograme môžeme použiť opäť príkaz GO SUB atď. Počet vložených podprogramov je obmedzený len veľkosťou voľnej pamäti pre zásobník.

Správne volanie podprogramov je ukázané na nasledujúcom obrázku:



Na riadkoch 40 a 60 sa volá podprogram 1.

V podprograme 1 sa volá na riadkoch 240 a 260 ďalší (vložený) podprogram 2.

Ak použijete viac podprogramov vložených do seba, príkazy GO SUB a RETURN sa môžu vyskytnúť v poradí GO SUB, GO SUB a RETURN, ale nie v postupnosti GO SUB, RETURN, RETURN.

Kvôli prehľadnosti sa umiestňujú podprogramy na koniec hlavného programu, i keď môžu byť umiestnené na ľubovoľnom mieste programu.

Na koniec si povedzme, že číslo riadku v príkaze GO TO a GO SUB môže byť zadané aritmetickým výrazom. Preto môžeme výstižným menom premennej označiť začiatok podprogramu a na toto meno sa potom odvolávať. Napr.

```
10 LET obsah=100: LET hlavicka=200: LET este=30
:
80 GO SUB hlavicka: GO SUB obsah: GO TO este
```

1.2.15. Práca s magnetofónom

Ak vypneme počítač, program uložený v počítači je pre nás stratený. Keby sme museli každý program vložiť cez klávesnicu do počítača, bolo by to veľmi zdĺhavé a nezáživné. Je preto vhodné užitočné programy niekam "odložiť" a v prípade potreby ich vložiť späť do počítača. Jedným z najjednoduchších spôsobov je uchovávanie programov a dát na obyčajnej magnetofónovej kazete.

Použiť môžete ľubovoľný magnetofón, pripojenie k počítaču bolo popísané v úvode príručky.

Pre prácu s magnetofónom sú určené príkazy SAVE, LOAD, VERIFY a MERGE.

Program z počítača na kazetu nahráme príkazom **SAVE**. Predpokladajme, že v počítači je uložený tento program:

```
 1 PRINT 1
 2 PRINT 2
10 PRINT 10
20 LET x=20
```

Tento program chcete uložiť na kazetu. Zadajte príkaz

```
SAVE "test"
```

Test je názov programu, pod ktorým bude uložený na pásku. Názov programu môže mať až 10 znakov (písmená alebo číslice). Po odoslaní príkazu počítač vypíše:

```
Start tape, then press any key
```

čo znamená v preklade "pusti magnetofón a stlač nejaký kláves". Po stlačení klávesu uvidíte na okraji obrazovky vodorovné bledomodré a červené pruhy široké asi 1 cm, ktoré sa pohybujú pomaly hore po dobu asi 5 sekúnd. Potom nasleduje krátky úsek modrých a žltých pruhov, nasleduje normálna obrazovka a hneď opäť bledomodré a červené pruhy a nakoniec modré a žlté pruhy.

Informácie sú z počítača na magnetofón vysielané v dvoch blokoch, kde každý blok má zavádzaciu časť (bledomodré a červené pruhy), blok (hlavička záznamu) obsahuje názov a ďalšie informácie o zázname, druhý blok sú vlastné informácie (program a premenné). Biela časť medzi blokmi je medzera.

Praktický postup pri nahrávaní programu bude:

1) pripojte magnetofón k počítaču
2) nastavte kazetu na úsek, na ktorý chcete nahrávať (pozor na zavádzaciu časť kazety)
3) zadajte SAVE "test" a stlačte ENTER
4) nastavte úroveň pre záznam podľa návodu pre magnetofón a spustíte magnetofón na nahrávanie záznamu
5) stlačte nejaký kláves
6) pozorujte na obrazovke priebeh nahrávania; po ukončení nahrávania sa vypíše hlásenie

```
0 OK, 0:1
```

Potom môžete zastaviť magnetofón.

Nahratý program na páske doporučujeme ihneď skontrolovať príkazom **VERIFY** nasledovným postupom:

1) previňte kazetu pred začiatok záznamu
2) zadajte príkaz
   VERIFY "test"
3) pustite magnetofón

Pokiaľ nie je na páske začiatok nahrávky, striedajú sa na okraji obrazovky farby červená a bledomodrá. Potom sa objaví rovnaký vzor pruhov ako pri zázname na pásku. Po prečítaní hlavičky sa objaví výpis

Program: test

a začína kontrola záznamu. Kontrola sa vykonáva tak, že záznam na páske sa porovnáva s obsahom pamäti v počítači (preto treba kontrolu vykonať ihneď bez zmeny programu v počítači).

Po úspešnej kontrole nasleduje známy výpis OK.

Ak sa tak nestalo, príčin môže byť viac. Skontrolujte prepojenie počítača a magnetofónu. Ak máte v magnetofóne reproduktor, mali by ste pri prehrávaní počuť zvuk podobný "morzeovke". Keď sa neobjavia vôbec pruhy ani výpis hlavičky, prepojenie je nesprávne alebo úroveň záznamu je zle nastavená.

Ak sa po výpise hlavičky prestanú zobrazovať pruhy, skontrolujte meno programu. Počítač totiž vypisuje všetky hlavičky na obrazovku, no čaká až na program zadaného mena.

Niekedy sa stane, že hlavička je vypísaná, ale objaví sa hlásenie

R Tape loading error

teda k chybe došlo v priebehu porovnávania. Nastavte kazetu pred záznam a skúste znovu príkaz VERIFY. Ak to nepomôže, nahrávka na páske je zlá (špinavá hlava, nekvalitná páska, ...) a treba skúsiť program nahrať znovu. Ak ani to nepomôže, skúste inú pásku.

Ak máte program nahratý na páske, pre nahratie do počítača použijeme príkaz **LOAD** v tvare:

LOAD "test"

Ak bol program uložený na pásku riadne skontrolovaný, nemal by byť problém pri použití príkazu LOAD. Problém môže byť však pri nahrávaní programu z pásky nehratej na inom magnetofóne. Postup v tomto prípade bol popísaný na začiatku príručky. Samozrejme, že programy pre iné typy počítačov (nekompatibilné so SINCLAIR) sa Vám nepodarí nahrať.

Upozorňujeme, že príkazom LOAD sa starý program a premenné v počítači zmažú.

Ak starý program nechceme zmazať, použijeme príkaz **MERGE** v tvare:

MERGE "test"

ktorý nahradí odpovedajúce riadky programu a hodnoty odpovedajúcich premenných z nového programu, nové riadky pridá (ako keby sme riadky zadávali postupne z klávesnice).

Doteraz uvedené príkazy pre prácu s magnetofónom sú:

SAVE - nahráva program a premenné na pásku
VERIFY - kontroluje program a premenné na páske s tými, ktoré sú v počítači
LOAD - maže starý program a premenné a nahradzuje ich novými, ktoré číta z pásky
MERGE - je podobné ako LOAD s výnimkou toho, že neruší starý program a premenné, pokiaľ sú riadky inak očíslované a premenné majú iné označenie

V každom z týchto príkazov je kľúčové slovo nasledované menom programu, čo môže byť ľubovolný reťazec (môže to byť teda aj reťazcový výraz, nie iba reťazcová konštanta). Pre SAVE je to meno programu, pod ktorým má byť na pásku uložený, zatiaľ čo ďalšie 3 "hľadajú" program na páske. Počas hľadania sa vypisujú mená záznamov, na ktoré sa narazilo. Identifikácia mena môže byť dvojaká. Pre VERIFY, LOAD a MERGE môžete použiť prázdny reťazec ako meno, ktoré hľadáte. Potom sa počítač nezaujíma o meno, ale berie prvý program, na ktorý narazí. Ak nechceme nahrať premenné spolu s programom (môžu podstatne predĺžiť záznam), zmažeme ich príkazom CLEAR pred použitím príkazu SAVE.

Druhá možnosť príkazu SAVE je táto:

SAVE meno LINE číslo

Program je uchovaný tak, že pri spätnom čítaní pomocou LOAD sa automaticky skáče na riadok s udaným číslom, odkiaľ začína program pracovať. Zatiaľ jediným druhom informácií, ktoré sme uchovávali na páske, boli programy spoločne s ich premennými. Sú tu však ďalšie možnosti pre pole a časť pamäti.

S poľom sa pracuje trochu odlišne. Pole môžete uchovať pri použití DATA v príkaze SAVE:

SAVE reťazec DATA meno poľa ()

Reťazec je meno o uchovanom zázname na páske a pracuje rovnako, ako keď uchovávame program alebo byte. Meno poľa špecifikuje pole, ktoré chcete uchovať, ide o jedno písmeno (pre číselné pole) alebo písmeno a znak $ (pre reťazcové pole). Pamätajte na zátvorky za menom poľa. Možno si myslíte, že sú zbytočné, ale počítaču uľahčujú prácu. Musí Vám byť jasný rozdiel medzi reťazcom a menom poľa. Pokiaľ napr. zadáte

```
SAVE "udaje" DATA b()
```

potom príkaz SAVE berie pole b a ukladá ho na pásku pod menom "udaje". Keď napíšete

```
VERIFY "udaje" DATA b()
```

počítač bude hľadať číselné pole uložené na páske pod menom "udaje". Keď ho počítač nájde, oznámí:

```
Number array: udaje
```

a kontroluje ho spätne proti poľu b v počítači.

```
LOAD "udaje" DATA b()
```

hľadá pole na páske a potom, pokiaľ je pre neho v počítači miesto, zruší v počítači pole b (pokiaľ existuje) a nahrá nové pole b. Príkaz MERGE nemôžete použiť pri uchovaní polí.

Takisto môžete uložiť na pásku reťazcové pole. Keď ho počítač nájde na páske, vypíše:

```
Character array: meno
```

Keď nahrávate do pamäti reťazcové pole, neruší to len reťazcové pole s týmto menom, ale tiež reťazec s týmto menom.

Uložiť časť pamäti je možné bez ohľadu na jej ďalšie použitie. Mohol by to byť televízny obraz, užívateľom definovaná grafika, atď. Obsah obrazovky uložíme príkazom

```
SAVE "obrazovka" CODE 16384,6912
```

Jednotkou pamäti je byte (číslo medzi 0 a 255), každý byte má svoju adresu (číslo medzi 0 a 65535). Prvé číslo po CODE je adresa prvého bytu, ktorý sa má nahrať na pásku a druhé je počet bytov, ktoré sa nahrajú. V našom prípade je 16384 adrea prvého bytu v pamäti obrazovky a 6912 je počet bytov v nej obsiahnutých. Tak uložíme kópiu televíznej obrazovky. Skúste to! Meno "obrazovka" je meno, podobne ako u programu. Na nahranie späť do počítača použijeme

```
LOAD "obrazovka" CODE
```

Môžete použiť aj tvar

LOAD meno CODE začiatok, dĺžka

Keď počítač zistí, že dĺžka záznamu na páske je dlhšia, než príkaze LOAD, vypíše sa správa

```
R Tape loading error
```

Pokiaľ sa dĺžka vynechá, bude počítač čítať pásku bez ohľadu na dĺžku. Začiatok udáva adresu, kam sa má uložiť prvý byte (môže byť rozdielny od adresy v príkaze SAVE). Pokiaľ sú adresy rovnaké, môže sa adresa v príkaze LOAD vynechať (začiatočnú adresu počítač načíta z pásky).

Príkaz SAVE meno CODE 16384,6912 pre uloženie pamäti obrazovky je tak užitočný, že môžete použiť jednoduchší príkaz

SAVE meno SCREEN$

a pre čítanie z pásky zase

LOAD meno SCREEN$

Tu je zriedkavý prípad, kedy VERIFY nebude pracovať, lebo VERIFY po prečítaní hlavičky z pásky napíše meno na obrazovku (tým zmení pamäť obrazovky) a tak pri verifikácii, kedy je obraz zmenený, dochádza k verifikačnej chybe. Vo všetkých iných prípadoch by ste mali použiť príkaz VERIFY po použití SAVE.

Meno v uvedených príkazoch je akýkoľvek reťazec, pod ktorým je informácia uchovaná na páske. Mal by sa skladať z ASCII znakov, avšak môže ich byť maximálne 10.

Poznáme 4 druhy informácii, ktoré môžu byť uchované na páske:

program a premenné spolu, číselné pole, reťazcové pole, časť pamäti.

Keď VERIFY, LOAD a MERGE hľadajú informácie na páske s uvedeným menom, potom sa na obrazovke vypisujú mená záznamov na páske, ktoré sa našli. Pred menom záznamu je informácia o type záznamu:

`program, number array, character array, bytes.`

Pokiaľ bolo meno prázdny reťazec, vezme sa do úvahy prvý záznam zadaného typu bez ohľadu na jeho meno.

SAVE uchováva informáciu na páske pod daným menom. Hlásenie

F Invalid file name

sa vypíše vtedy, keď je meno prázdny reťazec alebo má viac ako 10 znakov. SAVE vždy vyvolá správu:

Start tape, then press any key.

a počítač čaká na stlačenie klávesu.

Teraz uvedieme kompletný zoznam štyroch príkazov pre prácu s magnetofónom.

Príkaz SAVE

1. Program a premenné
   SAVE meno LINE číslo riadku
   uchováva program tak, že po LOAD automaticky nasleduje GO TO číslo riadku. Ak nie je daná časť LINE, program sa po LOAD iba nahrá.
2. Časť pamäti
   SAVE meno CODE začiatok, dĺžka
   uloží časť pamäti, ktorá začína na uvedenej adrese a má zadanú dĺžku
   SAVE meno SCREEN$
   je vlastne
   SAVE meno CODE 16384,6912
   a uloží televízny obraz
3. Pole
   SAVE meno DATA písmeno ()
   alebo
   SAVE meno DATA písmeno$ ()
   uloží pole, ktorého meno je zadané za DATA

Príkaz VERIFY

1. Program a premenné
   VERIFY meno
   kontroluje program na páske
2. Časť pamäti
   VERIFY meno CODE začiatok,dĺžka
   pokiaľ je záznam na páske, dlhší ako je zadaná dĺžka, dostávame chybu. Inak porovnáva záznam s časťou pamäti počítača od zadanej adresy.
   VERIFY meno CODE začiatok
   porovnáva záznam na páske s časťou pamäti a začína od uvedenej adresy, dĺžka sa berie z hlavičky.
   VERIFY meno CODE
   porovnáva záznam na páske s časťou pamäti, adresa začiatku a dĺžka sa berú z hlavičky
   VERIFY meno SCREEN$
   je to isté ako
   VERIFY meno CODE 16384,6912
   a bude iste chybné
3. Pole
   VERIFY meno DATA písmeno ()
   alebo
   VERIFY meno DATA písmeno$ ()
   kontrolujte pole na páske s poľom v pamäti.

Príkaz LOAD

1. Program a premenné
   LOAD meno
   ruší starý program a premenné, nahrá nový program a premenné z kazety. Pokiaľ bol program uchovaný príkazom
   SAVE meno LINE číslo
   program sa automaticky odštartuje. Chyba
   4 Out of memory
   sa objaví, keď nie je miesto pre nový program a premenné. V tomto prípade starý program a premenné nie sú zrušené.
2. Časť pamäti
   LOAD meno CODE začiatok,dĺžka
   pokiaľ je dĺžka v hlavičke na páske viac než je špecifikované, hlási sa chyba. Inak sa vykoná načítanie záznamu do pamäti.
   LOAD meno CODE začiatok
   nahrá časť pamäti z pásky do počítača, začína sa ukladať na začiatočnej adrese
   LOAD meno CODE
   nahrá časť pamäti z pásky do počítača podľa toho, ako bol špecifikovaný v príkaz SAVE.
   LOAD meno SCREEN$
   nahrá časť pamäti z pásky do video pamäti.
3. Pole
   LOAD meno DATA písmeno ()
   alebo
   LOAD meno DATA písmeno$ ()
   ruší akékoľvek pole nazvané písmenom alebo písmenom$ a vytvára nové pole z pásky. Chyba
   4 Out of memory
   sa objaví, pokiaľ nie je miesto pre nové pole. Staré pole potom nie je zrušené.

Príkaz MERGE

1. Program a premenné
   MERGE meno
   spojuje program z pásky s programom v počítači, pripisuje riadky programu alebo premenné do starého programu. Chyba
   4 Out of memory
   sa objaví, pokiaľ nie je dostatok miesta pre starý a nový program spolu.
2. Časť pamäti - nie je možné použiť
3. Pole - nie je možné použiť

1.2.16. Počítač meria čas

Dosť často budete chcieť, aby program nebežal po stanovenú dobu (čas na prezretie nakresleného obrázku, prečítanie textu, ...). Vtedy použijeme príkaz **PAUSE** v tvare:

PAUSE číslo

kde číslo je celé číslo od 0 do 65535.

Týmto príkazom sa zastaví vykonávanie programu na čas, ktorý je určený číslom. Jednotkou parametra číslo je 1/50 sekundy, teda čas zastavenia programu v sekundách bude číslo/50.

Číslo=0 znamená zastavenie vykonávania programu až do stlačenia nejakého klávesu.

Ukončenie príkazu PAUSE je možné stlačením nejakého klávesu.

Príkaz PAUSE využijeme v programe pre nakreslenie chodu hodín:

```
 10 REM prvy urobime cifernik
 20 FOR n=1 to 12
 30 PRINT AT 10-10*COS (n/6*PI), 16+10*SIN (n/6*PI);n
 40 NEXT n
 50 REM teraz nastartujeme hodiny
 60 FOR t=0 TO 200000: REM t je cas v sekundach
 70 LET a=t/30*PI: REM a je uhol sekundovej rucicky v radianoch
 80 LET sx=72*SIN a:LET sy=72*COS a
200 PLOT 128,88: DRAW OVER 1;sx,sy: REM kresli sekundovou rucicku
210 PAUSE 42
220 PLOT 128,88: DRAW OVER !;sx,sy: REM maze sekundovou rucicku
400 NEXT t
```

Hodiny pôjdu asi 55,5 hodín, čo je dané riadkom 60, ale môžete to ľahko zmeniť. Všimnite si, ako je čas na riadku 210 zadaný. Asi ste čakali PAUSE 50, ale výpočet tiež zaberie nejaký čas. Údaj v PAUSE budete musieť zrejme opraviť podľa chyby, ktorú hodiny vykazujú, ak sa porovnávajú s reálnym častom (hodiny idú s presnosťou asi 2 %).

Ak chcete presnejšie hodiny, použite nasledujúci postup, ktorý používa obsah určitej časti pamäti, ktorá sa nastavuje riadiacim programom Vášho počítača v závislosti od času. Uchované dáta sa vyberajú pomocou funkcie PEEK. Nasledujúci výraz

```
t=(65535*PEEK 23674+256*PEEK 23673+PEEK 23672)/50
```

je počet sekúnd od zapnutia počítača (po 3 dni a 21 hodín potom sa mení na 0 a počíta ďalej).

Upravený program pre hodiny bude potom:

```
 10 REM najskor urobime ciselnik
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS (n/6*PI),16+10*SIN (n/6*PI);n
 40 NEXT n
 50 DEF FN t()=INT ((65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50): REM
    pocet sekund od zaciatku
100 REM teraz nastartujeme hodiny
110 LET t1=FN t()
120 LET a=t1/30*PI : REM a je uhol sekundovej rucicky v radianoch
130 LET sx=72*SIN a: LET sy=72*COS a
140 PLOT 128,88: DRAW OVER 1=;sx,sy: REM nakresli rucicku
200 LET t=FN t()
210 IF t=t1 THEN GO TO 200: REM pockaj, az je cas na dalsiu rucicku
220 PLOT 128,88: DRAW OVER 1; sx,sy: REM maze staru rucicku
230 LET t1=t: GO TO 120
```

Vnútorné hodiny, používajúce túto metódu, majú presnosť asi 0,01%. Tieto hodiny však dočasne zastaví, ak použijete príkaz BEEP, alebo operácie s magnetofónom. Čísla na adresách 23674, 23673 a 23672 sú uložené v pamäti počítača a sú použité pre počítanie 50-nách sekundy. Každé je v rozmedzí od 0 do 255, po 255 sa vracia na 0. Najčastejšie sa mení číslo na adrese 23672 (najnižší rád). Vždy za 1/50 sekundy sa zväčší o 1. Keď je 255, v nasledujúcom kroku sa zmení na 0 a v rovnakom čase sa zvýši číslo na adrese 23673 o 1. Keď má toto hodnotu 255 a mení sa na 0, zároveň číslo na adrese 23674 sa zvýši o 1. Zväčšenie hodnôt sa vykonáva automaticky prerušením tak, že vykonávanie programu v jazyku BASIC sa zastaví, vykoná sa podprogram v strojovom kóde pre zmenu systémového času a vykonávanie programu v jazyku BASIC pokračuje. Toto prerušenie je časovo také malé, že ho nezaregistrujete.

Teraz starostlivo uvažujte. Predpokladajme naše tri čísla: 0 (pre adresu 23674), 255 (pre adresu 23673) a 255 (pre adresu 23672). To znamená, že čas v sekundách je

```
(65536*0 + 256*255 + 255)/50=1310.7
```

Ale je tu nebezpečenstvo. Práve sa ukončilo vyhodnotenie funkcie PEEK 23674 (hodnota 0) a nastalo nasledujúce prerušenie. Hodnoty v pamäti sa zmenia na 1, 0, 0 a až potom nasleduje vyhodnotenie funkcií PEEK 23673 a PEEK 23672 (teraz majú hodnoty 0, 0). Preto platí:

```
t = (65536*0 + 256*0+0)/50=0
```

čo je pochopiteľne chybne. Tomuto problému sa vyhneme tak, že budeme počítať čas 2 krát a vezmeme väčšiu hodnotu.

Tu je úprava pre uvedený postup. Definujte

```
10 DEF FN m(x,y)=(x+y+ABS(x-y))/2:REM vacsie z x a y
12 DEF FN u()=(65536*PEEK 23674 + 256*PEEK 23673+PEEK 23672)/50: REM cas
   moze byt chybny
14 DEF FN t()=FN m(FN u(),FN u()):REM spravny cas
```

Hodnoty v pamäti môžete však aj nastaviť, napr. nastavíme čas na 10 hodinu dopoludnia:

```
10*60*60*50=1800000=65536*27 + 256*119+64
```

Na nastavenie hodnôt na 27,119, 64 použijete:

```
POKE 23674,27: POKE 23673,119: POKE 23672,64
```

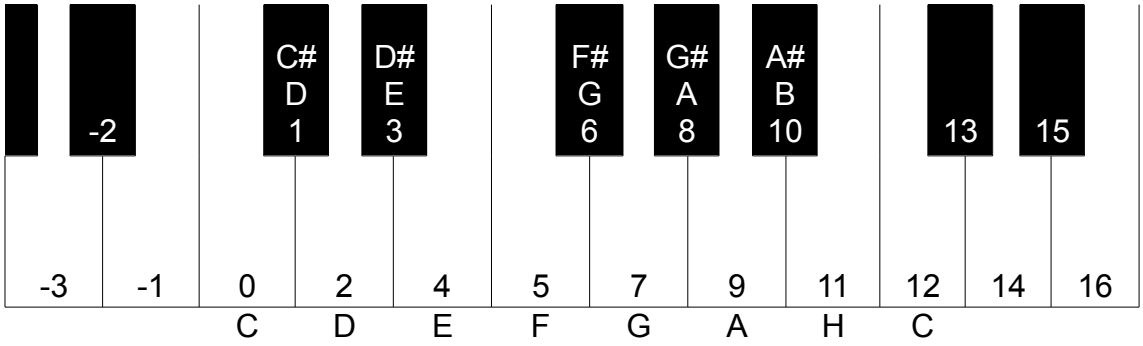
1.2.17. Počítač ako hudobník

Váš počítač má zabudovaný reproduktor, ktorý môžeme ovládať príkazom **BEEP** v tvare

BEEP dĺžka, tón

kde dĺžka a tón sú aritmetické výrazy. Dĺžka trvania sa zadáva v sekundách, tón sa zadáva v poltónoch nad stredným C. Pre nižšie noty sa používajú záporné čísla.

Nasledujúci diagram ukazuje hodnoty všetkých nót jednej oktávy na klávesnici klavíra:



Na získanie vyšších alebo nižších nôt musíme pridať alebo odčítať 12 pre každú oktávu hore i dole. Pokiaľ máte pri programovaní vedľa seba klavír, bude Vám stačiť k práci s ladením tento diagram.

Tu je jedna ukážka programu:

```
10 PRINT "Gustav"
20 BEEP 1,0: BEEP 1,2: BEEP .5,3: BEEP .5,2: BEEP 1,0
30 BEEP 1,0: BEEP 1,2: BEEP .5,3: BEEP .5,2; BEEP 1,0
40 BEEP 1,3: BEEP 1,5: BEEP 2,7
50 BEEP 1,3: BEEP 1,5: BEEP 2,7
60 BEEP .75,7:BEEP .25,8: BEEP .5,5: BEEP .5,3: BEEP .5,2: BEEP 1,0
70 BEEP .75,7:BEEP .25,8: BEEP .5,7: BEEP .5,5: BEEP .5,3: BEEP 1,0
80 BEEP 1,0: BEEP 1,-5: BEEP 2,0
90 BEEP 1,0: BEEP 1,-5: BEEP 2,0
```

Keď odštartujete program, mali by ste počuť pohrebný pochod z Mahlerovy prvej symfónie, časť, kde škriatkovia pochovávajú muža z americkej kavalérie. Predpokladajte, že Vaša melódia je písaná v kľúči C mol, podobne, ako Mahlerova.

Pokiaľ chcete zmeniť stupnicu, je najlepšie vložiť premennú stup pred každou hodnotou výšky tónu:

```
20 BEEP 1, stup+0: BEEP 1, stup+2: BEEP .5, stup+3: BEEP .5,stup+2:
   BEEP 1,stup+0
```

Pred spustením programu musíte dať vhodnú hodnotu pre stup. 0 pre C mol, 12 pre C mol zdvihnuté o jednu oktávu, atď. Môžete prinútiť počítač, aby znel ako iný nástroj, keď stup je vo forme zlomku. Takisto môžete upravovať dĺžku nót. Pretože tu išlo o pomalú skladbu, volili sme 1 sekundu pre štvrťovú notu, a založili na tom pravidlo, že napr. osminová nota znie 1/2 sekundy atď. Oveľa pružnejšie je použiť premennú cas tak, aby mal dĺžku štvrťovej noty a potom špecifikovať jej trvanie. Tak riadok 20 by teraz vyzeral:

```
20 BEEP cas, stup+0: BEEP cas, stup+2: BEEP cas/2,stup+3:
   BEEP cas/2,stup+2: BEEP cas,stup+0
```

Nastavte cas na rôzne hodnoty môžete meniť ľahko rýchlosť hrania. Uvedomte si, že v počítači je len jeden reproduktor a tak môžete hrať v jednom okamžiku len jednu notu. Skúste programovať svoje melódie, začnite najprv celkom jednoduchou. Pokiaľ nemáte klavír a nepoznáte noty, obstarajte si jednoduchý nástroj ako je píšťala a tvorte melódie. Počítač vytvorí každý tón ako tento nástroj.

Teraz zadajte:

```
FOR n=0 TO 1000: BEEP .5,n: NEXT n
```

Počítač bude hrať postupne zvyšujúce sa tóny a potom sa zastaví so správou:

```
B Integer out of range
```

lebo n je mimo rozsah. Skúste to isté pre klesajúce tóny. Najnižšie tóny budú znieť ako klepanie, i keď sa nízke tóny tvoria rovnako, ako vyššie tóny. Len uprostred povoleného rozsahu (od -60 do 69) sú tóny vhodné pre hudbu. Nízke tóny počujeme ako klepanie, vyššie sú slabé. Teraz vložte tento riadok:

```
10 BEEP .5,0: BEEP .5,2: BEEP .5,4: BEEP .5,5: BEEP .5,7: BEEP .5,9:
   BEEP .5,11: BEEP .5,12: STOP
```

Príklad hrá stupnicu C dur, ktorá používa všetky biele klávesy na klavíri od stredného C po nasledujúce C. Stupnica neznie rovnako ako na klavíri, nazýva sa temperovaná, pretože interval poltónu je rovnaký po celý čas. Huslista by hral stupnicu odlišne, tak, že by tóny zneli pre ucho príjemnejšie. Môže to urobiť tým, že jemne sunie prsty po strune, čo klavirista nemôže. Základná stupnica, ktorú hraje huslista, je asi nasledujúca:

```
20 BEEP .5,0: BEEP .5,2.039: BEEP .5,3.86: BEEP .5,4.98: BEEP .5,7.02:
   BEEP 5,8.84: BEEP .5,10.88: BEEP .5,12: STOP
```

Možno, že nebudete schopní poznať rozdiel medzi stupnicami. Prvý rozdiel je v tom, že tretí tón je nižší než v prirodzenej ladiacej stupnici. Pokiaľ ste špecialista, môžete v tejto tónine programovať. Nevýhodou je, že keď to pracuje dobre v tónine C, tak v niektorých stupniciach to možno urobiť len ťažko. V temperovanej stupnici to však môžete použiť vždy.

Niektoré hudby, obzvlášť indiánska, používajú intervaly menšie, než je časť tónu. Toto môžete programovať s použitím BEEP bez problémov, napr. štvrťtón hodnotou C má hodnotu .5.

Pokiaľ chcete počuť kvalitnejšie zvuky, musíte použiť iný než vstavaný reproduktor (signál je prítomný v zásuvke pre magnetofón). Ďalšou možnosťou je použiť zosilňovač s reprosústavou a s výsledným zvukom budete iste spokojní. Tóny môžete samozrejme nahrávať na magnetofón a potom prehrávať.

1.2.18. Použitie tlačiarne

Váš počítač nie je tak technicky vybavený, aby ste mohli k nemu okamžite pripojiť tlačiareň. Ak chcete nejakú tlačiareň pripojiť, budete zrejme potrebovať paralelný interface a tiež program pre ovládanie výpisu na určený typ tlačiarne. V prípade záujmu o pripojenie tlačiarne si môžete príslušný interface a ovládací program objednať vo v.d. Didaktik Skalica.

Ak máte tlačiareň pripojenú, môžete použiť príkazy pracujúce s tlačiarňou.

Prvé dva príkazy **LPRINT** a **LLIST** sú obdobné ako PRINT a LIST, ibaže namiesto obrazovky používajú tlačiareň. Ich použitie ukazuje nasledujúce program:

```
10 LPRINT "tento program"
20 LLIST
30 LPRINT "tlaci postupnost znakov"
40 FOR n=32 TO 255
50 LPRINT CHR n;
60 NEXT n
```

Príkaz na riadku 10 vypíše tlačiarní zadaný text.

Príkaz na riadku 20 vypíše listing programu, ďalej sa vypíše zadaný text (riadok 30).

Cyklus na riadkoch 40-60 vypíše postupne všetky znaky s kódom 32 až 255. Výsledok závisí od ovládacieho programu, ktorý používate. Zostáva posledný príkaz pre kópiu obrazovky

```
COPY
```

ktorý je určený špeciálne pre tlačiareň ZX Printer (je to jednoihličková tlačiareň, ktorá sa v zahraničí vyrábala).

Iste ste objavili na klávesnice i príkazy **MOVE**, **ERASE**, **CAT** a **FORMAT**. Sú určené pre vonkajšiu pamäť ZX Microdrive a pracujú až po pripojení tohto zariadenia. Nebudeme sa nimi zaoberať, pretože toto zariadenie sa už nevyrába.

Ďalšie dva príkazy **OPEN** a **CLOSE** sa používajú pre otvorenie a zatvorenie kanálov. Ich použitie bude podrobne vysvetlené v druhej časti príručky.

1.2.19. Vstupno/výstupné miesta počítača

Procesor počítača vie čítať a zapisovať do pamäti pomocou príkazov PEEK a POKE. Procesor sa nezaujíma, či pamäť je typu ROM, RAM, alebo či pamäť nie je, vie len, že je 65536 adries pamätí a môže čítať za každej adresy pamäti a zapísať 1 byte na 1 adresovateľné miesto. Podobne je k dispozícii 65536 miest, ktorým hovoríme **miesta I/O** (vstupno/výstupné). Tieto sú použité procesorom pre komunikáciu s vonkajším zariadením (klávesnica, tlačiareň,...) a môžu byť riadené v jazyku BASIC príkazom OUT a ich stav môže byť zisťovaný použitím funkcie IN. Upozorňujeme užívateľov, že ak sú úplní začiatočníci, bolo by vhodné pre ďalšie pochopenie preštudovať kapitolu "Číselné sústavy" z druhej časti príručky.

Funkcia **IN** je podobná funkcií PEEK a má tvar

```
IN adresa
```

kde adresa je v rozsahu 0 až 65535. Výsledok je hodnota bytu načítaná z vstupno/výstupného miesta s touto adresou.

Príkaz **OUT** je príkaz podobný POKE a má tvar

```
OUT adresa,hodnota
```

OUT zapíše danú hodnotu na miesto určené adresou. Ako je adresa chápaná, záleží na technickom usporiadaní počítača.

Adresa je určená pomocou 16 bitov, ktoré označujeme ako

```
A15, A14, A13, A12, ..... A2, A1, A0
```

A0 je prvý bit, A1 je druhý bit atď. Bity A0 až A4 sú najdôležitejšie. Normálne sú 1, ale pokiaľ niektorý z nich je 0, hovorí to počítaču, aby urobil niečo špecifického. Počítač nemôže robiť zároveň viac, ako jednu činnosť a tak nie viac ako jeden z týchto piatich bitov sú o 1 menšie než násobky 32, t. j. že A0....A1 sú 1. Bity A8, A9 sú niekedy použité pre zvláštne informácie. Prečítaný byte má 8 bitov a tie sa často označujú ako D7, D6, ... D1, D0. V práci so vstupmi/výstupmi je rad vstupných adries, ktoré čítajú klávesnicu a tiež signál z magnetofónu.

Klávesnica je rozdelená na 8 polovíc radov po 5 klávesoch.

IN 65278	číta polovicu rady od	CAPS SHIFT do V
IN 65022	-||-	A ..... G
IN 64510	-||-	Q ..... T
IN 63486	-||-	1 5
IN 61438	-||-	0 6
IN 57342	-||-	P 7
IN 49150	-||-	ENTER ..... H
IN 32766	-||-	SPACE ..... B

Uvedené adresy vypočítame podľa vzorca 254+256*(255-2^n) pre n=0 až 7. V čítanom byte znamenajú bity D0 až D4 päť klávesov. V danej polovici radu je D0 pre vonkajší kláves, D4 pre najbližší kláves u stredu. Bit je 0, keď je kláves stlačený a 1, keď stlačený nie je.

Pre ilustráciu funkcie IN vyskúšajte nasledujúci program:

```
10 FOR n=0 TO 7: REM polovičný rad
20 LET a=254+256*(255-2^n)
30 PRINT AT 0,0; IN a: GO TO 30
```

a skúšajte stláčať rôzne klávesy. Ak zistíte nejakú spojitosť, stlačte BREAK, potom zadajte NEXT n v priamom režime a pokračujte. Podrobnejšie informácie o vstupno/výstupných miestach sú uvedené v druhej časti príručky.

1.2.20. Práca so strojovým kódom

Ak sa zoznámite s počítačom dôkladnejšie, budete chcieť určite používať podprogramy v strojovom jazyku, vlastné tabuľky znakov, dokonca budete sami vytvárať programy a tabuliek v pamäti tak, aby ich operačný systém nezmazal, lebo on nevie, či danú časť pamäti využívate. Vy takisto musíte v takom prípade vedieť, ktoré časti pamäti využíva počítač pre svoju prácu. O tom sa dozviete podrobnejšie pri popise pamäti. Zatiaľ si povedzme, že pri programovaní v jazyku BASIC sa využíva časť pamäti od adresy 0 po adresu, ktorá sa nazýva **RAMTOP**. Po resetovaní počítača sa nastaví RAMTOP automaticky na hodnotu 65367, lebo nad touto adresou je uložená tabuľka znakov užívateľskej grafiky a táto je chránená. Dokonca príkaz NEW, ktorý maže pamäť RAM, pracuje len po RAMTOP, takže nemaže užívateľom definovanú grafiku. Aby ste si vytvorili priestor pre vlastné potreby, potrebujete nastaviť adresu RAMTOP. To umožňuje príkaz **CLEAR** v tvare

CLEAR adresa

kde adresa určuje novú adresu RAMTOP. Ostatná činnosť príkazu CLEAR ostáva, ako už bolo uvedené. Zopakujme si to:

1. vymaže všetky premenné
2. zmaže obrazovku
3. nastaví kurzor pre PLOT do ľavého spodného rohu
4. vykoná sa RESTORE
5. vymaže zásobník GO SUB

Ak použijete CLEAR týmto spôsobom, môžete buď posunúť RAMTOP, aby bolo viac miesta pre BASIC (prepísaním užívateľských grafických symbolov) alebo posunieme RAMTOP dole, aby nebola horná časť pamäti mazaná príkazom NEW a aby nebolo prepísaná pri práci v jazyku BASIC. Túto časť pamäti však môžeme meniť príkazom POKE a jej stav čítať funkciou PEEK.

Ak je zadaná adresa príliš malá (CLEAR 23000), ohlási sa chyba

M RAMTOP no good

Takže vytvoriť miesto pre program v strojovom kóde si už vieme. Nasledujúca časť je napísaná pre tých, ktorí poznajú strojový kód Z 80, súbor inštrukcií, ktorému rozumie procesorový čip Z 80. Pokiaľ máte záujem o podrobnejšie informácie, zožeňte si špeciálnu literatúru. Programy v strojovom kóde sú písané v jazyku, ktorý sa volá assembler. Ak program nie je príliš dlhý, môžete preklad previesť sami (inštrukcie strojového kódu sú uvedené v prílohe). Skúste tento program:

```
ld bc, 99
ret
```

ktorý uloží do dvojice registrov bc hodnotu 99. Program preloží na 4 byty 1, 99, 0 (pre ld bc, 99) a 201 (pre ret).

Ak máte program napísaný v strojovom kóde, ďalšou úlohou je vložiť ho do počítača. Musíte sa rozhodnúť, kam ho v pamäti uložiť a za tým účelom si preň rezervovať v pamäti dostatok miesta.

Na vytvorenie miesta a uloženie programu v strojovom kóde môžete použiť nasledovný program:

```
 5 CLEAR 32499
10 LET a=32500
20 FOR i=0 TO 3
30 READ n: POKE a+i,n
40 NEXT i
50 DATA 1,99,0,201
```

Na odštartovanie programu v strojovom kóde sa používa funkcia **USR** v tvare:

USR adresa

kde adresa je štartovacia adresa začiatku programu v strojovom kóde. Hodnota funkcie je hodnota registrov b, c procesora Z 80 pri návrate z tohto programu, preto je v rozsahu od 0 do 65535.

Pretože USR je funkcia, musí sa použiť v spojitosti s nejakým príkazom alebo môže byť časťou výrazov. Najčastejšie sa používajú tieto možnosti:

PRINT USR adresa

kedy sa vypíše hodnota tejto funkcie na obrazovku.

V našom prípade po zadaní

```
PRINT USR 32500
```

sa vypíše na obrazovku číslo 99, čo je obsah registrov b,c.

Ďalšou možnosťou je použitie

RANDOMIZE USR adresa

kedy sa hodnota funkcie uloží na adresu, na ktorú sa ukladá číslo z postupnosti náhodných čísiel.

Použiť môžete ešte

RESTORE USR adresa

kedy sa výsledok nikde neukladá.

Návrat do jazyka BASIC je zabezpečený po regulárnom ukončení inštrukciou RET.

Funkcia USR sa používa tiež na získanie adresy, na ktorej je uložený znak užívateľsky definovanej grafiky (udg). Funkcia sa používa v tvare:

USR "znak"

kde znak určuje znak užívateľskej grafiky. Napr. prvý znak udg je určený znakom "a", "A" alebo znakom zadaným stlačením klávesu A v grafickom móde klávesnice. Po resetovaní je hodnota

```
USR "a" = USR "A" = 65368
```

Definovanie znaku udg si ukážeme na príklade. Definujeme si prvý znak udg ako pole šachovnice rozmeru 8x8 bodov. Šachovnica bude daná hodnotami BIN 10101010 a BIN 01010101. Program na definovanie znaku bude:

```
1000 FOR n=0 TO 6 STEP 2
1010 POKE USR "a"+n,BIN 01010101: POKE USR "a"+n+1, BIN 10101010
1020 NEXT n
```

Ak tlačíte prvý znak udg (grafický mód a potom kláves A) červenou farbou na žltom podklade, je výsledná farba oranžová. Zmenou kombinácie farieb dostanete ďalšie "nové" farby pre farbenie väčších plôch obrazovky.

2. Druhá časť

Pre tých, ktorí už zvládli základnú komunikáciu so svojím počítačom a chceli by sa o ňom dozvedieť viac, je určená táto druhá časť príručky.

Dozviete sa v nej niečo o číselných sústavách, o orgánizácii pamäti počítača, ako pracovať s niektorými inštrukciami a vstupno-výstupnými kanálmi počítača a ďalej o niektorých úžitkových programoch. Venovať sa budeme tiež technickému popisu počítača a spôsobu pripojenia niektorých typických periférií.

2.1. Číselné sústavy

Iste ste už niekedy počuli, že počítač počíta v **dvojkovej sústave**. Čo to znamená. Počítač je zariadenie, ktoré vďaka svojej konštrukcii z elektronických obvodov rozoznáva len dva základné logické stavy, a to stav ÁNO - označme si ho logickou jedničkou, a stav NIE - ten si označíme logickou nulou. Iné stavy v počítači nastať nemôžu.

Všetka práca počítača potom prebieha len v týchto dvoch stavoch a ich kombináciách. Pretože tieto stavy sú dva, hovoríme, že počítač počíta v dvojkovej sústave a všetky príkazy, čísla a programy musia byť do tejto sústavy prevedené. O prevod sa však nestaráte vy, ale rôzne podprogramy v operačnom systéme počítača.

Naša sústava, ktorú bežne používame, sa volá **desiatková** preto, že v nej môže nastať desať rôznych základných stavov. Tieto stavy sú 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Všetky ostatné stavy sú potom vlastne odvodené od týchto desiatich základných. Teda napr. stav 18 je kombináciou stavov 1 a 8. Ale kombináciou stavov 1 a 8 je tiež stav 81. Záleží tu teda i na pozícii základných stavov. Taká číselná sústava sa potom volá **pozičná číselná sústava**.

A na rade je otázka: "Koľko číselných sústav vlastne môže existovať?". Existovať ich môže nekonečne mnoho a tak môžeme mať napríklad sústavu, v ktorej môže nastať 60 rôznych základných stavov (tu používali starí Mayovia).

Ak zhrnieme naše doterajšie poznatky, tak pod **číselnou sústavou so základom N** budeme rozumieť pozičnú číselnú sústavu, v ktorej rozoznávame N základných stavov. Stavom budeme hovoriť číslice, ich kombinácii potom čísla v čísle záleží na poradí číslic.

Ďalej sa budeme zaoberať len desiatkovou, dvojkovou a šestnástkovou sústavou.

2.1.1. Desiatkova sústava

Desiatková sústava je pre nás najprirodzenejšia, pretože v nej už od malička počítame. Na tejto sústave si vysvetlíme spôsob tvorby čísiel z číslic a počítanie s nimi. Tieto poznatky potom skúsime využiť pri tvorbe čísiel v dvojkovej a šestnástkovej sústave pri počítaní v týchto sústavách.

Na identifikáciu čísiel v desiatkovej sústave sa používa znak $_D$ (dekadické vyjadrenie) na konci čísla.

Uvažujme číslice 1, 5, 7 a 8. Tieto číslice napíšeme za sebou a to tak, že 1 bude prvá zľava a 8 posledná zľava.

Dostaneme číslo 1578_D. Toto číslo však môžeme napísať tiež takto:

$$1 \times 1000 + 5 \times 100 + 7 \times 10 + 8 \times 1 = 1578_D$$

Vidíme, že náš zápis 1578_D je vlastne skráteným zápisom tohto čísla. Skrátený zápis si môžeme dovoliť z jedného jediného dôvodu: vieme, že číslica na rôznom mieste (pozícii) môže vyjadrovať rôznu hodnotu. Číslica, umiestnená na prvom mieste sprava vyjadruje jednotky, na druhom desiatky, na treťom stovky atď.

Z matematiky vieme, že desať umocnené na nultú je jedna, zápis je nasledujúci:

$10^0 = 1$. Podobne $10^1 = 10$, $10^2 = 100$, $10^3 = 1000$ atď.

Naše číslo teda môžeme zapísať takto:

$$1 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 8 \times 10^0 = 1578_D$$

Tento zápis je najobecnejší a môžeme z neho odvodiť pravidlá zápisu pre ostatné číselné sústavy.

Číslica pred znamienkom krát môže nadobúdať hodnoty 0 až 9, teda hodnoty všetkých cifier desiatkovej sústavy. Číslo 10 sa nazýva základ sústavy a sústava je po ňom pomenovaná desiatková sústava. Základ sústavy je umocňovaný pozíciou cifry v zápise čísla. Prvá cifra sprava však nemá pozíciu jedna, ale nula. Druhá cifra sprava má pozíciu jedna atď.

Posledný príklad: číslo 1230569_D môžeme podľa vyššie uvedených pravidiel rozpísať takto:

$$1\times10^6+2\times10^5+3\times10^4+0\times10^3+5\times10^2+6\times10^1+9\times10^0 = 1230569_D$$

2.1.2. Dvojková sústava

Základom dvojkovej sústavy je číslo 2. Číslice tejto sústavy teda môžu nadobúdať hodnoty 0 a 1. Tvorba čísiel v tejto sústave je analogická ako v sústave desiatkovej. Na identifikáciu čísiel v dvojkovej sústave budeme používať na konci čísla znak $_B$ (binárne vyjadrenie).

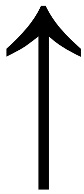
Skúsme si zapísať číslo v dvojkovej sústave: 11001100_B. Toto číslo teraz rozpíšeme:

$$1\times2^7 + 1\times2^6 + 0\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0 =$$
$$1\times128 + 1\times64 + 0\times32 + 0\times16 + 1\times8 + 1\times4 + 0\times2 + 0\times1 =$$
$$128 + 64 + 8 + 4 = 204_D$$

Podobným spôsobom sa dá vyčísliť ľubovolné číslo v dvojkovej sústave a previesť do sústavy desiatkovej.

Opačný prevod (z desiatkovej sústavy do dvojkovej) je trocha zložitejší. Skúsime previesť číslo 204_D z desiatkovej do dvojkovej sústavy. Prevod sa vykonáva delením prevádzaného čísla číslom 2, teda základom číselnej sústavy do ktorej prevádzame. Delenie sa vykonáva tak dlho, dokiaľ nie je výsledok delenie nula:

204 : 2 = 102 zvyšok: 0
102 : 2 = 51 zvyšok: 0
51 : 2 = 25 zvyšok: 1
25 : 2 = 12 zvyšok: 0
6 : 2 = 3 zvyšok: 0
3 : 2 = 1 zvyšok: 1
1 : 2 = 0 zvyšok: 1



Teraz si napíšeme zvyšky po delení (teda 0 a 1) vedľa seba. Začnite zdola v smere šípky, spodná číslica bude zapísaná prvá zľava. Za ňou vpravo napíšte druhú číslicu zdola atď. Dostanete číslo 11001100_B, teda naše pôvodné číslo.

Ostávajú nám ešte desatinné čísla. Tie sa prevádzajú obdobne, iba namiesto delenie dvomi sa číslo násobí dvomi. Pokiaľ výsledok prekročí číslo 1, do dvojkového zápisu sa zapíše 1 a od výsledku sa jednotka odčíta. Pokiaľ výsledok neprekročí číslo 1, do dvojkového zápisu sa zapíše 0 a pokračuje sa v násobení.

Výsledok sa zapíše v tom poradí, ako bol získaný. Teda za desatinnou bodkou sa zapíše prvý výsledok po násobení, potom druhý atď. Uvedieme si príklad. Chceme previesť číslo $4/5_D = 0.8_D$.

Postupujeme takto:

výpočet	**celá časť**	**desatinná časť**
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8

výpočet	celá časť	desatinná časť
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8
$0.8 \times 2 = 1.6$	1	0.6
$0.6 \times 2 = 1.2$	1	0.2
$0.2 \times 2 = 0.4$	0	0.4
$0.4 \times 2 = 0.8$	0	0.8

Dvojkový zápis $4/5_D$. je $0.11001100110011001100110011001100_B$ s presnosťou na 32 miest.

V ďalšom texte budeme nazývať jednotlivé číslice v dvojkovom zápise čísla **bity**. Každý bit čísla môže nadobúdať hodnoty nula alebo jedna. Osem za sebou idúcich bitov budeme nazývať **byte**. Byte môže nadobúdať hodnoty $00000000_B = 0_D$ až $11111111_B = 255_D$. Každý bit v byte má svoje poradové číslo. Prvý bit sprava má poradové číslo 0 (nultý bit, najnižší bit, bit s najmenšou váhou, LSB - Least Significant Bit), ďalší bit má poradové číslo 1 (prvý bit), až siedmy bit sprava má poradové číslo 7 (siedmy bit, najvyšší bit, bit s najväčšou váhou, MSB - Most Significant Bit).

Dva byty môžu vyjadriť číslo 0 až 65535 (256×256 = 64 kB), tri byty číslo 0 až 16777216 ($256 \times 256 \times 256$ = 16 MB) atď.

V prípade, že je číslo vyjadrené dvoma bytami (tiež sa hovorí uložené na dvoch bytoch), hovoríme o **"nižšom byte"** a **"vyššom byte"** čísla. Hodnotu čísla potom dostaneme spočítaním hodnoty nižšieho bytu s 256-násobkom hodnoty vyššieho bytu. V pamäti počítača sú tieto dva byty uložené nasledovne, poradie je dané zvyklosťami firmy **Intel** (u druhých firiem môže byť poradie opačné):

<table>
<tr><td>nižší byte</td><td>vyšší byte</td></tr>
<tr><td>adresa X</td><td>adresa (X+1)</td></tr>
<tr><td colspan="2">hodnota = nižší byte + 256*vyšší byte</td></tr>
</table>

2.1.3. Šestnástková sústava

Základom tejto sústavy je číslo 16. Číslice tejto sústavy môžu nadobúdať hodnoty 0 až 15. Číslice väčšie ako deväť sa však nedajú zapísať jedným znakom a preto bola vytvorená dohoda, že sa budú označovať prvými šiestimi písmenami abecedy. Čísla v šestnástkovej sústave budeme na konci označovať znakom $_H$ (hexadecimálny). Číslice šestnástkovej sústavy sú nasledujúce:

desiatkovo	šestnástkovo
0	0
1	1
2	2
3	3
4	4
5	5
6	6

desiatkovo	šestnástkovo
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Skúsme si previesť číslo $10AF_H$ do desiatkovej sústavy. Číslo sa dá rozpísať tato:

$1x16^3 + 0x16^2 + 10x16^1 + 15x16^0 =$
$1x4096 + 0x256 + 10x16 + 15x1 =$
$4096 + 160 + 15 = 4271_D$

Podobným spôsobom sa dá vyčísliť ľubovolné číslo v šestnástkovej sústave.

Opačný prevod je rovnaký ako u dvojkovej sústavy. Opäť postupným delením čísla v desiatkovej sústave dostaneme zvyšky po delení, ktoré reprezentujú zápis čísla v šestnástkovej do šestnástkovej sústavy:

4271 : 16 = 266 zvyšok: 16 = F
266 : 16 = 16 zvyšok: 10 = A
16 : 16 = 1 zvyšok: 0 = 0
1 : 16 = 0 zvyšok: 1 = 1

Opäť si zapíšeme zvyšky po delení zdola a dostaneme číslo v šestnástkovej sústave, zhodné s našim pôvodným, teda $10AF_H$.

Všetky aritmetické operácie (sčítanie, násobenie, odčítanie atď.) sa v týchto sústavách riadia rovnakými zákonmi ako v desiatkovej sústave.

V ďalšom texte budeme chápať všetky čísla bez označenia $_D$ ako čísla dekadické.

2.2. Vnútorná forma čísla

Každé číslo, s ktorým počítač pracuje, musí byť v jeho pamäti zapísané v nejakej štandardnej forme. BASIC počítača ukladá každé číslo na päť bytov, pričom rozlišuje tzv. malé celé čísla z intervalu **-65535** až **+65535** a ostatné čísla. Malé celé čísla sú uložené takto:

00_H	00_H pre kladné čísla FF_H pre záporné čísla	nižší byte čísla	vyšší byte čísla	00_H

Číslo -257 je uložené následovne $00FF010100_B$.

Číslo +257 obdobne: 0000010100_B.

Ostatné čísla sú uložené v **semilogaritmickom tvare**. Ako tento tvar vyzerá? Číslo je prevedené na tvar: $\pm m \times 2^e$, kde $\pm$ je znamienko, **m** je **mantisa**, ležiaca v rozsahu 1/2 až 1 (nesmie byť 1 a má rozsah 4 byty) a **e** je **exponent** v rozsahu -127 až 128.

Uloženie je nasledujúce:

byte	1. byte	2. byte	3. byte	4. byte
exponent	mantisa			

Najvyšší bit exponenta, či je exponent záporný (bit je nastavený na 0), či kladný (bit je nastavený na 1). Podobne najvyšší bit v prvom byte mantisy určuje, či je číslo záporné (bit je nastavený na 1), alebo kladné (bit je nastavený na 0).

Uvedieme si príklad. Chceme zapísať číslo 1/10. Najskôr ho musíme previesť do tvaru $\pm m \times 2^e$. 1/10 je menšie než 1/2 a preto je rozšírime 4/4. Dostaneme $1/10 \times 4/4 = 4/40 = 4/5 \times 2^{-3}$.

Platí: $4/5 = 0.11001100110011001100110011001100_B$; $-3 = 11111101_B$.

Ako sme k tomuto číslu prišli? Číslo +3 má v dvojkovej sústave zápis 00000011_B. Zmeníme všetky 0 na 1 a všetky 1 na 0. Dostaneme binárne číslo 11111100_B. K tomuto číslu pripočítame číslo 1 (teda 00000001_B) a dostaneme:

```
  11111100B
+ 00000001B
  11111101B
```

A to je práve binárny zápis čísla -3. Tomuto spôsobu vytvorenia čísla -3 sa hovorí, že sme vytvorili **dvojkový doplnok** čísla +3. Malo by teda platiť: -3 + 3 = 0. Skúsme to:

```
  00000011B
+ 11111101B
 100000000B
```

Vo výsledku sa objavil ešte deviaty bit, ktorý označuje prenos do vyššieho rádu a hovoríme, že prišlo k **pretečeniu**. Pri práci s osembitovými číslami tento bit zanedbávame.

Binárny zápis 11111101_B predstavuje číslo -3_D. Pretože je ale mantisa kladná, zmeníme prvý bit mantisy na 0. Podobne zmeníme prvý bit exponentu na 0 (exponent je záporný). Dostávame teda binárne číslo:

$01111101\ 01001100\ 11001100\ 11001100\ 11001100_B$

čo je vyjadrenie 1/10 v semilogaritmickom tvare. Podobne môžeme postupovať pri prevodu akéhokoľvek čísla do semilogaritmického tvaru.

2.3. Mapa pamäti

Jedným z najdôležitejších obvodov, ktoré počítač obsahuje, jeho **pamäť**. V nej je uložený interpreter jazyka BASIC, ukladajú sa do nej informácie o zobrazení na obrazovke, o programe a vlastný program.

Existujú dva základné typy pamäti. Jedným je pamäť typu **ROM** (Read Only Memory), teda pamäť, z ktorej sa dá len čítať. Druhým je pamäť **RWM** (Read Write Memory), teda pamäť, z ktorej sa dá čítať, a dá sa do nej zapisovať. Tento typ pamäti je tiež označovaný ako **RAM**.

Pamäť si môžeme predstaviť ako rad buniek (bytov), nasledujúcich bezprostredne za sebou. V každom byte môže byť zapísané číslo od 0 do 255. Toto číslo je v byte zapísané v dvojkovej sústave.

Veľkosť pamäti sa určuje v bytoch alebo **kilobytoch** (kB). Každý kilobyte má $1024 = 2^{10}$ bytov. Váš počítač má 64 kB pamäti. Pokiaľ by sme chceli vyjadriť veľkosť pamäti v bytoch, musíme vynásobiť $64 \times 1024 = 65536 = 2^{16}$ bytov. Prvý byte pamäti má poradové číslo 0, posledný byte má poradové číslo 65535 (nemôže mať 65536, pretože medzi 0 a 65535 je práve 65536 bytov).

Celú pamäť môžeme rozdeliť na dve časti. Prvá je pamäť ROM s veľkosťou 16 kB. V nej je uložený **operačný systém** počítača, ktorý zabezpečuje interpretáciu jazyka BASIC.

Druhá časť pamäti s veľkosťou 48 kB je typu RAM. V tejto časti sú uložené všetky informácie nutné pre prácu počítača. Pamäť RAM môžeme ešte rozdeliť na dve časti a to na tzv. **video RAM** (v nej sú uložené obrazové informácie) a **pamäť programu a dát**.

Celá pamäť počítača má nasledujúcu štruktúru:

pamäť programu a dát 42240 bytov	← $23296_D = 5B00_H$
pamäť video RAM 6912 bytov	← $16384_D = 4000_H$
pamäť ROM 16384 bytov	← $0_D = 0000_H$

2.3.1. Pamäť video RAM

Samotná pamäť je rozdelená na dve časti, v podstate na sebe nezávislé. Prvá časť je vyhradená pre kresbu - pamäť kresby, druhá pre farebné atribúty - **pamäť kresby**, druhá pre farebné atribúty - **pamäť atribútov**.

2.3.1.1. Pamäť kresby

Pamäť kresby obsahuje každý zobraziteľný bod obrazovky a určuje, či je bod "rozsvietený" (v pamäti kresby je na mieste pre tento bod uložená jednotka), alebo "zhasnutý" (v pamäti kresby je uložená nula).

Skúste zadať `POKE 16384,255`. V ľavom hornom rohu obrazovky sa rozsvieti vodorovná čiarka v rozsahu ôsmich bodov (bitov). Skúste zadať do počítača nasledujúci program a zadajte RUN:

```
10 FOR I=0 TO 7
20 POKE 16384,2^I
30 PAUSE 25
40 NEXT I
```

Tento program postupne ukladá na prvý byte pamäti kresby čísla 1, 2, 4, 8, 16, 32, 64, 128 a rozsvietený bod sa postupne presúva sprava doľava.

Pamäť pre kresbu má rozsah 6144 bytov. Toto číslo dostanete pomocou vzorca:

$$\text{pamäť kresby} = \frac{\text{počet\_mikroriadkov} \times \text{počet\_bodov\_na\_riadok}}{8}$$

pamäť kresby = (192 × 256) / 8 = 6144 bytov

Pamäť pre kresbu má v dôsledku použitého technického spôsobu zobrazovania trochu zvláštnu štruktúru, odlišné od štruktúry pamäti programu a dát. Pokúsime sa ju vysvetliť najskôr na príkladoch. Napíšte do počítača tento program:

```
10 FOR I=0 TO 31
20 POKE (16384+I),255
30 NEXT I
```

Odštartujte ho pomocou RUN. Na nultom riadku obrazovky sa postupne nakreslí čiara. Týmto programom ste uložili do pamäti kresby od adresy 16384 po adresu 16415 samé jednotky (celkom 32 bytov po ôsmich bitoch, teda 256 bitov). Týchto 256 bitov predstavuje nultý mikroriadok obrazovky. Dalo by sa očakávať, že ďalší mikroriadok obrazovky začína v pamäti kresby na adrese 16416, teda ihneď za prvým. To však nie je pravda. Prvý mikroriadok obrazovky začína v pamäti kresby až na adrese o 256 väčšej, než je adresa nultého mikroriadku, teda na adrese 16640. Druhý mikroriadok obrazovky začína v pamäti na adrese o 256 väčšej, než je adresa začiatku prvého mikroriadku, teda na adrese 16896 atď.

Napíšte tento program (poznámky REM sú len pre vašu orientáciu, nemusíte ich do programu písať):

```
10 FOR I=0 TO 7: REM POCET MIKRORIADKOV=8
20 FOR J=0 TO 31: REM POCET BYTOV V RIADKU=32
30 POKE (16384+i*256+j),255: REM ZACIATOK MIKRORIADKU JE POSUNUTY O 256
   BYTOV
40 NEXT J
50 NEXT I
```

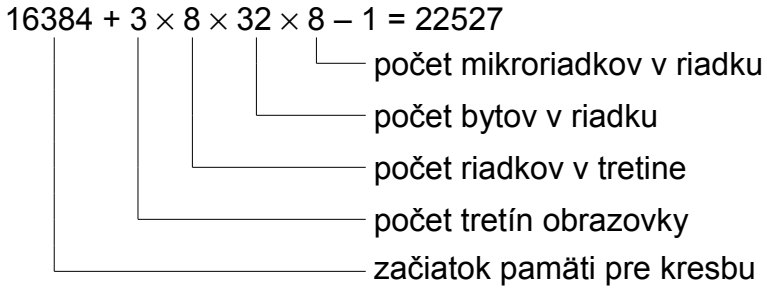
Po vykonaní programu v prvých ôsmich mikroriadkoch obrazovky nakreslí čiara. Iste si teraz položíte otázku: "Pokiaľ je prvý mikroriadok obrazovky posunutý oproti nultému o 256 bytov, tak čo je uložené na adrese za koncom nultého mikroriadku, teda na adrese 16384+32=16416?". Napíšte nasledujúci program:

```
10 FOR I=0 TO 31
20 POKE (16384+I),255
30 NEXT I: REM NA OBRAZOVKE SA VYKRESLILA V NULTOM MIKRORIADKU CIARA
40 PAUSE 50
50 FOR I=32 TO 63
60 POKE (16384+I),255
70 NEXT I: REM NA OBRAZOVKE SA VYKRESLI V OSMOM MIKRORIADKU CARA
```

Podobne platí, že na adrese 16384 + 32 + 32 = 16448 je uložených 0 + 8 + 8 = 16 mikroriadkov atď. Napíšte do počítača tento program:

```
10 FOR I=0 TO 32*8*8-1: REM 32 BYTOV NA MIKRORIADOK 8 MIKRORIADKOV=1 RIADOK
   OBRAZOVKY A ZAPLNIME 8 RIADKOV
20 POKE (16384+I),255
30 NEXT I
40 PAUSE 50
```

Uvedený program zaplní jednu tretinu obrazovky čiarami, výsledok je "začiernená" jedna tretina obrazovky. Prečo len jedna tretina? Aby toho nebolo málo, obrazovka sa ešte delí na tretiny. Prvá tretina začína na adrese 16384, druhá na adrese 18432 a tretia na adrese 20489. Posledný byte pamäti kresby je na adrese 22527. Táto adresa je vypočítaná nasledovne:



Spôsob zápisu kresby sa môže zdať trocha zložitejší. Z dôvodu jednoduchej konštrukcie počítača bola obrazovka rozdelená na tretiny. Každá tretina má 8 riadkov, každý riadok má 8 liniek. Toto rozdelenie si všimnite pri nahrávaní úvodného obrázku k nejakej hre. Najprv sa nahrávajú jednotlivé linky prvej tretiny, potom druhej a tretej. Nakoniec sa nahrávajú atribúty, ktoré výslednú kresbu zafarbia. Na určenie adresy bodu na obrazovke platí nasledujúca schéma:

bit	15	14	13	12	12	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	A		B			C			D				

$010\ 000000000000_B$ - binárne vyjadrenie adresy 16384_D

010 - pri adresovaní obrazovky sa prvé tri bity nemenia.

A - určenie tretiny počítané zhora. Pre A=4 ide o atribúty.

B - určenie čísla linky v jednom riadku (0=7)

C - určenie čísla riadku vo vybranej tretine (0-7).

D - určenie čísla stĺpca (0-31).

Pre všetky tretiny platia rovnaké pravidlá ukladania mikroriadkov obrazovky do pamäti pre kresbu. Môžete si to vyskúšať tak, že namiesto začiatku prvej tretiny (16384) dosadíte do predchádzajúcich programov začiatky druhej (18432) a tretej tretiny (20480). Napíšte do počítača posledný program tejto časti:

```
10 FOR I=0 TO 3*8*32*8-1
20 POKE (16384+I),255
30 NEXT I
40 PAUSE 100
```

Tento program postupne zaplní celú obrazovku (a teda i pamäť pre kresbu). Všimnite si, ako sa obrazovka zaplňuje. Najskôr sa zaplní prvá tretina, potom druhá a tretia.

Tým sme skončili popis pamäti pre kresbu a budeme sa venovať pamäti atribútov (pamäti pre farby).

2.3.1.2. Pamäť atribútov

Pamäť atribútov je druhá časť pamäti video **RAM**. V nej sú obsiahnuté všetky informácie o farbách na obrazovke, o blikaní a zvýšenom jase.

Pamäť atribútov začína na adrese 22528 a je dlhá 768 bytov, teda prvý voľný byte za touto oblasťou je na adrese 23296. Dĺžka vyplýva z toho, že pre celú obrazovku, ktorá má 24 riadkov a 32 stĺpcov, potrebujeme $24 \times 32 = 768$ bytov na atribúty a každý byte prináleží práve jednému štvorčeku obrazovky s rozmermi 8×8 bodov.

Skúste zadať POKE 22528,0. Štvorček v ľavom hornom rohu sa zafarbí na čierno.

Zadajte:

```
FOR I=0 TO 31: POKE (22528+I),0: NEXT I
```

a odošlite riadok. Nultý riadok obrazovky sa zafarbí na čierno. Atribúty pre jeden riadok sú teda v pamäti uložené za sebou.

Ale čo ďalší riadok? Sú atribúty druhého riadku uložené bezprostredne za atribútami nultého riadku? Napíšte program:

```
10 FOR I=0 TO 31
20 POKE (22528+I),0
30 NEXT I: REM NULTY RIADOK SA ZAFARBI NA CIERNO
40 PAUSE 50
50 FOR I=32 TO 63
60 POKE (22528+I),0
70 NEXT I:REM PRVY RIADOK SA ZAFARBI NA CIERNO
```

Z uvedeného príkladu vyplýva, že atribúty pre jednotlivé riadky sú v pamäti uložené za sebou v tom istom poradí, ako riadky na obrazovke. Napíšte ešte tento program:

```
10 FOR I=0 TO 767
20 POKE (22528+I),0
30 NEXT I
```

Po spustení sa postupne zafarbí celá obrazovka na čierno. Zresetujte počítač.

Vieme teda, ako sú atribúty pre riadky v pamäti uložené. Zatiaľ sme na ich miesto v pamäti ukladali len hodnotu nula príslušný štvorček obrazovky sa zafarbil na čierno. Ako ale dosiahnuť iné farby, popr. blikanie a zvýšený jas? Do atribútových bytov je treba túto informáciu podľa určitých pravidiel uložiť. Pravidlá vyplývajú zo štruktúry **atribútového bytu** (ATB):

FLASH	BRIGHT	**PAPER**			**INK**		
7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit

V počítači je každá farba vyjadrená určitým číslom podľa nasledujúcej tabuľky:

farba	**kód dekadicky**	**kód binárne**
čierna	0	000
modrá	1	001
červená	2	010
fialová	3	011
zelená	4	100
bledomodrá	5	101
žltá	6	110
biela	7	111

Z uvedenej tabuľky vyplýva, že ľubovoľná farba sa dá získať skladaním troch základných farieb, a to modrej, červenej a zelenej. Tak napr. žltá farba (kód 6) vznikne zložením červenej (kód 2) a zelenej (kód 4) farby.

Predpokladajme, že chceme mať na obrazovke modrý papier (modrá farba má hodnotu 1) so žltým atramentom (žltá farba má hodnotu 6), neblikajúci, bez zvýšeného jasu. Do pamäti atribútov musíme uložiť hodnotu: $00001110_B = 14$.

Na zelený papier, čierny atrament a zvýšený jas uložíme do pamäti atribútov hodnotu: $01010000_B = 96$.

Obecne môžeme atribútový byte definovať nasledovne:

```
ATB = FLASH×128 + BRIGHT×64 + PAPER×8 + INK.
```

V našich dvoch predchádzajúcich príkladoch majú atribútové byty túto hodnotu:

- `ATB = 0×128 + 0×64 + 1×8 + 6 = 14`
- `ATB = 0×128 + 1×64 + 4×8 + 0 = 96`

Takto si môžete vypočítať a dosadiť do pamäti pre atribúty ľubovoľnú hodnotu a meniť tak zafarbenie obrazovky. Tento spôsob priameho dosadenia atribútov do pamäti sa používa predovšetkým v programoch v strojovom kóde, pretože je to najrýchlejšia cesta, ako zmeniť zafarbenie celej, popr. časti obrazovky. Polohu atribútu na obrazovke možno vypočítať podľa vzťahu:

ATB = 22528 + 32 × R + S
ATB - vypočítaná adresa atribútu
22528 - adresa prvého atribútu na obrazovke
R - poradové číslo riadku (0 - 23)
S - poradové číslo stĺpca (0 - 31)

Po prepísaní adresy do binárneho tvaru je všetko zrejmé:

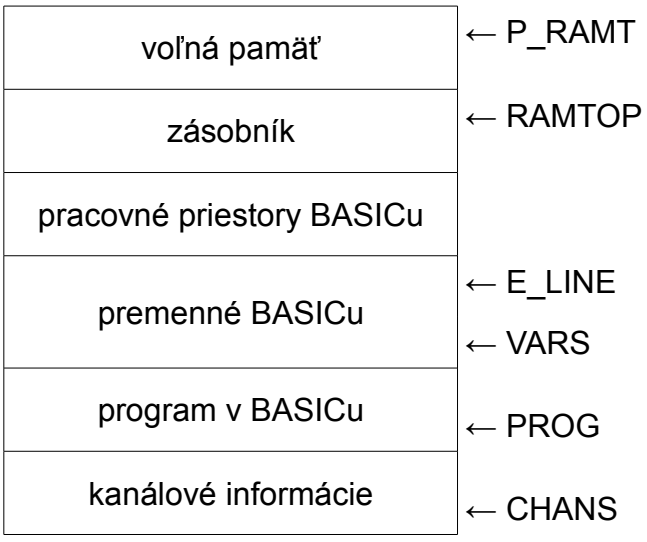
bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hodnota	0	1	0	1	1	0	R	R	R	R	R	S	S	S	S	S
adresa	22528						riadok					stĺpec				

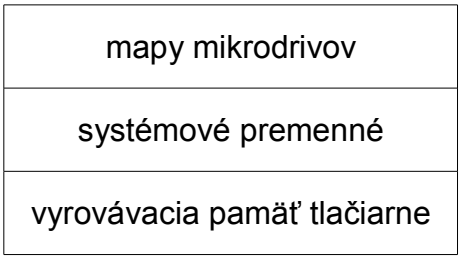
2.3.2. Pamäť programu a dát

Pamäť programu a dát sa rozkladá od adresy 23296 po adresu 65535. V jej spodnej časti sú uložené dôležité informácie pre prácu systému, za nimi je program v jazyku BASIC a ďalšie informácie pre systém. Pamäť môžeme rozdeliť do niekoľkých skupín:

- vyrovnávacia pamäť tlačiarne
- systémové premenné
- mapy mikrodrivov
- kanálové informácie
- program v BASICu
- premenné v BASICu
- pracovné priestory BASICu
- zásobník
- užívateľská grafika
- voľná pamäť

Obsadenie pamäti môžeme znázorniť nasledovne:





Každej časti pamäti sa v tejto kapitole budeme venovať zvlášť, len užívateľskú grafiku popíšeme v kapitole samostatne. V diagrame nie je zakreslená, pretože jej umiestnené v pamäti môže byť ľubovoľné. Názvy hraníc medzi oblasťami sú vysvetlené v prílohe D. Teraz len krátko:

CHANS - adresa začiatku kanálových informácií

PROG - adresa začiatku programu v BASICu

VARS - adresa začiatku premenných programu

E_LINE - adresa konca premenných programu v BASICu a začiatku pracovnej oblasti

RAMTOP - adresa konca pamäti vyhradené pre BASIC

P_RAMT - adresa fyzického konca pamäti (ukazuje na poslednú adresu, z ktorej sa dá bez chyby čítať a zapisovať, väčšinou na adresu 65535)

2.3.2.1. Vyrovnávacia pamäť tlačiarne

Vyrovnávacia pamäť tlačiarne je uložená od adresy 23296 a dlhá je 256 bytov. Používa sa pri tlači na ZX Printer (dvojihličková tlačiareň so špeciálnym papierom, ktorá sa vyrábala v zahraničí pre počítače ZX Spectrum). V prípade tlače na ZX Printer sa do nej ukladá tlačený riadok, ale rozložený na mikroriadky (podobne ako na obrazovke). Dá sa využívať na uloženie vlastných programov v strojovom kóde.

Pri použití príkazu COPY dôjde k prepísaniu tejto oblasti, preto pozor na jeho používanie, pretože prípadný strojový kód v buffere je po zadaní príkazu COPY zrušený.

2.3.2.2. Systémové premenné

Systémové premenné sú uložené od adresy 23552 a zaberajú oblasť 182 bytov. Slúžia na riadenie práce systému. Ich rozloženie v pamäti a presný význam je uvedený v prílohe D, ktorú vám doporučujeme pozorne preštudovať.

Do tejto oblasti sa nepokúšajte zadávať hodnoty pomocou POKE, pokiaľ presne neviete, čo zadanie novej hodnoty spôsobí. Zadanie nesprávnej hodnoty obvykle skončí stlačením tlačidla RESET, popr. počítač túto činnosť vykoná za Vás.

2.3.2.3. Mapy mikrodrivov

Táto oblasť má začínať na adrese 23734 a má nenulovú dĺžku len v prípade, že je mikrodrive fyzicky pripojený a zapnutý. Viac sa s mikrodrivom nebudeme zaoberať, pretože je to neperspektívne zariadenie a v zahraničí sa už dlhšiu dobu nevyrába.

2.3.2.4. Kanálové informácie

V oblasti kanálových informácií je uložená tabuľka, obsahujúca **adresy vstupných a výstupných podprogramov** (rutín) v strojovom kóde, predstavujúcich obsluhu jednotlivých **kanálov** (typov periférií). Tabuľka je po zapnutí počítača uložená od adresy 23734 a má dĺžku 20 bytov. V prípade použitia mikrodrivov sa jej začiatok úmerne posúva k vyšším adresám. Jej dĺžka je tiež premenná v závislosti od počtu kanálov, ktorých môže byť až 16. Začiatok tabuľky je uložený v systémovej premennej CHANS. Tabuľka má nasledujúci tvar, začiatok tabuľky je od adresy uloženej v CHANS:

kanál 3	typ tlačiareň (Printer)	P	← PROG
	adresa vstupnej rutiny	$15C4_H$	
	adresa výstupnej rutiny	$09F4_H$	
kanál 2	typ BASIC editor (woRkspace)	R	
	adresa vstupnej rutiny	$15C4_H$	
	adresa výstupnej rutiny	$0F81_H$	
kanál 1	typ obrazovka (Screen)	S	
	adresa vstupnej rutiny	$15C4_H$	
	adresa výstupnej rutiny	$09F4_H$	
kanál 0	typ Klávesnica	K	
	adresa vstupnej rutiny	$10A8_H$	
	adresa výstupnej rutiny	$09F4_H$	← CHANS

V tejto tabuľke je možné meniť adresy výstupných a vstupných rutín a tak použiť svoje vlastné rutiny napr. pre vstup z klávesnice alebo pre výstup na obrazovku a tlačiareň. Viac sa o vstupno/výstupných kanálov dozviete v kapitole 2.7.

2.3.2.5. Program v BASICu

Adresa začiatku programu v BASICu je uložená v systémovej premennej PROG a po zapnutí počítača je nastavený na hodnotu 23755 teda bezprostredne za tabuľku kanálových informácií. So zväčšením tabuľky kanálových informácií, popr. pri použití mikrodrivov sa začiatok programu posúva k vyšším adresám. Koniec oblasti je označený bytom s hodnotou 80_H. Priestor pre program vyzerá takto:

	← VARS
80_H	označenie konca programu
program v BASICu	← PROG

Každý riadok programu je v pamäti uložený nasledovne:

číslo riadku	dĺžka riadku	riadok programu	koncový znak
2 byty	2 byty	x bytov	1 byte = $0D_H$

Číslo riadku je uložené tak ako je napísané v programe. Teda riadok s číslom 10 má v prvom byte uloženú nulu a v druhom byte uložené číslo 10. Dĺžku riadku vypočíta po odoslaní riadku operačný systém a dosadí ju na vyhradené miesto za číslo riadku. Dĺžka riadku je počítaná mimo dva byty, určené pre číslo riadku. Dĺžka riadku je počítaná mimo dva byty, určené pre číslo riadku (obecne platí: dĺžka riadku (v bytoch) = 2 + X + 1).

Dĺžka je už uložená v pamäti štandardne, teda najskôr nižší byte a za ním vyšší byte. Potom nasleduje vlastný riadok programu, ktorý je zakončený znakom CR (Carriage Return - návrat vozíka) s kódom $0D_H = 13_D$. Za posledným riadkom programu (a teda i za posledným znakom CR) je ešte byte, ktorý obsahuje hodnotu $80_H = 128_D$ a označuje koniec programu v pamäti.

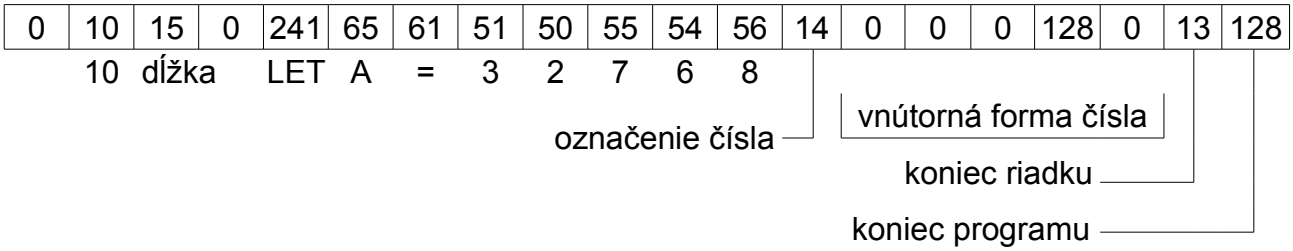
V ďalšom texte budeme používať značku obdĺžnika, ktorý nie je uzatvorený aspoň z jednej strany na znázornenie jedného bitu.

 = jeden byte = dva byty

Riadok programu je v pamäti uložený v trochu inej forme, než ako ho zadáte na obrazovke. Predpokladajme, že ste zadali počítaču len jeden riadok a to:

```
10 LET A=32768
```

Tento riadok sa uloží do pamäti nasledovne (všetky čísla sú v desiatkovej sústave):



Uloženie riadku v pamäti začína číslom riadku, za ktorým nasleduje dĺžka riadku. Všetky kľúčové slová sú zakódované do jedného bytu, tzv. **tokenu** (priradenie kódu kľúčovým slovám (tokenom) je uvedené v prílohe C) a ostatné znaky v riadku sú uložené v **kóde ASCII** (viď príloha C). Za každým číslom v riadku nasleduje byte s hodnotou 14, ktorý označuje, že nasleduje zápis tohto čísla vo vnútornej forme v dĺžke piatich bytov. Koniec riadu je označený znakom CR.

Z uvedeného vyplýva, že za každým číslom, použitým v programe, nasleduje byte s hodnotou 14, ktorý označuje, že nasleduje zápis tohto čísla vo vnútornej forme v dĺžke piatich bytov. Koniec riadku je označený znakom CR.

Z uvedeného vyplýva, že za každým číslom, použitým v programe, nasleduje byte s hodnotou 14, označujúci výskyt čísla, a za ním päť bytov, na ktorých je uložené číslo vo vnútornej forme. Pri vykonávaní programu už BASIC pracuje len s číslami vo vnútornej forme.

Tento spôsob ukladania čísiel má jednu veľkú nevýhodu a tou je nepríjemné narastanie dĺžky programu pri používaní väčšieho množstva čísiel. Proti tomu sa však možno brániť. Stačí len ako päť bytov. Niekoľko výrazov vám ponúkame, na iné iste postupom času prídete sami:

miesto	použite	ušetríte
0	`NOT PI`	5 bytov
1	`SGN PI`	5 bytov
3	`INT PI`	5 bytov
32	`CODE " "`	4 byty
255	`CODE "COPY"`	5 bytov

Namiesto akéhokoľvek čísla môžete použiť výraz VAL"cislo"

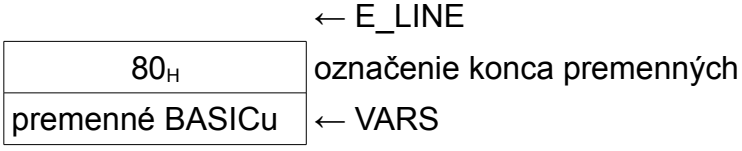
a tým ušetríte 3 byty. Ďalšia možnosť je používanie konštánt namiesto niektorých často sa vyskytujúcich čísiel vo vašom programe. Tak napr. do prvého riadku programu dáte príkaz:

```
LET K10=10
```

a všade v programe budete používať namiesto čísla 10 konštantu K10 a tým ušetríte pri každom jej použití 5 bytov. Nevýhoda použitia konštánt a výrazov je spomalenie vykonávanie programu.

2.3.2.6. Premenné BASICu

Premenné, použité v programe, sú uložené ihneď za posledným riadkom programu. Ich začiatok je uložený v systémovej premennej VARS a koniec označený bytom s hodnotou 80_H. Priestor pre premenné vyzerá takto:



Začiatok premenných nie je vždy na tom istom mieste. V prípade, že do programu pridáte alebo vypustíte riadok, zmení sa i dĺžka programu a začiatok premenných sa posunie.

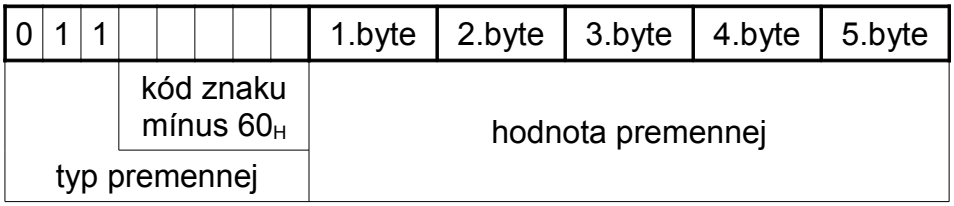
Veľkosť tejto oblasti je tiež premenná. Po príkaze RUN sa oblasť vyčistí. Pri vykonávaní programu sa každá nová premenná zaradí do oblasti premenných. Za menom premennej je potom uložená jej aktuálna hodnota a všetky aritmetické a logické operácie sú vykonávané s touto aktuálnou hodnotou.

Pri vykonávaní programu a priradení hodnoty premennej sa postupuje nasledovne:

- v programe sa zistí meno premennej
- vezme sa zo systémovej premennej VARS začiatok oblasti premenných a od tejto adresy sa hľadá meno premennej
- v prípade, že sa meno v oblasti nachádza, zistí sa z programu hodnota, ktorá má byt v premennej uložená a uloží sa do nej
- v prípade, že sa premenná v oblasti nenachádza, tak sa pre ňu vyhradí miesto, zapíše sa naň meno premennej a hodnota, ktorá sa zistí z programu.

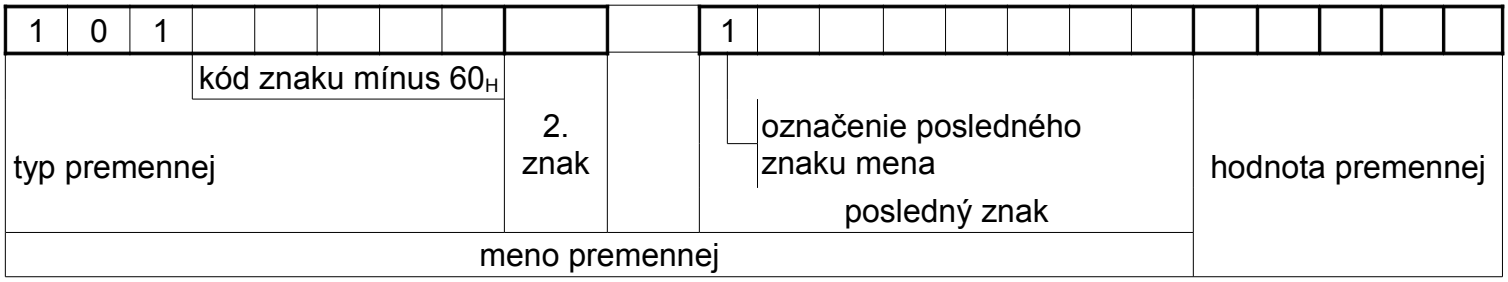
Teraz uvedieme spôsob uloženia mien a hodnôt niektorých typov premenných v oblasti premenných.

Číselná premenná (jednoznaková):

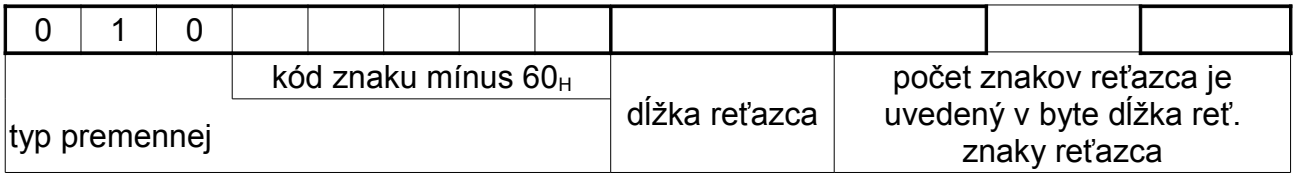


Na miesto označené "kód znaku minus 60_H" sa uloží kód znaku (písmena), ktorým je premenná označená, od ktorého sa odčíta hodnota 60_H. Napr. pre premennú, označenú písmenom "i" sa na dané miesto uloží: CODE("i") - 60_H = 69_H - 60_H = 09_H.

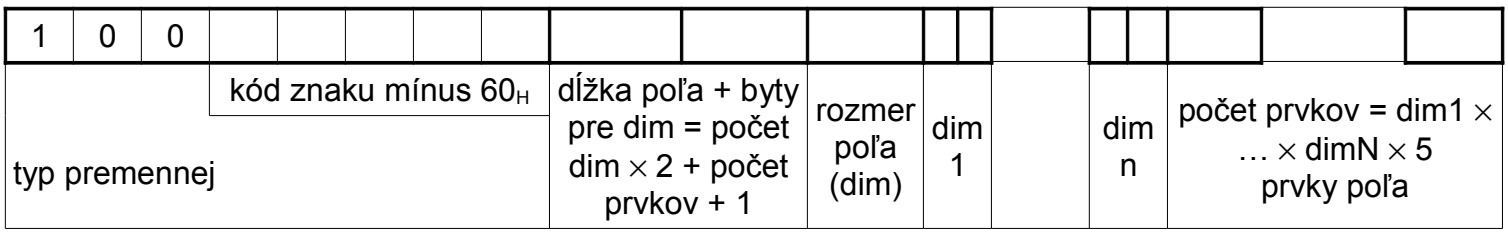
Číselná premenná (viacznaková):



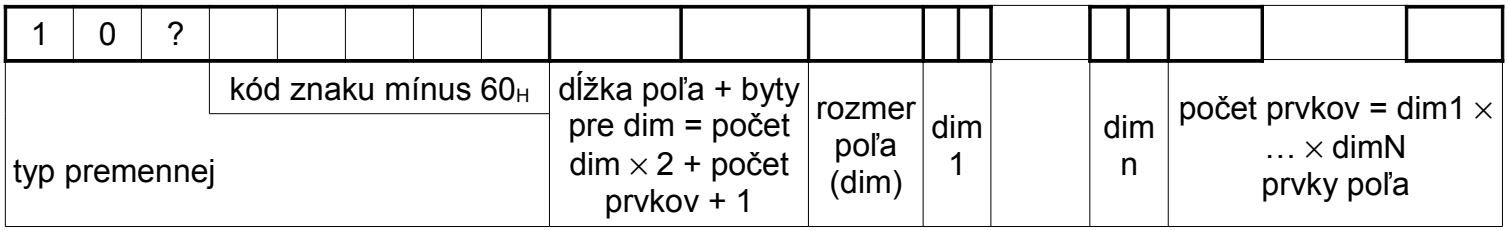
Reťazec:



Číselné pole (n-rozmerné, kde n = 1 až 255):



Znakové pole (n-rozmerné, kde n = 1 až 255):



Premenná cyklu FOR .. NEXT:

1	1	1																					
			kód znaku mínus 60_H					koncová hodnota					začiatoč. hodnota					krok cyklu				riadok NEXT		príkaz NEXT
typ cyklus								parametre cyklu														koniec cyklu		

Byty označené ako "riadok NEXT" a "príkaz NEXT" určujú umiestnenie príkazu NEXT v programe (teda na ktorom riadku sa nachádza a aké má poradové číslo v riadku). Pri vykonávaní programu sa potom pokračuje (po vykonaní cyklu) príkazom nasledujúcim za príkazom NEXT.

2.3.2.7. Pracovné priestory BASICu

Pre svoju prácu používa BASIC ešte ďalšie časti pamäti RAM, umiestnenej nad koncom premenných. V tejto časti sa vykonávajú všetky výpočty **kalkulátora**, je tu **zásobník kalkulátora** a prenáša sa sem práve vykonávaný riadok programu. Rozloženie oblasti pre vyššie uvedené činnosti môžeme znázorniť nasledovne:

	← SP (Stack Pointer)
voľná oblasť	← STKEND
zásobník kalkulátora	← STKBOT
pracovný priestor	← WORKSP
80_H	koniec editačnej oblasti
editačná oblasť	← E_LINE

Ani jedna z týchto troch oblastí nemá svoje pevné miesto v pamäti. Ich začiatky a konce sa neustále pri vykonávaní programu presúvajú podľa toho, ako veľkú ktorú oblasť počítač potrebuje. Z tohto dôvodu sú ich začiatky a konce uložené v systémových premenných E_LINE (adresa začiatku editačnej oblasti), WORKSP (adresa začiatku pracovného priestoru), STKBOT (adresa dna zásobníka kalkulátoru) a STKEND (adresa vrchu zásobníka pre kalkulátor). Vrch **zásobníka procesora Z-80** je označený SP (Stack Pointer), viac sa o ňom dozviete v časti venovanej iba jemu.

2.3.2.8. Editačná oblasť

Táto časť pamäti slúži na editáciu riadkov a ich nasledovnému vykonávaniu. Znaky, ktoré sú zadávané z klávesnice sú vyhodnotené systémom. Znaky a tokeny sú uložené do editačnej oblasti v poradí, ako boli stlačené. Editačné klávesy sú vyhodnotené zvlášť a potom je v editačnej oblasti vykonaná činnosť, ktorú označujú (zrušenie znaku, posun po riadku doľava, doprava apod.). Po stlačení každého klávesu je riadok v editačnej oblasti zobrazený na obrazovku do editačnej zóny (spodná časť obrazovky). Tam až vidíte, čo ste napísali.

Po stlačení klávesu ENTER prebehne **syntaktická kontrola** riadku. V prípade, že je syntax správna, riadok sa z editačnej oblasti zaradí do programu (pokiaľ má číslo) alebo sa vykoná.

Zaradenie riadku do programu vykoná systém veľmi jednoducho. Zistí si, či je voľné miesto pre riadok v oblasti pre program. Pokiaľ nie, oznámi vám to "zatrúbením". V tomto prípade neostáva nič iné, než riadok stlačením EDIT vymazať z editačnej oblasti a urobiť preň miesto pomocou príkazu CLEAR (vysvetlíme v ďalšej kapitole) alebo program skrátiť výmenou čísiel za niektoré výrazy ako sme uviedli v kapitole 2.3.2.5.

V prípade, že voľné miesto je, systém zistí, či sa v programe riadok s rovnakým číslom už nevyskytuje. Pokiaľ nie, tak je nájdený riadok programu s číslom už nevyskytuje. Pokiaľ nie, tak je nájdený riadok programu s číslom najbližším vyšším, než je číslo nového riadku. Oblasť, začínajúca týmto "vyšším" riadkom sa až po koniec programu presunie o toľko bytov smerom ku koncu pamäti, koľko predstavuje dĺžka nového riadku. Na takto vzniknuté miesto sa prenesie z editačnej oblasti nový riadok.

Pokiaľ sa už v programe vyskytuje riadok s rovnakým číslom ako má nový zadaný riadok, starý riadok sa zruší takto: oblasť nového riadku až po koniec programu sa prenesie o toľko bytov smerom k začiatku pamäti, koľko predstavuje dĺžka starého riadku. Tým prestal riadok s rovnakým číslom ako má nový riadok v programe existovať a nový riadok sa do programu zaradí.

Pri vykonaní riadku (bez čísla) vykonáva systém jednotlivé príkazy uložené v editačnej oblasti, pokiaľ nenarazí na koncový byte 80_H. Potom riadok z editačnej oblasti vymaže púhym presunom bytu 80_H na začiatok editačnej oblasti (teda tam, kam ukazuje systémová premenná E_LINE).

Pri vykonaní programu (príkazom RUN, GO TO) sa postupne prenášajú riadky programu do editačnej oblasti (už bez čísiel) a tam sú vykonané. Vo všetkých prípadoch práce s editačnou oblasťou sa menia systémové premenné začínajúce VARS a STKEND.

2.3.2.9. Pracovný priestor, zásobník kalkulátora

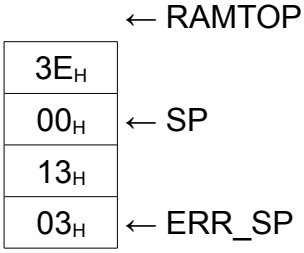
Pracovný priestor a zásobník kalkulátora sú využívané pri práci **kalkulátora**, t. j. tej časti operačného systému, ktorá má na starosti všetky aritmetické a logické operácie a prácu s reťazcami.

Umiestnenie obidvoch oblastí sa počas vkladanie programu do počítača a pri vykonávaní programu mení. Voľná oblasť medzi dnom zásobníka kalkulátora a vrchom zásobníka procesora Z-80 je vyhradená práve pre tieto posuny (pracovná oblasť a zásobník kalkulátora sa pri zväčšovaní dĺžky programu posunujú smerom k vyšším adresám).

Pri vykonávaní príkazov je tiež testovaná veľkosť voľnej pamäti medzi dnom zásobníka kalkulátora a vrchom zásobníka procesora Z-80.

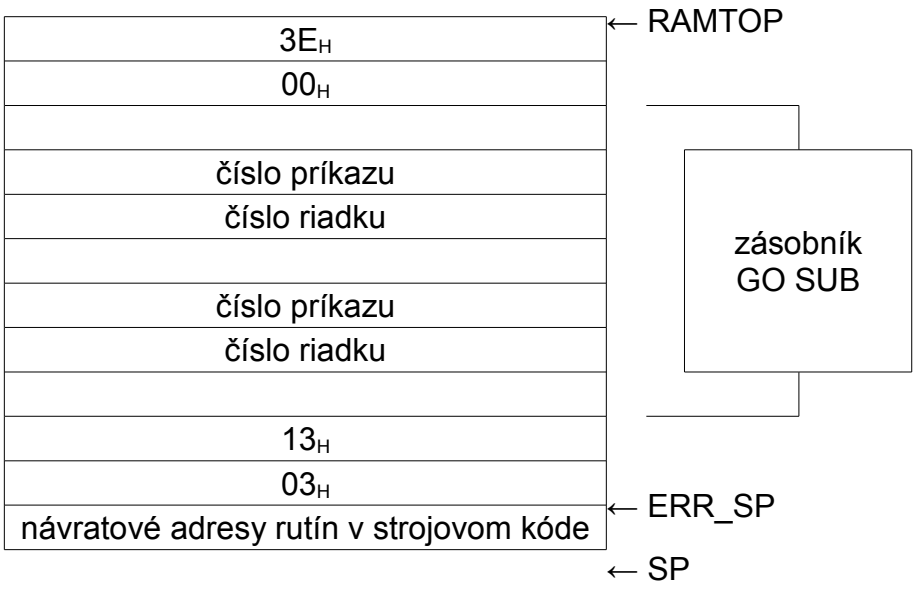
2.3.2.10. Zásobník

Pre prácu celého systému je veľmi dôležitá tá časť pamäti, kde sú uložené tzv. **návratové adresy** volania **podprogramov**. Nazývame ju zásobník. Návratové adresy sa vytvárajú automaticky ako pri volaní podprogramov v strojovom kóde procesora Z-80, tak pri volaní podprogramov pomocou GO SUB v BASICu. Štruktúra zásobníka po resete počítača je nasledujúca:



Systémová premenná ERR_SP ukazuje na položku zásobníka, na ktorej je uložená adresa obsluhy ľubovoľnej chyby, SP je ukazovateľ vrchu zásobníka a RAMTOP ukazovateľ konca pamäti pre BASIC. Koniec zásobníka (jeho dno) je označený bytom obsahujúcim hodnotu $3E_H$. Tento byte je tiež posledným bytom využiteľnej pamäti pre BASIC. Pri vkladaní hodnôt do zásobníka sa hodnota SP zmenšuje (zásobník sa zväčšuje smerom k začiatku pamäti RAM), pri vyberaní hodnôt zo zásobníku sa hodnota SP zväčšuje (zásobník sa zmenšuje smerom ku koncu pamäti RAM).

Pri výskyte nejakej chyby, ktorá vyvolá výpis chybového hlásenie, vyzdvihne systém obsah dvoch bytov adresovaných premennou ERR_SP a na túto adresu predá riadenie. Zmenou obsahu tejto položky zásobníka je možné dosiahnuť napr. reset počítača po stlačení BREAK pri nahrávaní programu, tak ako to poznáme u niektorých hier. Stačí len na tieto dva byty uložiť nuly. Pri vykonávaní riadku v editačnej oblasti zásobník vyzerá nasledovne:



Adresa obsluhy chyby je nastavená na hodnotu 1303_H. Na tejto adrese je podprogram obsluhujúci všetky možné chyby a vypisujúci v spodnej časti obrazovky chybové hlásenie.

Ďalej sa v zásobníku objavila položka **zásobník GO SUB**. Tá slúži na určenie miesta návratu v programe pri použití príkazu GO SUB, teda pri volaní podprogramu v BASICu. Na zásobník sa uloží číslo aktuálneho riadku (dva byty) a číslo príkazu (jeden byte) nasledujúceho za príkazom GO SUB, teda miesto v programe, kde sa dá po vykonaní podprogramu (po príkaze RETURN) pokračovať. Zásobník GO SUB môže obsahovať viacej ukazovateľov návratových miest v programe, pričom táto štruktúra zodpovedá vnorenému volanie podprogram GO SUB. V prípade, že nebol volaný žiadny podprogram GO SUB, zásobník GO SUB je prázdny.

Ukazovateľ vrchu zásobníka môže pri vykonávaní programu meniť svoju hodnotu podľa toho, aké podprogramy v strojovom kóde systém práve využíva a tiež v prípade použitia užívateľských podprogramov v strojovom kóde (podprogramy v strojovom kóde budeme v ďalšom texte nazývať rutiny). Po skončení týchto rutín však musí byť ukazovateľ nastavený na to isté miesto, aké mal pred ich vyvolaním.

Systémová premenná RAMTOP označuje koniec pamäti pre BASIC. Jej hodnota sa dá meniť príkazom:

```
CLEAR číslo
```

kde číslo je nová hodnota RAMTOP (konca pamäti). Zmenou RAMTOPu môžete pamäť pre BASIC zmenšovať alebo zväčšovať, v závislosti na dĺžke vášho programu. Po resete počítača je RAMTOP nastavený na hodnotu 65367_D.

Zmena RAMTOPu môže niekedy priniesť problémy. skúste zadať počítaču CLEAR 23850. Teraz skúste do editačného riadku niečo napísať. Odpoveďou vám bude len zatrúbenie editora, ktorý vám takto signalizuje, že niečo nie je v poriadku. Zadaním hodnoty 23850 ste totiž vymedzili oblasť pre BASIC na púhych 95 bytov (za predpokladu, že program v BASICu začína na adrese 23755). Do tejto oblasti musí počítač umiestniť všetky vyššie spomenuté pracovné oblasti a neostáva mu miesto na editačnú oblasť, teda do nej nemôže umiestniť vami zadávaný kláves. Nestáva nič iné než počítač zresetovať.

Pokiaľ sa vám stane táto vec pod odoslaní nejakého riadku programu, musíte ho z programu vypustiť, zmeniť RAMTOP, popr. skrátiť program a riadok do neho znovu zadať.

Príkaz **CLEAR** vykonáva tieto činnosti:

- vymaže všetky premenné z oblasti premenných
- vymaže obrazovku
- v systémových premenných nastaví súradnica posledného kresleného bodu na hodnoty 0,0
- vykoná RESTORE
- vymaže zásobník GO SUB.
- v prípade použitia tvaru CLEAR číslo nastaví RAMTOP (pokiaľ je to možné) na hodnotu číslo. Nová hodnota RAMTOPu musí ležať medzi STKEND a P_RAMT.

2.3.2.11. Voľná pamäť

Na adresu o jedna väčšej, než je hodnota RAMTOP, začína voľne použiteľná pamäť, ktorá je zhora ohraničená adresou uloženou v systémovej premennej P_RAMT.

Túto časť pamäti BASIC nepoužíva, je však prístupná cez príkaz POKE, PEEK a môžete do nej nahrávať súbory uložené na páske príkazom SAVE "meno" CODE X,Y. Má jednu veľkú výhodu; jej obsah nemaže príkaz NEW (ten maže pamäť len po RAMTOP). Väčšinou sa používa na uloženie programov v strojovom kóde a na uloženie užívateľskej grafiky.

2.3.2.12. Umiestnenie užívateľskej grafiky

Počítač disponuje okrem **základnej sady znakov**, uloženej v pamäti ROM, ešte ďalšou sadou znakov označovaných ako **užívateľská grafika** (User Define Graphic - **udg**). Táto sada môže byť umiestnená kdekoľvek v pamäti RAM počítača a obsahuje 21 znakov, ktorých tvar si môže užívateľ sám definovať. Adresa začiatku udg je umiestnená v systémovej premennej UDG, dĺžka je 168 bytov.

Po resete počítača je premenná UDG nastavená na hodnotu 65368, teda na adresu prvého bytu za RAMTOPom. Pokiaľ chcete vo svojich programoch používať užívateľskú grafiku, doporučujeme umiestniť ju vždy až za RAMTOP, aby nebola premazaná BASICom, popr. príkazom NEW. Viacej sa o tvorbe znakov užívateľskej grafiky dozviete až po zoznámení sa s generátorom znakov umiestneným v pamäti ROM.

Týmto sme skončili popis pamäti počítača.

2.4. Znakový generátor

Pri výpise znaku musí byť do pamäti kresby umiestnená na miesto, kde má byť znak vypísaný, **matica znaku** obsahujúca rozkreslenie znaku na jednotlivé body. Matica znaku má rozmery 8x8 bitov a v každom bite je umiestnená informácia, či daný bod (priradený bitu) má byť rozsvietený (bit obsahuje hodnotu 1) alebo (bit obsahuje hodnotu 0).

Matice znakov musia byť vytvorené pre všetky zobraziteľné znaky, teda od znaku medzery (kód 32) až po znak copyright (kód 127) a uložené v pamäti, odkiaľ sa pri výpise znaku kopírujú do pamäti kresby. Celková dĺžka pamäti, nutná pre matice všetkých znakov, je 96 * 8 = 768 bytov a oblasť vyhradená pre matice znakov sa nazýva znakový generátor (pomocou neho sú generované znaky na obrazovke).

Po resete počítača je znakový generátor umiestnený v pamäti ROM od adresy 3D00h = 15616 a táto adresa, zmenšená o 256, je umiestnená v systémovej premennej CHARS. Z toho tiež vyplýva, že znakový generátor môže byť umiestnený i v pamäti RAM, ale do systémovej premennej CHARS musí byť vložená adresa začiatku generátora znakov, zmenšená o 256.

Skúste teraz zadať POKE 23606,255: POKE 23607,0. Po odoslaní riadku sa nevypíše žiadne hlásenie, len v editačnej oblasti sa objaví riadok zložený z akýchsi hieroglyfov. Vložením týchto dvoch čísiel do premennej CHARS ste totiž zmenili začiatok generátora znakov na adresu nula. A na tejto adrese je uložená rutina v strojovom kóde, žiaden generátor znakov. Zresetujte počítač.

Ostáva len povedať, ako taký vlastný generátor znakov vytvoríte. Základom je vytvoriť matice všetkých znakov. Tie potom za sebou uložíte po jednotlivých bytoch do pamäti.

Maticu znaku si môžete predstaviť ako šachovnicu s veľkosťou 8×8. Do nej musíte zakresliť znak. Predpokladajme, že chcete vytvoriť maticu znaku s kódom 33, čo je výkričník. Nakreslite si šachovnicu:

byte	7	6	5	4	3	2	1	0	bit
0	0	0	0	1	1	0	0	0	BIN 00011000=24
1	0	0	0	1	1	0	0	0	BIN 00011000=24
2	0	0	0	1	1	0	0	0	BIN 00011000=24
3	0	0	0	1	1	0	0	0	BIN 00011000=24
4	0	0	0	1	1	0	0	0	BIN 00011000=24
5	0	0	0	0	0	0	0	0	BIN 00000000=0
6	0	0	0	1	1	0	0	0	BIN 00011000=24
7	0	0	0	0	0	0	0	0	BIN 00000000=0

Týmto sme vytvorili maticu znaku "!". Predpokladajme, že generátor znakov má byť uložený od adresy 60000. Od adresy 60000 po adresu 60007 je uložená matica znaku medzery (8 bytov), od adresy 60008 po adresu 60015 je uložená matica znaku výkričník, od adresy 60016 po adresu 60023 matica znaku úvodzovky hore atď.

Časť pamäti vyzerá nasledovne:

?	← 60016 začiatok uloženia matice znaku úvodzovky
0	← 60015 koniec uloženia matice znaku výkričník
24	
0	
24	
24	
24	
24	
24	← 60008 začiatok uloženia matice znaku výkričník
0	← 60007 koniec uloženia matice znaku medzera

Podobným spôsobom môžete vytvoriť matice všetkých znakov od medzery až po copyright a uložiť ich na príslušné adresy v generátore znakov.

Adresu začiatku osmice bytov vyjadrujúcich maticu znaku dostanete týmto výpočtom:

AdresaOsmiceBytuZnaku=AdresaGeneratoraZnaku + 8 × (CODE"znak" - 32)

V našom prípade: `Adresa = 60000 + 8 × (CODE"!" - 32) = 60000 + 8 × (33 - 32) = 60000 + 8 = 60008`

2.5. Užívateľská grafika (udg)

Počítač disponuje okrem základnej sady znakov ešte ďalšou sadou. Tá sa nazýva užívateľsky definovaná grafika (udg) a slúži na výpis znakov, ktoré si sami vytvoríte pre použitie vo vlastných programoch.

Znak užívateľskej grafiky je možné vypísať po prechode do grafického módu (stlačením CAPS SHIFT + 9) stlačením jedného z klávesov "A" až "U". Užívateľská grafika má teda rozsah 21 znakov a zaberá v pamäti miesto 168 bytov. Po resete počítača je uložená od adresy 65368 a na túto adresu je nastavená systémová premenná UDG. Zmenou jej hodnoty je možné meniť adresu začiatku udg, vždy ju však doporučujeme umiestniť nad RAMTOP.

Znaky udg sa po resete počítača rovnajú stlačenému klávesu. Pri tvorbe vlastných znakov udg postupujeme úplne rovnako, ako pri vytváraní nových znakov pre znakový generátor. Jediný rozdiel je pri vypočítavaní adresy začiatku matice znaku v pamäti, vpre ktorej vyčíslenie môžete použiť funkciu USR, ktorá pri použití tvaru PRINT USR "znak" vracia **adresu začiatku matice znaku udg v pamäti**.

Uvažujme, že chcete nájsť adresu začiatku matice znaku uloženého pod klávesom "C". Po resete počítača napíšte nasledujúci riadok: PRINT USR "C" a na obrazovku sa vypíše adresa 65384, čo je začiatok matice udg znaku uloženého pod klávesom "C". Podobným spôsobom získate adresy všetkých matíc udg znakov.

2.6. Funkcia USR, rutiny z ROM

V tejto krátkej časti sa budeme venovať niektorým zaujímavým rutinám (podprogramom) z pamäti ROM, teda z operačného systému počítača, ktoré by vám mohli uľahčiť vaše počiatočné kroky pri práci v strojovom kóde.

Na začiatok pripomeňme funkciu USR. Táto funkcia slúži na volanie podprogramov v strojovom kóde. Pokiaľ jej argument ako **x**, tak x môže byť ľubovoľný celočíselný výraz z intervalu 0 až 65535 (v prípade použitia u udg môže byť argumentom znak uložený pod klávesom "A" až "U". V tejto časti budeme uvažovať ako argument **x** a riadenie je predané na adresu pamäti x.

Návrat zo strojového kódu musí byť zakončený inštrukciou RET a v prípade, že bolo zakázané prerušenie inštrukciou DI, musí byť opäť pred návratom do BASICu povolené inštrukciou EI.

Pri volaní funkcie USR je hodnota argumentu x predaná do registrového páru BC a po návrate je výsledkom funkcie posledná hodnota tohto registrového páru.

Počiatočnú prácu v strojovom kóde vám môžu uľahčiť nasledujúce rutiny z ROM:

- vymazanie celej obrazovky - okrem horných desiatich riadok:

```
RANDOMIZE USR 3652
```

- posun celej obrazovky o jeden riadok hore:

```
RANDOMIZE USR 3280
```

- posun celej obrazovky o 22 riadkov smerom hore:

```
RANDOMIZE USR 3330
```

- výpis dĺžky voľnej pamäti pre program v BASICu:

```
PRINT 65535 - USR 7962
```

- reset počítača:

```
RANDOMIZE USR 0
```

- čakanie na stlačenie ľubovoľného klávesu; po jeho stlačení sa vypíše dĺžka čakania v 1/50 sek.:

```
PRINT USR 7997 - USR 7997
```

- výpis textového reťazca - $203C_H$ (8252):

```
…
LD A,02h (kanál pre výpis na obrazovku)
CALL 1601h (otvorenie kanála)
LD DE, počiatočná adresa textu v pamäti
LD BC, dĺžka textu v bytoch
CALL 203Ch
…
```

- pípnutie - $03B5_H$ (949)

```
…
LD DE, dĺžka tónu
LD HL, výška tónu
CALL 0385h
…
```

- save na pásku - $04C2_H$ (1218):

```
…
LD IX, počiatočná adresa úseku pamäti pre SAVE
LD DE, dĺžka úseku v bytoch
LD A, značkový byte (obvykle hlavička 0, telo 255)
CALL 04C2h
…
```

- load z pásky - 0556_H (1366):

```
…
LD IX, počíatočná adresa
LD DE, dĺžka
LD A, značkový byte (obvykle hlavička 0, telo 255)
SCF
CALL 0556h
…
```

- zmena farby bordera - 2298_H (8859):

```
…
LD A, farba borderu (môže byť z intervalu <0-7>)
CALL 2298H
…
```

- vykreslenie bodu so súradnicami X, Y - $22E5_H$ (8933):

```
…
LD B,Y (súradnica môže byť z intervalu <0-175>)
LD C,X (súradnica môže byť z intervalu <0-255>)
CALL 22E5H
…
```

- cls (vyčistenie obrazovky) - $0D68_H$ (3432):

```
…
LD A,02h (kanál pre výpis na obrazovku)
CALL 1601h (otvorenie kanála)
CALL OD68H
…
```

- zistenie výskytu bodu so súradnicami X,Y - $22CE_H$ (8910):

```
…
LD B, Y (súradnica môže byť z intervalu <0-175>)
LD C, X (súradnica môže byť z intervalu <0-255>)
CALL 22CEH (v A reg. je výsledok: 1/0 - bod je/nie je)
…
```

Veríme, že postupom času prídete na rad rozmanitých rutín z pamäti ROM, ktoré budete s úspechom využívať v programoch.

2.7. Práca s V/V kanálmi

V tejto časti sa budeme viacej venovať práci s V/V kanálmi počítača, pomocou ktorých sa dajú obsluhovať rôzne **periférne zariadenie** (niekedy ich označujeme tiež ako vstupno-výstupné zariadenia, V/V zariadenia, periférie).

Systém, ktorý umožňuje pripojenie týchto periférií, je úplne univerzálny. K počítaču totiž možno pripojiť V/V zariadenie s ľubovoľným druhom komunikácie. Pre toto prepojenie musíte poznať, ako pracujú **kanály** a linky vášho počítača, pretože ten so svojimi perifériami komunikuje práve pomocou týchto kanálov a liniek.

Linka je cesta ktorou sa posielajú jednotlivé spracovávané znaky z počítača do periférie, alebo sú perifériou vysielané a počítač ich prijíma. Typ periférie je označovaný ako kanál, pričom kanál nemá nič

spoločné s hardwareovou konštrukciou periférie a ide len o programové zaistenie komunikácie medzi hlavným programom a vstupno/výstupnými programami pre obsluhu periférií. Ako už viete z kapitoly 2.3.2.4. každý kanál obsahuje adresu vstupnej rutiny, výstupnej rutiny a typ periférie pre ktorú je kanál určený.

Tabuľka liniek (jej začiatok je uložený v systémovej premennej STRMS = 23568) má dĺžku 38 bytov (2 × 19 dvojbytových hodnôt) a sú v nej uložené informácie o všetkých linkách. Prvé tri hodnoty prináležia linkám -3 až -1, ktoré používa BASIC (pre editor, tlač hlásenia o chybe atď.). Tieto linky nie sú obvyklými príkazmi prístupné a nie je dobré do nich zasahovať ani príkazom POKE.

Štvrtá až devätnásta hodnota (pre linky nula až pätnásť) určujú, na ktorý kanál je linka napojená nasledovne: k hodnote uloženej v systémovej premennej CHANS (začiatok kanálov) sa pripočíta hodnota uložená v tabuľke liniek na mieste prislúchajúcom danej linke a výsledkom je adresa kanála, na ktorý je linka pripojená. Tabuľka liniek po resete obsahuje tieto hodnoty:

linka	posun	typ kanálu	
15	0	uzatvorený	
.	.	.	
.	.	.	
4	0	uzatvorený	
3	16	tlačiareň	
2	6	obrazovka	
1	1	klávesnica	
0	1	klávesnica	
-1	11	BASIC editor	
-2	6	obrazovka	
-3	1	klávesnica	← STRMS

Pre prácu s linkami slúžia dva príkazy BASICu. Prvý z nich je príkaz **OPEN#**. Pomocou neho môžete priradiť ľubovoľnej linke 0 až 15 ľubovoľný dostupný kanál. Po príkaze OPEN #5,"s" je linke s poradovým číslom 5 priradený kanál typu obrazovka. Tabuľka liniek po tomto príkaze vyzerá nasledovne:

linka	posun	typ kanálu	
15	0	uzatvorený	
.	.	.	
.	.	.	
5	6	obrazovka	
4	0	uzatvorený	
3	16	tlačiareň	
2	6	obrazovka	
1	1	klávesnica	
0	1	klávesnica	
-1	11	BASIC editor	
-2	6	obrazovka	
-3	1	klávesnica	← STRMS

Z uvedeného príkladu vyplýva, že na jeden kanál môže byť nasmerovaných viacej liniek, ale jedna linka nemôže súčasne ukazovať na viacero kanálov.

Druhým príkazom pre prácu s linkami je príkaz **CLOSE#**. Tento príkaz uzatvára linku, teda vlastne dosadí v tabuľke liniek na miesto vyhradené danej linke nuly a tým linka prestane byť aktívna. Pokiaľ zadáte CLOSE #5, tabuľka liniek vyzerá rovnako, ako pred otvorením linky. Teraz môžete linku opäť otvoriť, napr. pre typ kanála tlačiareň príkazom OPEN #5,"p".

Ostáva už len popísať, akým spôsobom sa dá napojiť do daného kanálového systému ľubovoľná vstupná alebo výstupná rutina. Je treba uložiť jej adresu na patričné miesto v tabuľke kanálov.

Uvedieme príklad na napojenie výstupnej rutiny pre tlačiareň na tretiu linku (tá je tlačiarní štandardne vyhradená).

```
open_p  ld    hl, (2*6+STRMS)   ;posun od CHANS do HL
        ld    (de), CHANS       ;začiatok kanálov do DE
        add   hl, de            ;adresa kanála v HL
        push  hl                ;uschovaj ju
        ld    hl, outp - open_p ;relatívna adresa OUTP oproti začiatku tejto rutiny
        add   hl, bc            ;adresa OUTP v HL
        pop   de                ;späť adresa kanála
        ex    de, hl            ;vymeň DE a HL
        ld    (hl), d           ;vyšší byte OUTP do tabuľky kanálov
        dec   hl
        ld    (hl), e           ;nižší byte OUTP do tabuľky kanálov
        ret

outp    …                       ;vyslanie znaku na periférne zariadenie
        ret                     ;návrat
```

Po vykonaní tejto rutiny (RANDOMIZE USR OPEN_P) je nastavená adresa výstupnej rutiny tretieho kanála na adresa OUTP a tá zaisťuje výpis všetkých znakov (poslaných cez tretiu linku) na tlačiareň.

Zatiaľ sme sa zaoberali len možnosťou presmerovať nejakú linku na už existujúci kanál. Čo je však treba urobiť, pokiaľ chceme otvoriť ešte ďalšie kanály? Odpoveďou by vám mohla byť nasledujúci rutina, ktorá vytvorí miesto pre ďalšie dva kanála (kanály 4 a 5) a vykoná ich inicializáciu. Po vykonaní rutiny môžete používať tvary PRINT #3, PRINT #4 a PRINT #5, pričom výstupné rutiny týchto kanálov sú odlišné a v praxi to znamená, že môžete mať pripojené tri rôzne zariadenia typu tlačiareň pre výstup (napr. na kanále 3 tlačiareň, na kanále 4 dierovač diernej pásky a na kanále 5 zapisovač). Táto rutina je relokovateľná, otvorí štvrtý kanál pre vstup/výstup a piaty kanál tiež pre vstup/výstup.

```
zac     push  bc                ;v reg. páre BC adresa začiatku rutiny
        ld    hl, (23631)       ;v HL adresa začiatku kanálov
        ld    bc, #14           ;14h=20d, teda posun pre 4. kanál
        add   hl, bc            ;adresa 4. kanála
        push  hl                ;uschovaj ju
        ld    a, (h1)           ;obsah prvého bytu 4. kanála
        cp    #80               ;je kanál už vytvorený?
        jr    nz, obn           ;pokiaľ áno, obnov len adresy rutín
        ld    bc, 10            ;miesto pre dva kanály = 2 × 5 = 10 bytov
        call  #1655             ;vytvor miesto pre dva kanály
obn     pop   de                ;v DE adresa 4. kanálu
        pop   hl                ;v HL adresa začiatku tejto rutiny
        ld    bc, tabk - zac    ;v BC rel. posun tabuľky oproti začiatku
        add   hl, bc            ;v HL adresa tabuľky nových hodnôt kanál.
        ld    bc, 10            ;v BC dĺžka tabuľky - 10 bytov
        ldir                    ;prenos tabuľky do kanálovej oblasti
        ld    hl, 23568         ;v HL začiatok oblasti liniek
        ld    bc, 14            ;v BC posun pre 4. linku (2 × 7 = 14)
        add   hl, bc            ;v HL adresa 4. linky
        ld    bc, 21            ;4.linka smeruje na 4. kanál (21 = 1 + 4 × 5)
```

```
            ld    (hl), c          ;inicializuje 4. linku pre 4. kanál
            inc   hl
            ld    (h1), b
            ret                    ;koniec inicializačnej rutiny

tabk        dw    outpunch         ;adresa rutiny výstupu pre 4. kanál
            dw    inpunch          ;adresa rutiny vstupu pre 4. kanál
            db    "P"              ;typ tlačiareň
            dw    outplott         ;adresa rutiny výstupu pre 5. kanál
            dw    inplott          ;adresa rutiny vstupu pre 5. kanál
            db    "P"              ;typ tlačiareň
```

Týmto príkladom skončíme prácu s V/V kanálmi. Na ich lepšie pochopenie vám doporučujeme vytvoriť si niekoľko vlastných programov pre vstup a výstup. Nepotrebujete k tomu žiadne periférne zariadenie, stačí len vytvoriť rutiny, z ktorých jedna niekam do pamäti zapisuje (napr. príkazom PRINT #3; text) a druhá tento text zasa z pamäti číta (napr. príkazom INPUT#3;a$).

2.8. Práca s V/V miestami

Okrem zápisu a čítania hodnôt z pamäti môže procesor zapisovať a čítať ešte z ďalších miest, ktoré sa nazývajú **V/V miesta**. Celkom ich je (podobne ako pamäťových miest) 65536 a používajú sa pri komunikácii procesora s perifériami. Inštrukcie určené pre prácu s V/V miestami sú inštrukcie OUT a IN, veľmi podobné inštrukciam POKE a PEEK. Inštrukcia OUT zapisuje na V/V miesto danú hodnotu. Jej tvar je: **OUT adresa**, **hodnota** kde adresa je z rozsahu 0 až 65535 (teda dva byty) a hodnota z rozsahu 0 až 255 (teda jeden byte). Čo sa pri použití tejto inštrukcie stane? Počítač má celkom 16 adresových liniek, ktorými adresuje pamäť a porty od 0 do 65535. Ďalej má 8 dátových liniek, na ktorých sa môže objaviť hodnota 0 až 255 (týchto liniek je 8, pretože váš počítač je osembitový, u šestnásťbitového počítača je 16 dátových liniek atď.) a pomocou ktorých sa uskutočňuje výmena dát medzi pamäťou a procesorom a medzi V/V miestami a procesorom. V/V miesto, na ktoré je pripojené fyzické V/V zariadenie sa, nazýva **port**.

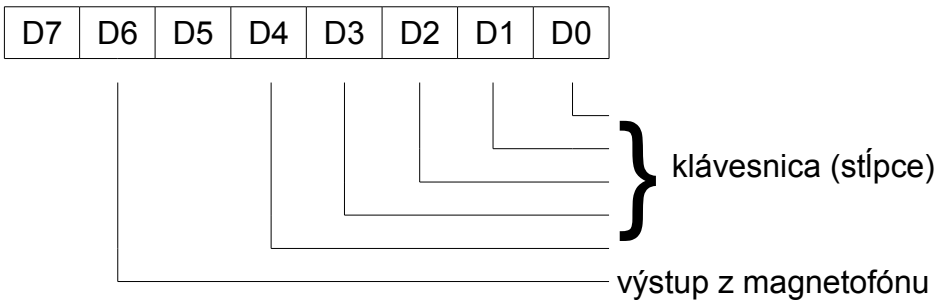
Po inštrukcii **OUT adresa**, **hodnota** sa na adresových linkách objaví zadaná adresa a na dátových linkách zadaná hodnota. Periféria, ktorá je na port s vyslanou adresou pripojená, si prečíta hodnotu na dátových linkách a ďalej ju (už nezávisle na počítači) spracuje. Týmto spôsobom, teda zápisom nejakých hodnôt na port, ku ktorému je pripojená nejaká periféria, môže počítač s touto perifériou komunikovať.

Veľmi podobne pracuje aj funkcia **IN**, ktorá sa používa v tvare: **IN adresa**. Po dekódovaní inštrukcie IN procesor načíta hodnotu z dátových liniek práve v čase, kedy sa na adresových linkách objaví adresa V/V miesta. Takto môže periféria pripojená na nejaký port zasielať počítaču hodnoty. Pomocou portov teda prebieha komunikácia počítača s vonkajšími perifériami. Okrem toho počítač pomocou portu komunikuje s jedinou vnútornou perifériou, so **zákazníckym obvodom U-106-47**.

2.8.1. Zákaznícky obvod

Zákaznícky obvod zaisťuje zobrazovanie na obrazovke, komunikáciu počítača s magnetofónom, zvuk reproduktora a načítanie klávesnice. Zákaznícky obvod má adresu 254.

Funkcia IN načíta s portu 254 hodnotu, v ktorej význam jednotlivých bitov je nasledujúci.



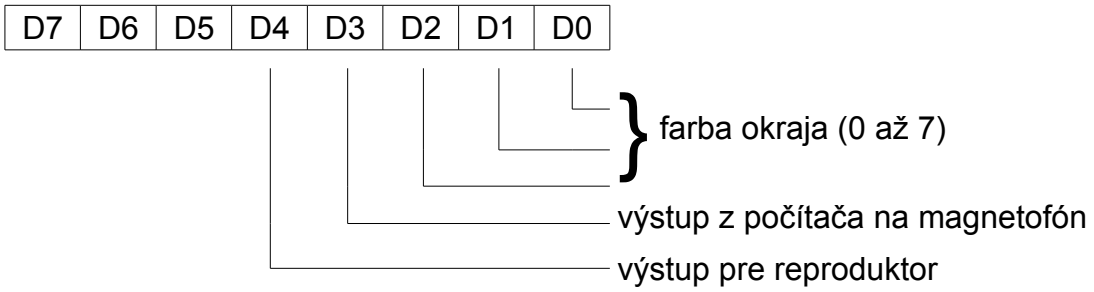
Celá klávesnica je rozdelená do ôsmich radov po piatich klávesoch (päť stĺpcov). Každý rad je možno čítať na inej adrese, pričom päť stĺpcov predstavuje päť nižších dátových bitov (D0 až D4), na bitoch D5 a D7 je hodnota 1, hodnota bitu D6 môže byť 0, alebo 1. Klávesnicu si môžeme znázorniť nasledovne:

port	vyšší byte polriadok	nižší byte stĺpec	D4	D3	D2	D1	D0
65278	11111110_B	11111110_B	V	C	X	Z	CAPS SHIFT
65022	11111101_B	11111110_B	G	F	D	S	A
65410	11111011_B	11111110_B	T	R	E	W	Q
63486	11110111_B	11111110_B	5	4	3	2	1
61438	11101111_B	11111110_B	6	7	8	9	0
57342	11011111_B	11111110_B	P	O	I	U	Y
49150	10111111_B	11111110_B	H	J	K	L	ENTER
32766	01111111_B	11111110_B	B	N	M	SYMBOL SHIFT	SPACE

Jednotlivé bity v načítanom byte sú nastavené na 0, pokiaľ bol niektorý kláves v stĺpci stlačený, a na 1 pokiaľ žiaden kláves v danom stĺpci stlačený náboj.

Bit D6 slúži k nahrávaniu dát z magnetofónu do počítača. Signál, ktorý bol zaznamenaný na magnetofónovú kazetu, sa sním a podľa jeho hodnoty sa nastaví bit D6. Pri vstupe logickej nuly sa bit D6 nastaví na hodnotu 0, pri vstupe logickej jednotky na hodnotu 1. Pokiaľ sa hodnota bitu D6 po určitý čas nezmení, znamená to, že bol načítaný jeden bit. Z ôsmich načítaných bitov sa vytvorí jeden byte a takto sa do počítača nahrá celý blok dát.

Príkaz OUT na port 254 slúži k ovládaniu farieb okraja, reproduktora a pre nahrávanie dát z počítača na magnetofón. Pri inštrukcii OUT je význam bitov vo vyslanom byte nasledujúci:



Farba okraja môže byť v rozsahu čierna (hodnota 0) až biela (hodnota 7).

Pre nahrávanie z počítača na magnetofón slúži bit D3. Spôsob nahrávania z počítača je obdobný ako pri nahrávaní z magnetofónu. Hodnota každého bitu z bytu je vysielaná po určitý čas na bit D3. Logické hodnoty 0, 1 z tohto bitu sú upravené na signály s príslušnou napäťovou úrovňou, ktoré sa vyšlú na vstup magnetofóna pre nahrávanie. Týmto postupom sa všetky byty nahrávaného bloku dát zapíšu na pásku.

Reproduktor je ovládaný striedaním jednotky a nuly. Správnym striedaním núl a jednotek je dosiahnutý žiadaný tón.

2.9. Interface

Pre styk počítača s perifériami sa môže použiť technický alebo programový prostriedok, ktorý prácu s periférioou uľahčí. Tento prostriedok sa nazýva **interface** (interfejs), čo môžeme preložiť "medzitvár", u nás sa používa termín **rozhranie**. Anglické slovo však lepšie vystihuje účel tohto prostriedku.

Interface je teda akýmsi rozhraním medzi počítačom a periférioou, pričom zabezpečuje to, že počítaču sa javí určitá skupina periférií rovnaká, má rovnakú tvár. Ich technická konštrukcia však môže byť úplne odlišná.

Podobne interface zabezpečuje, že periférii sa javí istá skupina počítačov rovnako. Potom môžeme hovoriť o **type interface**, umožňujúceho komunikáciu medzi určitým typom počítačov a periférii. Najrozšírenejšími typmi sú **interface paralelný** a **interface sériový**. Nazývajú sa podľa typu prenosu dát medzi počítačom a periférioou.

Paralelný interface sa vyznačuje tým, že všetkých osem bitov dátového bytu je prenášaných naraz. Výhodou je jednoduchá obsluha zo strany programátora a rýchlosť prenosu, nevýhodou nutnosť mať pre prenos osem dátových vodičov a u niektorých typov paralelných interfaceov môže byť prenos uskutočnený iba do malej vzdialenosti, asi 2 m. Pri väčšej vzdialenosti sa objavujú pri prenose chyby (malá odolnosť voči rušeniu). Existuje niekoľko typov paralelných interfaceoch, medzi najpoužívanejšie patrí pravdepodobne CENTRONIX. Elektronický obvod, ktorý sa v paralelných interfaceoch najviac používa, je obvod MHB 8255 alebo Z80PIO, ale i ďalšie.

Sériový interface prenáša osem bitov jedného bytu postupne jeden za druhým. Používa sa väčšinou tam, kde je treba prenášať dáta na väčšiu vzdialenosť (podľa typu sériového interfacea na niekoľko desiatok metrov, v prípade počítačových sietí spojených cez telekomunikačnú sieť až na vzdialenosť niekoľko tisíc kilometrov) s vylúčením možných chýb.

Na prenos sa používajú dva a viacej vodičov. Typy sériových interfaceov môžu byť napr. RS 232 a V24. Elektronický obvod zaisťujúci sériovú komunikáciu je obvod MHB 8251, alebo Z80SIO, ale i ďalšie.

Okrem technického interface môže existovať ešte **programový interface**. Ten môže byť použitý všade tam, kde je treba pripraviť nejaké dáta do formy určenej určitému typu periférie. Typickým príkladom môže byť súbor rutín v strojovom kóde, ktoré umožnia nahrávať dáta z počítača na magnetofón a späť.

Váš počítač v sebe neobsahuje žiaden zo spomínaných technických interfaceov (ku svojej činnosti ich nepotrebuje), je však možné k nemu interface pripojiť cez systémovú sbernicu. Paralelný interface typu Centronix sa predáva v maloobchodnej sieti, a taktiež je možné si interface objednať priamo vo v. d. Didaktik Skalica.

2.10. Úžitkové programy

Pre počítač ZX Spectrum (a teda i pre Váš počítač) existuje veľké množstvo rôznych programov. Dajú sa rozdeliť do niekoľkých skupín, medzi ktorými dominujú predovšetkým hry. Tejto skupine sa venovať nebudeme, pretože u užívateľov, ktorí sa dostali až po túto kapitolu, predpokladáme záujem o iný druh programov.

Programovacie jazyky:

- veľmi dobrý kompilátor PASCALu HP4TM od firmy HISOFT. Svojimi možnosťami uspokojí i náročných užívateľov a môže sa stať výborným pomocníkom predovšetkým pre stredoškolských studentov.
- programovací jazyk LOGO, určený pre menšie deti, umožňujúci osvojenie základov štruktúrovaného programovania.
- MEGABASIC - rozšírený BASIC, umožňujúci snáď najlepšiu prácu s grafikou zo všetkých interpreterov BASICu pre Váš počítač. Má veľké možnosti po stránke animovanej grafiky.
- rozšírené interpretery BASICu BETABASIC a SIGMABASIC. Obsahujú oproti štandardnému BASICu v ROM mnohé nové príkazy.

Kompilátory BASICu:

- celočíselný kompilátor COLT. Prevádza programy z BASICu do strojového kódu tak, že sú 2-krát až 800-krát rýchlejšie než pôvodný program.
- celočíselný kompilátor MCODER-2.
- celočíselný kompilátor HIBASIC. Medzi celočíselnými kompilátormi je zrejme najlepší.
- reálny kompilátor TOBOS FP, ktorého autori sa zamerali predovšetkým na zrýchlenie operácií s reálnymi číslami. Vo svojej kategórii jeden z najlepších.

Assemblery a monitory:

- assembler GENS3 a monitor MONS3 patria medzi najstaršie programy tohto typu. Ich prednosťou je relokovateľnosť.
- MRS (Memory Resident System). Program združujúci assembler, monitor a debugger (prostriedok pre odladovanie programov). Vo svojej triede zrejme najlepší na vytváranie a odladovanie programov v strojovom kóde procesora Z-80.

Textové editory:

- TASWORD TWO a jeho modifikácie. Veľmi rozšírený editor s celým radom funkcií, spríjemňujúcich prácu s textom. Veľmi jednoduchá inštalácia obslužných programov pre rôzne typy tlačiarní.
- D-WRITER. Editor, približujúci sa svojim ovládaním a príkazmi k editorom pre počítače typu IBM.

Grafické programy:

- ART STUDIO. Program umožňujúci veľmi jednoducho vytvárať obrázky, ktoré užívateľ môže začleniť do vlastných programov.
- LEONARDO, PENCIL.

Výučbové programy:

- programy na výuku cudzích jazykov, zemepisu, matematiky. Umožňujú užívateľovi zvládnuť základy týchto predmetov a pomôžu pri príprave ku skúškam napr. na stredné typy škôl.
- Z-80 TUTOR je program, ktorý vás zoznámi so základmi programovania v strojovom kóde procesora Z-80. Celý program je rozdelený do viacej než 30 lekcií, venovaných vždy jednému typu inštrukcií.
- MAKE A CHIP vás zoznámia so základmi číslicovej techniky.

Všetky uvedené programy existujú medzi užívateľmi počítača ZX Spectrum, v kluboch a staniciach mladých technikov, popr. je možné niektoré z nich kúpiť v nasledujúcich organizáciách:

- Didaktik Skalica, Nálepkova 22, 90901 Skalica
- ZenitCentrum Beroun, Hostímská 1, 26601 Beroun
- 602. ZO Svazarmu Praha, Dr. Z. Wintra 8, 16041 Praha 6
- 666. ZO Svazarmu Praha, P. S. 64 16900 Praha 69

2.11. Hardware

Na nasledujúcom obrázku je bloková schéma zapojenia mikropočítača Didaktik M.

Hlavné časti počítača sú:

- - centrálna procesorová jednotka CPU
- - zákaznícky obvod Didaktik U-106-47
- - pamäť ROM 16 kB
- - pamäť RAM 48 kB
- - vstupno-výstupný blok obsahujúci:
    - klávesnicu
    - interface pre magnetofón
    - obvody video a vf
    - akustický výstup
- - napájací zdroj

Centrálnu procesorovú jednotku predstavuje 8-bitový mikroprocesor Z-80. Komunikáciu procesora s ostatnými časťami mikropočítača zabezpečujú 3 zbernice: adresná, dátová a riadiaca. Datová zbernica je osembitová, obojsmerná a sprostredkúva prenos dát medzi jednotlivými časťami systému. Adresná zbernica je 16-bitová a umožňuje adresovať 64 kB pamäti. Riadiaca zbernica obsahuje signály vystupujúce z procesora (výstupné) ako aj vystupujúce do procesora (vstupné).

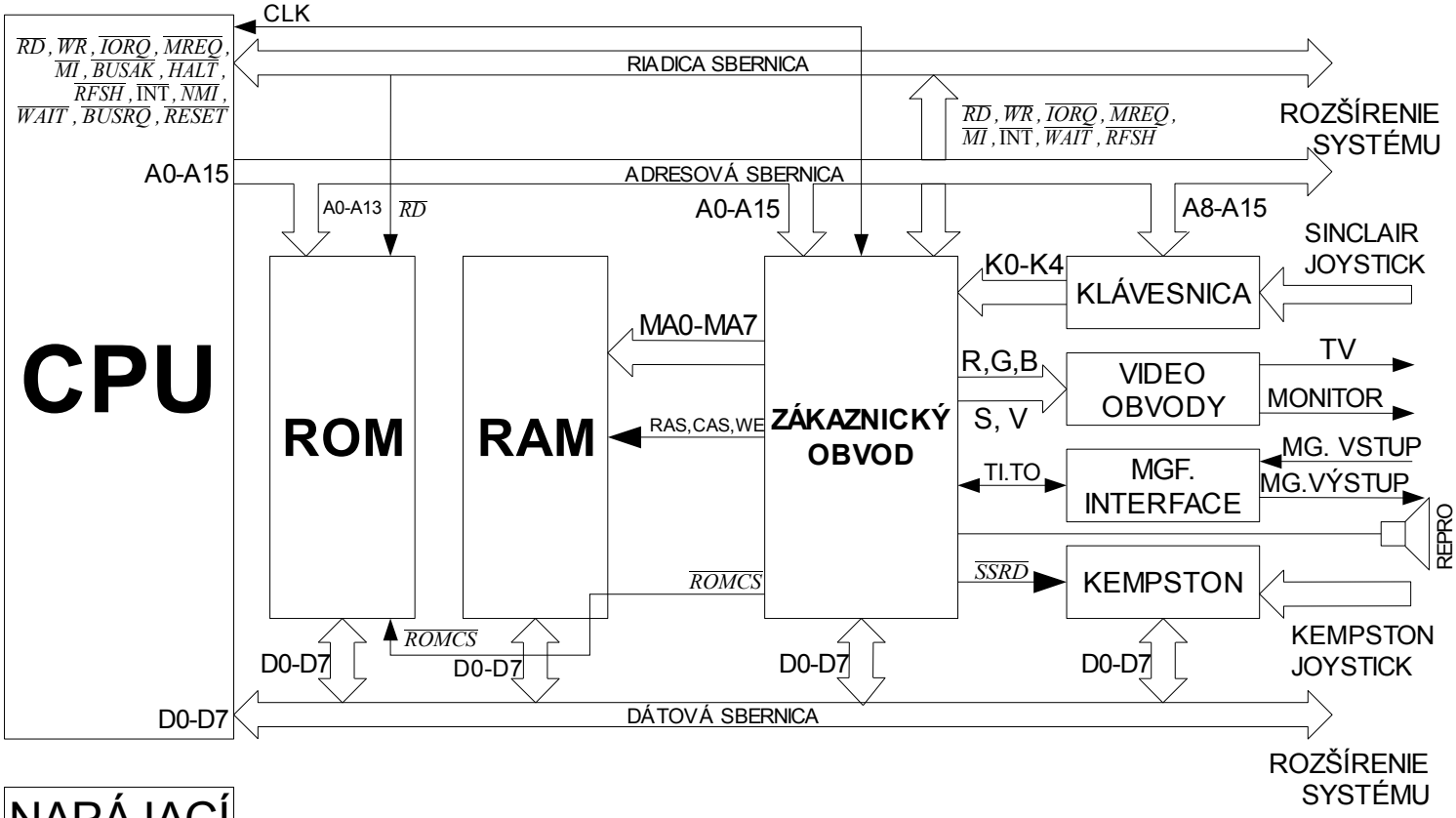
Výstupné riadiace signály sú:

$\overline{WR}$ - zápis
$\overline{RD}$ - čítanie
$\overline{IORQ}$ - komunikácia s vstupno-výstupným zariadením
$\overline{MERQ}$ - komunikácia s operačnou pamäťou
$\overline{BUSAC}$ - signalizuje odpojenie procesora od zbernic

Vstupné signály sú:

$\overline{INT}$ - požiadavka na prerušenie činnosti procesora
$\overline{NMI}$ - požiadavka na prerušenie činnosti procesora (nemaskovateľná)
$\overline{WAIT}$ - požiadavka na zastavenie činnosti procesora
$\overline{BUSRQ}$ - požiadavka na odpojenie procesora od zbernic

CPU pracuje s frekvencou 4 MHz generovanou zákazníckym obvodom.



NAPÁJACÍ ZDROJ

BLOKOVÁ SCHÉMA DIDAKTIK M

Zákaznícky obvod vykonáva niekoľko funkcií:

- zabezpečuje cyklické čítanie pamäti video RAM a generuje video signály R, G, B a synchronizačný signál S. Signály R, G, B sú generované frekvenciou 8 MHz, čo pri pravidelnom striedaní svetlých a tmavých bodov môže v systéme PAL spôsobiť výskyt farebných pruhov
- generuje všetky potrebné signály pre činnosť pamätí RAM
- generuje periodický signál WAIT, pomocou ktorého dovoľuje prístup do pamätí RAM len vtedy, keď zákaznícky obvod nečíta dáta pre zobrazovanie. Zákaznícky obvod teda ovláda prideľovanie dátovej zbernice a tým zabezpečuje zdieľanie pamäti RAM. Tento spôsob prideľovania zberníc nezaručuje presné generovanie časových slučiek.
- generuje taktovaciu frekvenciu 4 MHz pre CPU
- generuje signál INT s periódou 20 ms - umožňuje obsluhu klávesnice
- generuje signál ROM CS pre výber pamäti ROM
- obsahuje vstupno-výstupné brány pre ovládanie klávesnice, pre komunikáciu s magnetofónom a akustický výstup

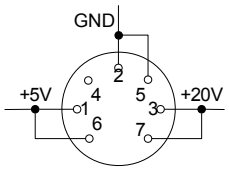
Pamäť ROM má kapacitu 16 kB. Je možné ju odpojiť signálom ROMCS vyvedeným na systémový konektor (pripojením úrovne log.1) a predať riadenie vonkajšej pamäti ROM (ROM modul).

Pamäť RAM je dynamická s kapacitou 64 kB osadená obvodom typu 4164. Využitých je len 48 kB tejto pamäti. Generovanie adresy, riadiacich signálov ako aj občerstvovanie pamäti (refresh) zabezpečuje zákaznícky obvod.

Ďalej sa budeme venovať jednotlivým častiam podrobnejšie.

Napájací zdroj

Je umiestnený v samostatnej krabičke a je určený na pripojenie k sieti 220 V / 50 Hz. Zdroj dodáva stabilizované napätie +5V / 1A a nestabilizované napätie cca 20V / 200mA vytvorené zdvojovačom napätie - toto napätie sa stabilizuje v počítači na 12 V. Popis vývodov konektora je na nasledovnom obrázku:



Zapojenie konektora napájenia pri pohľade zozadu na počítač

Stabilizovaný zdroj obsahuje 2 poistky: jednu v primárnej časti 0.16 A a druhú v sekundárnej časti 1.6 A.

Každá úprava v napájacom zdroji môže mať vážne následky na funkciu počítača, preto jediný oprávnený zásah je výmena poistky.

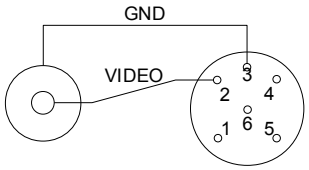
Klávesnica

Klávesnica počítača je zapojená do matice 5×8 na horných 8 bitov adresnej zbernice a na 5 dátových bitov zákazníckeho obvodu. Klávesnice je osadená 45 klávesami. 5 klávesov (CAPS SHIFT a šípky) je dvojkontaktných. Prvý kontakt klávesu CAPS SHIFT je sériovo spojený s klávesom RESET, čím je zabezpečené, že resetovanie počítača je možné docieliť len súčasným stlačením týchto dvoch klávesov. Samostatné klávesy so šípkami sú zapojené tak, že simulujú súčasné stlačenie klávesov CAPS SHIFT a jedného z klávesov 5-8.

TV a VIDEO

Video signál je určený pre pripojenie monitora (farebného aj monochromatického) alebo TV prijímača pomocou vstupu pre videomagnetofón. Úroveň video signálu je minimálne 0.8 V.

Príslušný kábel nie je univerzálny, pretože konektory pre videomagnetofón bývajú zapojené rôzne. Na obr. je popis zapojenia video signálu k TV prijímaču ORAVAN.



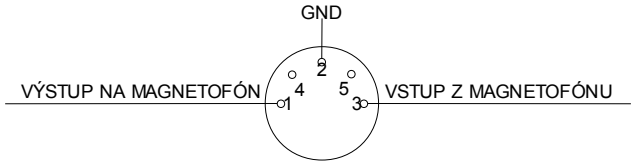
Zapojenie konektora pre pripojenie videosignálu k TVP Oravan.

Vf signál je určený pre pripojenie k anténemu vstupu TV prijímača káblom, ktorý je súčasťou dodávky. Signál je v pásme UHF kanál č. 48. Po vyladení je potrebné nastaviť vhodný jas a kontrast.

Magnetofón

Na nasledujúcom obrázku je popis konektora určeného na pripojenie magnetofónu.

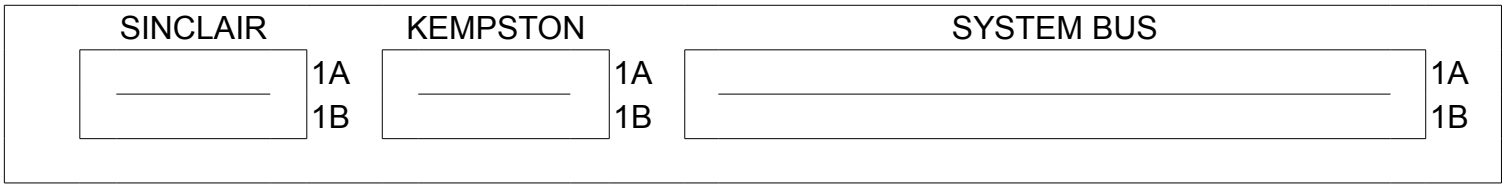
Minimálne vstupné napätie z magnetofónu do počítača je 220 mV. Minimálne výstupné napätie z počítača do magnetofóna je 1 V.



Zapojenie vstupno-výstupného konektora pre magnetofón při pohľade zozadu na počítač

Pripojenie joystickov

V počítači sú na zadnom čele dva krátke obdĺžnikové otvory, do ktorých je možné priamo zapojiť dva joysticky, jeden typu Sinclair joystick B a druhý typu Kempston. Oba konektory pre joysticky sa nachádzajú na zadnej strane počítača. Sú riešené ako priame konektory s 2 × 6 vývodmi s rozostupom 2,54 mm. Ich umiestnenie na zadnom čele je nasledovné:



Konektor pre Sinclair joystick je paralelne pripojený k časti klávesnice vyberanej adresou A12 (klávesy 6 - 0). Pripojenie Sinclair joysticka je v tabuľke 1, v zátvorkách sú uvedené klávesy na klávesnici, ktorých stlačenie vyvolá rovnakú činnosť, ako pohyb joystickom. V prípade, že ľubovoľný program má možnosť ovládania definovaním klávesov, stačí tieto definovať podľa popísaného pripojenia, a program je potom možné ovládať Sinclair joystickom.

Tabuľka 1

strana spojov	číslo vývoda	strana súčiastok
NC	1	spoločný vodič
výrez	2	výrez
NC	3	vpravo (7)
vľavo (6)	4	k sebe (8)
od seba (9)	5	akce (0)
NC	6	NC

Pripojenie Kempston joysticku je v tabuľke 2, v zátvorkách sú uvedené hodnoty načítané po pripojeniu Kempston joysticku z portu 31 inštrukciou IN.

Tabuľka 2

strana spojov	číslo vývoda	strana súčiastok
NC	1	spoločný vodič
výrez	2	výrez
NC	3	vpravo (1)
vľavo (2)	4	k sebe (4)
od seba (6)	5	akce (16)
aktivácia	6	aktivácia

Načítanie hodnôt z portu s adresou 31 si môžete skúsit odštartovaním nasledujúceho programu:

```
10 PRINT AT 0,0;"    ";AT 0,0;IN 31:GO TO 10
```

Vstupný port pre Kempston joystick je aktivovaný v prípade, že je Kempston joystick zapojený v počítači. Toto musí byť zabezpečené prepojením pinov 6A a 6B na konektore joysticku. Zasunutím joysticku sa premostia piny 6A a 6B na priamom konektore počítača, čím sa aktivuje port na adrese 31. Tento spôsob aktivácie portu pre joystick bol realizovaný preto, aby bolo možné použiť iné vonkajšie zariadenie s tou istou adresou ako Kempston joystick (31).

Kovodružstvo Náchod vyrába joystick s označením Didaktik M, ktorý je určený priamo na pripojenie k počítaču Didaktik M.

Joystick zakončený iným konektorom je treba príslušne upraviť (použitý konektor je časť z konektora typu WK 46580).

Systémový konektor

Systémový konektor sa nachádza na zadnej strane počítača. Sú na ňom vyvedené adresná, dátová, riadiaca zbernica a pomocné signály. Je riešený ako priamy konektor s 2 x 25 vývodmi s rozostupom 2,54 mm. Protikusom je časť konektora WK 46580.

Vývody sú na obidvoch stranách počítačovej dosky. Vrchnú stranu dosky (počítač klávesnicou hore) pri pohľade zhora budeme nazývať **strana súčiastok** (strana A) a spodnú stranu dosky budeme nazývať **strana spojov** (strana B).

Začiatok číslovania konektora je pri pohľade zozadu (počítač opäť klávesnicou hore) vpravo (viď. umiestnenie na zadnom čele podľa predchádzajúceho obrázku).

Rozmiestnenie signálov na systémovom konektore (SYSTEM BUS):

strana spojov	číslo vývodu	strana súčiastok
A14	1	A15
A12	2	A13
+5V	3	D7
NC	4	NC
výrez	5	výrez
GND	6	D0
GND	7	D1
CLK	8	D2
A0	9	D6
A1	10	D5
A2	11	D3
A3	12	D4
NC	13	/INT
GND	14	/NMI
NC	15	/HALT
NC	16	/MREQ
NC	17	/IORQ
NC	18	/RD
/BUSRQ	19	/WR
/RESET	20	/ROMCS-D
A7	21	/WAIT
A6	22	NC
A5	23	NC

strana spojov	číslo vývodu	strana súčiastok
A4	24	/M1
ROMCS	25	/RFSH
/BUSAK	26	A8
A9	27	A10
A11	28	NC

Význam jednotlivých signálov je nasledujúci, / pred signálom znamená negáciu signálu:

signál	význam signálu
A0-A15	adresová zbernica vyvedená z mikroprocesora
/BUSAK	potvrdenie žiadosti o zbernicu
/BUSRQ	žiadosť o zbernicu
D0-D7	datová zbernica vyvedená z mikroprocesora
GND	zem
/HALT	informácie o vykonávaní inštrukcie HALT
/INT	vstup maskovateľného prerušenia
/IORQ	žiadosť o prístup k I/O zariadeniu
/MREQ	žiadosť o prístup do pamäti
/M1	indikácia priebehu strojného cyklu číslo 1
NC	nezapojené vývody konektora
/NMI	nemaskovateľné prerušenie
/RD	read - indikácia čítacieho cyklu zbernice
/RFSH	refresh - indikácia cyklu obnovenia **DRAM**
/RESET	reset mikropočítača. Je tiež ovládaný tlačidlom z klávesnice
/ROMCS-D	signál pre výber pamäti ROM vyvedený priamo zo zákazníckeho obvodu
ROMCS	signál pre výber pamäti ROM vyvedený priamo zo zákazníckeho obvodu
/WAIT	signál na vloženie čakacieho cyklu mikroprocesora
/WR	write - indikácia zápisového cyklu zbernice
+5V	stabilizované napätie 5V. Pri jeho použití je třeba brať ohľad na dovolené zaťaženie napájacieho zdroja
výrez	miesto, do ktorého sa vsúva zámok konektora

Prílohy

Príloha A: Zoznam hlásení o chybách

Hlásenia o chybách sú oznamy, ktoré počítač vypisuje na spodnú časť obrazovky (editačnú zónu) v prípade riadneho ukončenia programu, alebo pri jeho zastavení z dôvodu chyby pri jeho vykonávaní. Pomáhajú nám určiť dôvod zastavenia programu.

Hlásenie o chybe sa skladá z kódu hlásenia, anglického vyjadrenia významu a dvoch čísiel oddelených dvojbodkou. Prvé číslo udáva číslo riadku a druhé poradové číslo príkazu na riadku, pri spracovaní ktorého došlo k vypísaniu hlásenia o chybe.

Hlásenie	Význam
0 OK	V tomto prípade nejde vlastne o chybu. Počítač oznamuje, že ukončil riadne beh programu vykonaním posledného príkazu na poslednom programovom riadku.
1 NEXT without FOR	NEXT bez FOR. Bolo použité kľúčové slovo NEXT bez predchádzajúceho použitia zodpovedajúceho FOR. Premenná za NEXT musí byť zhodná s premennou za FOR.
2 Variable not found	Premenná nebola nájdená. V programe sa odvolávame na premennú, ktorá nebola doteraz definovaná (nebola jej priradená hodnota). U indexovanej premennej nebol použitý príkaz DIM.
3 Subscript wrong	Chybný index. Index použitý na určenie prvku poľa bol mimo rozsah definovaný príkazom DIM. Najmenšia hodnota indexu je číslo 1.
4 Out of memory	Mimo rozsah pamäti. Počítač nemá pre požadovanú operáciu dostatok miesto v pamäti.
5 Out of screen	Príkaz požadoval zápis mimo pracovnú oblasť obrazovky.
6 Number too big	Číslo je príliš veľké. Výsledkom matematickej operácie je číslo väčšie, než aké počítač dokáže zobraziť. Obvykle to signalizuje delenie nulou.
7 RETURN without GO SUB	RETURN bez GO SUB. Bolo použitých viac príkazov RETURN než bolo volaných podprogramov príkazom GO SUB. Počítač vyčerpal zásobník návratových adries.
8 END of FILE	Koniec súboru. Nie je využité.
9 STOP statement	Hlásenie STOP. Program sa zastavil na príkaze STOP. Po vypísaní CONTINUE je možné pokračovať ďalším príkazom.
A Invalid argument	Nevhodný argument. Funkcii sme predali argument, ktorý nie je schopná spracovať.
B Integer out of range	Celé číslo mimo rozsah. Číslo, ktoré sme predali ako argument funkcie alebo príkazu, je mimo povolený rozsah.
C Nonsense in BASIC	Nezmysel v jazyku BASIC. Zápis nezodpovedá povoleným pravidlám syntaxe.
D BREAK - CONT repeats	BREAK - vypísaním CONT pokračuje. Program bol zastavený klávesom BREAK. V programe je možno pokračovať odoslaním príkazu CONT. Počítač dokončí nedorobený príkaz a pokračuje v programe.
E Out of DATA	Pokus o čítanie dát príkazom READ za koncom zoznamu poľa DATA. Svedčí to o malom počte položiek v poli DATA, alebo o nevykonanej inicializácii príkazom RESTORE pri opätovnom spustení programu príkazom GO TO.
F Invalid file name	Nevhodné meno súboru. Príkazy pre spoluprácu s magnetofónom majú chybne uvedené meno súboru. Meno je dlhšie než 10 znakov, alebo sú u príkazu SAVE uvedené prázdne úvodzovky. Takisto pri zlej špecifikácii zariadenia v príkaze OPEN #.
G No room for line	Nie je priestor pre riadok. V pamäti nie je miesto pre vkladaný riadok. Program je príliš dlhý (vrátane premenných), alebo RAMTOP je príliš nízko.
H STOP in INPUT	STOP v INPUTe. Program bol zastavený zadaním príkazu STOP, alebo v INPUT LINE bol stlačený kláves šípka dole.
I FOR without NEXT	FOR bez NEXT. Príkaz cyklu nie je ukončený kľúčovým slovom NEXT, alebo u FOR nie je použitá rovnaká premenná ako u NEXT, alebo chýba kľúčové slovo STEP v prípade, kedy je nevyhnutné.
J Invalid I/O device	Chybná špecifikácia I/O zariadení. Napr. pri pokuse čítať príkazom INPUT z obrazovky.

Hlásenie	Význam
K Invalid colour	Argument príkazu pre ovládanie atribútu a borderu (PAPER, BRIGHT, ...) je mimo povolený rozsah. Môže sa tiež objaviť v prípade snahy o vypísanie programu v jazyku BASIC, kde je za REM uložený strojový kód.
L BREAK into program	Zastavenie programu pri stlačení klávesu BREAK. Vypísaním CONT program pokračuje ďalším príkazom.
M RAMTOP no good	Hodnota RAMTOP nie je správna. Snaha o zadanie nevhodného parametra v príkaze CLEAR. Hodnota je buď príliš malá, alebo príliš veľká.
N Statement lost	Bol požadovaný skok na neexistujúci príkaz.
O Invalid stream	Odvolávame sa na komunikáciu s kanálom, ktorý nebol doteraz otvorený.
P FN without DEF	FN bez DEF. Pokus o použitie užívateľskej funkcie, ktorá nebola doteraz definovaná
Q Parametr error	Chybný parameter. Užívateľskej funkcii bol predaný parameter, ktorý nezodpovedá jej definícii, alebo bol predaný chybný počet parametrov.
R Tape loading error	Chyba pri nahrávaní.

Príloha B: Zoznam kľúčových slov jazyka BASIC

V tejto časti je uvedený abecedný zoznam kľúčových slov. Strana udáva číslo strany, na ktorej začína podrobný popis tohto kľúčového slova.

Kláves označuje, stlačením ktorého klávesu dostaneme kľúčové slovo (v príslušnom móde).

Význam obsahuje stručný popis významu kľúčového slova.

V zozname kľúčových slov budú mať nasledujúce symboly tento význam:

x, y, z - aritmetický výraz
x$ - reťazec
v - číselná premenná
v$ - reťazcová premenná
c - zoznam príkazov pracujúcich s farbou oddelených čiarkou alebo bodkočiarkou (PAPER 1; INK 3; FLASH 1; ...)

Ak sa vyžaduje niekde celé číslo, z hodnoty aritmetického výrazu sa automaticky berie celá časť.

Príkaz	Strana	Kláves	Význam
ABS x	32	G	funkcia; absolútna hodnota x
ACS x	33	W	funkcia; arkuskosínus x
AND	37	Y	logický operátor; x AND y = x , ak y<>0 0 , ak y=0 x$ AND y = x$, ak y<>0 prázdny reťazec, ak y=0
ASN x	33	Q	funkcia; arkussínus x
AT x,y	43	I	položka v príkaze PRINT; nastaví ukazovateľ pre výpis na riadok x a stĺpec y
ATN x	33	E	funkcia; arkustangens x
ATTR (x,y)	49	L	funkcia; vracia hodnotu atribútu pre pozíciu (x,y)
BEEP x,y	62	Z	príkaz; tón dĺžky x sekúnd,výška je y poltónov nad c^1 pre y>0 alebo pod pre y<0
BIN cislod	32	B	funkcia; cislod je dvojkové číslo, z ktorého vracia desiatkové číslo

Príkaz	Strana	Kláves	Význam
BORDER x	46	B	príkaz; farba okraja obrazovky
BRIGHT x	47	B	príkaz; určuje jas pri zobrazovaní
CAT	64	9	príkaz; pre mikrodrive
CHR$	30	U	funkcia; vracia znak s kódom x
CIRCLE x,y,z	51	H	príkaz; kreslí kružnicu so stredom v bode (x,y) s polomerom z
CLEAR	22	x	príkaz; ruší premenné,vykoná CLS a RESTORE, pozíciu pre PLOT nastaví do ľavého dolného rohu
CLEAR x	65	x	príkaz; ako CLEAR a nastaví RAMTOP na hodnotu x (ak sa dá)
CLOSE # x	89	5	príkaz; zatvára kanál x
CLS	15	V	príkaz; maže obrazovku
CODE x$	30	I	funkcia; kód prvého znaku v x$
CONTINUE	18	C	príkaz; pokračuje v programe tam, kde bol program prerušený inou správou ako 0; ak bola správa 9 alebo L, pokračuje ďalším príkazom; inak opakuje posledný príkaz
COPY	64	Z	príkaz; kopíruje obsah obrazovky na tlačiareň ZX Printer
DATA zoznam výrazov	42	D	k príkazu READ; hodnoty výrazov sa priraďujú premenným z príkazu READ
DEF FN a(zoznam)=x	52	1	príkaz; užívateľskej aritmetickej funkcie a
DEFFN a$(zoznam)=x$	53	1	príkaz; definícia užívateľskej aritmetickej funkcie a
DIM v(x1,...,xk)	25	D	príkaz; definovanie číselného poľa v s dimenziami x1,...,xk
DIM v$(x1,...,xk)	25	D	príkaz; definovanie reťazcového poľa v$ s dimenziami x1,...,xk
DRAW x,y	50	W	príkaz; kreslí úsečku zo súčasnej pozície do bodu (x,y) v relatívnych súradniciach
DRAW x,y,z	50	W	príkaz; kreslí kruhový oblúk daný uhlom z do bodu (x,y) zadaného relatívne
ERASE	64	7	príkaz; pre mikrodrive
EXP x	33	X	funkcia; exponenciálna funkcia x
FLASH x	47	V	príkaz; určuje blikanie
FN a(zoznam)	52	2	volanie užívateľom definovanej aritmetickej funkcie a
FN a$(zoznam)	53	2	volanie užívateľom definovanej reťazcovej funkcie a$
FOR v=x TO y STEP z	39	F	príkaz; príkaz cyklu od x do y s krokom x
FORMAT	64	0	príkaz; pre mikrodrive
GO SUB x	54	H	príkaz; volanie podprogramu na riadku x
GO TO x	18	G	príkaz; volanie podprogramu na riadku x
IF x THEN príkaz	34	U	príkaz podmieňovací; ak x<>0, pokračuje sa za THEN; inak sa pokračuje na nasledujúcom riadku
IN x	64	I	funkcia; číta klávesnicu, výsledkom je znak v móde L alebo C
INK x	47	X	funkcia; premenným v zozname priradí hodnoty zadané z klávesnice
INT x	32	R	funkcia; celá časť x

Príkaz	Strana	Kláves	Význam
INVERSE x	47	M	príkaz; inverzné zobrazenie znakov
LEN x$	28	K	funkcia; dĺžka reťazca x$
LET v=x	21	L	príkaz; priradí premennej v hodnotu x
LET v$=x$	23	L	príkaz; priradí premennej v$ hodnotu x$
LINE	45, 57	3	súčasť príkazu SAVE alebo INPUT
LIST	17	K	príkaz; výpis programu od začiatku na obrazovku
LIST x	17	K	príkaz; výpis programu od riadku x na obrazovku
LLIST	64	V	príkaz; výpis programu od začiatku na tlačiareň
LLIST x	64	V	príkaz; výpis programu od riadku x na tlačiareň
LN x	33	Z	funkcia; prirodzený logaritmus
LOAD x$	57	J	príkaz; nahrá program s menom x$ z pásky do počítača
LOAD x$ DATA v()	58	J	príkaz; nahrá záznam x$ z pásky do poľa ()
LOAD x$ DATA v$()	60	J	príkaz; nahrá záznam x$ z pásky do poľa v$()
LOAD x$ CODE	58	J	príkaz; nahrá záznam x$ z pásky do pamäti, adresa a dĺžka sa berú z pásky
LOAD x$ CODE x	60	J	príkaz; nahrá záznam x$ z pásky do pamäti od adresy x
LOAD x$ CODE x,y	58	J	príkaz; nahrá záznam x$ od adresy x, uloží sa najviac y bytov
LOAD x$ SCREEN$	58	J	príkaz; nahrá záznam x$ z pásky do video pamäti (obrazovka)
LPRINT zoznam	64	C	príkaz; výpis hodnôt zo zoznamu na tlačiareň
MERGE x$	57	T	príkaz; spojí program a premenné z pásky x$ s programom v počítači
MOVE	64	6	príkaz; pre mikrodrive
NEW	19	A	príkaz; maže program a premenné
NEXT v	39	N	príkaz ukončenie cyklu
NOT	37	S	logický operátor; NOT x = 1 , ak x=0 0 , ak x<>0
OPEN # x	89	4	príkaz; otvorenie kanála x
OR	37	U	logický operátor; x OR y = 1 , ak y<>0 x , ak y=0
OUT x,y	64	O	príkaz; vyšle na port s adresou x hodnotu y
OVER x	47	N	príkaz; spôsob prepisovania znakov
PAPER x	46	C	príkaz; farba podkladu pri výpis
PAUSE x	61	M	príkaz; pozastaví vykonávanie programu čas x*0.02 sekundy
PEEK x	32	O	funkcia; číta hodnotu bytu v pamäti s adresou x
PI	33	M	funkcia; má hodnotu 3.1415927
PLOT c;x,y	50	Q	príkaz vykreslí na dané súradnice (x,y) bod farbou INK. V mieste parametru c môžu byť rôzne kombinácie modifikačných slôv PAPER, FLASH, alebo BRIGHT.
POINT (x,y)	51	8	príkaz; má hodnotu 1, ak má bod (x,y) farbu podľa INK; 0, ak má farbu podľa PAPER

Príkaz	Strana	Kláves	Význam
POKE x,y	32	O	príkaz; na adresu x v pamäti sa uloží hodnota y
PRINT zoznam	10, 13	P	príkaz; výpis hodnôt zoznamu na obrazovku
RANDOMIZE	34	T	príkaz; nastavuje generátor náhodných čísiel pseudonáhodne
RANDOMIZE x	34	T	príkaz; nastavuje generátor náhodných čísiel na určité miesto
READ zoznam premenných	42	A	príkaz; premenným v zozname priradí hodnoty z DATA
REM text	18	E	príkaz; pre komentár, je nevýkonný
RESTORE	42	S	príkaz; nastaví ukazateľ pre príkaz READ na začiatok programu
RESTORE x	42	S	príkaz; nastaví ukazateľ pre príkaz READ na riadok x
RETURN	54	Y	príkaz; koniec podprogramu
RND	34	T	funkcia; generuje náhodné číslo
RUN	17	R	príkaz; odštartuje program od začiatku
RUN x	17	R	príkaz; odštartuje program od riadku x
SAVE x$	56	S	príkaz; uloží na pásku pod menom x$ program premenné
SAVE x$ LINE x	57	S	príkaz; uloží na pásku pod menom x$ program a premenné, po nahratí do počítača sa automaticky program odštartuje od riadku x
SAVE x$ DATA v()	58	S	príkaz; uloží na pásku pod menom x$ pole v()
SAVE x$ DATA v$()	59	S	príkaz; uloží na pásku pod menom x$ pole v$()
SAVE x$ CODE x,y	58	S	príkaz; uloží na pásku pod menom x$ časť pamäti od adresy x s dĺžkou y
SAVE x$ SCREEN$	58	S	príkaz; uloží na pásku pod menom x$ video pamäť
SCREEN$ (x,y)	43	K	funkcia; vracia znak napísaný na pozícii (x,y) obrazovky
SGN x	32	F	funkcia; signum x, je -1 , ak x<0 0 , ak x=0 1 , ak x>0
SIN x	33	Q	funkcia; sínus x
SQR x	33	H	funkcia; druhá odmocnina z x
STEP	39	D	súčasť príkazu FOR
STR$ x	28	Y	funkcia; reťazec z hodnoty x
STOP	19	A	príkaz; prerušenie programu
TAB x	43	P	položka v príkaze PRINT, nastaví ukazovateľ pre výpis na stĺpec x
TAN	33	E	funkcia; tangens x
THEN	34	G	súčasť podmieneného príkazu IF
TO	23, 39	F	súčasť príkazu FOR
USR x	66	L	funkcia; odštartuje program v strojovom kóde od adresy x a vracia hodnotu dvojice registrov bc
USR x$	66	L	funkcia; vracia adresu vzoru udg pre jednoznakový reťazec x$
VAL x$	29	J	funkcia; mení reťazec na číslo
VAL$ x$	29	J	funkcia; mení reťazec na reťazec

Príkaz	Strana	Kláves	Význam
VERIFY	56, 59	R	príkaz; analogický ako LOAD; porovnáva záznam na páske s obsahom pamäti

Príloha C: Tabuľka kódov znakov

Každý znak v mikropočítači má svoje číselné vyjadrenie. Pretože Didaktik M nepoužíva štandardný ASCII kód, uvádzame kompletnú tabuľku znakov.

dekad	hexa	znak
0	00_H	nevyužité
1	01_H	nevyužité
2	02_H	nevyužité
3	03_H	nevyužité
4	04_H	nevyužité
5	05_H	nevyužité
6	06_H	výpis
7	07_H	EDIT
8	08_H	kurzor vľavo
9	09_H	kurzor vpravo
10	$0A_H$	kurzor dole
11	$0B_H$	kurzor hore
12	$0C_H$	DELETE
13	$0D_H$	ENTER
14	$0E_H$	číslo
15	$0F_H$	GRAPHICS
16	10_H	atrament
17	11_H	podklad
18	12_H	blikanie
19	13_H	jas
20	14_H	inverzia
21	15_H	prepisovanie
22	16_H	pozícia
23	17_H	tabulátor
24	18_H	nevyužité
25	19_H	nevyužité
26	$1A_H$	nevyužité
27	$1B_H$	nevyužité
28	$1C_H$	nevyužité
29	$1D_H$	nevyužité
30	$1E_H$	nevyužité
31	$1F_H$	nevyužité
32	20_H	medzera
33	21_H	!

dekad	hexa	znak
34	22_H	"
35	23_H	#
36	24_H	$
37	25_H	%
38	26_H	&
39	27_H	`
40	28_H	(
41	29_H	)
42	$2A_H$	*
43	$2B_H$	+
44	$2C_H$	,
45	$2D_H$	-
46	$2E_H$	.
47	$2F_H$	/
48	30_H	0
49	31_H	1
50	32_H	2
51	33_H	3
52	34_H	4
53	35_H	5
54	36_H	6
55	37_H	7
56	38_H	8
57	39_H	9
58	$3A_H$	:
59	$3B_H$	;
60	$3C_H$	<
61	$3D_H$	=
62	$3E_H$	>
63	$3F_H$	?
64	40_H	@
65	41_H	A
66	42_H	B
67	43_H	C

dekad	hexa	znak
68	44_H	D
69	45_H	E
70	46_H	F
71	47_H	G
72	48_H	H
73	49_H	I
74	$4A_H$	J
75	$4B_H$	K
76	$4C_H$	L
77	$4D_H$	M
78	$4E_H$	N
79	$4F_H$	O
80	50_H	P
81	51_H	Q
82	52_H	R
83	53_H	S
84	54_H	T
85	55_H	U
86	56_H	V
87	57_H	W
88	58_H	X
89	59_H	Y
90	$5A_H$	Z
91	$5B_H$	[
92	$5C_H$	\
93	$5D_H$	]
94	$5E_H$	^
95	$5F_H$	_
96	60_H	£
97	61_H	a
98	62_H	b
99	63_H	c
100	64_H	d
101	65_H	e
102	66_H	f
103	67_H	g
104	68_H	h
105	69_H	I
106	$6A_H$	j
107	$6B_H$	k

dekad	hexa	znak
108	$6C_H$	l
109	$6D_H$	m
110	$6E_H$	n
111	$6F_H$	o
112	70_H	p
113	71_H	q
114	72_H	r
115	73_H	s
116	74_H	t
117	75_H	u
118	76_H	v
119	77_H	w
120	78_H	x
121	79_H	y
122	$7A_H$	z
123	$7B_H$	{
124	$7C_H$	|
125	$7D_H$	}
126	$7E_H$	~
127	$7F_H$	©
128	80_H	
129	81_H	
130	82_H	
131	83_H	
132	84_H	
133	85_H	
134	86_H	
135	87_H	
136	88_H	
137	89_H	
138	$8A_H$	

dekad	hexa	znak
139	$8B_H$	
140	$8C_H$	
141	$8D_H$	
142	$8E_H$	
143	$8F_H$	
144	90_H	(A) UDG
145	91_H	(B) UDG
146	92_H	(C) UDG
147	93_H	(D) UDG
148	94_H	(E) UDG
149	95_H	(F) UDG
150	96_H	(G) UDG
151	97_H	(H) UDG
152	98_H	(I) UDG
153	99_H	(J) UDG
154	$9A_H$	(K) UDG
155	$9B_H$	(L) UDG
156	$9C_H$	(M) UDG
157	$9D_H$	(N) UDG
158	$9E_H$	(O) UDG
159	$9F_H$	(P) UDG
160	$A0_H$	(Q) UDG
161	$A1_H$	(R) UDG
162	$A2_H$	(S) UDG
163	$A3_H$	(T) UDG
164	$A4_H$	(U) UDG
165	$A5_H$	RND
166	$A6_H$	INKEY$
167	$A7_H$	PI
168	$A8_H$	FN
169	$A9_H$	POINT
170	AA_H	SCREEN$
171	AB_H	ATTR
172	AC_H	ATTR
173	AD_H	TAB
174	AE_H	VAL$

dekad	hexa	znak
175	AF_H	CODE
176	$B0_H$	VAL$
177	$B1_H$	LEN
178	$B2_H$	SIN
179	$B3_H$	COS
180	$B4_H$	TAN
181	$B5_H$	ASN
182	$B6_H$	ACS
183	$B7_H$	ATN
184	$B8_H$	LN
185	$B9_H$	EXP
186	BA_H	INT
187	BB_H	SQR
188	BC_H	SGN
189	BD_H	ABS
190	BE_H	PEEK
191	BF_H	IN
192	$C0_H$	USR
193	$C1_H$	STR$
194	$C2_H$	CHR$
195	$C3_H$	NOT
196	$C4_H$	BIN
197	$C5_H$	OR
198	$C6_H$	AND
199	$C7_H$	<=
200	$C8_H$	>=
201	$C9_H$	<>
202	CA_H	LINE
203	CB_H	THEN
204	CC_H	TO
205	CD_H	STEP
206	CE_H	DEF FN
207	CF_H	CAT
208	$D0_H$	FORMAT
209	$D1_H$	MOVE
210	$D2_H$	ERASE
211	$D3_H$	OPEN #
212	$D4_H$	CLOSE #
213	$D5_H$	MERGE
214	$D6_H$	VERIFY

dekad	hexa	znak
215	$D7_H$	BEEP
216	$D8_H$	CRICLE
217	$D9_H$	INK
218	DA_H	PAPER
219	DB_H	FLASH
220	DC_H	BRIGHT
221	DD_H	INVERSE
222	DE_H	OVER
223	DF_H	OUT
224	$E0_H$	LPRINT
225	$E1_H$	LLIST
226	$E2_H$	STOP
227	$E3_H$	READ
228	$E4_H$	DATA
229	$E5_H$	RESTORE
230	$E6_H$	NEW
231	$E7_H$	BORDER
232	$E8_H$	CONTINUE
233	$E9_H$	DIM
234	EA_H	REM
235	EB_H	FOR

dekad	hexa	znak
236	EC_H	GO TO
237	ED_H	GO SUB
238	EE_H	INPUT
239	EF_H	LOAD
240	$F0_H$	LIST
241	$F1_H$	LET
242	$F2_H$	PAUSE
243	$F3_H$	NEXT
244	$F4_H$	POKE
245	$F5_H$	PRINT
246	$F6_H$	PLOT
247	$F7_H$	RUN
248	$F8_H$	SAVE
249	$F9_H$	RANDOMIZE
250	FA_H	IF
251	FB_H	CLS
252	FC_H	DRAW
253	FD_H	CLEAR
254	FE_H	RETURN
255	FF_H	COPY

Príloha D: Prehľad systémových premenných

Písmená v stĺpci **Pozn.** majú nasledujúci výraz:

X - pokiaľ zmeníte obsah tejto premennej, systém sa môže zrútiť
N - zmenou hodnoty tejto premennej nezískate žiaden efekt.

Číslice v stĺpci **Pozn.** udávajú veľkosť premennej v bytoch v stĺpci **Obsah** je uvedená hodnota premennej po zapnutí počítača. Pokiaľ nie je hodnota uvedená, nemá táto premenná po zapnutí rozumný obsah.

DB b - označuje obsadenia pamäťového miesta jednobytovou hodnotou **b**, ktorá môže nadobúdať hodnoty 0 až 255.

DW w - označuje obsadenie dvoch pamäťových miest dvojbytovou hodnotou **w**, ktorá môže nadobúdať hodnoty 0 až 65535.

DS s - označuje obsadenie **s** pamäťových miest.

Pozn	Adresa		Meno	Obsah	Význam
N8	$5C00_H$	23552	KSTATE	DS 8	miesto je využívané pri čítaní kláves
N1	$5C08_H$	23560	LAST_K	DB 0	uložený kód posledného stlačeného klávesu
1	$5C09_H$	23561	REPDEL	DB 35	čas (v 1/50 s), po ktorom sa začne kláves opakovať
1	$5C0A_H$	23562	REPPER	DB 5	čas (v 1/50 s), medzi opakovaním kláves
N2	$5C0B_H$	23563	DEFADD	DW 0	adresa argumentu užívateľskej funkcie
N1	$5C0D_H$	23565	K_DATA	DB 0	druhý byte riadenia farieb (z klávesnice)
N2	$5C0F_H$	23566	TVDATA	DW 0	byty farieb, AT a TAB (na obrazovku)
X38	$5C10_H$	23568	STRMS	DS 38	adresy kanálov

Pozn	Adresa		Meno	Obsah	Význam
2	$5C36_H$	23606	CHARS	DW $3C00_H$	adresa generátora znakov, zmenšená o 256
1	$5C38_H$	23608	RASP	DB 64	dĺžka varovného bzučiaka
1	$5C39_H$	23609	PIP	DB 0	dĺžka pípnutia po stlačení klávesu
1	$5C3A_H$	23610	ERR_NR	DB 255	o jedna menej než číslo hlásenia (chyby)
X1	$5C3B_H$	23611	FLAGGS	DS 1	rôzne príznaky pre riadenie systému BASIC
X2	$5C3D_H$	23613	ERR_SP	DW $FF54_H$	adresa položky zásobníka, v ktorej je uložená adresa programu na obsluhu chyby
N2	$5C3F_H$	23615	LIST_SP	DW $FF54_H$	adresa položky zásobníka, v ktorej je uložená adresa programu na obsluhu návratu z automatického listingu
N1	$5C41_H$	23617	MODE	DB 0	určuje režim kurzora (K,L,C,E,G)
2	$5C42_H$	23618	NEWPPC	DS 2	číslo riadku, na ktorý sa má skočiť príkazom GO TO
1	$5C44_H$	23620	NSPPC	DS 2	číslo príkazu v riadku, určenom NEWPPC
2	$5C45_H$	23621	PPC	DS 2	číslo práve vykonávaného riadku
1	$5C47_H$	23623	SUBPPC	DS 1	číslo práve vykonávaného príkazu
1	$5C48_H$	23624	BORDER	DB 56	farba pozadia*8 + atribúty spodnej časti obrazovky
2	$5C49_H$	23625	E_PPC	DS 2	číslo riadku, na ktorom je nastavený kurzor
X2	$5C4B_H$	23627	VARS	DW $5CCB_H$	adresa začiatku premenných
N2	$5C4D_H$	23629	DEST	DS 2	adresa premennej, do ktorej sa priraďuje
X2	$5C4F_H$	23631	CHANS	DW $5CB6_H$	adresa začiatku informácii o kanáloch
X2	$5C51_H$	23633	CURCHL	DW $5CBB_H$	adresa kanála pre vstup/výstup
X2	$5C53_H$	23635	PROG	DW $5CCB_H$	adresa začiatku programu v BASICu
X2	$5C55_H$	23637	NXTLIN	DS 2	adresa ďalšieho riadku v programe
X2	$5C57_H$	23639	DATADD	DS 2	adresa naposledy čítanej položky príkazu DATA
X2	$5C59_H$	23641	E_LINE	DW $5CCC_H$	adresa práve napísaného príkazu
2	$5C5B_H$	23643	K_CUR	DS 2	adresa kurzora
X2	$5C5D_H$	23645	CH_ADD	DS 2	adresa znaku v programe, ktorý sa bude práve interpretovať
2	$5C5F_H$	23647	X_PTR	DS 2	adresa výskytu syntaktickej chyby
X2	$5C61_H$	23649	WORKSP	DW $5D08_H$	adresa dočasnej pracovnej pamäti
X2	$5C63_H$	23651	STKBOT	DW $5D08_H$	adresa dna zásobníka kalkulátora
X2	$5C65_H$	23653	STKEND	DW $5D08_H$	adresa vrcholu zásobníka a začiatok voľnej pamäti
N1	$5C67_H$	23655	BREG	DS 1	register B kalkulátora
N2	$5C68_H$	23656	MEM	DW $5C92_H$	adresa pamäti pre kalkulátor
1	$5C6A_H$	23658	FLAGS2	DS 1	ďalšie príznaky pre prácu systému
X1	$5C6B_H$	23659	DF_SZ	DB 2	počet riadkov dolnej časti obrazovky
2	$5C6C_H$	23660	S_TOP	DS 2	počet riadkov vypisovaných pri automatickom výpise programu
2	$5C6E_H$	23662	OLDPPC	DS 2	číslo riadku, na ktorý skočí CONTINUE
1	$5C70_H$	23664	OSPPC	DS 1	číslo príkazu, na ktorý skočí CONTINUE
N1	$5C71_H$	23665	FLAGX	DS 1	rôzne príznaky pre prácu systému
N2	$5C72_H$	23666	STRLEN	DS 2	dĺžka práve vyhodnocovaného reťazca
N2	$5C74_H$	23668	T_ADDR	DS 2	adresa ďalšej položky v tabuľke syntaxe
2	$5C76_H$	23670	SEED	DW 0	inicializácia generátora náhodných čísiel
3	$5C78_H$	23672	FRAMES	DS 3	čítač prerušení, je zväčšený o jedna pri každom povolenom prerušení (každú 1/50 s)
2	$5C7B_H$	23675	UDG	DW $FF58_H$	adresa prvého znaku udg
1	$5C7D_H$	23677	COORDS	DS 1	súradnica x naposledy vykresleného bodu

Pozn	Adresa		Meno	Obsah	Význam
1	$5C7E_H$	23678		DS 1	súradnica y naposledy vykresleného bodu
1	$5C7F_H$	23679	P_POSN	DS 1	pozícia výpisu na ZX Printer
2	$5C80_H$	23680	PR_CC	DS 2	používa program pre tlač na tlačiareň
2	$5C82_H$	23682	ECHO_E	DS 2	adresa v editačnej časti obrazovky, za ktorú už nie je možné posunúť kurzor
2	$5C84_H$	23684	DF_CC	DS 2	adresa bytu obrazovky, na ktorú sa vypíše ďalší znak príkazom PRINT
2	$5C86_H$	23686	DF_CCL	DS 2	podobne ako DF CC, ale pre spodnú časť obrazovky
X1	$5C88_H$	23688	S_POSN	DS 1	pozícia stĺpca pre výstup znaku príkazom PRINT
X1	$5C89_H$	23689		DS 1	pozícia riadku pre výstup znaku príkazom PRINT
X2	$5C8A_H$	23690	S_POSNL	DS 2	ako S POSN, ale pre editačnú časť obrazovky
1	$5C8C_H$	23692	SCR_CT	DB 23	počet riadkov + 1, po vypísaní ktorých sa výpis zastaví
1	$5C8D_H$	23693	ATTR_P	DB 56	nastavené farby
1	$5C8E_H$	23694	MASK_P	DB 0	príznak transparentných farieb
N1	$5C8F_H$	23695	ATTR_T	DB 56	dočasne platné farby
N1	$5C90_H$	23696	MASK_T	DB 0	ako MASK P, ale dočasne
1	$5C91_H$	23697	P_FLAG	DS 1	ďalšie príznaky pre prácu systému
N30	$5C92_H$	23698	MEMBOT	DS 30	oblasť vyhradená pre kalkulátora
2	$5CB0_H$	23728	NMI	DW 0	adresa obslužného programu pre NMI
2	$5CB2_H$	23730	RAMTOP	DW $FF57_H$	adresa posledného bytu pamäti využiteľnej pre BASIC
2	$5CB4_H$	23732	P_RAMT	DW $FFFF_H$	adresa posledného bytu fyzickej pamäti

Príloha E: Strojový kód procesora Z80

Inštrukcie mikroprocesora Z-80 môžu byť jednobytové až štvorbytové a do dĺžky inštrukcie sa počíta tiež dĺžka operandu. V kódovaní inštrukcií sa používajú 4 prefixy (prvý byte inštrukcie). Tvar inštrukcie pre prefix $CB_H = 203_D$ a $ED_H = 237_D$ je uvedený v tabuľke. Teda číslo $7A_H$ vyjadruje inštrukciu LD A,D. Ak však toto číslo nasleduje po čísle CB_H (v pamäti sú uložené po sebe číslo CB_H a $7A_H$), potom vyjadruje inštrukciu BIT 7,D. Po čísle ED_H (v pamäti sú po sebe uložené čísla ED_H a $7A_H$) vyjadruje inštrukciu ADC HL, SP.

Prefix $DD_H = 221_D$ určuje, že inštrukcia pracuje s registrom IX, prefix $FD_H = 253_D$ určuje prácu s registrom IY. Teda číslo $23_H = 35_D$ vyjadruje inštrukciu INC HL. Ak však toto číslo nasleduje po prefixe DD_H, potom vyjadruje inštrukciu INC IX. Obdobne úvahu platí i pre prefix inštrukcie IY. Prefix DD_H alebo FD_H môže byť tiež pred prefixom CB_H alebo ED_H a inštrukcia opäť pracuje s registrami IX alebo IY.

V tabuľke sú zoradené inštrukcie mikroprocesora Z-80 vzostupne podľa ich kódov. V stĺpci 'dekad' je uvedený dekadický kód inštrukcie, v stĺpci 'hexa' šestnástkový kód inštrukcie, v stĺpci 'inštrukcia' je uvedený mnemonický kód jednobytovej inštrukcie a v stĺpcoch 'prefix CB_H' a 'prefix ED_H' mnemonických kód týchto inštrukcií. Ďalej používame v tabuľke písmeno N pre označenie jednobytového operandu a číslo NN pre označenie dvojbytového operandu.

dekad	hexa	inštrukcia	prefix CB_H	prefix ED_H
0	00	NOP	RLC B	
1	01	LD BC, NN	RLC C	
2	02	LD (BC),A	RLC D	
3	03	INC BC	RLC E	
4	04	INC B	RLC H	
5	05	DEC B	RLC L	
6	06	LD B,N	RLC (HL)	
7	07	RLCA	RLC A	
8	08	EX AF, AF'	RRC B	
9	09	ADD HL,BC	RRC C	
10	0A	LD A,(BC)	RRC D	
11	0B	DEC BC	RRC E	
12	0C	INC C	RRC H	
13	0D	DEC C	RRC I	
14	0E	LD C,N	RRC (HL)	
15	0F	RRCA	RRC A	
16	10	DJNZ N	RL B	
17	11	LD DE,NN	RL C	
18	12	LD (DE),A	RL D	

dekad	hexa	inštrukcia	prefix CB_H	prefix ED_H
19	13	INC DE	RL E	
20	14	INC D	RL H	
21	15	DEC D	RL L	
22	16	LD D,N	RL (HL)	
23	17	RLA	RL A	
24	18	JR N	RR B	
25	19	ADD HL,DE	RR C	
26	1A	LD A,(DE)	RR D	
27	1B	DEC DE	RR E	
28	1C	INC E	RR H	
29	1D	DEC E	RR L	
30	1E	LD E,N	RR (HL)	
31	1F	RRA	RR A	
32	20	JR NZ,N	SLA B	
33	21	LD HL,NN	SLA C	
34	22	LD (NN),HL	SLA D	
35	23	INC HL	SLA E	
36	24	INC H	SLA H	
37	25	DEC H	SLA L	
38	26	LD H,N	SLA (HL)	
39	27	DAA	SLA A	
40	28	JR Z,N	SRA B	
41	29	ADD HL,HL	SRA C	
42	2A	LD HL,(NN)	SRA D	
43	2B	DEC HL	SRA E	
44	2C	INC L	SRA H	
45	2D	DEC L	SRA L	
46	2E	LD L,N	SRA (HL)	
47	2F	CPL	SRA A	
48	30	JR NC,N		
49	31	LD SP,NN		
50	32	LD (NN),A		
51	33	INC SP		
52	34	INC (HL)		
53	35	DEC (HL)		
54	36	LD (HL),N		
55	37	SCF		
56	38	JR C,N		
57	39	ADD HL,SP		
58	3A	LD A,(NN)		
59	3B	DEC SP		
60	3C	INC A		
61	3D	DEC A		
62	3E	LD A,N		
63	3F	CCF		
64	40	LD B,B	BIT 0,B	IN B,(C)
65	41	LD B,C	BIT 0,C	OUT (C),B
66	42	LD B,D	BIT 0,D	SBC HL,BC
67	43	LD B,E	BIT 0,E	LD (NN),BC
68	44	LD B,H	BIT 0,H	NEG
69	45	LD B,L	BIT 0,L	RETN
70	46	LD B,(HL)	BIT 0,(HL)	IM 0
71	47	LD B,A	BIT 0,A	LD I,A
72	48	LD C,B	BIT 1,B	IN C,(C)
73	49	LD C,C	BIT 1,C	OUT (C),C
74	4A	LD C,D	BIT 1,D	ADD HL,BC
75	4B	LD C,E	BIT 1,E	LD BC,(NN)
76	4C	LD C,H	BIT 1,H	
77	4D	LD C,L	BIT 1,L	RETI
78	4E	LD C,(HL)	BIT 1,(HL)	
79	4F	LD C,A	BIT 1,A	LD R,A
80	50	LD D,B	BIT 2,B	IN D,(C)
81	51	LD D,C	BIT 2,C	OUT (C),D
82	52	LD D,D	BIT 2,D	SBC HL,DE
83	53	LD D,E	BIT 2,E	LD (NN),DE
84	54	LD D,H	BIT 2,H	
85	55	LD D,L	BIT 2,L	
86	56	LD D,(HL)	BIT 2,(HL)	IM 1
87	57	LD D,A	BIT 2,A	LD A,I
88	58	LD E,B	BIT 3,B	IN E,(C)
89	59	LD E,C	BIT 3,C	OUT (C),E
90	5A	LD E,D	BIT 3,D	ADC HL,DE
91	5B	LD E,E	BIT 3,E	LD DE,(NN)
92	5C	LD E,H	BIT 3,H	
93	5D	LD E,L	BIT 3,L	
94	5E	LD E,(HL)	BIT 3,(HL)	IM 2
95	5F	LD E,A	BIT 3,A	LD A,R
96	60	LD H,B	BIT 4,B	IN H,(C)
97	61	LD H,C	BIT 4,C	OUT (C),H
98	62	LD H,D	BIT 4,D	SBC HL,HL
99	63	LD H,E	BIT 4,E	LD (NN),HL
100	64	LD H,H	BIT 4,H	
101	65	LD H,L	BIT 4,L	
102	66	LD H,(HL)	BIT 4,(HL)	
103	67	LD H,A	BIT 4,A	RRD
104	68	LD L,B	BIT 5,B	IN L,(C)
105	69	LD L,C	BIT 5,C	OUT (C),L
106	6A	LD L,D	BIT 5,D	ADC HL,HL
107	6B	LD L,E	BIT 5,E	LD (HL),NN
108	6C	LD L,H	BIT 5,H	
109	6D	LD L,L	BIT 5,L	
110	6E	LD L,(HL)	BIT 5,(HL)	
111	6F	LD L,A	BIT 5,A	RLD
112	70	LD (HL),B	BIT 6,B	IN R,(C)

dekad	hexa	inštrukcia	prefix CB_H	prefix ED_H
113	71	LD (HL),C	BIT 6,C	
114	72	LD (HL),D	BIT 6,D	SBC HL,SP
115	73	LD (HL),E	BIT 6,E	LD (NN),SP
116	74	LD (HL),H	BIT 6,H	
117	75	LD (HL),L	BIT 6,L	
118	76	HALT	BIT 6,(HL)	
119	77	LD (HL),A	BIT 6,A	
120	78	LD A,B	BIT 7,B	IN A,(C)
121	79	LD A,C	BIT 7,C	OUT (C),A
122	7A	LD A,D	BIT 7,D	ADC HL,SP
123	7B	LD A,E	BIT 7,E	LD SP,(NN)
124	7C	LD A,H	BIT 7,H	
125	7D	LD A,L	BIT 7,L	
126	7E	LD A,(HL)	BIT 7,(HL)	
127	7F	LD A,A	BIT 7,A	
128	80	ADD A,B	RES 0,B	
129	81	ADD A,C	RES 0,C	
130	82	ADD A,D	RES 0,D	
131	83	ADD A,E	RES 0,E	
132	84	ADD A,H	RES 0,H	
133	85	ADD A,L	RES 0,L	
134	86	ADD A,(HL)	RES 0,(HL)	
135	87	ADD A,A	RES 0,A	
136	88	ADC A,B	RES 1,B	
137	89	ADC A,C	RES 1,C	
138	8A	ADC A,D	RES 1,D	
139	8B	ADC A,E	RES 1,E	
140	8C	ADC A,H	RES 1,H	
141	8D	ADC A,L	RES 1,L	
142	8E	ADC A,(HL)	RES 1,(HL)	
143	8F	ADC A,A	RES 1,A	
144	90	SUB B	RES 2,B	
145	91	SUB C	RES 2,C	
146	92	SUB D	RES 2,D	
147	93	SUB E	RES 2,E	
148	94	SUB H	RES 2,H	
149	95	SUB L	RES 2,L	
150	96	SUB (HL)	RES 2,(HL)	
151	97	SUB A	RES 2,A	
152	98	SBC A,B	RES 3,B	
153	99	SBC A,C	RES 3,C	
154	9A	SBC A,D	RES 3,D	
155	9B	SBC A,E	RES 3,E	
156	9C	SBC A,H	RES 3,H	
157	9D	SBC A,L	RES 3,L	
158	9E	SBC A,(HL)	RES 3,(HL)	
159	9F	SBC A,A	RES 3,A	
160	A0	AND B	RES 4,B	LDI
161	A1	AND C	RES 4,C	CPI
162	A2	AND D	RES 4,D	INI
163	A3	AND E	RES 4,E	OUTI
164	A4	AND H	RES 4,H	
165	A5	AND L	RES 4,L	
166	A6	AND (HL)	RES 4,(HL)	
167	A7	AND A	RES 4,A	
168	A8	XOR B	RES 5,B	LDD
169	A9	XOR C	RES 5,C	CPD
170	AA	XOR D	RES 5,D	IND
171	AB	XOR E	RES 5,E	OUTD
172	AC	XOR H	RES 5,H	
173	AD	XOR L	RES 5,L	
174	AE	XOR (HL)	RES 5,(HL)	
175	AF	XOR A	RES 5,A	
176	B0	OR B	RES 6,B	LDIR
177	B1	OR C	RES 6,C	CPIR
178	B2	OR D	RES 6,D	INIR
179	B3	OR E	RES 6,E	OTIR
180	B4	OR H	RES 6,H	
181	B5	OR L	RES 6,L	
182	B6	OR (HL)	RES 6,(HL)	
183	B7	OR A	RES 6,A	
184	B8	CP B	RES 7,B	LDDR
185	B9	CP C	RES 7,C	CPDR
186	BA	CP D	RES 7,D	INDR
187	BB	CP E	RES 7,E	OTDR
188	BC	CP H	RES 7,H	
189	BD	CP L	RES 7,L	
190	BE	CP (HL)	RES 7,(HL)	
191	BF	CP A	RES 7,A	
192	C0	RET NZ	SET 0,B	
193	C1	POP BC	SET 0,C	
194	C2	JP NZ,NN	SET 0,D	
195	C3	JP NN	SET 0,E	
196	C4	CALL NZ,NN	SET 0,H	
197	C5	PUSH BC	SET 0,L	
198	C6	ADD A,N	SET 0,(HL)	
199	C7	RST 0	SET 0,A	
200	C8	RET Z	SET 1,B	
201	C9	RET	SET 1,C	
202	CA	JP Z,NN	SET 1,D	
203	CB	PREFIX	SET 1,E	
204	CC	CALL Z,NN	SET 1,H	
205	CD	CALL NN	SET 1,L	
206	CE	ADC A,N	SET 1,(HL)	

dekad	hexa	inštrukcia	prefix CB_H	prefix ED_H
207	CF	RST 8	SET 1,A	
208	D0	RET NC	SET 2,B	
209	D1	POP DE	SET 2,C	
210	D2	JP NC,NN	SET 2,D	
211	D3	OUT (N),A	SET 2,E	
212	D4	CALL NC,NN	SET 2,H	
213	D5	PUSH DE	SET 2,L	
214	D6	SUB N	SET 2,(HL)	
215	D7	RST 10h	SET 2,A	
216	D8	RET C	SET 3,B	
217	D9	EXX	SET 3,C	
218	DA	JP C,NN	SET 3,D	
219	DB	IN A,(N)	SET 3,E	
220	DC	CALL C,NN	SET 3,H	
221	DD	PREFIX IX	SET 3,L	
222	DE	SBC A,N	SET 3,(HL)	
223	DF	RST 18h	SET 3,A	
224	E0	RET PO	SET 4,B	
225	E1	POP HL	SET 4,C	
226	E2	JP PO,NN	SET 4,D	
227	E3	EX (SP),HL	SET 4,E	
228	E4	CALL PO,NN	SET 4,H	
229	E5	PUSH HL	SET 4,L	
230	E6	AND N	SET 4,(HL)	
231	E7	RST 20h	SET 4,A	

dekad	hexa	inštrukcia	prefix CB_H	prefix ED_H
232	E8	RET PE	SET 5,B	
233	E9	JP (HL)	SET 5,C	
234	EA	JP PE,NN	SET 5,D	
235	EB	EX DE,HL	SET 5,E	
236	EC	CALL PE,NN	SET 5,H	
237	ED	PREFIX	SET 5,L	
238	EE	XOR N	SET 5,(HL)	
239	EF	RST 28h	SET 5,A	
240	F0	RET P	SET 6,B	
241	F1	POP AF	SET 6,C	
242	F2	JP P,NN	SET 6,D	
243	F3	DI	SET 6,E	
244	F4	CALL P,NN	SET 6,H	
245	F5	PUSH AF	SET 6,L	
246	F6	OR N	SET 6,(HL)	
247	F7	RST 30h	SET 6,A	
248	F8	RET M	SET 7,B	
249	F9	LD SP,HL	SET 7,C	
250	FA	JP M,NN	SET 7,D	
251	FB	EI	SET 7,E	
252	FC	CALL M,NN	SET 7,H	
253	FD	PREFIX IY	SET 7,L	
254	FE	CP N	SET 7,(HL)	
255	FF	RST 38h	SET 7,A	

Príloha F: Vysvetlenie niektorých pojmov

Výpočtová technika je odbor, ktorý sa veľmi rýchle rozvíja. Väčšina odborných pojmov je anglických. To spôsobuje, že snahy o preklad do iného jazyka nesú nebezpečenstvo skreslenia a nedorozumenia. Medzi odborníkmi tento problém väčšinou nehrozí, pretože používajú vo svojej reči pôvodné anglické pojmy, ktoré maximálne nadobudnú podobu slangovú. Problém nastáva u verejnosti, ktorá je takisto nútená niektoré pojmy používať. Tu je nebezpečenstvo nedorozumenia značné, pretože obvykle nie je všeobecne známy presný obsah pojmu, ktorý sa používa. Preto je na nasledujúcich stránkach uvedený stručný slovníček pojmov použitých v tomto manuáli.

Nájdete tu i slangové výrazy. Sú uvedené zámerne, pretože užívateľov mikropočítačov kompatibilných so ZX-Spectrem je u nás niekoľko sto tisíc. Ak pripočítame ostatné typy počítačov, bude tento počet ešte vyšší. Ti všetci používajú tento slang. Umožňuje im krátke a jasné vyjadrenie pojmov, ktoré by bola inak nutné vysvetľovať dlhým opisom.

Ak je za heslom text v zátvorke, ide o približnú výslovnosť. ak sú v zátvorke slová, ktorých niektoré písmeno je veľké, ide o rozpísanie skratky (nie výslovnosť).

Do tohto prehľadu boli okrem pojmov, ktoré sa vyskytujú v tomto manuáli, zahrnuté tiež najčastejšie sa vyskytujúce pojmy, s ktorými sa možno stretnúť v programoch.

adresa

Číslo označujúce bunku pamäti, v ktorej je uložené informácia. Čísluje sa od nuly po jednotke vyššie. Adresa sa udáva obvykle v šestnástkovej (hexadecimálnej),dvojkovej (binárnej) alebo desiatkovej (dekadickej) číselnej sústave.

Používa sa tiež adresa portu

Najvyššia adresa, ktorú je u osembitového mikroprocesora s šestnásťbitovou adresovou zbernicou možné adresovať, je $FFFF_H$, t.j. 65535_D. Ak má počítač viac pamäti, je realizovaná ako stránkovaná pamäť.

alfanumerický

Označenie pre informáciu obsahujúce okrem číslic 0 - 9 tiež všetky abecedné znaky (písmená) a niektoré špeciálne symboly.

algoritmus

Obecný postup riešenia problému, ktorý vedie k cieľu v konečnom počte krokov.

aplikačný software

Programy vytvorené podľa požiadavky konkrétneho užívateľa alebo pre veľkú triedu typických úloh (textové editory, databanky, tabuľkové editory apod.).

architektúra

Výraz pre vyjadrenie štruktúry počítača s ohľadom na jeho zapojenie (veľkosť pamäti, šírka zbernice, typ procesora apod.). Používa sa tiež pri vyjadrení štruktúry zložitých IO (mikroprocesora,....).

ASCII (American Standard Code for Information Interchange)

Čítaj asky. Súbor alfanumerických znakov vo forme osembitového kódu. Jeho využitie umožňuje vzájomnú prenositeľnosť dát medzi rôznymi počítačmi na logickej úrovni.

BASIC (Beginners All-purpose Symbolic Instruction Code)

Programovací jazyk, ktorý vznikol v 60-tych rokoch v USA. Je dodávaný ako základné programové vybavenie mikropočítačov. Pretože nebol nikdy dôsledne štandardizovaný, existuje niekoľko desiatok rôznych dialektov.

binárna sústava viď dvojková sústava

bit skratka b (BInary digiT - binary didžit) - dvojková číslica

Je to základná jednotka informácie, ktorá nadobúda dva rôzne stavy. Obvykle hodnoty 0 a 1. Tiež sa používa označenie L (low - nízký) a H (high - vysoký) alebo FALSE - nepravda a TRUE - pravda.

bus (bas) - zbernice

Vodiče, ku ktorým sú paralelne pripojené jednotlivé funkčné celky mikropočítača (mikroprocesor, pamäti, I/O zariadenia). Podľa významu vodičov nájdeme v mikropočítači dátovú zbernicu (data bus), adresovú zbernicu (address bus) a riadiacu zbernicu (control bus). Podľa šírky dátovej zbernice (8 vodičov tzn. 8 bit) hovoríme, že Didaktik M je osembitový mikropočítač. Je to dané typom použitého mikroprocesora (Z-80).

buffer (bafr)

Označenie pre dočasnú vyrovnávaciu pamäť. Používa sa obvykle pre spoluprácu s pomalšími zariadeniami, napr. tlačiarňou. Pri spolupráci s tlačiarňou môže počítač dodávať jednotlivé znaky po bytoch a čakať, až ich tlačiareň vytlačí, alebo naplniť veľmi rýchlo vyrovnávaciu pamäť tlačiarne (buffer) dlhú napr. 4 kB. Tlačiareň si znaky postupne odoberá z tohto bufferu. Keď je buffer prázdny, oznámi to počítaču a ten tam môže v prípade potreby preniesť ďalšie údaje. V dobe, kde tlačiareň tlačí, sa počítač môže venovať inej činnosti. Buffer nemusí byť len súčasť periférneho zariadenia, ale i vlastného počítača.

byte skratka B (bajt) - slabika

Byte = 8 bitov. Jednotka bit je príliš malá, preto sa vo väčšine prípadov používa byte. Číslo 8 a jeho násobky sú pre svet výpočtovej techniky charakteristické.

delete (dylít) - škrtnúť

Označenie funkčného klávesu na vymazanie znaku.

display (dysplei)

Zariadenie pre viditeľné zobrazenie informácií na optoelektronickom princípe.

display file (fajl) - rad, súbor

Časť pamäti video RAM dlhá 6144 B (byte), ktorá slúži na uchovanie obrazovej informácie. Obraz sa skladá z bodov (pixelu), každý je buď zhasnutý (hodnota 0), alebo rozsvietený (hodnota 1). Pre vyjadrenie, či bod svieti, či nie, potrebujeme informáciu 1 bit. Pre osem bodov je to 1 byte. Display file obsahuje 256 × 192 bodov, teda (256 × 192) / 8 = 6144 byte.

drop out (drop aut)

Strata informácie spôsobená výpadkom, chybou. Na magnetofónovej páske sa takto označuje miesto, kde je nekvalitná magnetická vrstva, ktorá nie je schopná niesť informáciu.

dvojková sústava, dvojková číselná sústava

Číselná sústava so základom 2. Koeficienty jednotlivých rádov môžu nadobúdať len hodnoty 0 a 1.

dvojstavový

Majúci dva stavy. Dvojstavový signál - signál, ktorý nadobúda dve rôzne hodnoty (0,1; H,L; zapnuté, vypnuté).

edit, editovať, editácia

Proces modifikácie (zmeny) údajov alebo programu.

EPROM (Erasable Programmable Read Only Memory)

porovnaj RAM, ROM

Pamäť svojimi vlastnosťami podobná pamäti ROM. Rozdiel je v tom, že ju možno dodatočne naprogramovať a v prípade potreby opäť vymazať (najčastejšie UV žiarením). Programovanie nového obsahu pamäti sa vykonáva v špeciálnych prípravkoch (programátoroch pamäti).

epromka

slangové označenie pre pamäť EPROM.

error

Chyba, odchýlka, porucha.

false (fóls) - nepravdivý

fault (fólt) - porucha, závada

field (fíld) - pole, oblasť

file (fail) - súbor

Ucelená sústava dát (program), ktorá má svoju štruktúru.

fill (fil) - plniť

Plniť obecne. Zaplniť pamäť určitými údajmi, vyplniť obrázok nejakou štruktúrou (napr. rastrom, začierniť apod.).

floating point (flouting point) - plávajúca desatinná čiarka

Spôsob vyjadrenia čísla v počítači, ktoré je rozdelené na mantisu a exponent. Jazyk BASIC ukladá čísla taktiež v tomto tvare. Pre mantisu má vyhradené 4 byty a pre exponent 1 byte.

format (format) - rozmer, velikosť

Usporiadanie, veľkosť, rozmer. Používa sa v spojení formát zobrazenia, formát záznamu údajov na diskete apod.

full (ful) - plný

Označenie nedostatku miesta, zaplnenie. Disk is full, directory is full. Súčasť hlásenia rôznych programov.

game (geim) - hra

hard copy (hárd kopy)

Výraz pre tlač obsahu obrazovky na tlačiareň alebo režim paralelnej tlače všetkých údajov tak na obrazovku, ako aj na tlačiareň počítača.

hexadecimálny

Šestnástkový, vyjadrenie v šestnástkovej sústave.

hardware (hardvér) - doslova tvrdý tovar, technické vybavenie

porovnaj software

Je to všetko, na čo si možno "siahnuť". Počítač, všetky periférie ako tlačiarne, prevodníky, monitory, myši, floppy diskové jednotky apod. Nie sú to programy.

character (keriktr) - znak

Znak, symbol (napr. ASCII), ktoré sú použité v tlačiarni alebo počítači. Môže ich byť i viac. U Didaktiku M je znaková sada uložená v pamäti ROM. Adresu ich začiatku ukazuje systémová premenná CHARS. Ak vytvoríme novú sadu znakov (napr. ich nový tvar), možno ich umiestniť do pamäti a novú adresu začiatku uložiť do premennej CHARS.

check sum (ček sam) - kontrolná suma, kontrolný súčet

Číslo, ktoré sa používa na indikáciu správnosti nahraného súboru. Pre jeho výpočet sa používa rad algoritmov. Výsleodk býva číslo dlhé jeden až dva byty. Pretože prostý súčet bytov by bol vo väčšine prípadov väčší než číslo, ktoré možno zobraziť v jednom alebo dvoch bytoch, výsledok sa delí modulo FFh alebo FFFFh. Často je súčasťou nahrávaného súboru. Počítač si pri nahrávaní počíta vlastný check sum. Pri nesúhlase s hodnotou zaznamenanou v súboru ohlási chybu pri nahrávaní.

integrovaný obvod (skratka IO)

Elektronická súčiastka združujúca na čipe rad aktívnych a pasívnych prvkov. Ich počet môže byť podľa hustoty integrácie až niekoľko sto tisíc na ploche niekoľko štvorcových milimetroch. Bez porušenia ho nie je možné rozobrať.

inteligentná periféria, zariadenie

Označenie pre periférium (tlačiarne, prevodníky, analyzátory, ...), u ktorej riadenie jednotlivých častí na tej najnižšej hardwareovej úrovni nevykonáva mikropočítač, ale vlastná elektronika. Nezriedka taká periféria obsahuje vlastný mikropočítač, ktorý riadi jej činnosť. Taká periféria sa potom vyznačuje tým, že má rad módov činnosti, ktoré sa dajú programovať. Vonkajší mikropočítač, ku ktorému je taká periféria pripojená, hovorí, "čo sa má robiť", nie však, "ako sa to má robiť".

interface (interfejs) - rozhranie

1. Obvykle je myslené hardwareové rozhranie medzi počítačom a periférium.
2. V prenesenom význame môže byť chápané i ako softwareové rozhranie, napr. pre prenos parametrov medzi rôznymi navzájom spolupracujúcimi programami.

interpreter

porovnaj kompilátor, prekladač

Typ prekladača, ktorý vzhľadom k svojmu spôsobu práce umožňuje dialógový režim pri odladňovaní programu. Program nie je preložený raz pred spustením do strojového kódu ako u kompilátoru, ale

každý príkaz programovacieho jazyka je vykonávaný (interpretovaný) až v okamžiku, kedy má byť vykonaný. Z tohto dôvodu je tiež beh programu omnoho pomalší než u kompilátora. Ak vykonáme napr. cyklus, interpreter si odoberá jednotlivé príkazy a prekladá ich toľkokrát, koľkokrát sa cyklus vykonáva. Kompilátor taký cyklus (vlastne celý program) preloží len raz pred spustením programu. Interpreter naopak umožňuje program v ľubovoľnom bode zastaviť, vykonať zmeny a opäť ho z ľubovolného miesto spustiť. Ako interpretery sú písané prekladače jazyka BASIC, Forth, Lisp, Prolog, Logo apod. Mnohokrát sú spôsoby spracovania programu kombinované.

IO - viď integrovaný obvod

I/O zariadenie (Input/Output - Vstup/Výstup, tiež V/V)

Zariadenie, pomocou ktorého komunikuje mikropočítač so svojim okolím ako klávesnica, monitor, tlačiareň, magnetofón, apod. Je vidno, že niektoré zariadenie zabezpečujú komunikáciu len jedným smerom (klávesnica, tlačiareň) a niektoré obidvoma (magnetofón).

joystick (džojstyk)

Periférne zariadenie na komunikáciu programu s užívateľom. Má tvar škatuľky s krátkou pákou, ktorou možno pohybovať v štyroch smeroch: hore, dole, vľavo, vpravo. Okrem toho má ešte jedno alebo dve tlačítka. Umožňuje pohybovať kurzorom po obrazovke. Je veľmi obľúbený na ovládanie (hranie) počítačových hier. Jednotlivé objekty nie je treba ovládať klávesmi, ktoré obvykle v zápale hry značne trpia, ale ovládajú sa pohybmi joysticku. Pretože ovládanie je často dosť nevyberavé, je ich konštrukcia robustná.

key (ký) - kláves, kľúč

keyboard (kýbód) - klávesnica, tastatúra

keyword (kýwód) - kľúčové slovo

kilo skratka k

Predpona znamenajúca tisíc, v počítačovej terminológii 1024:

1 kB = 1024 bytov
1 kb = 1024 bitov

kľúčové slovo

Slovo s pevne definovaným významom. Býva najčastejšie v programovacom jazyku, ale tiež v aplikačnom programe. Napr. v jazyku BASIC Didaktiku M nie je možné použiť premennú FOR, pretože FOR je kľúčové slovo príkazu cyklu. Hovorí sa im tiež rezervované slová.

kompatibilita - zlučiteľnosť

kompilátor

porovnaj interpreter, prekladač

Typ prekladača. Prevádza zdrojový text programu do strojového kódu, ktorý je procesor schopný priamo vykonávať. Preklad sa prevedie raz pred spustením programu. Výhodou kompilovaných programov je ich väčšia rýchlosť oproti interpretovaným. Nevýhodou je väčšia doba potrebná pri ich odlaďovaní vzhľadom na to, že i pri nepatrnej zmene zdrojového textu je treba preložiť celý program. Ako kompilátor je riešená väčšina programovacích jazykov, ako Pascal, FORTRAN, COBOL, ADA, PL/1 apod.

komunikácia - vzájomná výmena informácií.

konfigurácia - zostava, zoskupenie, usporiadanie

Konfigurácia počítača - zahŕňa informácie o zostave počítač (typ a veľkosť pamäti, typ monitora, spôsob zobrazenia, počet floppy diskových mechaník, možnosť pripojenia prídavných pamätí apod.).

Konfigurácia pamäti - zahŕňa informácie o mape pamäti: veľkosť pamäti ROM, RAM, EPROM, čo je vyhradené pre systém, zobrazenie (VIDEO), užívateľa apod.

kopírák
Slangový výraz pre kopírovací program. Slúži na zhotovovanie kópií programov a údajov.

kurzor
Indikátor polohy znaku na obrazovke. Môže niesť informáciu o móde činnosti počítača (programu). Je realizovaný rôznymi symbolmi (písmenom, šípkou, podčiarknutím apod.). Kvôli zvýrazneniu obvykle bliká.

ladenie, ladenie programu
Označenie pre optimalizáciu zápisu programu. Pri ladení sa testuje program, hľadajú sa chyby. U rozsiahlych programových celkov zaberá ladenie prevažnú časť vývoja programu.

LED (Light Emitting Diode)
Medzinárodná skratka pre luminiscenčnú diódu. Našla širokú oblasť použitia ako zdroj viditeľného infračerveného žiarenia.

ledka
Slangový výraz pre diódy LED, ktoré sa používajú na indikáciu prevádzkových stavov rôznych zariadení.

letter (letr) - znak, písmeno

line (lajn) - riadok

listing
Akýkoľvek výpis dát na periférnom zariadení. Najčastejšie sa používa na označenie výpisu programu na tlačiarni.

loudovať program
Slangový výraz pre nahranie programu z magnetofónu do pamäti počítač príkazom LOAD.

machine code (mešin koud) - strojový kód

manual (menjuel) - príručka, ručný, manuálny

medium (midjem) - médium, prostriedok
Prostriedok na záznam údajov. Napr. disketa, magnetofónová kazeta, dierna páska apod.

mega skratka M
Predpona znamenajúca milión, v počítačovej terminológii je to 1024 × 1024:

1 MB = 1024 × 1024 bytov
1 Mb = 1024 × 1024 bitov

memory (memory) - pamäť
porovnaj RAM, ROM, EPROM

Pamäť slúži na prechodné alebo trvalé uchovanie informácie. Možno je rozdeliť na vnútornú (je súčasťou počítača) alebo vonkajšiu (je pripojená ako periférna jednotka) napr. magnetofón, floppy disk apod.

menu (menju) - ponuka, jedálniček
Súbor ponúkaných variánt, ponuka. Spôsob riešenia komunikácie programu s užívateľom. Voľba varianty sa vykonáva obvykle stlačením počiatočného alebo zvýrazneného písmena, prechodom kurzora (myšou, svetelným perom) na požadovanú voľbu apod.

mikroprocesor
Zložitý integrovaný obvod, ktorý tvorí srdce mikropočítača (riadi činnosť). Mikroprocesor je tvorený mnoho tisícmi tranzistormi. Jeho konkrétna činnosť je určená zapojením a programom v pamäti mikropočítač. So svojim okolím je spojený dátovou, adresovou a riadiacou zbernicou.

modify (modifai) - zmeniť, upraviť, modifikovať

monitor

Toto slovo má niekoľko rozdielnych významov.

1. displej s vákuovou elektrónkou (podobný TV prijímaču, ale s kvalitnejším obrazom).
2. základný program uložený v pamäti ROM, ktorý umožňuje komunikáciu s užívateľom. Obvykle zabezpečuje natiahnutie operačného systému, ktorý potom prevezme riadenie počítača.
3. tiež debugger, program na ladenie a oživovanie programu v strojovom kóde

mouse (maus) - myš

Periférne zariadenie, ktoré umožňuje programu komunikovať s užívateľom. Má tvar malej krabičky (myši) do dlane. Hlavnou súčasťou myši je guľôčka, ktorá sa otáča, keď s myšou pohybujeme po stole alebo inej podložke. Pohyb tejto guľôčky sa sníma a prenáša na pohyb kurzora na obrazovke. Práca s myšou je veľmi pohodlná a rýchla. Preto si získala značnú obľubu u profesionálnych počítačov.

naloudovať - viď loudovať

Slangový výraz

name (neim) - meno (programu, súboru)

nasejvovať - viď sejvovať

negácia

Zmena významu (hodnoty) na opačnú. O negácii má zmysel hovoriť tam, kde sú povolené dva rozdielne stavy.

Negácia signálu - zmena hodnoty signálu na opačný (z H na L a naopak.).

number (nambr) - číslo

numerický - číselný

odladiť - viď ladenie

open (oupn) - otvoriť

output (autput) - výstup (dát zo systému)

pamäť viď memory

Pascal

Štruktúrovaný programovací jazyk vhodný na riešenie väčšiny problémov. Jeho tvorcom je profesor Nicolas Wirth z Eidgenossiche Technische Hochschule v Zurichu. Tento jazyk napísal ako jazyk na výuku štruktúrovaného programovania. Kvôli svojim prednostiam si získal značnú obľubu.

periféria

Technické zariadenie, ktoré komunikuje s počítačom (tlačiareň, myš, joystick, prevodníky, magnetofón apod.). Počítač s perifériou je obvyklé prepojený interfaceom.

pixel (pixel)

Bod, najmenšia zobraziteľná jednotka na displeji alebo rastri.

plotter (plotr) - súradnicový zapisovač

point - bodka

pointer (pointer) - ukazovateľ

port - brána

Port znamená latinsky prístav. Vo výpočtovej technike pod pojmom port - brána je myslený fyzický obvod na pripojenie periférie. Port je určený svojou adresou. Mikroprocesor Z-80 umožňuje adresovať 65536 portov, na ktoré je možné poslať jednobytovú hodnotu (číslo) alebo ju naopak prečítať.

poukať, poukovať, napoukať, napoukovať

Slangový výraz na zmenu obsahu pamäti počítača využitím príkazu POKE. Obvykle sa používa v spojení s prepisom systémových premenných.

printer (printr) - tlačiareň

program

Postupnosť príkazov zodpovedajúcich syntaxi použitého jazyka, ktorá rieši zadaný problém.

programovací jazyk

Súbor príkazov s definovanou syntaxou slúžiaci na zápis programu. Napr. ADA, assembler, C, BASIC, FORTRAN, Forth, Lisp, Logo, Pascal, PROLOG ...

prekladač

Názov pre program, ktorý prevedie (preloží) program do formy zrozumiteľnej procesoru počítača. Podľa spôsobu práce ich delíme na dve veľké skupiny, kompilátory a interpretry.

QWERTY

Označenie anglickej normy pre rozmiestnenie klávesov na klávesnici. Názov pochádza z prvých šiestich klávesov hornej rady klávesnice. Nemecká norma má prehodené klávesy Y a Z. Táto klávesnica sa označuje ako QWERTZ.

RAM (Random Access Memory)

Pamäť umožňujúca opakovateľný zápis a čítanie informácie behom činnosti počítača. Nahrávajú sa do nej programy z vonkajšieho dátového média (magnetofón, floppy disk apod.).

ramka

Slangové označenie pre pamäť RAM.

random (rendom) - náhodný

range (reindž) - rozsah

read (ríd) - čítanie

real (rýl) - skutočný, ozajstný

U označenia čísiel sú myslené reálne čísla.

record (rykód) - záznam, zaznamenať

register

Pamäť malého rozsahu (obvykle 1 - 2 byty) na uchovanie špecifickej informácie. Programovateľné IO používajú registre na uchovanie dočasnej informácie. Podľa spôsobu použitia sa k nim pridáva rozlišujúce prídavné meno. Napr. stavový register, indexregister, register vektora prerušenia (interrupt register) apod.

report (rypót) - správa, oznam

restart (restart) - opakované spustenie programu

restore (rystó) - obnoviť

ROM (Read Only Memory)
Pamäť určená len na čítanie. Informáciu (program a údaje) do nej raz navždy zapíše výrobca pri výrobe pamäti a nie je možné ich už zmeniť. Obsahuje obvykle krátky program, ktorý umožňuje zaviesť do pamäti RAM počítača ľubovolný iný program z vonkajšieho dátového média. Mnoho mikropočítačov má v tejto pamäti umiestnený i interpreter jazyka BASIC.

romka
Slangové označenie pre pamäť ROM.

screen (skrín) - obrazovka

sejvovať program
Slangový výraz pre nahranie programu z pamäti počítača na magnetofónovú pásku príkazom SAVE.

software - (softvér - mäkký tovar) programové vybavenie, program
porovnaj hardware

Program je postupnosť príkazov, ktoré počítač vykonáva. Môže byť buď nedeliteľnou súčasťou počítača napr. uložený v pamäti ROM, alebo je nahrávaný do pamäti zvonka z kaziet, diskiet.

súbor - viď file

spadnutie systému
Slangový výraz prevzatý užívateľmi mikropočítačov zo slangu veľkých (sálových) počítačov. Znamená hrubú chybu v činnosti počítača (softwareovú alebo hardwareovú), ktorá obvykle vedie (u mikropočítačov) k resetu počítača. Slovom systém bol pôvodne myslený operačný systém počítač, neskôr samotný mikropočítač s programom.

stroják
Slangový výraz pre strojový kód.

"Píšem ve strojáku", nepresné tvrdenie, ktoré znamená: "Píšem programy v mnemonickom kóde (assembler) procesora".

strojový kód
Postupnosť jednotiek a núl, ktoré zapísané do pamäti počítač tvoria inštrukcie (program) pre počítač. Len tomuto tvaru príkazu rozumie mikroprocesor. Písanie programu v tomto tvare je veľmi obťažné. Preto sa takéto programu píšu v jazyku symbolických adries a do strojového kódu sú preložené assemblerom.

syntax
Súbor pravidiel prípustného zápisu (program, textu apod.)

systémový konektor
Miesto, kde je zakončená systémová zbernica. U Didaktiku M je to na zadnej strane počítača.

systémová zbernica
Miesto, kam sú vyvedené signály dátovej, adresovej a riadiacej zbernice.

terminál
Periférne zariadenie na vstup a výstup údajov. Obvykle obsahuje monitor (displej) a alfanumerickú klávesnicu.

textový editor
Program na písanie a editáciu textu (programu).

true (trú) - pravdivý

user (júzr) - užívateľ

Video RAM (skratka VRAM)
Pamäť RAM slúžiaca na uloženie informácií, ktorá sa premieta na obrazovke. Na zobrazenie sa využíva len oblasť s veľkosťou 6912 B (video file).

word (vód) - slovo

write (rait) - písať, zapísať, zaznamenať

záznamová rýchlosť
Rýchlosť záznamu dát na záznamové médium (magnetofónová kazeta, disketa apod.). Je dané kvalitou záznamového média, spôsobom záznamu (kódovania) a zariadením, na ktorom je realizované. U magnetofónu sa podľa typu počítača pohybuje zhruba v rozmedzí od 2 kB do 20 kB za minútu. U Didaktiku M sa pohybuje zhruba okolo 11 kB za minútu. Zvyšovanie rýchlosti záznamu u magnetofónu je limitované spoľahlivosťou záznamu a jeho citlivosťou na iných magnetofónoch, ktorých parametre sú mierne odlišné od toho, na ktorom bol záznam zaznamenaný.

zbernica viď bus

zdroják
Slangový výraz pre zdrojový text.

zdrojový text
Označenie pre výpis programu, ktorý zodpovedá syntaxi zápisu daného programovacieho jazyka, ale nie je v tvare, ktorému rozumie daný počítač. Do tohto tvaru ho je treba preložiť kompilátorom. Existuje teda zdrojový text programu v mnemonike mikroprocesora Z-80 alebo jazyka Pascal, C apod. Až po preložení získame zo zdrojového textu spustiteľný program.

zrútenie počítača, zrútenie systému
Slangový výraz vyjadrujúci hrubú chybu v programe, alebo zlyhanie technických prostriedkov, ktoré spôsobili nefunkčnosť počítača. Dá sa vždy odstrániť resetom počítača alebo jeho vypnutím a opätovným zapnutím. K zrúteniu môže dôjsť i vplyvom technickej závady, napr. pri neodbornom pripojovaní periférií na konektory apod. Tu však už môže dôjsť k trvalému poškodeniu počítača.

Z-80
Označenie osembitového mikroprocesora firmy Zilog (čítaj zajlog). Pre svoje vlastnosti sa stal neoficiálne uznávaným štandardom a vo svete celkom nahradil mikroprocesor I 8080 firmy Intel (u nás vyrábaný pod označením MHB 8080).

Príloha G: Doporučená literatúra

[3] Bernard,J.M.Hugon,J.-Le Corvec,R.: Od logických obvodů k mikroprocesorům, SNTL 1988

[4] Dědina,B.- Valášek,P.: Mikroprocesory a mikropočítače, Knižnice výpočetní techniky SNTL, Praha 1983.

[5] Dlabola,F.-Starý J.: Systémy s mikroprocesory a přenos dat, NADAS 1986.

[6] Jenne,D.: Kompletní výpis ZX ROM, Zenitcentrum Beroun, 1989.

[7] Jinoch,J.-Muller,K.-Vogel.J.: Programování v jazyku Pascal, SNTL 1988

[8] Kokeš,J.: Přenos a zpracování informací v mikropočítačích, NADAS 1986.

[9] Kroka,P.-Slavík,P.: BASIC pro začátečníky, SNTL 1988

[10] Machačka,I.-Pavlů,J.: Programování v jazyku basic, S NTL 1987

[11] Molnar,L.: Programování v jazyku Pascal, Alfa 1989

[12] Olehla,J.-Dauth,M.: BASIC u mikropočítačů, NADAS 1988

[13] Sobotka,Z.: Otázky a odpovede z mikroprocesorov a mikropočítačov - aplikácie, Alfa 1988

[14] Starý,J.: Mikropočítač a jeho programování, SNTL 1988

[15] Šritter,J.-Dauth,M.: Rutiny ROM ZX Spectrum, členskáknihovna 602. ZO Svazarmu 1987

[16] Twiehaus,J.: Kľúč k počítaču - Programové vybavenie, Alfa 1988

[17] Vejmola,S.: Hry s počítačem, SPN 1988.

[18] Zajíček,L.: Bity do bytu - základy programování ve strojovém kódu - assembleru Z-80, Mladá Fronta 1988

[19] Literatura z organizací:
ZenitCentrum Beroun, Hostímská 1, 266 01 Beroun.
602.ZO Svazarmu, Dr. Z. Wintra 8, 160 41 Praha 6.