PERSONÁLNY POČÍTAČ PP 01

SOU dopravní Holice v Č.

PRÍRUČKA UŽÍVATEĽA
811 282 46 011

ZVT

PERSONÁLNY POČÍTAČ PP 01

PRÍRUČKA UŽÍVATEĽA
811 282 46 011

ZVT

PRÍRUČKA UŽÍVATEĽA

811 282 46 011

Obsah

ÚVOD

Príručka, ktorú ste práve otvorili, Vám bude na začiatku učiteľom, neskôr poradcom pri práci s PP01 (personálnym počítačom). Príručka je písaná tak, aby umožnila každému, kto si sadne k PP01, s ním okamžite pracovať (bez ohľadu na jeho vzdelanie alebo predchádzajúce skúsenosti s počítačmi).

Celá orientácia príručky je venovaná tomuto cieľu. Už v 1. kapitole obdrží používateľ všetky potrebné informácie, aby uviedol PP01 do chodu v najjednoduchšej konfigurácii a mohol na ňom začať pracovať. Pre rýchle splnenie tohoto cieľa nie je v tejto kapitole kompletný technický popis jednotlivých častí, ktorý bude používateľa určite v ďalšom štádiu zaujímať, ale len základné informácie potrebné pre uvedenie zariadenia do činnosti.

V druhej kapitole sú uvedené základné informácie pre ovládanie PP01 v priamom režime a v tretej kapitole už jednoduché programy.

Prvé tri kapitoly bez nároku na úplnosť informácii dávajú dostatok podkladov pre písanie jednoduchých programov.

Štvrtá kapitola naučí používateľa pracovať s magnetofónom ako vonkajšou pamäťou PP01 a nahrávať a spúšťať programy z kazety.

V piatej kapitole, ktorá už pôjde viac do hĺbky, sa oboznámime s klávesnicou, ovládacími a indikačnými prvkami PP01. Podrobný popis niektorých ovládacích a indikačných prvkov však v tomto štádiu znalostí presahuje možnosti začiatočníka. Preto sú v kapitole 5 uvedené odkazy na ďalšie kapitoly a prílohy.

Kapitola 6 už podrobne rozoberá aritmetické a logické výrazy, prvky týchto výrazov a matematické funkcie. Všetky nové znalosti sú priebežne demonštrované na príkladoch.

Siedma až deviata kapitola podrobne rozoberá jednotlivé príkazy GBASICu, podprogramy, polia a reťazce.

Grafické príkazy spolu s príkladmi, v ktorých sa grafika využíva sú uvedené v kapitole 10.

Kapitola 11 popisuje strojovo orientované príkazy a funkcie.

1. UVEDENIE DO CHODU

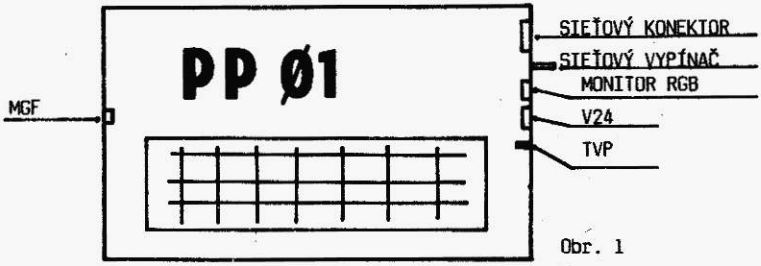
Personálny počítač 01 je kompaktný konštrukčný celok, ktorý zahrňuje:

- vlastný personálny počítač
- zdroj so všetkými potrebnými napätiami
- klávesnicu
- ovládacie a indikačné prvky
- konektory pre pripojenie ďalších zariadení

V minimálnej konfigurácii potrebuje pre svoju prácu jediné ďalšie zariadenie - televízor, ktorý slúži pre zobrazovanie informácií, ktoré vstupujú do počítača a informácií, ktoré počítač posiela používateľovi. Výhodou PP01 oproti niektorým iným čs. personálnym počítačom je skutočnosť, že ako zobrazovaciu jednotku možete použiť ľubovoľný televízor (teda aj ten, ktorý máte doma, za predpokladu, že Vám to manželka, rodičia, či deti pri ostrej konkurencii programov čs. televízie dovolia).

1.1. Pripojenie napájania

```
**********
* POZOR ! ************************************************
* Pri akejkoľvek manipulácii, pri pripájaní na sieť, pri *
* pripojovaní ďalších zariadení sa presvedčte, že PP01 je *
* vypnutý.                                               *
**********************************************************
```



Obr. 1

PP01 je vypnutý, ak sieťový vypínač na pravej strane je v polohe dole (pozri obr.1). Sieťovú šnúru zapojíme do sieťového konektora (obr.1.) a potom do zásuvky 220 V.

1.2. Pripojenie televízora

Bežný televízor pripojíme k PP01 pomocou kábla pripojenia televízora. Jednotlivé káble sa možu líšiť podľa typu televízora (typ anténnej zdierky televízora). Kábel zasunieme do zdierky "TVP" v PP01 a do anténnych zdierok pre 1. televízne pásmo televízora. Zapneme televízor, stiahneme zvuk a naladíme televízor na 4. kanál. Na obrazovke televízora vidíme škvrny pripomínajúce husté sneženie.

1.3. Zapnutie počítača

Sieťový vypínač prepneme smerom hore (do polohy zapnuté). Na televízore sa v ľavom hornom rohu objaví nápis:

```
GBASIC Vn.m                kde n.m je číslo verzie interpretu
READY
```

Ovládacími prvkami televízora - ladenie, jas, kontrast dostavíme televízor tak, aby bol výpis čo najzretelnejší.

Ak máte v ľavom hornom rohu televízora uvedený výpis, možete si gratulovať, prvú úlohu ste zvládli na výbornú.

Počítač je READY, tzn. pripravený pre prácu. Túto skutočnosť nám bude výpisom READY oznamovať veľmi často.

Pozrime sa na televízor.

Pod výpisom READY vidíme blikajúci štvorček. Tomuto znaku, ktorý ukazuje, na ktoré miesto na obrazovke sa bude zapisovať, hovoríme ukazovátko (KURZOR).

1.4. Klávesnica

Klávesnica počítača je rozdelená na dve časti: štandardnú alfanumerickú časť podobnú klávesnici písacieho stroja a numerickú časť (vpravo). V tejto kapitole budeme používať len časť alfanumerickú (v strede počítača). Podobne ako u písacieho stroja možu mať niektoré klávesy dva významy. Jednotlivé klávesy budeme v tejto príručke znázorňovať medzi dvomi znakmi ! - !A! znamená "klávesa A". Po zatlačení klávesy počítač vydá zvukové znamenie ("pípne"), čo signalizuje, že zaregistroval zatlačenie klávesy a zobrazí príslušný znak na mieste kurzora. Kurzor sa zároveň posunie o jedno miesto doprava. Ak nezaznel zvukový signál, je nutné klávesu zatlačiť znova.

Stlačte klávesu:	Na obrazovke vidíme:
!A!	A

Ak podržíme klávesu zatlačenú dlhšie, pípanie sa opakuje a znak sa opakovane zobrazí na obrazovke.

Podržíme klávesu	Na obrazovke vidíme
!B!	ABBBBBB

Počet zobrazených znakov B je závislý od toho, ako dlho sme držali klávesu !B ! stlačenú. Pri chybnej manipulácii s klávesnicou (napr. nedovolené súčasné stlačenie dvoch kláves) počítač vstup ignoruje.

Ak urobíme pri písaní chybu, môžeme ju opraviť zatlačením !DEL!. Jej stlačenie spôsobí zmazanie posledného zapísaného znaku a posun kurzora o jedno miesto doľava.

Zatlačíme klávesu	Na obrazovke vidíme:
!DEL!	ABBBBB

Ak podržíme !DEL!, postupne sa vymažú všetky znaky a kurzor sa vráti až na začiatok riadku.

Niektoré klávesy sú označené dvoma znakmi. Pri zatlačení klávesy sa na obrazovke zobrazí spodný znak z klávesy. Ak chceme zapísať horný znak, musíme, podobne ako na písacom stroji, zatlačiť klávesu !SHIFT! a súčasne významovú klávesu. Pri hornom znaku počítač pípne vyšším tónom. Slovom SHIFT budeme označovať klávesu ⇧.

Stlačte klávesu	Na obrazovke vidíme
!SHIFT!!%!	%
!SHIFT!!?!	%?

Podobne ako na písacom stroji, sa na klávesnici nachádza !SHIFT! dvakrát.

Klávesy odlíšené farebne majú špeciálny význam, ktorý bude postupne vysvetlený v ďalších kapitolách.

Vyskúšajme si prácu s klávesnicou.
Pomocou !DEL! vymažme všetky znaky v riadku (ak nejaké sú).
Znakom ! ! je označená klávesa medzera.

Stláčajte postupne klávesy:	Na obrazovke vidíme:
!P!	P
!R!	PR
!I!	PRI
!N!	PRIN
!T!	PRINT
! !	PRINT
!SHIFT!!"!	PRINT "

!A!	PRINT "A
!H!	PRINT "AH
!O!	PRINT "AHO
!J!	PRINT "AHOJ
!SHIFT!!"!	PRINT "AHOJ"

Na obrazovke vidíme prvý príkaz pre počítač - slovo PRINT znamená pre počítač TLAČ a to, čo je medzi dvoma úvodzovkami je text, ktorý má počítač vytlačiť. Pod pojmom vytlačiť budeme ohápať zobrazenie na televízore. To, že naše zadanie úlohy pre počítač je skončené a že má úlohu vykonať, mu oznámime stlačením klávesy !CR!. Túto klávesu budeme vždy používať na ukončenie príkazu alebo riadku. Pri práci v priamom režime sa po vykonaní každého príkazu vypíše READY.

Stlačte	Na obrazovke
!CR!	PRINT "AHOJ"
	AHOJ
	READY

To znamená, že počítač vykonal úlohu, ktorú sme mu zadali, napísal na nový riadok AHOJ a výpisom READY opäť signalizuje, že je pripravený vykonávať naše ďalšie príkazy. V jednom riadku môžeme napísať aj niekoľko príkazov. Jednotlivé príkazy oddelíme od seba dvojbodkou.

Napíšte:
PRINT "AHOJ":PRINT "PPO1"!CR!
Na obrazovke vidíme:
AHOJ
PPO1
READY

2. JEDNODUCHÉ OPERÁCIE

V tejto kapitole si bez nároku na úplnosť informácií vysvetlíme a na príkladoch ukážeme, ako môžeme PP01 používať na vykonanie jednoduchých výpočtov zadaných priamo z klávesnice.

2.1. Kalkulačka

Vyskúšajte postupne riešiť jednotlivé úlohy:

Úloha	Napíšte	Na obrazovke
2+7	PRINT 2+7!CR!	PRINT 2+7 9 READY
2^2	PRINT 2^2!CR!	PRINT 2^2 4 READY
3x4	PRINT 3*4!CR!	PRINT 3*4 12 READY
$\sqrt{4489}$	PRINT SQR(4489)!CR!	PRINT SQR(4489) 67 READY
sin (π/4)	PRINT SIN(PI/4)	PRINT SIN(PI/4) 0.707103 READY

Všimnime si, že PP01 pozná Ludolfovo číslo PI = 3.14159 a je možné s touto hodnotou pracovať.

2.2. Výmaz obrazovky

Klávesa CLEAR (v prílohe č.3 je označená F0) vymaže obrazovku a umiestni kurzor do ľavého horného rohu obrazovky.

2.3. Chybové hlásenia

V prípade, že zadáme chybný príkaz, PP01 pípne vysokým tónom a na obrazovke sa na nový riadok vypíše:

CHYBA číslo

kde čísla jednotlivých chýb s vysvetlením sú uvedené v prílohe 1.

Uvedené chybové hlásenie sa vypíše v priamom i príkazovom režime pri nájdení chyby, pričom v príkazovom režime vypíše BASIC ešte riadok, v ktorom bola objavená chyba a približné miesto výskytu chyby označí znakom ? a zvyšok príkazového riadku vypíše do ďalšieho riadku na obrazovke.

Príklad:
```
5 A=7
10 PRINT 5.5**A
RUN
```

na displeji:
```
CHYBA 1
10 PRINT 5.5**
             A
```

Chybu pri chybovom výpise treba hľadať vľavo i vpravo od chybového výpisu (t.j. otáznik je umiestnený v okolí chybového výpisu).

2.4. Premenné

Niekedy môže byť užitočné označiť si niektoré hodnoty, ktoré používame vo výpočtoch písmenom (písmenami) a potom používať toto písmeno vo výrazoch. V programe môže byť takto

označená hodnota menená - odtiaľ názov premenná.

Príklad.
Vypočítajte plochu kruhu pre polomer R=2.
Plochu možeme vypočítať podľa vzťahu $P=\pi r^2$ (2.4.1)
Jedna možnosť, ako dospieť k riešeniu zadanej úlohy, je:

Napíšte:	Na obrazovke:
PRINT 3.1415*2^2!CR!	PRINT 3.1415*2^2
	12.566

čiže jednotlivé označenia premenných a konštánt z výrazu sme nahradili konkrétnymi hodnotami.

Druhá možnosť riešenia:

Napíšte:	Na obrazovke:
R=2!CR!	R=2
	READY
P=PI*R^2!CR!	P=PI*R^2
	READY
PRINT P!CR!	PRINT P
	12.5663
	READY

hodnotu 2, s ktorou chceme počítať, sme priradili premennej R a ďalší príkaz je vlastne prepísanie vzťahu (2.4.1.) do tvaru, ktorému počítač rozumie. Tretí príkaz vytlačí vypočítaný výsledok.

V tomto príklade sa použitie premenných zdá byť zbytočnou komplikáciou. Skutočnosť, že premenným priradené hodnoty sú zapamätávané a môžu sa použiť k ďalším výpočtom, však nadobúda význam v ďalšom pri písaní programov.

3. PROGRAM

Slovo program vo Vás vzbudzuje posvätnú úctu - musí to byť niečo nesmierne zložité, čo ja nikdy nedokážem napísať. Táto kapitola Vás má presvedčiť, že to s písaním programov až také zlé nebude. Programovací jazyk GBASIC je jednoduchý na používanie a dovolí nám formou ľahkej konverzácie (musíme sa naučiť niekoľko anglických slov) zadávať počítaču úlohy pre riešenie. Program, tak ako ho budeme z klávesnice do počítača zadávať po jednotlivých príkazoch, sa ukladá do pamäti. Po uložení celého programu môžeme spustiť výpočet.

3.1. Príprava pamäti pre vstup

Na začiatku, skôr než začneme vkladať program do pamäti, musíme pamäť pripraviť na vstup programu. Na tento účel použijeme príkaz:

Napíšte:	Na obrazovke:
NEW!CR!	NEW
	READY

```
*************
* Dôležité ! ***********************************************
* Príkaz NEW musíme použiť vždy pred vkladaním nového prog-*
* ramu                                                     *
************************************************************
```

NEW spôsobí zrušenie skôr vloženého programu.

Každý riadok v jazyku GBASIC je označený číslom. Postupnosť riadkov tvorí program. Po spustení sú jednotlivé riadky programu postupne vykonávané.

3.2. Vstup programu

V odstavci 2.4. sme vypočítali postupným zadávaním príkazov, ktoré sa okamžite po zadaní vykonávali, plochu kruhu. Pomocou rovnakých príkazov s malým zobecnením si napíšeme program, ktorý bude počítať plochu kruhu pre ľubovoľný, z klávesnice zadaný polomer R.

Zapíšte program do pamäti presne podľa návodu:

1. Napíšte NEW!CR!
2. Vlastný program:

```
10 INPUT "ZADAJ POLOMER",R!CR!
20 P=PI*R^2!CR!
30 PRINT "PLOCHA KRUHU=";P!CR!
```

Príkazy v riadkoch číslo 20, 30 už poznáme z kapitoly 2. Príkaz vstupu INPUT spôsobí, že na obrazovke sa zobrazí text, ktorý je v úvodzovkách a program bude očakávať vstup čísla z klávesnice. Hodnota prečítaná z klávesnice sa uloží do premennej R. Na skutočnosť, že je očakávaný vstup z klávesnice, program upozorní výpisom znaku "?" na obrazovke.

3.3. Spustenie programu.

Program uložený v pamäti spustíme príkazom RUN.

Napíšte:	Na obrazovke:
RUN!CR!	RUN
2!CR!	ZADAJ POLOMER?2
	PLOCHA KRUHU= 12.5663
	READY
RUN!CR!	RUN
2.5!CR!	ZADAJ POLOMER?2.5
	PLOCHA KRUHU= 19.6347

```
                                    READY
RUN!CR!                             RUN
3.2!CR!                             ZADAJ POLOMER?3.2
                                    PLOCHA KRUHU= 32.1695
                                    READY
```

3.4. Prerušenie programu

Bežiaci program môžeme prerušiť stlačením klávesy !CTRL!!C!. K prerušeniu dôjde medzi vykonaním dvoch príkazov BASICu.

Basic vypíše:
BREAK AT LINE x
kde x je číslo riadku, ktorého vykonávanie bolo prerušené. Ak v riadku x je len jeden príkaz, potom tento príkaz ešte nebol vykonaný.

3.5. Pozastavenie programu

Bežiaci program môžeme pozastaviť stlačením klávesy !CTRL!!S!. K pozastaveniu dôjde medzi vykonaním dvoch príkazov v Basicu. O pozastavení nie je vypisovaná žiadna správa, Basic iba čaká na stlačenie ďalšej klávesy na klávesnici (musí to byť iná klávesa ako !CTRL!!S!). Ak stlačíme ďalšiu klávesu, potom Basic pokračuje vo vykonávaní príkazov.

Poznámka: O tom, že Basic očakáva vstup z klávesnice, nás informuje textový kurzor.

Ak textový kurzor bliká, potom Basic očakáva vstup z klávesnice (toto neplatí iba vtedy, ak pre vstup použijeme funkciu KEY).
Ak textový kurzor nebliká, alebo ak vôbec nie je zobrazený, potom Basic nepožaduje vstup z klávesnice (výnimku tvorí funkcia KEY).

3.6. Zobrazenie programu

Program, ktorý je uložený v pamäti PP01 je možné zobraziť príkazom:
LIST!CR!

Príklad: Zobrazte program uložený do pamäti v odstavci 3.2.
Príklad: Po príkazoch

```
NEW!CR!
LIST!CR!
```

vidíme, že pamäť programu je prázdna, príkaz NEW pripravil pamäť pre vstup nového programu.

4. PRÁCA S MAGNETOFÓNOM

Malý program, aký sme si napísali v predchádzajúcej kapitole, si môžeme kedykoľvek vložiť do pamäti počítača z klávesnice. Programy, ktoré obsahujú desiatky alebo stovky príkazov však nie je efektívne vkladať z klávesnice vždy, keď ich chceme opakovane použiť. K lacnému počítaču je neúnosné pripájať relatívne drahé vonkajšie pamäti, na ktoré by sa dal program zaznamenať. Z tohoto dovodu je ako vonkajšia pamäť použitý bežný komerčný magnetofón, ktorý máte doma.

4.1. Pripojenie magnetofónu

Magnetofón pripojíme k PP01 bežnou prepojovacou šnúrou, ktorá sa používa pre nahrávanie na magnetofón z iného magnetofónu, rozhlasového prijímača, gramofónu alebo televízora.

V magnetofóne pripojíme šnúru do konektora pre nahrávanie a v PP01 do konektora "MGF" (obr.1).

4.2. Organizácia záznamov na kazete

Každý záznam, ktorý nahrávame na magnetofón, je rozdelený na bloky, pričom dĺžka každého bloku je 133 bytov. Každý blok obsahuje 128 datových bytov, kontrolnú sumu, poradové číslo bloku rámci záznamu, číslo záznamu, ktoré zadal operátor a informačný byte, ktorý nesie informáciu, ktorým príkazom bol daný záznam vytvorený.

Pri nahrávaní je každý blok nahratý na mgf. pásku dvakrát za sebou.

Každý záznam na kazete má svoje číslo, podľa ktorého môžeme konkrétny záznam vyhľadať. Programy alebo údaje, ktoré zapisujeme na kazetu, číslujeme vzostupne od 1 do 79. Medzi jednotlivé záznamy môžeme nahrať slovnú informáciu, ktorá charakterizuje nasledujúci záznam.

Zvyknite si po založení pásky a jej pretočení na začiatok vynulovať počítadlo na magnetofóne a pri nahrávaní programu si poznačiť stav počítadla v medzere medzi záznamami. Tento údaj Vám v budúcnosti pomôže orientovať sa, kde je čo na kazete uložené.

4.3. Nahrávanie programu do pamäti

Prenahrávanie programu použijeme príkaz

KLOAD $ číslo záznamu!CR!

kde "číslo záznamu" je číslo, pod ktorým je program nahratý na kazete.
Skôr, ako zatlačíme !CR!, spustíme magnetofón. Na obrazovke sa pri čítaní záznamu zobrazí jeho číslo. Táto informácia nám pomôže zorientovať sa v záznamoch na kazete. Ak sa začne čítať požadovaný záznam od počiatku, na obrazovke vidíme číslo záznamu a obrazovka sa začne posúvať po polriadku smerom nahor. V prípade, že sa na obrazovke zobrazí len číslo záznamu, alebo sa nezobrazí ani číslo záznamu, ale počujeme z magnetofónu, že magnetofón sníma program, je nutné pretočiť pásku tak, aby sa dostala do medzery pred hľadaným záznamom. Pretáčanie pásky a vyhľadávanie záznamu musíme urobiť ovládacími tlačítkami magnetofónu ručne.

4.4. Záznam programu na kazetu

Postup pri nahrávaní na kazetu:

1. Pripojíme magnetofón (pozri 4.1)
2. Vložíme do magnetofónu kazetu, na ktorú chceme nahrávať.
3. Ak je kazeta prázdna, pretočíme ju na začiatok a vynulujeme počítadlo magnetofónu.
   Ak kazeta nie je prázdna, čiže na kazete už máme skôr

nahrané programy, pred nahratím ďalšieho programu kazetu pretočíme do medzery za posledným nahratým programom a pokračujeme od bodu 5.

4. Kazetu pretočíme tak, aby hlava bola na magnetickom médiu.
5. Poznačíme si stav počítadla na magnetofóne, číslo záznamu a meno programu.
6. Spustíme nahrávanie a cez mikrofón nahráme na magnetofón číslo záznamu a meno programu.
7. Zatlačíme tlačítko PAUSE na magnetofóne.
8. Napíšeme príkaz
   KSAVE $číslo záznamu
   kde číslo záznamu môže byť číslo z intervalu 1 až 79.
9. Uvoľníme tlačítko PAUSE
10. Zatlačíme !CR!.

Program, ktorý je v pamäti, sa nahráva na magnetofón a po ukončení nahrávania počítač pípne a na obrazovke sa vypíše READY.
Zastavíme magnetofón.

Záznamy na páske je vhodné číslovať vzostupne.

Príkazy KSAVE a KLOAD sa možu vykonávať v priamom a príkazovom režime.

Ak napíšeme príkaz KSAVE do príkazového režimu, potom je vhodné umiestniť pred neho príkaz PRINT, ktorý upozorní operátora, že má pripraviť mgf na nahrávanie.

Príkaz KLOAD v príkazovom režime na rozdiel od priameho režimu aj spustí program (ktorý bol jeho prostredníctvom zosnímaný z pásky) a to od riadku s najnižším číslom (čiže, bez zásahu operátora sa vykoná RUN 1).

Ak sa pri snímaní z pásky objaví výpis: FILE IS NO COMPLETE!, znamená to, že záznam nebol zosnímaný celý správne z pásky, ale stratil sa jeden alebo viac blokov. Basic vykoná príkaz NEW, čiže nemáme prístup k takémuto neúplnému záznamu.

Ak pri snímaní z pásky stlačíme klávesu CONTROL-C,

prerušíme tým snímanie z pásky, Basic vypíše:
BREAK
a vykoná NEW a nedovolí tým prístup ku neúplnému záznamu.

Ak stlačíme CONTROL-C pri nahrávaní na mgf., potom je prerušené nahrávanie na mgf., program vypíše BREAK a výpisom READY sa očakáva ďalší príkaz.

5. KLÁVESNICA, OVLÁDACIE A INDIKAČNÉ PRVKY

Klávesnica mikropočítača je znázornená v prílohe č.3. Zásady práce boli vysvetlené v časti 1.4. V tejto kapitole sa sústredíme na popis niektorých klávies a zásady práce s nimi.

Po súčasnom zatlačení klávesy RESET a SHIFT sa generuje signál RESET, to znamená, že činnosť, ktorú vykonáva BASIC je prerušená a prebieha nasledujúca činnosť:

- nastavia sa biele výpisy na čiernom pozadí
- ak prebiehala tlač na tlačiareň pomocou príkazu LPRINTER, potom je tlač prerušená a výstup na tlačiareň zrušený (t.j. ak chceme znovu kopírovať výpisy na tlačiareň, potom musíme znovu vykonať príkaz LPRINTER)
- BASIC vymaže obrazovku, vypíše READY a očakáva príkaz od operátora.

To znamená, že program, ktorý napísal užívateľ, zostane nezmenený.

Po súčasnom zatlačení klávies RESET, SHIFT a medzera sa generuje tiež signál RESET, ale toto zatlačenie klávies má taký istý účinok, ako zapnutie mikropočítača (pozri 11.7). V tomto prípade je program, ktorý napísal užívateľ vymazaný!

Klávesa označená F0 je klávesa CLEAR, po zatlačení tejto klávesy je displej vymazaný a kurzor je prestavený do ľavého horného rohu. Zatlačenie tejto klávesy počas behu programu alebo v príkaze INPUT, EDIT nemá žiadnu odozvu.

Klávesy označené F1 až F14 sú tzv. funkčné klávesy. Tieto klávesy majú rozdielny účinok podľa toho, či ich stlačíme s klávesou !SHIFT! alebo bez nej.
Klávesy F1 a F14 bez stlačenia !SHIFT! generujú po zatlačení vyhradené slová BASICu a to:

F1 - LIST	F2 - CONT	F3 - RUN	F4 - AUTO
F5 - EDIT	F6 - MONIT	F7 - PRINT	F8 - GOTO
F9 - INPUT	F10 - GOSUB	F11 - RETURN	F12 - DRAW
F13 - CHR$	F14 - SQR		

Tieto slová sa po zatlačení vygenerujú do vstupného riadku, ak ich chceme okamžite vykonať, potom stlačíme príslušnú funkčnú klávesu a klávesu !CR!, napríklad:

Stlačíme:	Na obrazovke:
!F1!!CR!	LIST
	výpis programu

Po zatlačení funkčnej klávesy nemusíme stláčať klávesu !CR!, ale možeme dopisovať potrebné parametre a prípadne príkazy uložiť s číslom riadku a vykonať ich až v nepriamom režime.

Ak stláčame klávesy F1 až F14 spolu s klávesou SHIFT, potom tieto klávesy generujú text, ktorý si možeme nadefinovať nasledujúcim príkazom:

SETKEY CK reťazec

kde CK je číslo klávesy a musí tu byť uvedené číslo z intervalu 1 až 14 a reťazec sú znaky uzavreté v apostrofoch alebo úvodzovkách, pričom počet znakov môže byť maximálne 15.

Príklad:
SETKEY 2 "AB+CD 34"!CR!

Po zatlačení klávesy F2 spolu s klávesou SHIFT sa vygeneruje do vstupného riadku text AB+CD 34.

Príkaz SETKEY je možné použiť v priamom i nepriamom režime, t.j., tá istá klávesa môže mať v rôznych častiach programu rozdielny význam (ak ju v programe počas behu programu predefinujeme). Význam jednotlivých klávies je možné meniť

iba predefinovaním na nový reťazec, tieto klávesy nemaže žiadny príkaz.
Po zapnutí počítača, alebo po stlačení klávies tomu ekvivalentných (t.j. RESET, SHIFT, MEDZERA) sú funkčné klávesy nastavené na znak % (percento). To znamená, že ak stlačíme napr. klávesy SHIFT a F5 a zobrazí sa % vieme, že F5 ešte nebola nadefinovaná.

Význam klávies F1 až F14 stlačených bez !SHIFT! je nezávislý od stlačenia tých istých klávies so !SHIFT!.
Indikačné prvky sú popísané v technickom popise PP01.

6. VÝRAZY A FUNKCIE

V tejto kapitole si povieme niečo o výrazoch, ktoré môžeme použiť pri výpočtoch. Výrazy obecne pozostávajú z premenných, čísel, funkcií a operátorov. V tejto kapitole sa budeme zaoberať len aritmetickými a logickými výrazmi. Reťazcom, reťazcovým premenným a funkciám je venovaná samostatná kapitola (9).

6.1. Aritmetické výrazy

Príklady aritmetických výrazov sme videli už v odstavci 2.1. Obecne môžeme povedať, že aritmetický výraz je predpis pre výpočet číselnej hodnoty. Postupne sa oboznámime s aritmetickými operátormi, číslami, premennými a funkciami, čiže so všetkými prvkami, ktoré tvoria aritmetický výraz.

6.1.1. Aritmetické operátory

GBASIC dovoľuje použiť tieto aritmetické operátory:

- sčítanie (+)
- odčítanie (-)
- násobenie (*)
- delenie (/)
- umocnenie (^)
- celočíselné delenie (\)

Vyskúšajme si jednoduché aritmetické operácie s číslami.
V kapitole 2.1. sme si uviedli príklady, výsledky ktorých sme tlačili príkazom PRINT.
Priblížiť použitie PP01 kalkulačke môžeme, keď napíšeme jednoduchý program:

Napíšte:	Na obrazovke:
NEW	NEW READY
10 INPUT X:PRINT X:GOTO 10!CR!	10 INPUT X:PRINT X:GOTO 10
RUN	RUN

Program je spustený a na obrazovke sa vypíše otáznik, ktorý signalizuje, že počítač očakáva vstup.

3+2!CR!	?3+2 5
5*4!CR!	?5*4 20

Pri používaní umocnenia (^) si treba uvedomiť, že hodnota výrazu uvedeného vľavo od znaku musí byť väčšia ako nula. Vyplýva to zo spôsobu realizácie umocnenia v mikropočítači.

Chyba na poslednom desatinnom mieste je spôsobená nepresnosťou pri výpočte čísel v pohyblivej radovej čiarke.

2^3!CR!	?2^3 8,00006
16\3!CR!	?16\3 5

Operácia celočíselného delenia odrezáva desatinnú časť čísla, po delení, tzn. výsledkom je vždy celé číslo.

17\3!CR!	?17\3 5
3\17!CR!	?3\17 0

V prípade, že sa vo výraze nachádza viac aritmetických operátorov, je poradie vykonávania jednotlivých operátorov dané ich prioritou:

1. ^	Umocnenie
2. \, *, /	Celočíselné delenie, násobenie, delenie
3. +,-	Sčítanie, odčítanie

Ak sa vo výraze nachádza niekoľko operátorov rovnakej prioritnej úrovne, vykonávajú sa v poradí, v akom sú napísané. Ako uvidíme v príkladoch, je nutné si pri písaní aritmetických výrazov uvedomiť tieto pravidlá.

Napíšte:	Na obrazovke:
3^2+2*3!CR!	?3^2+2*3
	14.9999
3^2*3+2!CR!	?3^2*3+2
	28.9997
5*4/2^2!CR!	?5*4/2^2
	5
5/2*4^2!CR!	?5/2*4^2
	40.0002
5+2*2!CR!	?5+2*2
	9

Ak pomocou uvedených pravidiel nedokážeme zapísať aritmetický výraz, môžeme použiť zátvorky, ktoré menia prioritu operátorov.

Napíšte:	Na obrazovke:
(5+2)*2!CR!	?(5+2)*2
	14

V aritmetických výrazoch môžeme použiť len okrúhle zátvorky (). Pri vyhodnocovaní aritmetického výrazu majú zátvorky najvyššiu prioritu. Pokiaľ sa vo výraze nachádzajú zátvorky do seba vnorené, majú najvyššiu prioritu vnútorné zátvorky.

Napíšte:	Na obrazovke:
3*(2+3*(2+3))!CR!	?3*(2+3*(2+3))
	51

Vo výraze sa musí vždy nachádzať rovnaký počet pravých a ľavých zátvoriek. V opačnom prípade PP01 hlási chybu.

Výsledok - posledné číselné vyhodnotenie jednotlivých aritmetických výrazov v našom príklade (spomeňme si, že v pamäti je uložený program
10 INPUT X:PRINT X:GOTO 10)
je vždy uložený v premennej X. Túto premennú s uloženým medzivýsledkom môžeme použiť k ďalšiemu výpočtu.
Napríklad po poslednom príklade je v premennej X uložená hodnota 51.

Napíšte:	Na obrazovke:
X+X-2!CR!	X+X-2
	100

To znamená, že pri nastavovaní premennej pomocou príkazu INPUT, nemusíme zadávať len číslo, ale môžeme zadať aj aritmetický výraz.

6.1.2. Čísla

Doteraz sme pracovali s číslami bez toho, aby sme si definovali rozsah čísel, presnosť zobrazenia a tvary zápisu čísel. V tomto odstavci si uvedieme všetky tieto informácie.

ROZSAH ČÍSEL, ktoré môžu vstúpiť alebo byť uložené je:

$\langle -0.999999E63; -0.1E-63 \rangle$

$\langle 0.1E-63; 0.999999E63 \rangle$

To znamená, že čísla sú v PP01 uložené s presnosťou 6 platných miest.

Vo všetkých ďalších príkladoch musí byť v pamäti uložený a spustený kalkulačkový program z predchádzajúceho odstavca.

TVARY ČÍSEL, ktoré môžeme zapísať v programe, resp. ktoré môžu vstúpiť ako vstupné údaje:

Celé čísla - postupnosť číslic

Napíšte:	Na obrazovke:
12345!CR!	?12345
	12345
-2378!CR!	?-2378
	-2378

Ako celočíselné konštanty môžu v GBASICu vstupovať aj hexadecimálne čísla. Každé hexadecimálne číslo musí byť ukončené znakom H a musí začínať číslicou.

T.j.:

FOH je nesprávne zapísaná hexadecimálna konštanta
OFOH je správne zapísaná hexadecimálna konštanta

Program, ktorý máme doteraz uložený v pamäti, môžeme použiť na prevod čísel zo šestnástkovej sústavy do sústavy desiatkovej, pretože PRINT v uvedenom programe vypisuje čísla v desiatkovej sústave.

Napíšte:	Na obrazovke:
0AH	?0AH
	10
20H	?20H
	32
FH	?FH
	0

Výpis nuly v poslednom príklade je dôsledkom toho, že chýba nula pred číslom a BASIC považuje FH za identifikátor premennej.

Desatinné čísla - postupnosť číslic.postupnosť číslic

Napíšte:	Na obrazovke:
12.345!CR!	?12.345
	12.345

V prípade, že chceme vstúpiť s desatinným číslom, ktoré má nulovú celú časť, nie je nutné písať nulu pred desatinnou bodkou.

Napíšte:	Na obrazovke:
0.123!CR!	?0.123
	0.123
.123!CR!	?.123
	0.123

Semilogaritmický tvar čísla - mantisa E exponent
kde mantisa je celé alebo desatinné číslo a exponent je celé číslo z intervalu $\langle -63,63 \rangle$. Je to vlastne upravený zápis, ktorý používame v matematike:

$$\text{mantisa} * 10^{\text{exponent}}$$

Napíšte:	Na obrazovke:
123E2!CR!	?123E2
	12300
123E-2!CR!	?123E-2
	1.23
-1.23E4!CR!	?-1.23E4
	-12300
	?

Z príkladu vidíme, že rovnaké číslo možeme zapísať niekoľkými možnými spôsobmi.

ZOBRAZENIE ČÍSEL pri výstupe na obrazovku sme už čiastočne videli v predchádzajúcich príkladoch. Presnejšie pravidlá si uvedieme:

- všetky čísla sa zobrazujú na 6 platných miest
- kladné číslo má na začiatku medzeru, záporné číslo má na začiatku znamienko mínus (-)
- celé čísla v absolútnej hodnote menšie ako 10^6 sa zobrazujú v tvare celého čísla
- nezobrazujú sa nevýznamné nuly vľavo v celej časti čísla

Napíšte:	Na obrazovke:
00123!CR!	?00123
	123
-0135!CR!	?-0135
	-135
	?

- čísla z intervalu (-1;1) sa zobrazujú v desatinnom tvare, ak sa možu zobraziť na 6 platných miest
- desatinné čísla v absolútnej hodnote väčšie ako jedna a menšie ako 10^6 sa zobrazujú v desatinnom tvare
- nevýznamné nuly v desatinnej časti čísla sa nezobrazujú
- všetky ostatné čísla sa zobrazujú v semilogaritmickom tvare, pričom v celej časti mantisy je jedna číslica

Napíšte:	Na obrazovke:
.1234567!CR!	?.1234567
	0.123456
12345.543!CR!	?12345.543
	12345.5
21.3E12!CR!	?21.3E12
	2.13E13
-0123!CR!	?-0123
	-123
0.00000000234!CR!	?0.00000000234
	2.34E-09

6.1.3. Premenné

Ďalším prvkom, ktorý sa vyskytuje v aritmetickom výraze

je premenná. S premennými sme sa zoznámili už v odstavci 2.4., na tomto mieste si tento pojem upresníme. GBASIC rozlišuje premenné, ktoré možu nadobúdať číselné hodnoty a ďalej reťazcové premenné. Reťazcové premenné možu nadobúdať hodnoty reťazcov znakov. Podrobnejšie budeme o reťazcoch a reťazcových premenných hovoriť v odstavci 6.3 a kapitole 9.

Číselná premenná určuje miesto v pamäti, v ktorom je uložená číselná hodnota. Rozoznávame dva druhy premenných:

- jednoduché premenné
- polia (indexované premenné)

Jednoduché premenné

V GBASICu je možné tvoriť mená premenných nasledujúcim spôsobom:

- meno musí začínať písmenom
- ako ďalšie znaky mena môžu byť použité ľubovoľné písmeno, alebo číslica
- dĺžka mena je obmedzená na 15 znakov
- meno premennej nesmie byť zhodné s kľúčovým slovom GBASICu (príkazy, funkcie a pod.)

Príklady:

Správne utvorené mená premenných:

X, X1, ABC, A1B1C2

Nesprávne utvorené mená premenných:

1X, PRINT, A 2

Premennej je možné priradiť hodnotu priraďovacím príkazom.

Napíšte:	Na obrazovke:
X=10!CR!	X=10
	READY
XQUADRAT=X^2!CR!	XQUADRAT=X^2
	READY

```
PRINT X,XQUADRAT!CR!          PRINT X,XQUADRAT
                               10            99.9985
                              READY
```

Vidíme, že v priraďovacom príkaze môže na pravej strane stáť aritmetický výraz. Výraz sa vyčísli podľa pravidiel, ktoré sme si uviedli a jeho hodnota sa priradí premennej, ktorá stojí na ľavej strane. Hodnotu premennej môžeme zmeniť ďalším priraďovacím príkazom.

Napíšte:	Na obrazovke:
X=X+10!CR!	X=X+10
PRINT X!CR!	PRINT X
	20

Ako sme už uviedli, GBASIC dovoľuje použiť okrem jednoduchých aj indexované premenné. O týchto budeme hovoriť v kapitole 8.

6.1.4. Matematické funkcie

GBASIC dovoľuje v programoch, ale aj v kalkulačkovom režime, použiť celý rad preddefinovaných funkcií. Okrem toho, ako uvidíme v ďalšom, dovoľuje používateľovi definovať a nasledovne aj využívať svoje vlastné funkcie.

Funkcia je predpis, ako pre zadané hodnoty argumentov vypočítať hodnotu funkcie. Rôzne funkcie môžu mať rôzny počet argumentov, aj keď v štandardných funkciách sa najviac vyskytujú funkcie s jedným argumentom alebo bez argumentu.

Funkcie môžeme rozdeliť do niekoľkých skupín:
- matematické funkcie
- reťazcové funkcie
- užívateľom definované funkcie
- špeciálne funkcie

V tomto odstavci sa budeme zaoberať len funkciami, ktorých argumentom je aritmetický výraz, alebo sú bez argumentu a výsledkom je číselná hodnota. Ďalšími skupinami funkcií sa

budeme zaoberať v ďalších kapitolách.
Prehľad matematických funkcií GBASICu je v tabuľke 1.

Tabuľka 1. Matematické funkcie

Funkcia a argument	Význam
ABS(X)	Absolútna hodnota X
TRUNC(X)	Celočíselná časť X
FRC(X)	Desatinná časť X
SQR(X)	Kladná druhá odmocnina X
SGN(X)	-1 ak X<0, 0 ak X=0, 1 ak X>0
SIN(X)	sínus X
COS(X)	cosínus X
TAN(X)	tangens X
ATAN(X)	arcustangens X (v 1. alebo 4. kvadrante)
LN(X)	prirodzený logaritmus X
EXP(X)	e^X
RND	náhodné číslo z intervalu <0,1)
PI	hodnota 3.14159 (Ludolfovo číslo)
INF	hodnota 9.99999E+62(najväčšie zobraziteľné číslo)
EPS	hodnota 1E-64 (najmenšie zobraziteľné číslo)

Jednotlivé funkcie si, pokiaľ to bude nutné, vysvetlíme v ďalšom texte. Pokiaľ je činnosť funkcie jasná z názvu, uvedieme si len príklady. V jednotlivých príkladoch predpokladáme, že v pamäti počítača je uložený jednoriadkový program z odstavca 6.1.1., ktorý dovoľuje použiť PP01 ako kalkulačku. Argument X funkcií uvedených v tabuľke 1 (okrem posledných troch, ktoré argument nemajú), môže byť ľubovoľný aritmetický výraz.

Absolútna hodnota

Napíšte:	Na obrazovke:
ABS(-35)!CR!	?ABS(-35) 35
ABS(82)!CR!	?ABS(82) 82

Celá časť čísla

Napíšte:	Na obrazovke:
TRUNC(2.3)!CR!	?TRUNC(2.3) 2
TRUNC(-2.3)!CR!	?TRUNC(-2.3) -2

Funkcia orezáva desatinnú časť čísla.

Desatinná časť čísla

Napíšte:	Na obrazovke:
FRC(2.234)!CR!	?FRC(2.234) 0.234
FRC(-7.89)!CR!	?FRC(-7.89) -0.89

Funkcia odrezáva celú časť čísla. Výsledok má rovnaké znamienko ako argument.

Napíšte:	Na obrazovke:
TRUNC(4.567)+FRC(4.567)!CR!	?TRUNC(4.567)+FRC(4.567) 4.567

Druhá odmocnina

Vypočíta druhú odmocninu z hodnoty aritmetického výrazu X. Hodnota aritmetického výrazu X musí byť väčšia alebo rovná 0.

Napíšte:	Na obrazovke:
SQR(2.345)!CR!	?SQR(2.345) 1.53133
SQR(28910)!CR!	?SQR(28910) 170.029

Znamienko výrazu

Výsledkom funkcie je hodnota 1 ak je argument kladný, 0 ak je nulový a -1 ak je argument záporný.

Napíšte:	Na obrazovke:
SGN(-5)!CR!	?SGN(-5) -1
SGN(120)!CR!	?SGN(120) 1
SGN(0)!CR!	?SGN(0) 0

Trigonometrické funkcie

Argument trigonometrických funkcií môže byť uhol v stupňoch alebo v radianoch. Po zapnutí mikropočítača je nastavený výpočet v radiánoch.
Prepnutie do módu, keď je možné zadávať uhol v stupňoch, je možné pomocou príkazu:

DEG

Spätné prepnutie do módu, keď sa interpretujú uhly v radiánoch, je možné pomocou príkazu:
RAD

Napíšte:	Na obrazovke:
SIN(2.34)!CR!	?SIN(2.34) 0.718471
COS(SQR(2))!CR!	?COS(SQR(2)) 0.15596

Logaritmické funkcie

Pomocou funkcie prirodzený logaritmus môžeme vyčísliť logaritmus pri ľubovoľnom základe pomôcou vzťahu:

$$\ln_a X = \frac{\ln X}{\ln a}$$

Vypočítajte $\ln_2 175$, $\ln_{10} 100$, $e^{\sin(2)}$

Napíšte:	Na obrazovke:
LN(175)/LN(2)!CR!	?LN(175)/LN(2) 7.45115
LN(100)/LN(10)!CR!	?LN(100)/LN(10) 2
EXP(SIN(2))!CR!	?EXP(SIN(2)) 2.48257

Náhodné čísla

V niektorých aplikáciach (štatistika, teória hromadnej obsluhy, simulácie) je dôležité generovať náhodné čísla. Generátor, ktorý je implementovaný v GBASICu, generuje pseudonáhodné čísla väčšie alebo rovné 0 a menšie ako 1. Po zapnutí mikropočítača sa postupnosť náhodných čísel generuje vždy od toho istého čísla.

Napíšte:	Na obrazovke:
RND!CR!	?RND XXXXXX
RND!CR!	?RND YYYYYY

Preddefinované konštanty

GBASIC dovoľuje použiť tri preddefinované konštanty, ktoré sa v programe alebo pri výpočte môžu napísať menom.

Napíšte:	Na obrazovke:
PI!CR!	?PI 3.14159
EPS!CR!	?EPS 1E-64
INF!CR!	?INF 9.99999E+62

6.2. Logické výrazy

Logické výrazy môžu nadobúdať dve logické hodnoty "true" alebo "false", čiže logický výraz môže byť pravdivý alebo nepravdivý. Základným logickým výrazom je relácia, kde relácia sú dva aritmetické výrazy porovnávané relačným operátorom.

6.2.1. Relačné operátory

Relačné operátory určujú vzťah medzi dvoma aritmetickými výrazmi.

Symbol	Význam
=	rovný
<	menší než
>	väčší než
<=	menší alebo rovný
>=	väčší alebo rovný
<>	rôzny

Príklady relácií:

A<B

C>3

A<=C

6.2.2. Logické operátory

Zložitejšie logické výrazy môžeme tvoriť z jednoduchých logických výrazov pomocou logických operátorov. GBASIC dovoľuje používať tri logické operátory pre konjunkciu (AND), disjunkciu (OR) a negáciu (NOT).

Ak použijeme v jednom riadku viacero logických operátorov, potom sa vyhodnocujú podľa priority a to:

NOT
AND
OR

pričom NOT má najvyššiu prioritu a OR najnižšiu prioritu. Ak chceme zmeniť prioritu pri vyhodnocovaní, potom musíme použiť hranaté zátvorky.

Konjunkcia

AND spája dva logické výrazy. Ak sú oba pravdivé (T), je výsledok pravdivý. Ak je jeden alebo oba nepravdivé (F) je výsledok nepravdivý.

X	Y	X AND Y
T	T	T
T	F	F
F	T	F
F	F	F

Disjunkcia

OR spája dva logické výrazy. Ak je jeden alebo oba výrazy pravdivé, je výsledok pravdivý. Ak sú oba nepravdivé, je výsledok nepravdivý.

X	Y	X OR Y
T	T	T
T	F	T
F	T	T
F	F	F

Negácia

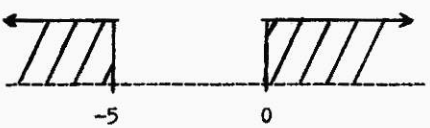
NOT vráti obrátenú logickú hodnotu logického výrazu.

X	NOT X
T	F
F	T

Príklady logických výrazov:
X^2+Y^2<1 AND X>0

Výraz je pravdivý, ak bod o súradniciach X,Y leží v prvom alebo druhom kvadrante jednotkovej kružnice.

X>0 OR X<-5



Platí pre všetky X z obrázku.

Použitie logických výrazov bude zrejmé pri výklade podmienených príkazov.

6.3. Reťazce

Postupnosť znakov ohraničenú úvodzovkami nazývame reťazec.

Príklad: "DOBRY DEN"

Dĺžka reťazca sa udáva v počte znakov, ktoré reťazec obsahuje. Úvodzovky sa do tohto počtu znakov nepočítajú. Medzery vo vnútri reťazca sú významové znaky. To znamená, že reťazec z príkladu má dĺžku = 9.

Reťazcová premenná je taká premenná, ktorá odkazuje na reťazec. Označujeme ich podobne ako numerické premenné menom, ktoré však musí byť ukončené znakom $.

Príklad: A$,XYZ$,JANOMA$

Podobne ako sme priraďovali číselné hodnoty premenným, môžeme priradiť reťazcovým premenným reťazec.

Príklad: A$="AHOJ"

Potom príkazy

```
     PRINT "AHOJ"
a    PRINT A$
```

majú rovnaký efekt.

7. ZÁKLADY PROGRAMOVANIA V GBASIC-U

Prvé informácie o vstupe programu, jeho spustení a zobrazení programu boli uvedené už v kapitole 3. V tejto kapitole budú tieto informácie rozvedené podrobnejšie a doplnené ďalšími príkazmi GBASICu. GBASIC je implementácia štandardného jazyka BASIC, ktorý je známy z iných počítačov, rozšírená o grafické príkazy. Program v GBASICu je postupnosť príkazov, ktoré určujú, aké činnosti má počítač vykonať. V jednoduchých programoch v kapitole 3 sme videli, že jednotlivé príkazy môžu obsahovať jeden alebo niekoľko "rezervovaných slov", ktoré majú v GBASICu špeciálny význam. Tieto rezervovane slová určujú operáciu, ktorú má počítač vykonať alebo dávajú informáciu potrebnú pre vykonanie iných príkazov.

Príklady rezervovaných slov:

PRINT	GOTO
INPUT	GOSUB
READ	DIM
FOR	REM
NEXT	DATA

Úplný zoznam rezervovaných slov je uvedený v prílohe 2.

Každý riadok zapísaný v jazyku GBASIC musí začínať číslom príkazu. Pri zadaní čísla riadku a vlastného príkazu ukončeného znakom !CR! sa príkaz nevykoná, ale sa len uloží do pamäti počítača. Väčšinu príkazov je však možné vykonať v tzv. priamom režime tzn., že pred vlastným príkazom nie je uvedené číslo riadku. Po ukončení príkazu bez čísla riadku znakom !CR! sa príkaz okamžite vykoná a neuloží sa do pamäti.

Okrem príkazov jazyka GBASIC rozoznáva PP01 ešte tzv. systémové príkazy, ktoré sa väčšinou vykonávajú v priamom režime. Niektoré z nich, napr. NEW, RUN, LIST sme už poznali

v predchádzajúcich kapitolách.

Čísla riadkov

Číslo riadku môže byť ľubovoľné celé číslo z intervalu $\langle 1,32767 \rangle$. Riadky sú v pamäti uložené podľa čísla riadku od riadku s najnižším číslom po riadok s najvyšším číslom, bez ohľadu na poradie, v akom riadky vstupovali. Je výhodné riadky číslovať počnúc od 10 s prírastkom čísla príkazu 10. V prípade, že bude neskôr nutné do už hotového programu pridávať ďalšie riadky, je vytvorená rezerva na vloženie ďalších deviatich riadkov medzi ľubovoľné dva riadky.

Príprava pamäti pre vstup programu

Pred vstupom nového programu musíme starý program vymazať z pamäti. Pri vstupe programu z vonkajšej pamäti (napr. z magnetofónu) sa starý program vymaže automaticky. Vypnutie počítača znamená stratu obsahu pamäti programu.

Pri vstupe nového programu z klávesnice je potrebné starý program vymazať príkazom NEW!CR!.

Vstup programu

Skôr než si podrobne vysvetlíme vstup programu, je potrebné uviesť niekoľko údajov o automatickom číslovaní riadkov, medzerách v príkazoch a dĺžke príkazového riadku.

Automatické číslovanie (AUTO)

Pre prepnutie do režimu automatického číslovania riadkov slúži príkaz:

AUTO
AUTO počiatočná hodnota
AUTO počiatočná hodnota, prírastok

Vykonanie príkazu AUTO spôsobí, že na obrazovke sa zobrazí

číslo riadku 10 a po zadaní príkazu sa automaticky zobrazí číslo riadku 20 atď.

Príklad:
Vyskúšajte automatické číslovanie riadkov pri vstupe programu z odstavca 3.2.

Napíšte:	Na obrazovke:
NEW!CR!	NEW
AUTO!CR!	AUTO
INPUT"ZADAJ POLOMER",R!CR!	10 "INPUT ZADAJ POLOMER",R
P=PI*R^2!CR!	20 P=PI*R^2
PRINT "PLOCHA KRUHU=",P!CR!	30 PRINT "PLOCHA KRUHU=",P
!CTRL!!C!	40
	READY

Počiatočnú hodnotu čísla riadku a prírastok, o ktorý sa bude číslo riadku zvyšovať je možné z hodnôt (10,10) meniť ľubovoľne.

Napríklad:

AUTO 50,5

Spôsobí, že číslo prvého riadku bude 50, ďalšieho 55 atď.

Prepnutie do režimu bez automatického číslovania riadkov je možné:

- súčasným stlačením kláves !CTRL! a !C!
- vykonaním príkazu v priamom režime
  napr. !DEL!!DEL!....!DEL!!CR!
  alebo !DEL!!DEL!....!DEL! príkaz bez čísla !CR!

"Počiatočná hodnota" a "prírastok" musia byť celé čísla volené tak, aby číslo riadku pri automatickom číslovaní bolo z intervalu $\langle 1,32767 \rangle$. Chyba sa hlási, ak je parameter v príkaze AUTO mimo tohoto intervalu, ako aj v prípade, že pri

automatickom číslovaní sa má zobraziť číslo riadku, ktoré presiahlo uvedený interval.

Medzery v príkaze

Pre lepšiu čitateľnosť príkazového riadku je možné jednotlivé časti príkazu oddeľovať od seba znakmi "medzera". Medzera však nemôže byť vo vnútri čísla riadku, mena premennej, vyhradeného slova a pod., ale môže oddeľovať tieto prvky príkazu od seba navzájom. Nakoľko by pri syntaktickej analýze mohlo dojsť k nejednoznačnému dekódovaniu vyhradených slov GBASICu, musí byť kľúčové slovo oddelené od nasledujúcej premennej, funkcie alebo konštanty medzerou.

Príklad:

PRINTX - nesprávne
PRINT X - správne

Dĺžka riadku

V jednom riadku programu je možné umiestniť aj niekoľko príkazov. Príkazy v jednom riadku oddeľujeme od seba dvojbodkou !:! . V prípade, že pri zadávaní príkaz presiahne dĺžku riadku obrazovky, príkazový riadok sa môže zobraziť na niekoľkých riadkoch obrazovky. Maximálna dĺžka príkazového riadku je 97 znakov (včítane !CR!).

Príklad:
Jednoduchý program z odstavca 3.2 môžeme napísať aj vo forme jednoriadkového programu.

Napíšte:
```
10 INPUT"ZADAJ POLOMER",R:P=PI*R^2:PRINT"PLOCHA KRUHU=";P!CR!
```

Spustenie programu (RUN)

Program, ktorý je uložený v pamäti spustíme napísaním

RUN!CR!
alebo RUN číslo riadku!CR!

V prvom prípade sa spustí program od riadku s najnižším číslom, v druhom prípade, od riadku s číslom uvedeným v príkaze RUN. Ak takéto číslo neexistuje, program sa spustí od riadku s najbližším vyšším číslom, ak ani taký príkaz neexistuje, vypíše sa READY.

Príkazy sa vykonávajú v poradí, v akom sú za sebou napísané, s výnimkou príkazov, ktoré možu poradie vykonávania zmeniť (napr. GOTO).

Ak je program spustený príkazom RUN, vymaže sa tabuľka premenných a funkcií definovaných používateľom, to znamená, že premenné nemajú definovanú hodnotu (napr. z predchádzajúceho behu programu), ale ich hodnoty sa definujú počas behu programu.

Po spustení programu pomocou príkazu RUN, bude príkaz READ čítať od počiatku.

Príkaz RUN je možné použiť len v priamom režime, to znamená, že sa nemôže nachádzať v programe.

Prerušenie vykonávania programu

Prerušiť bežiaci program je možné tromi sposobmi:

- súčasným stlačením !CTRL! a !S!, ktoré sposobí pozastavenie vykonávania bez výpisu. Pokračovanie vo vykonávaní pozastaveného programu je možné zatlačením ľubovoľného znaku na klávesnici okrem CTRL-C a CTRL-S.

- použitím príkazu STOP zapísanom v programe.
  Tento príkaz sposobí prerušenie vykonávania programu a výpis STOP AT LINE X, kde X je číslo riadku v programe, na ktorom bolo napísané STOP.

- Súčasným stlačením kláves !CTRL! a !C!, čo sposobí prerušenie programu a výpis:
  BREAK AT LINE číslo riadku

Pokračovanie vo vykonávaní prerušeného programu (CONT)

Príkaz CONT umožní pokračovať vo vykonávaní programu prerušeného príkazom STOP alebo zatlačením !CTRL! a !C!.

V prvom prípade pokračuje program príkazom, ktorý nasleduje bezprostredne za STOP, v druhom prípade pokračuje vo vykonávaní príkazu, ktorý bol prerušený. Výnimkou je príkaz WAIT, v ktorom, ak počas čakania stlačíme !CTRL!!C!, je čakanie prerušené a po príkaze CONT už nie je vykonaný zvyšok čakania, ale pokračuje sa príkazom nasledujúcim po príkaze WAIT.

Pokračovať v prerušenom programe príkazom CONT nie je možné, ak:

- bol program vykonávaný v priamom režime
- bol po prerušení program zmenený (oprava, vsunutie alebo vypustenie riadku)
- po vykonaní príkazu MEMEND
- po chybovom výpise
- po dobehnutí programu do konca

Zobrazenie programu (LIST)

Program, ktorý je uložený v pamäti, je možné zobraziť príkazom:

```
1.  LIST
2.  LIST     OD
3.  LIST     OD , DO      ;kde OD a DO sú čísla riadku
4.  LIST        , DO
```

V prvom prípade príkaz zobrazí celý program.
V druhom prípade zobrazí program počnúc od uvedeného čísla riadku.
V treťom prípade zobrazí časť programu s číslami riadkov z intervalu OD,DO .
V štvrtom prípade zobrazí program od počiatku až po uvedené číslo riadku.

Zobrazenie programu je možné pozastaviť (aby pro dlhšom výpise neprebehol po obrazovke) súčasným zatlačením kláves !CTRL! a !S! a opätovne spustiť zatlačením ľubovoľnej klávesy. Prerušiť zobrazenie je možné zatlačením !CTRL! a !C!.

Editovanie programu

Niekedy je potrebné modifikovať (opraviť) program, ktorý je už uložený pamäti. V predchádzajúcom texte sme videli, ako možeme vložiť, vymazať alebo prepísať celý riadok. GW BASIC však dovoľuje editovať aj vo vnútri už existujúceho riadku pomocou príkazu

EDIT číslo riadku !CR!

Po vykonaní príkazu EDIT sa príslušný riadok vypíše na obrazovke a používateľ má k dispozícii tieto editačné možnosti:

!→! alebo !CTRL!!X!	-	posun kurzora doprava
!←! alebo !CTRL!!H!	-	posun kurzora doľava
!↠! alebo !CTRL!!I!	-	posun kurzora o 8 znakov vpravo
!↖!	-	posun kurzora o 8 znakov vľavo

!↑! alebo !CTRL!!Z!	-	posun kurzora na začiatok editovaného riadku
!↓! alebo !CTRL!!K!	-	posun kurzora na koniec editovaného riadku
!CTRL!!D! alebo !DEL!	-	výmaz znaku, na ktorom je kurzor
!CTRL!!F!	-	vloženie znaku "medzera" na pozíciu kurzora; všetky znaky, počnúc znakom na ktorý ukazoval kurzor, sa posunú doprava
!ľubovoľná klávesa!	-	ľubovoľný znak, na ktorý ukazuje kurzor, môžeme prepísať iným znakom
!CR!	-	upravený riadok je uložený do pamäti a editácia je ukončená
!CTRL!!C!	-	ukončená editácia s tým, že zmeny sa neuložia, v pamäti zostane pôvodný tvar riadku

Príklad:

Napíšte:	Na obrazovke:
EDIT 20!CR!	EDIT 20
	20 A=SIN(B)*2
!→!!→!!→!X!→!COS!→!!→!!CTRL!!D!	
!→!!→!!CTRL!!F!)!CR!	20 X=COS(B*2)

7.1. Základné príkazy

7.1.1. Poznámky v programe (REM)

Pri písaní programu býva zvykom jednotlivé časti programu popísať, tzn. umiestniť medzi výkonné príkazy programu poznámky, ktoré sa pri vykonávaní programu ignorujú, ale zlepšujú "čitateľnosť" programu. Pre umiestnenie poznámky slúži príkaz

REM ľubovoľné znaky!CR!

Všetky znaky, ktoré sú umiestnené za REM až po znak !CR!,

ktorý ukončuje príkazový riadok, sú pri vykonávaní programu ignorované. Vyplýva z toho skutočnosť, že za príkazom REM sa v príkazovom riadku už nesmie nachádzať ďalší príkaz.

Opoznámkujme si program pre výpočet plochy kruhu, ktorý sme už niekoľkokrát použili. Predpokladajme, že program máme v pamäti a dopíšme ďalšie riadky programu.

Napíšte:

```
5 REM **PROGRAM PRE VYPOCET PLOCHY KRUHU**!CR!
15 REM R JE POLOMER KRUHU!CR!
25 REM PI JE LUDOLFOVO CISLO!CR!
35 REM VYPOCITANA PLOCHA JE V PREMENNEJ P!CR!
LIST
```

a na obrazovke vidíte opoznámkovaný program, ktorý po spustení vykonáva tú istú činnosť, akú sme si ukázali v odstavci 3.2.

7.1.2. Zobrazenie výsledkov (PRINT)

Pre zobrazenie výsledkov používame príkaz

PRINT zoznam

kde zoznam môže obsahovať aritmetické výrazy (tvorené pomocou číselných premenných, číselných funkcií, číselných konštánt), reťazcové výrazy (tvorené pomocou reťazcových premenných, reťazcových funkcií, reťazcových konštánt).

Jednotlivé výrazy v príkaze PRINT možeme od seba oddeliť pomocou oddeľovačov čiarka !,!, bodkočiarka !;!, TAB alebo AT. Ak nepoužijeme medzi výrazmi žiaden oddeľovač alebo výrazy oddelíme iba medzerou, má to taký istý účinok, ako keby sme ich oddelili bodkočiarkou.

Pre výstup hodnoty aritmetického výrazu platí:

Číslo vždy vpredu začína znamienkom, pričom pri kladnom čísle namiesto znamienka "+" sa zobrazuje medzera, pri zápornom čísle je znamienko zobrazené znakom "-" (mínus). Za číslom je vždy umiestnená jedna medzera.
Hodnota reťazcového výrazu nie je nijako zmenená ani dopĺňaná žiadnymi znakmi.

Napíšte:	Na obrazovke:
A=123:B=456!CR!	A=123:B=456
	READY
PRINT A;B !CR!	PRINT A;B
	123 456
	READY
PRINT A,B !CR!	PRINT A,B
	123 456
	READY
PRINT A TAB(9)B !CR!	PRINT A TAB(9)B
	123 456
	READY

Z príkladu vidieť, že jednotlivé oddeľovače spôsobia pri zobrazení rôzny počet medzier medzi položkami.

Oddeľovač bodkočiarka:
Ak sú jednotlivé položky zoznamu v príkaze PRINT oddelené bodkočiarkou, potom medzi jednotlivé zobrazované položky nie sú vkladané žiadne znaky. To znamená napr., že výstup z dvoch reťazcových výrazov oddelených bodkočiarkou na obrazovke bude súvislý reťazec znakov, ale že výstup z dvoch číselných výrazov bude oddelený:

- ak je hodnota druhého výrazu záporná, potom jednou medzerou
- ak je hodnota druhého výrazu kladná, potom dvoma medzerami

Ak dve položky v zozname oddelíme viac ako jednou bodkočiarkou, má to ten istý výsledok ako jedna bodkočiarka!

Taký istý výstup, ako pri oddelení položiek bodkočiarkou, dostaneme vtedy, keď jednotlivé položky oddelíme od seba jednou alebo viacerými medzerami alebo v prípade jednoznačnosti nie sú potrebné pre oddelenie žiadne medzery.

Napíšte:	Na obrazovke:
	READY
PRINT 50;-100;27.5!CR!	PRINT 50;-100;27.5
	50 -100 27.5
	READY
PRINT "VYSLEDOK JE";A!CR!	PRINT "VYSLEDOK JE";A
	VYSLEDOK JE 123

Oddeľovač čiarka:
Oddeľovač čiarka má za následok, že nasledujúca položka bude vypisovaná od začiatku ďalšej zóny. Pri výstupe s oddeľovačom čiarka je každý riadok obrazovky rozdelený na dve zóny po 16 znakov. Prvá zóna začína na prvej pozícii v riadku, druhá zóna začína na 17 pozícii v riadku. Pri výpise viacerých položiek oddelených čiarkou sa prechádza do ďalšieho riadku a začína sa zase od zóny č. 1.

Výstup pri oddelení položiek čiarkou prebieha nasledovne:
Keď Basic pri výstupe narazí na oddeľovač čiarka, potom:
- ak je kurzor na pozícii 1 až 16, potom sa presunie na pozíciu 17 v tom istom riadku a Basic testuje ďalšiu položku v zozname
- ak je kurzor na pozícii 17 až 32, potom sa presunie na pozíciu 1 v nasledujúcom riadku a Basic testuje ďalšiu položku v zozname.

Dve položky v zozname môžeme oddeliť aj viacerými čiarkami. pričom si treba uvedomiť, že každá čiarka znamená presun kurzora na začiatok ďalšej zóny.
Napríklad: ak je kurzor po výpise predchádzajúcej položky na pozícii 17 až 32, potom použitie troch čiarok pre oddelenie od nasledujúcej položky má za následok jeden voľný riadok medzi výstupmi.

Napíšte:	Na obrazovke:
	READY
PRINT 123,234,-345!CR!	PRINT 123,234,-345
	123 234
	-345

Oddeľovač TAB
syntax:
TAB (číslo pozície)

Tento oddeľovač premiestni kurzor v danom riadku na pozíciu, ktorú má výraz po vyčíslení. Pri tomto oddeľovači sú jednotlivé pozície v riadku označované 0 až 31.
Oddeľovačom TAB presunieme kurzor iba vtedy, ak číslo novej pozície je väčšie ako číslo súčasnej pozície kurzora. V opačnom prípade je oddeľovač TAB ignorovaný.

Ak číslo novej pozície v oddeľovači TAB je väčšie ako 31, potom je kurzor presunutý na pozíciu 31. Jednotlivé presuny pri oddeľovačoch čiarka alebo TAB sú uskutočňované pomocou kódu 18H (t.j. presuň kurzor bez výpisu medzery). To znamená, že ak je niečo zapísané alebo nakreslené na pozíciách, cez ktoré kurzor posúvame, potom toto zostane zachované. Pri kopírovaní výpisov na tlačiareň (pomocou príkazu LPRINTER) je kód 18H nahradený pri tlačení kódom 20H (to znamená, na tlačiareň sa vypisujú medzery).

Napíšte:	Na obrazovke:
PRINT "PRIEMER="TAB(9)20!CR!	PRINT "PRIEMER="TAB(9)20
	PRIEMER= 20
	READY

Napíšte:	Na obrazovke:
NEW!CR!	NEW
AUTO!CR!	AUTO
INPUT "DEN",D,"MESIAC",	10 INPUT "DEN",D,"MESIAC",
M,"ROK",R!CR!	M,"ROK",R

```
PRINT "DEN"TAB(10)
"MESIAC"TAB(22)"ROK"!CR!
PRINT D TAB(10)M TAB
(22)R!CR!
!CTRL!!C!

RUN!CR!
10!CR!
2!CR!
1984!CR!
```

```
20 PRINT "DEN"TAB(10)
"MESIAC"TAB(22)"ROK"
30 PRINT D TAB(10)M TAB
(22)R
40
READY
RUN
DEN:10
MESIAC:2
ROK:1984

DEN        MESIAC        ROK
 10          2           1984
```

Príkaz PRINT ukončuje zobrazovanie položiek vyslaním dvoch znakov (CR,LF), ktoré prestavia kurzor na začiatok ďalšieho riadku.

Napíšte:

```
NEW!CR!
AUTO!CR!
PRINT!CR!
PRINT!CR!
PRINT!CR!
PRINT "KOLKO RIADKOV JE
PRAZDNYCH?"!CR!
!CTRL!!C!

RUN!CR!
```

Na obrazovke:

```
READY
NEW
AUTO
10 PRINT
20 PRINT
30 PRINT
40 PRINT "KOLKO RIADKOV JE
PRAZDNYCH?"
50
READY
RUN


KOLKO RIADKOV JE  PRAZDNYCH?
```

Každý príkaz PRINT na riadkoch 10,20,30 spôsobí len prestavenie kurzora na nový riadok, teda tri prázdne riadky a konečne PRINT na 40. riadku spôsobí na 4. riadku obrazovky

tlač vlastného textu.

Niekedy je vhodné, aby nový príkaz PRINT pokračoval v tlači na tom istom riadku, kde predchádzajúci PRINT skončil.

Ukončenie príkazu PRINT bodkočiarkou, čiarkou, funkciou TAB alebo AT spôsobí, že nový PRINT tlačí na rovnakom riadku, kde predchádzajúci prestal, pričom si treba uvedomiť, že okrem potlačenia prechodu na nový riadok sa vykoná činnosť určená príslušným oddeľovačom.

To znamená, že uvedenie niektorého oddeľovača na konci príkazu PRINT potlačuje prechod na nový riadok (t.j. znaky CR, LF nie sú vyslané).

Napíšte:	Na obrazovke:
	READY
NEW!CR!	NEW
AUTO!CR!	AUTO
PRINT "JEDNA ";!CR!	10 PRINT "JEDNA ";
PRINT "DVA ";!CR!	20 PRINT "DVA ";
PRINT "TRI"!CR!	30 PRINT "TRI"
!CTRL!!C!	40
	READY
RUN!CR!	RUN
	JEDNA DVA TRI

Pomocou funkcie AT je možné umiestniť výstupnú položku na ľubovoľné znakové miesto na obrazovke. Každé miesto na obrazovke je určené dvoma parametrami: číslo riadku z intervalu $\langle 0,31 \rangle$ a číslo stĺpca z intervalu $\langle 0,31 \rangle$. Funkcia

AT X,Y kde $X,Y \rangle = 0$

spôsobí, že položka nasledujúca za AT bude zobrazená počnúc od pozície X,Y. Ak je X alebo Y väčšie ako 31, automaticky sa nastaví na 31.

Príklad:

Napíšte:	Na obrazovke:
A=123!CR!	A=123

```
                                READY
B=456!CR!                       B=456
                                READY
PRINT AT 0,0 A AT 10,0 B        123




                                456
                                READY

PRINT AT 0,30 A                                        23
                                READY

PRINT AT 0,0 "-123"             -123
                                READY

PRINT AT 90,90 A                .
                                .
                                .
                                                       123
                                READY

PRINT AT 0,90 A                                        123
                                READY

PRINT AT 90,0                   .
                                .
                                .
                                456
                                READY
```

Ak pred niektorú číselnú položku v príkaze PRINT napí-

šeme znak &, potom hodnota príslušnej číselnej položky je orezaná o desatinnú časť a ak je z intervalu 0 až 65535 je potom vypísaná v číslenej sústave, ktorá bola naposledy zvolená príkazom BASE. Ak hodnota položky nie je z uvedeného intervalu, potom je hlásená chyba.

Po zapnutí PP01 je pre výpis pomocou znaku & zvolená desiatková sústava.

Príkaz pre nastavenie sústavy:

BASE X

kde X je výraz, ktorého hodnota po orezaní desatinnej časti musí byť z intervalu 2 až 16.
Po vykonaní príkazu BASE platí zvolená sústava dovtedy, kým nie je vykonaný nový príkaz BASE.

Príklad pre výpis čísiel 0 až 20 v dvojkovej až šestnástkovej sústave:

```
10 FOR X=2 TO 16
20 BASE X
30 FOR CIS=0 TO 20
40 PRINT &CIS
50 NEXT CIS,X
```

Kopírovanie výstupov na tlačiareň

Basic obsahuje aj obslužný program pre tlačiareň, ktorá je pripojená cez medzistyk IRPR.

Po vykonaní príkazu LPRINTER sú všetky výstupy (okrem grafických) kopírované aj na tlačiareň.

Výstup na tlačiareň zrušíme príkazom CONSOLE.

Príklad:

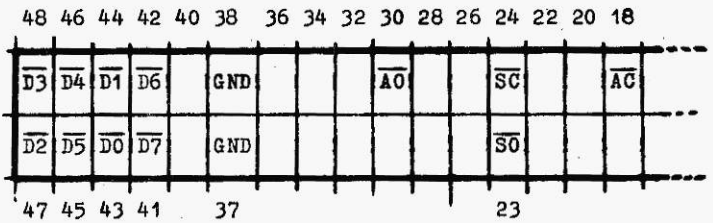
```
5 LPRINTER
10 FOR X=0 TO 50
20 PRINT X,X*X
30 NEXT X
40 CONSOLE
```

Tento príklad vypíše na zobrazovaciu jednotku (ďalej ZJ) aj na tlačiareň čísla od 0 do 50 a ich druhé mocniny.

Príkazy LPRINTER aj CONSOLE môžu byť vykonané v priamom aj príkazovom režime.

Ak použijeme príkaz LPRINTER a tlačiareň nie je pripojená alebo pripravená, potom voľba tlačiarne je zrušená a Basic vypíše chybu 22.

Rozloženie signálov na konektore IO pre pripojenie tlačiarne s medzistykom IRPR:



7.1.3. Priraďovací príkaz (LET)

Jednotlivým premenným môžeme v GBASICu priradiť konkrétne hodnoty pomocou priraďovacieho príkazu, ktorý má tvar:

LET premenná = aritmetický výraz
LET reťazcová premenná = reťazcový výraz

Hneď úvodom je treba povedať, že kľúčové slovo LET je v priraďovacom príkaze nepovinné - je možné ho vynechať.

Príklady:

```
LET X = SQR (Y^2+Z^2)
X=SQR (Y^2+Z^2)
A$="SLOVO"
```

Ak je vo výraze použitá numerická premenná skôr, ako jej bola priradená hodnota, počíta sa s hodnotou premennej 0. Ak je vo výraze použitá reťazcová premenná, ktorá nemala priradenú hodnotu, počíta sa s prázdnym reťazcom. Pri počítaní mocniny pomocou ^ nesmie byť hodnota argumentu rovná nule. Pokiaľ nasleduje za sebou niekoľko priradení, môžu byť zapísané do riadku a oddelené od seba čiarkou. Priradenia budú vykonané rýchlejšie ako pri oddelení dvojbodkou alebo ako pri písaní každého príkazu na nový riadok.

```
LET X=2, Y=75, A$="AUTO"
```

7.1.4. Príkaz skoku (GOTO)

V jednotlivých príkladoch sme videli, že príkazy programu sú v zásade vykonávané v poradí od riadku s najnižším číslom riadku až po riadok s najvyšším číslom riadku. GBASIC však poskytuje niekoľko možností, ako zmeniť toto prirodzené poradie vykonávania príkazov. Najjednoduchší z týchto príkazov je:

```
GOTO výraz
```

Najčastejšie sa namieste výraz zapíše priamo číslo riadku, ale obecne tam môže stáť ľubovoľný aritmetický výraz. Výraz sa vyčísli a prevedie na celočíselný (odrezaním prípadnej desatinnej časti čísla). Výsledok sa interpretuje ako číslo riadku a riadenie sa odovzdá na riadok s uvedeným číslom. Ak

riadok s číslom uvedeným za GOTO neexistuje, pokračuje sa na riadku s najbližším vyšším číslom, ak ani taký neexistuje, potom je hlásená chyba.

Príklad:
Vytlačte tabuľku druhých mocnín prirodzených čísel. Napíšte:

```
10 N=1
20 P=N*N
30 PRINT N,P
40 N=N+1
50 GOTO 20
RUN
```

1	1
2	4
3	9
4	16
5	25

!CTRL!!C!

V ďalšom príkaze využijeme možnosť, že za GOTO môže byť výraz na vetvenie programu.

Príklad:

```
10 X=100
20 X=X-10
30 GOTO X
40 STOP
50 PRINT " ?";:GOTO 20
60 PRINT " JASNE";:GOTO 20
70 PRINT " TO";:GOTO 20
80 PRINT " VAM";:GOTO 20
90 PRINT "JE";:GOTO 20
RUN
```

Na obrazovke:

```
JE VAM TO JASNE ?
 STOP AT LINE 40
```

7.1.5. Vstup údajov z klávesnice INPUT

Pre vstup údajov z klávesnice počas výpočtu programu slúži príkaz INPUT, ktorý má obecný tvar:

```
INPUT zoznam
```

kde položkou zoznamu môže byť meno premennej alebo reťazec. Jednotlivé položky zoznamu sú oddelené od seba čiarkou.

```
NEW
10 INPUT "POCET CISEL",N
```

Príkaz spôsobí výpis reťazca a výzvy pre vstup. Výzvou pre vstup je v prípade číselnej premennej otáznik (?) a v prípade reťazcovej premennej dvojbodka (:). Výpis výzvy môžeme potlačiť, ak za kľúčovým slovom INPUT uvedieme znak "%".

Napíšte:	Na obrazovke:
RUN!CR!	RUN
10	POCET CISEL? 10

Po vypísaní výzvy sú očakávané údaje z klávesnice. V prípade vstupu do číselnej premennej je možné zadať ľubovoľný aritmetický výraz, v prípade vstupu do reťazcovej premennej môže vstúpiť reťazec znakov. Ak sa vo vstupe dopustíme chyby (napr. ak aritmetický výraz nie je správne zapísaný), potom GBASIC vypíše číslo chyby, ktorá bola odhalená a požaduje údaj znovu.
Každý údaj pri vstupe musí byť odsúhlasený klávesou !CR!

Ak v príkaze INPUT nie je pred premennou reťazec, vypíše sa len výzva pre vstup.

NEW!CR!	NEW
	READY

```
10 INPUT A$!CR!                    10 INPUT A$
RUN!CR!                            RUN
 AHOJ                              : AHOJ
```

V prípade, že vstupuje reťazec do reťazcovej premennej, potom ako reťazec sú chápané všetky znaky, ktoré boli odsúhlasené klávesou !CR!.
Ak je chybne zadaný vstupný údaj, vypíše sa číslo chyby a nová výzva o vstup.
Reťazec, ktorý je položkou zoznamu príkazu INPUT sa po chybe vypíše len v prípade, že reťazec a nasledujúce meno premennej neboli oddelené čiarkou.
Každý vstupný údaj sa ukončí znakom !CR!.

```
NEW!CR!                            NEW
10 INPUT A,B,C!CR!                 10 INPUT A,B,C
RUN!CR!                            RUN
10!CR!                             ? 10
20!CR!                             ? 20
30!CR!                             ? 30
```

```
NEW
10 INPUT "MENO",A$,"PRIEZVISKO",B$
RUN
MENO: JAN!CR!
PRIEZVISKO: SMREK!CR!
```

Ak pri vykonávaní príkazu INPUT po vypísaní výzvy "?" alebo ":" stlačíme iba klávesu !CR!, má to za následok chybové hlásenie a znovu očakáva vstup.

S príkazom INPUT súvisí funkcia ERR. Táto funkcia nadobúda hodnotu 0 alebo 1.
Ak je hodnota funkcie ERR=1, potom v príkaze INPUT, ktorý sa vykonával naposledy, došlo ku nejakej chybe, t.j. že údaje, ktoré boli zadané z klávesnice, neboli správne a došlo ku chybovému výpisu.
Ak je hodnota funkcie ERR=0, potom v príkaze INPUT, ktorý sa

vykonával naposledy nedošlo ku chybe.

Okrem príkazu INPUT môžeme pre vstup znakov z klávesnice použiť aj funkciu KEY.

Tvar funkcie:
KEY

Funkcia pri svojom vyvolaní otestuje klávesnicu a ak je na nej stlačená nejaká klávesa, potom vráti jej kód, ináč vráti hodnotu 128.
Pri používaní funkcie KEY si musíme uvedomiť, že funkcia KEY vracia kód veľkých aj malých písmen (podľa toho, či držíme klávesu SHIFT alebo nie).

7.1.6. Čítanie údajov READ, DATA

Okrem príkazu vstupu z klávesnice je možné vstup dát realizovať pomocou dvojice príkazov READ, DATA. Obecný tvar týchto príkazov je:

READ zoznam premenných
DATA zoznam aritmetických alebo reťazcových výrazov

Príkaz READ špecifikuje premenné, ktorým sa budú priradzovať hodnoty. V zozname premenných môžu byť jednoduché, indexované aj reťazcové premenné. Oddelené sú od seba čiarkou.
Príkaz DATA obsahuje zoznam aritmetických alebo reťazcových výrazov, ktorých hodnoty budú priradené premenným z príkazu DATA. Každá položka z príkazu DATA musí čo do typu korešpondovať premennej z príkazu READ.

```
READY
AUTO
10 DATA 2.5,3
20 DATA 2.7E4,7
30 READ A,B
```

```
40 PRINT "A=";A,"B=";B
50 READ C,D
60 PRINT "C=";C,"D=";D
70 !CTRL!!C!
READY
RUN
A= 2.5           B= 3
C= 27000         D= 7

READY
```

V príklade boli použité dva príkazy READ a ku každému z nich príkaz DATA. Usporiadanie príkazov READ a DATA v programe však môže byť daleko voľnejšie. V systéme je definované interné ukazovátko, ktoré označuje prvok dát, ktorý bude čítaný. Na začiatku ukazuje na prvý prvok toho príkazu DATA, ktorý má najnižšie číslo riadku. Po prečítaní sa ukazovátko presunie na najbližší ďalší prvok príkazu DATA. Ak sú vyčerpané všetky prvky, presunie sa ukazovátko na prvý prvok nasledujúceho príkazu DATA. Ak už taký neexistuje, hlási sa chyba.

```
NEW
10 FOR I=1 TO 5
20 READ N
30 PRINT N,N^2
40 NEXT I
50 DATA 3,7,11
60 DATA 17,19
RUN
 3            8.99992
 7            48.9994
 11           120.996
 17           288.996
 19           360.99

READY
```

Ako vidíme z príkladu, údaje môžu byť usporiadané v jednom alebo niekoľkých príkazoch DATA. Príkaz DATA môže byť v programe kdekoľvek, pred alebo za príkazom READ.

```
NEW
10 DATA "PACI"
20 READ A$,B$
30 DATA "FIK"
40 PRINT A$+B$
RUN
PACIFIK

READY
```

7.1.7. Príkaz RESTORE

Ukazovátko, ktoré ukazuje na jednotlivé položky v príkaze DATA, môže byť prestavené na prvú položku ľubovoľného príkazu DATA v programe príkazom

RESTORE výraz

Výraz sa interpretuje ako číslo riadku. Ukazovátko sa prestaví na dané číslo riadku, ak takýto riadok neexistuje, potom sa prestaví na riadok s najbližším vyšším číslom, ak ani taký neexistuje, potom je hlásená chyba. To, či daný riadok obsahuje príkaz DATA, sa kontroluje až vtedy, keď príkaz READ ide priradovať hodnotu. Ak riadok, na ktorý sme postavili ukazovátko pomocou RESTORE, neobsahuje príkaz DATA, potom sa hľadá najbližší vyšší riadok s príkazom DATA. Ak taký neexistuje, potom je hlásená chyba. V prípade, že číslo riadku nie je uvedené, ukazovátko sa nastaví na prvý príkaz DATA v programe.

```
NEW
10 READ I,J,K
20 DATA 100,200,300,400,500,600
30 RESTORE
```

```
40 READ L,M,N,Q
50 PRINT I;J;K;L;M;N;Q
RUN
 100  200  300  100  200  300  400

READY
NEW
10 DATA 100,200,300
20 DATA 400,500,600
30 DATA 700,800,900
40 DATA 1000,1100,1200
50 FOR I=20 TO 40 STEP 10
60 READ A,B
70 RESTORE I
80 PRINT A,B
90 NEXT I
RUN
 100            200
 400            500
 700            800

READY
```

7.1.8. Časové zdržanie WAIT

Príkaz WAIT výraz spôsobí časové zdržanie výpočtu na počet časových jednotiek daných výrazom za WAIT. Výraz musí mať hodnotu z intervalu ⟨0-65535⟩. Časová jednotka je 100 milisekúnd. Hodnota výrazu=0 alebo prípad, že výraz za WAIT vobec nie je uvedený, sposobí maximálnu dobu čakania t.j. 1 hod 49 minút 13,6 sekundy. Časové zdržanie je realizované programovo a môže byť ovplyvnené z klávesnice klávesami !CTRL!!S! a !CTRL!!C!. Po zatlačení !CTRL!!S! program očakáva stlačenie ľubovoľnej dalšej klávesy a nepočíta čas čakania (čiže čakanie sa predlžuje). Zatlačenie ľubovoľnej ďalšej klávesy vráti riadenie príkazu WAIT. Po zatlačení

!CTRL!!C! je čakanie prerušené a na obrazovke vidíme výpis BREAK AT LINE číslo. Príkazom CONT je možné pokračovať vo vykonávaní programu s tým, že zbytok čakania je ignorovaný a pokračuje sa za príkazom WAIT.

Príklad: Hodiny

```
10 REM Hodiny sa menia každých 5 sekúnd
20 INPUT "VLOZ CAS (HOD,MIN,SEK)"H,M,S
30 PRINT H;":";M;":";S
40 S=S+5
50 IF S =60 THEN GOSUB 100
60 IF M =60 THEN GOSUB 200
70 IF H =24 THEN H=H-24
80 WAIT 50
90 GOTO 30
100 S=S-60:M=M+1:RETURN
200 M=M-60:H=H+1:RETURN
```

7.1.9. Zvukový generátor BEEP

Mikropočítač PP01 dokáže aj generovať tóny pomocou reproduktora umiestneného v klávesnici. Na túto skutočnosť nás upozorní aj to, že pípnutie sa ozve po každom správnom zatlačení klávesy. Ak súčasne stlačíme aj klávesu !SHIFT!, potom PP01 pípne vyšším tónom. V prípade, že nás pípanie po zatlačení klávies znervózňuje, možeme ho zakázať a to príkazom BEEP NO. Po vykonaní tohto príkazu je pípanie zakázané, až kým nepoužijeme príkaz BEEP. T.j. BEEP znovu povoľuje pípanie.

Pomocou príkazu BEEP výraz1,výraz2 možeme generovať tóny, pričom výraz1 je výraz, ktorý po vyčíslení určuje frekvenciu tónu a výraz2 po vyčíslení určuje dĺžku tónu. Počet dvojíc výraz1,výraz2 v jednom príkaze BEEP nie je obmedzený.

Pretože výraz1 určuje v skutočnosti dĺžku periódy a

výraz2 určuje počet periód, musíme si uvedomiť, že tón, ktorý má vyššiu frekvenciu bude trvať kratšie ako tón, ktorý má zadanú rovnakú dĺžku, ale nižšiu frekvenciu. Hodnota výraz1 a výraz2 musí byť z intervalu $\langle 0;255 \rangle$ a pre frekvenciu platí, že číslo 1 je najvyššia frekvencia a číslo 255 je najnižšia frekvencia, pričom 0 je tiež najnižšia frekvencia.

Varianty príkazu BEEP:

BEEP NO
BEEP
BEEP výraz1,výraz2

Príkaz BEEP NO neovplyvňuje príkaz BEEP výraz1,výraz2, to znamená, že môžeme generovať tóny, aj keď je zakázané pípanie po stlačení klávies.

7.1.10. Príkaz pre zastavenie bežiaceho programu STOP

Syntax: STOP

Po vykonaní príkazu STOP je program prerušený a vypíše sa správa:

STOP AT LINE X

kde X je číslo riadku, v ktorom je daný príkaz STOP umiestnený. Príkaz STOP musí byť posledným príkazom v riadku. Program môže obsahovať viacero príkazov STOP.

Príkaz STOP môže mať aj nasledujúcu syntax:

STOP reťazec

kde reťazec je reťazec znakov uzavretých v úvodzovkách alebo apostrofoch. Tento reťazec je po vykonaní príkazu STOP vypísaný a potom vypísaná správa:

STOP AT LINE X.

7.2. Vetvenie programu

7.2.1. Podmienený príkaz IF ... THEN

Podmienený príkaz dovoľuje viazať vykonanie príkazu alebo postupnosti príkazov na splnenie podmienky.

Obecný tvar podmieneného príkazu je:

IF logický výraz THEN príkaz

Ak je logický výraz za IF pravdivý, vykoná sa príkaz uvedený za THEN a ak sú uvedené ešte ďalšie príkazy v riadku, potom sa vykoná celý riadok až do konca. Ak je logický výraz nepravdivý, vo vykonávaní sa pokračuje na ďalšom riadku.

Príklad:

```
10 INPUT A$
20 IF A$="A" THEN PRINT "ANO":GOTO 40
30 PRINT "NIE":GOTO 10
40 STOP
```

Program sa pýta na znak a vypisuje NIE tak dlho, kým nie je na vstupe !A!. Vtedy vypíše ANO a zastaví sa.

Za THEN môže nasledovať ďalšie IF.

Napr.
IF X>3 THEN IF Y<5 THEN Z=8

Ak za slovom THEN nasleduje skok na konkrétne číslo riadku, je možné vynechať kľúčové slovo GOTO.

IF Y<3 THEN GOTO 70
možeme napísať v tvare:

IF Y<3 THEN 70
ale v príkaze
IF Y<4 THEN GOTO Z
už nie je možné vynechať GOTO.

7.2.2. Príkaz cyklu

V programe sa často vyskytne potreba opakovať určitú postupnosť operácií s meniacim sa parametrom. Jazyková konštrukcia, ktorá nám dovoľuje jednoduchým sposobom zapísať takúto postupnosť operácií, sa nazýva príkaz cyklu a v GBASICu má tvar:

```
FOR premenná = výraz1 TO výraz2 STEP výraz3
:
:
príkazy
:
:
NEXT premenná
```

Výraz 1, výraz 2 a výraz 3 musia byť aritmetické výrazy.

Na obrazovke:

```
10 FOR I=1 TO 10 STEP 2
20 PRINT I;
30 NEXT I
RUN                          1  3  5  7  9
                            READY
```

Z príkladu je vidieť, ako sa vykonáva príkaz cyklu.

1. Premenná za FOR (parameter cyklu) sa na začiatku nastaví na hodnotu výraz 1
2. Vykoná sa telo príkazu cyklu - všetky príkazy až po príkaz NEXT.
3. Parameter cyklu sa zvýši o hodnotu výraz 3.
4. Ak hodnota parametra cyklu presiahla hodnotu výraz 2,

pokračuje sa ďalším príkazom za NEXT.

5. V opačnom prípade sa pokračuje bodom 2.

V prípade, že výraz 3 má hodnotu 1, môžeme napísať príkaz cyklu takto:

FOR premenná = výraz 1 TO výraz 2

Príklad: Vytlačte čísla 1...9

```
10 FOR I=1 TO 9
20 PRINT I;
30 NEXT I
RUN
                                   1  2  3  4  5  6  7  8  9
                                  READY
```

Príklad:

```
10 FOR X=90 TO 40 STEP -10
20 GOTO X
40 STOP
50 PRINT " ?":GOTO 100
60 PRINT " JASNE";:GOTO 100
70 PRINT " JE TO";:GOTO 100
80 PRINT " UZ";:GOTO 100
90 PRINT "A TERAZ";:GOTO 100
100 NEXT X
RUN                                A TERAZ UZ JE TO JASNE ?
                                   READY
```

Je prepísanie príkladu z odstavca 7.1.4. Vidíme, že krok, o ktorý sa mení parameter cyklu môže byť aj záporné číslo.

Premennú, ktorá sa používa ako parameter cyklu, nemôžeme meniť vo vnútri cyklu priraďovacím príkazom bez toho, že by sa ovplyvnil priebeh vykonávania cyklu.

Príklad:

```
10 FOR I=1 TO 10
20 I=11
30 PRINT I
40 NEXT I
```

Tento cyklus sa vykoná len raz a vytlačí hodnotu 11.

Vloženie cyklu

Medzi príkazmi FOR - NEXT sa môžu obecne nachádzať ľubovoľné príkazy, teda aj ďalší príkaz cyklu.

Príklad:

```
10 FOR I=0 TO 1
20 FOR J=0 TO 1
30 PRINT I;J;I*J
40 NEXT J
50 NEXT I
RUN
```

```
 0  0  0
 0  1  0
 1  0  0
 1  1  1
```

V cykle, v ktorom je parameter premenná I (začína na riadku 10 a končí na riadku 50) je vložený cyklus s parametrom J (riadky 20 až 40). Pre každú hodnotu parametra vonkajšieho cyklu sa vykoná celý vnútorný cyklus. Dôležité je tu poradie príkazov NEXT, ktoré musí byť také, aby sa cykly neprekrývali. Prekrývanie cyklov nie je dovolené a vyvolá chybové hlásenie.

Príklad:

```
10 FOR I=0 TO 1
20 FOR J=0 TO 1
30 PRINT I;J;I*J                CHYBNÉ
40 NEXT I
```

```
50 NEXT J
RUN
 0  0  0
 1  0  0
 CHYBA 5
 50 NEXT J
 ?
 READY
```

GBASIC dovoľuje vnorenie cyklov do seba až do hĺbky 35. V prípade, že v programe končia dva alebo viac cyklov na rovnakom mieste, je možné zapísať v programe jediný príkaz NEXT s vymenovaním parametrov tých cyklov, ktoré na danom mieste končia.

Príklad:

```
10 FOR I=1 TO 10
20 FOR J=1 TO 10
:
:
50 NEXT J,I:REM POZOR NA PORADIE PREMENNYCH!
```

Poznámky k príkazu cyklu.

1. Nie je dovolené vstúpiť do vnútra cyklu mimo príkaz FOR

Príklad:

```
        10 FOR I=1 TO 5
        :
        40
        :
        60 NEXT I
        :
        100 GOTO 40
```

Vykonanie príkazu GOTO 40 sposobí chybové hlásenie, keď sa riadenie dostane na príkaz NEXT.

2. Z príkazu cyklu je možné vyskočiť (napr. podmieneným príkazom) aj pred vykonaním celého cyklu. Parameter cyklu má potom hodnotu definovanú posledným prechodom cyklu a do cyklu je možné sa z vonku vrátiť (neplatí predchádzajúce pravidlo).

7.2.3. Prepínač ON GOTO

Prepínač dovoľuje vetviť výpočet na niekoľko miest v programe. Obecný tvar príkazu:

ON výraz GOTO výraz1,výraz2,.....výrazN

Príkaz sa vykonáva tak, že sa vyčísli hodnota výrazu za ON. Táto hodnota musí byť z intervalu $\langle 1,N \rangle$, kde N je počet výrazov za GOTO. Riadenie sa odovzdá na číslo riadku vyhodnoteného ako K-ty v poradí za GOTO (kde K je hodnota výrazu za ON).

Napr.

```
50 ON X GOTO 70,120,200
```

Príkaz spôsobí odovzdanie riadenia na číslo riadku 70 ak X=1, 120 ak X=2, 200 ak X=3 a chybové hlásenie, ak X nie je z intervalu $\langle 1,3 \rangle$.

Príklad:

Pracovníci degustačného oddelenia pivovaru majú hodinovú mzdu závislú od počtu piv vypitých za týždeň podľa nasledujúcej tabuľky:

Počet piv	hodinová mzda
do 40	10 Kčs
41 - 50	12 Kčs
51 - a viac	14 Kčs

Nadčasové hodiny za týždeň, teda hodiny, ktoré presahujú 42,5 hodiny, sú zvýhodnené 50% príplatkom.

V mzdovej účtarni si zostavili pre výpočet mzdy pracovníkov oddelenia tento program:

```
vstupy: meno priezvisko
        počet piv
        počet hodín

príklad výpočtu

10 INPUT "MENO" M$, "PIVA" P, "HODINY" H
20 ON (SGN(TRUNC((P-45.5)/5))+2) GOTO 30,40,50
30 X=10:GOTO 60
40 X=12:GOTO 60
50 X=14
60 IF H<42.5 THEN PLAT=X*H:GOTO 80
70 PLAT=42.5*X+(H-42.5)*X*3/2
80 PRINT M$,PLAT:GOTO 10
```

7.3. Podprogramy a funkcie

Pri písaní programov sa častokrát používajú určité postupnosti príkazov, ktoré sa v danom programe alebo v rôznych programoch opakujú. Takéto postupnosti príkazov alebo operácií je vhodné zapísať vo forme podprogramov alebo funkcií a tieto potom opakovane využívať. GBASIC ma definovaný celý rad štandardných funkcií, ktoré už boli popísané napr. v odstavci 6.1.4. Štandardné funkcie nie je potrebné v programe definovať, stačí ich využívať.

7.3.1. Funkcie definované používateľom DEF FN

Predpokladajme, že v programe potrebujeme na rôznych miestach vypočítavať plochu kruhu pre rozne polomery. Budeme preto definovať funkciu, ktorá bude plochu kruhu počítať.

```
DEF FN P(R)=PI*R*R
```

a potom v programe túto funkciu používať.

```
A=FN P(10)
:
:
PRINT FN P(12.5)
:
:
X=SIN (A)+FN P(A)^2
:
:
```

Obecný tvar funkcie je:

DEF FN meno funkcie (parameter)=výraz

kde: - meno funkcie môže byť písmeno A-Z
- parameter je identifikátor premennej
- výraz je ľubovoľný aritmetický výraz

Príkaz DEF FN musí byť jediný v riadku.
Funkciu vyvolávame zapísaním

FN meno funkcie (skutočný parameter)

kde skutočný parameter môže byť ľubovoľný aritmetický výraz. Hodnota tohto aritmetického výrazu je dosadená do výrazu (určeného pri definovaní funkcie) ako hodnota premennej.

Používateľom definovaná funkcia musí byť pred prvým vyvolaním v programe definovaná. Po odštartovaní programu príkazom RUN sú všetky predtým definované funkcie zrušené a program musí znovu prechádzať cez definovanie funkcie. Funkcia raz definovaná môže byť v programe predefinovaná novým príkazom DEF FN.

7.3.2. Podprogramy GOSUB RETURN

V prípade, že sa bude v programe opakovane používať niekoľko príkazov, je výhodné vytvoriť podprogram. Podprogram je časť programu, ktorá je ukončená príkazom RETURN. Prvým príkazom podprogramu môže byť ľubovoľný príkaz, okrem príkazu NEXT. Podprogram vyvolávame príkazom

GOSUB výraz

Výraz sa vyhodnotí a riadenie sa odovzdá na príslušné číslo riadku, dané hodnotou výrazu. Všetky premenné, ktoré sú prístupné v hlavnom programe sú prístupné aj v podprograme a podprogram ich môže využívať, alebo meniť ich hodnoty.

Príklad: Nájdite najväčší spoločný deliteľ troch čísel (X,Y,Z).
Podprogram 100 nájde najväčší spoločný deliteľ dvoch čísel X,Y.

```
NEW
AUTO
10 INPUT "ZADAJ TRI CISLA",X,Y,Z
20 GOSUB 100
30 X=Z
40 GOSUB 100
50 PRINT "NAJVACSI SPOLOCNY DELITEL JE",Y
60 GOTO 10
100 IF X>=Y THEN 140
110 POM=X
120 X=Y
130 Y=POM
140 POM=X-X\Y*Y
150 IF POM> 0 THEN 130
160 RETURN
```

Podobne ako existuje príkaz ON GOTO, pozná GBASIC aj príkaz

ON GOSUB výraz

s jediným rozdielom, že riadenie sa odovzdá podprogramu.

Príklad: Napíšte program, ktorý po zadaní známky študenta číslom napíše známku slovom.

```
NEW
AUTO
10 INPUT "NAPISTE ZNAMKU (1 AZ 4)",X
20 ON X GOSUB 40,50,60,70
30 GOTO 10
40 PRINT "VYBORNE":RETURN
50 PRINT "VELMI DOBRE":RETURN
60 PRINT "DOBRE":RETURN
70 PRINT "NEDOSTATOCNE":RETURN
```

Príkazy ON GOSUB a ON GOTO musia byť posledné v riadku, pretože príkazy za nimi uvedené sa ignorujú.

8. POLIA

Jednoduché premenné, o ktorých sme hovorili v kapitole 6, mohli nadobúdať číselné hodnoty resp. v prípade reťazcovej premennej bola hodnota premennej reťazec. Často však pri spracovaní údajov pracujeme s celou množinou hodnot, ktorá je určitým spôsobom usporiadaná. Táto množina môže byť usporiadaná v jednom rozmere.

Napr.

a1 a2 a3 a4 a5

alebo v dvoch rozmeroch

Napr.

b11 b12 b13 b14
b21 b22 b23 b24

Každý prvok takejto množiny je jednoznačne určený svojim indexom (indexami). Takto definovanú množinu prvkov nazývame pole a jednotlivé prvky poľa sa nazývajú indexované premenné. GBASIC dovoľuje pracovať s jednorozmernými poľami (vektormi) a dvojrozmernými poľami (maticami).

8.1. Indexované premenné

S indexovanými premennými pracujeme rovnakým spôsobom ako s obyčajnými premennými. Index sa zapisuje za meno premennej v okrúhlych zátvorkách.

Napr.
```
A(1) = 1
A(2) = 4
A(3) = 9
A(4) = 16
```

V prípade, že sa jedná o indexovanú premennú z dvojrozmerného poľa, sú indexy oddelené od seba čiarkou.

Napr.

```
B(2,3) = 73
```

8.2. Definovanie poľa DIM

Skôr ako sa začne pracovať s indexovanými premennými, je nutné pole definovať a vymedziť hranice pre jeho indexy. Spodná hranica indexu je u polí vždy 1, takže pri definovaní poľa stačí určiť hornú hranicu indexu.

Obecný tvar definovania poľa je:

```
DIM zoznam
```

kde zoznam pozostáva z mien polí, za ktorými sú v zátvorkách horné hranice indexov.

Napr.

```
DIM A(5),B(2,4)
```

Tento príkaz vyhradí miesto v pamäti pre jednorozmerné pole A, ktoré má 5 prvkov a dvojrozmerné pole B, ktoré má 8 prvkov v dvoch riadkoch a štyroch stĺpcoch. Vzhľadom na skutočnosť, že pri definícii poľa sa vyhradzuje miesto v pamäti pre všetky prvky tohoto poľa, môže sa stať, že sa bude signalizovať chyba, ktorá signalizuje nedostatok pamäti, ak definujeme veľké polia. Pri definovaní poľa sú číselné premenné nastavené na hodnotu 0, reťazcové majú dĺžku reťazca = 0.

Podobne ako indexované číselné premenné GW-BASIC pracuje aj s indexovanými reťazcovými premennými. Pri vymedzení indexovanej reťazcovej premennej príkazom DIM sú všetky prvky premennej nastavené na nulovú dĺžku. Každý prvok indexovanej premennej zaberá potom toľko miesta, na koľko bol počas behu programu nastavený.

Každý prvok indexovanej reťazcovej premennej môže mať dĺžku najviac 255 bytov. Pri pokuse o prekročenie tejto dĺžky Basic ohlási chybu.

Príkaz práce s reťazcovými premennými:

```
DIM    A(5),A$(5)

A$(1)="ABC"
A$(2)="123456789ACEFH"
A$(3)="59A"
A$(4)="XYZ"
A$(5)="JA NO"
```

Premenné A,A(3),A$,A$(2) sú štyri úplne rozne premenné.

8.3. Nahranie obsahu premenných na magnetofón DSAVE, DLOAD

Podobným spôsobom, akým nahrávame program na mgf. pásku, možeme nahrať aj obsah poľa na mgf. pásku.

```
príkaz: DSAVE
syntax: DSAVE $čís,ip(
```

kde čís je číslo záznamu, ktorý bude vytvorený na páske a ip je identifikátor poľa, ktorého obsah chceme nahrať. Z uvedeného vyplýva, že jedným príkazom DSAVE môžeme nahrať obsah jedného poľa a vytvoríme tým jeden súbor na páske.

```
Príklad:
DSAVE $5,ABC$(
```

Týmto príkazom nahráme obsah indexovanej premennej ABC$(na pásku do súbgru č.5.

príkaz: DLOAD
syntax: DLOAD §čís

Týmto príkazom zosnímeme obsah poľa z pásky do pamäti PP01.

POZOR!
Tento príkaz zruší tabuľku premenných, čiže všetky doteraz použité premenné sa stanú nedefinované a obsah nahrávaný z pásky je prístupný pod tým istým identifikátorom, pod ktorým bol nahrávaný na pásku. To znamená, že príkaz DLOAD by mal byť jedným z prvých príkazov programu.

POZOR!
Ak po zosnímaní obsahu premennej do pamäti sa pokusíme príkazom DIM deklarovať premennú s tým istým identifikátorom, potom je hlásená chyba.

Príklad:
do záznamu č.5 zapíšeme obsah premennej, ktorá bola deklarovaná príkazom DIM ako TA§(90)

príkaz pre zápis:
DSAVE §5,TA§(
Príkazom
DLOAD §5
potom prečítame obsah záznamu č. 5 do tej istej premennej, z ktorej bol záznam č.5 zapísaný t.j. TA§(90).

Príklad:
Treba napísať nasledovný program:

```
10 DIM B§(26)
15 REM NAPLNENIE PREMENNYCH POLA PISMENAMI
20 FOR X=1 TO 26
30 B§(X)=CHR§(40H+X)
40 NEXT X
45 REM NAHRATIE POLA B§ NA PASKU
50 DSAVE §8,B§(
```

Spustíme magnetofón pre nahrávanie a stlačíme RUN. Po výpise READY previnieme magnetofón na začiatok nahrávky a zapíšeme nasledujúci program:

```
NEW
5 REM ZOSNIMANIE POLA Z PASKY DO PAMATI
10 DLOAD Ø8
15 CLEAR
17 REM VYPIS OBSAHU PREMENNYCH POLA
20 FOR Y=1 TO 26
30 PRINT BØ(Y);
40 NEXT Y
```

Program vypíše do riadku písmená:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
READY
```

9. PRÁCA S REŤAZCAMI

V odstavci 6.3 sme sa dozvedeli, že základnými reťazcovými výrazmi sú reťazec (t.j. text uzavretý medzi úvodzovkami alebo apostrofami) a reťazcová premenná.
Reťazce však možeme ďalej vytvárať:
- spájaním reťazcov
- reťazcovými funkciami

9.1. Spájanie reťazcov

Pre spojenie reťazcov používame operátor +.

Napr.

```
A$="PI"
B$=A$+"VO"
PRINT B$
PIVO
```

9.2. Reťazcové funkcie

9.2.1. Dĺžka reťazca

Dĺžku reťazca je možné určiť pomocou funkcie LEN.

Obecný tvar: LEN (premenná$ alebo reťazec)
Napr.

```
A$="ABS123"
PRINT LEN (A$)
 6
```

9.2.2. Podreťazce

V niektorých aplikáciách je potrebné vybrať časť reťazca. Pre uvedenú operáciu sú v GBASICu k dispozícii funkcie LEFT, RIGHT a MID.

Funkcia LEFT$ (premenná$ alebo reťazec,výraz) vráti reťazec, ktorý obsahuje prvých n znakov povodného reťezca, kde n je celočíselná hodnota výrazu.

Napr. ak použijeme premenné z predchádzajúceho príkladu

```
PRINT LEFT$(B$,2)
PI
```

Hodnota n výrazu musí byť z intervalu ⟨1,255⟩.

Funkcia RIGHT$ (premenná$ alebo reťazec, výraz) vráti reťazec, ktorý obsahuje posledných n znakov povodného reťazca, kde n je celočíselná hodnota výrazu. Ak je reťazec alebo premenná$ prázdna, potom je hlásená chyba.

Napr.

```
PRINT RIGHT$(B$,3);LEFT$(B$,2);B$
IVOPIPIVO
```

Funkcia MID$ (premenná$ alebo reťazec, výraz 1, výraz 2) vráti reťazec, ktorý pozostáva z výraz 2 znakov povodného reťazca počnúc od znaku výraz 1.

```
PRINT MID$(B$,2,2)
IV
```

Ak je hodnota výraz1 väčšia ako dĺžka reťazca, funkcia vráti nulový reťazec.
Ak hodnota výraz2 je väčšia ako zvyšok znakov po znaku výraz1, potom vráti všetky znaky reťazca od znaku výraz1.

Ak je výraz1=0, potom je hlásená chyba.
Ak je výraz2=0, potom vráti nulový reťazec.
Ak reťazec alebo premenná$ je prázdna, potom je hlásená chyba.

```
A$="PI"
B$="PIVO"
E$=" "
C$=A$+"JE"
D$=MID$(B$,2,3)+E$+RIGHT$(C$,2)+E$+"A"+E$+C$+E$+B$
PRINT D$
IVO JE A PIJE PIVO
```

Príklad:

Zistite, koľko písmen M sa nachádza v reťazci ABC$.

```
80 INPUT ABC$
90 K=0
100 FOR I=1 TO LEN (ABC$)
110 IF MID$ (ABC$,I,1)="M" THEN K=K+1
120 NEXT I
130 PRINT K
```

Príklad:

Obrátenie poradia znakov v reťazci. Reťazec B$ bude mať obrátené poradie znakov ako reťazec A$.

```
NEW
10 A$="KOBYLAM"
20 B$=""
30 FOR I=LEN(A$) TO 1 STEP-1
40 B$=B$+MID$(A$,I,1)
50 NEXT I
60 PRINT A$+"A"+B$
RUN
KOBYLAMAMALYBOK
```

9.3. Hodnoty reťazca CHR$,ASC

Funkcia CHR$(výraz) vráti jednoznakový reťazec, ktorého ASCII hodnota je hodnota výrazu. Táto hodnota musí byť z intervalu $\langle 0,255 \rangle$.

Funkciu je možné použiť napr. pre výstup nezobraziteľných znakov (napr. riadiace znaky) alebo výstup znakov s nastaveným najvyšším bitom.

Zobrazovacia jednotka reaguje na nasledovné riadiace znaky:

Hodnota znaku dekadicky	Hodnota znaku hexadecimálne	Význam
7	07	Pípnutie
8	08	Posun vľavo o znak
10	0A	Posun na nový riadok s rolovaním
11	0B	Posun na nový riad. bez rolovania
13	0D	Posun na začiatok riadku
24	18	Posun vpravo o znak
26	1A	Posun nahor o riadok
29	1D	Posun do ľavého horného rohu
30	1E	Výmaz riadku do konca
31	1F	Výmaz obrazovky do konca
01 až 06	01 až 06	výpis znakov z užívateľského generátora
32 až 95	20H až 5FH	výstup bežných znakov

Hodnoty znaku 1-6 sú rezervované pre užívateľom definované tvary znakov a sú tiež použiteľné pomocou funkcie CHR$ (pozri príkaz SETCHAR).

Výmaz obrazovky môžeme okrem príkazu CLEAR urobiť aj príkazom

```
PRINT CHR$(29)+CHR$(31)
```

Funkcia ASC (premenná$ alebo reťazec) vráti ASCII hodnotu prvého znaku reťazca. V prípade, že argumentom je prázdny reťazec, vráti hodnotu 256=100H.

```
PRINT ASC ("PRINT")

 80
```

9.4. Prevody VAL, STR$

Funkcie STR$ a VAL umožňujú konvertovanie reťazca na číselnú hodnotu a naopak. Majú význam najmä pri použití s inými reťazcovými funkciami pri úprave číselných, resp. reťazcových výstupov.

STR$ (výraz) transformuje numerickú hodnotu výrazu do reťazcového ekvivalentu. Reťazec sa vytvára podobným spôsobom ako v príkaze PRINT, ale medzery vpredu nevytvára.

```
NEW
10 INPUT "CELKOVA SUMA",X
20 KCS=TRUNC (X)
30 HAL=TRUNC (100*(X-KCS))
40 A$=STR$(KCS)+" KORUN A "+STR$(HAL)+" HAL "
50 PRINT "SUMA ";X;" JE ";A$
60 GOTO 10
RUN
CELKOVA SUMA? 310.30
SUMA 310.3 JE 310 KORUN a 30 HAL
```

Funkcia VAL (premenná$ alebo reťazec) konvertuje reťazcovú reprezentáciu čísla na číselnú konštantu.

```
PRINT VAL ("12")^2
143.996
```

Reťazec, ktorý konvertuje na číslo, musí začínať znakom "+" alebo "-" alebo " " alebo platná časť čísla. Konverzia končí, keď narazí na znak, ktorý nie je súčasťou čísla, alebo keď narazí na koniec reťazca.

10. GRAFIKA

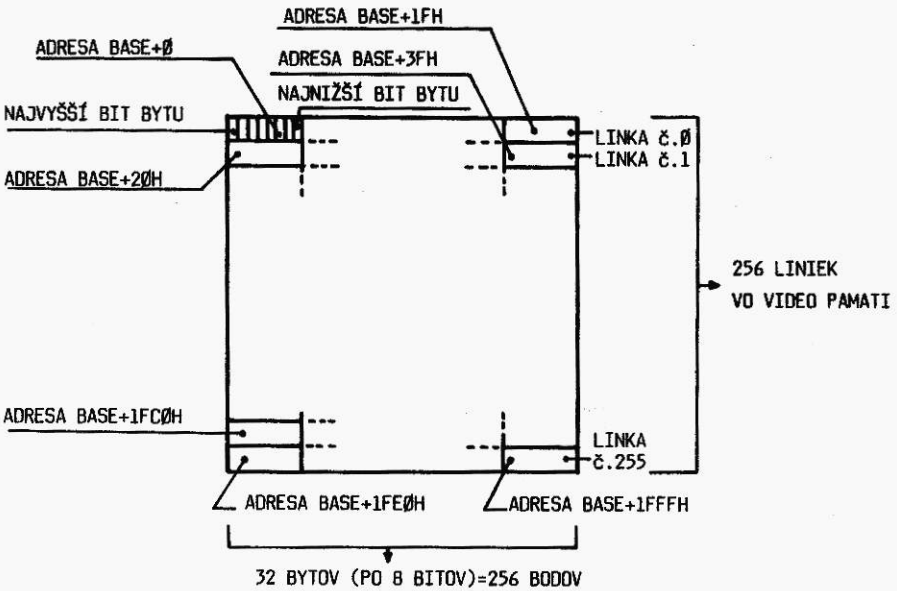
10.1. Úvod

Interpret GBASIC umožňuje spracovávať aj grafické príkazy. Všetky grafické príkazy sa môžu vykonávať v priamom i nepriamom režime. Grafickými príkazmi možno ovládať i voľbu farby na výpis i voľbu farby pozadia. Užívateľ má k dispozícii osem farieb. Číslovanie farieb pre pozadie a popredie je rovnaké. Pozadie je celá plocha obrazovky, na ktorú sa kreslia obrazce alebo vypisujú znaky. Popredie je vlastný výpis znakov alebo kreslenie obrazcov na obrazovku.

Po zapnutí osobného počítača PP01 je pozadie čierne a popredie biele, tzn. na čiernu plochu obrazovky sa vypisuje alebo kreslí bielou farbou. Pri výpisoch sa používa textový kurzor (zobrazuje sa ako štvorec), pri kreslení sa používa grafický kríž (nezobrazuje sa). Poloha textového kurzora a grafického kríža sa ovláda nezávisle.

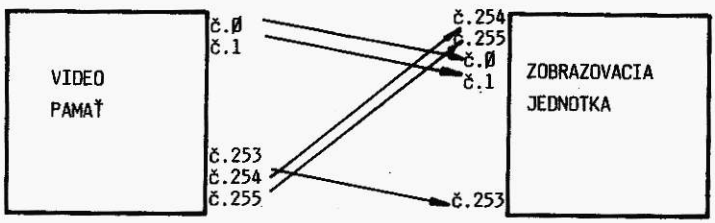
Technické vybavenie PP01 obsahuje video pamäť, ktorej obsah sa zobrazuje na monitore v rastri 256x256 bodov. Obsah tejto videopamäti nie je možné jednoduchým spôsobom čítať. Zápis do videopamäti prebieha nasledovne:
Bity, ktoré sú v zapisovanom byte nastavené na 1 sa zobrazia ako farebné body, pričom farba bodu je daná aktuálnou hodnotou tzv. registra farby. Bity, ktoré sú v zapisovanom byte nastavené na 0, pamäť neovplyvňujú. V Basicu register farby ovládame prostredníctvom príkazov INK a PAPER.

Pamäť video je organizovaná nasledovným spôsobom:



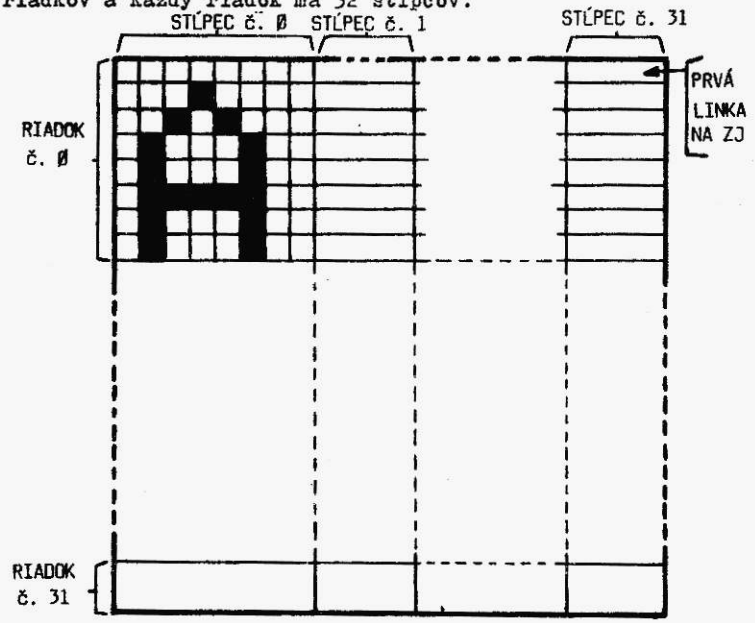
V PPO1 môžeme prostredníctvom registra ovládať, ktorá linka z video pamäti sa bude zobrazovať na ZJ ako prvá, čiže linka č.0 z video nemusí byť na ZJ zobrazená najvyššie (prvá).

Môže nastať napríklad situácia:



Z uvedeného vyplýva, že ak pomocou príkazu PEEK alebo prostredníctvom Z príkazu monitora budeme prezerať video pamäť, nemusí byť počiatok video pamäti zobrazený v ľavom hornom rohu monitora.

Pre výpisy je pamäť video organizovaná tak, že jeden znak zaberá 8 liniek na monitore a šírku jeden byte. Z toho potom vyplýva, že pri výpisoch je pamäť organizovaná: 32 riadkov a každý riadok má 32 stĺpcov.



Jeden znak pri výpisoch zaberá teda na šírku 8 bitov (1 byte) a na výšku 8 bytov. Vlastný znak zaberá najviac 5x7 bodov, zvyšok sa používa na oddelenie medzi riadkami a medzi písmenami v riadku. Tvary znakov sú zakódované v pamäti EPROM, nie je ich teda možné meniť jednoduchým spôsobom, ale užívateľ má možnosť vytvoriť si 6 vlastných znakov v plnom rastri (t.j. 8x8).

Príklad znaku z generátora Basicu:

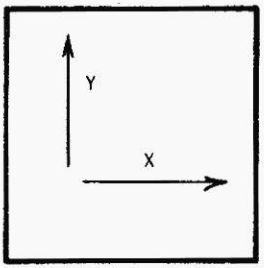
písmeno A je kódované:

00H, 10H, 28H, 44H, 44H, 7CH, 44H, 44H (pozri obrázok)

kde: 00H - hodnota bytu č.0 písmena A
10H - - " - č.1 - " -
28H - - " - č.2 - " -
44H - - " - č.3 - " -
44H - - " - č.4 - " -
7CH - - " - č.5 - " -
44H - - " - č.6 - " -
44H - - " - č.7 - " -

Pre kreslenie je pamäť organizovaná:
256 bodov na šírku a 256 bodov na výšku.

Súradnica x rastie v smere doprava, súradnica y rastie v smere nahor.



Basic obsahuje tzv. generátor priamky. Je to podprogram, ktorý vo video pamäti dokáže spojiť dva body priamkou. Oba body musia mať zadané súradnice x,y počiatku a konca priamky.

Generátor priamky používajú grafické príkazy, ktoré sú implementované v Basicu.

Príkaz pre vytvorenie vlastného znaku SETCHAR

syntax:
SETCHAR kód výraz0,výraz1,výraz2,výraz3,výraz4,výraz5,
výraz6,výraz7

kde kód je výraz, ktorého hodnota je číslo, pomocou ktorého sa budeme na daný znak odvolávať. Hodnota výrazu musí byť po orezaní desatinnej časti z intervalu 1 až 6.

výraz0 je výraz, ktorý po vyčíslení a orezaní desatinnej časti určí hodnotu bytu č.0 znaku

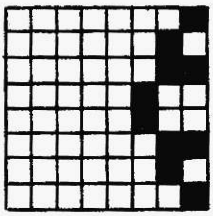
.

.

výraz7 je výraz, ktorý po vyčíslení a orezaní desatinnej časti určí hodnotu bytu č.7 znaku

Takto zadefinované znaky môžeme vypisovať pomocou funkcie

CHR$(1) až CHR$(6)



TVAR ZNAKU
č. 4

Príklad:

```
10 SETCHAR 4 1,2,3,4,4,3,2,1
20 PRINT CHR$(4)
```

Po zapnutí počítača sú hodnoty byte 0 až byte 7 každého zo znakov s kódom 1 až 6 nulové, to znamená, že reprezentujú medzeru. Pomocou príkazu SETCHAR môžeme v priamom alebo príkazovom režime nadefinovať nové hodnoty a tým nové tvary týchto znakov. Tvar daného znaku platí dovtedy, kým nie je ďalším príkazom SETCHAR s tým istým kódom predefinovaný!

Znaky zadefinované pomocou SETCHAR možu byť súčasťou reťazca a potom ich môžeme vypisovať pomocou príkazu PRINT.

```
10 SETCHAR 3 10H,10H,10H,10H,10H,10H,10H,10H
20 T$="ABC"+CHR$(3)+"DEF"
30 PRINT T$
```

10.2. Príkazy pre grafiku

Všetky príkazy pre grafiku sa môžu používať v priamom i v príkazovom režime. Na obrazovku je možné "súčasne" vypiso-

vať abecedno-číslicové znaky i používať príkazy pre grafiku, tzn. nevyžaduje sa inicializácia obrazovky pre kreslenie alebo pre výpisy.

Pri výpise sa na označenie pozície, kde sa na obrazovke bude vypisovať, používa textový kurzor zobrazený ako blikajúci štvorček. Pozícia pri kreslení je určená grafickým krížom, pričom grafický kríž sa môže nachádzať v dvoch stavoch, buď ako zdvihnutý alebo ako priklopený. Ani v jednom stave sa však grafický kríž nezobrazuje.

Pri použití grafického príkazu textový kurzor zhasne. Textový kurzor sa na obrazovke objaví až pri použití príkazu, ktorý spôsobí výpis alfanumerických znakov. Pozície textového kurzora a grafického kríža sú ovládané nezávisle.

Vymazanie obrazovky CLEAR

Tvar príkazu: CLEAR

CLEAR sa môže realizovať v príkazovom alebo priamom režime. Príkaz CLEAR vymaže displej bez toho, aby zmenil pozíciu grafického kríža alebo textového kurzora.

Určenie mierky SCALE

Príkaz SCALE patrí medzi najdôležitejšie príkazy pri grafickom zobrazovaní informácie. Príkaz realizuje nastavenie merítka na obrazovke v dvoch rôznych smeroch: vodorovný smer - os x, zvislý smer - os y. Jednotlivé parametre definujú minimálne a maximálne hodnoty v obidvoch smeroch, pričom musia byť splnené nasledujúce podmienky:

Xmin < Xmax
Ymin < Ymax
Xmax - Xmin < najväčšie zobraziteľné číslo v počítači
Ymax - Ymin < najväčšie zobraziteľné číslo v počítači

Tvar príkazu:

```
SCALE Xmin,Xmax,Ymin,Ymax
```

Ak sa pri vykonaní tohoto príkazu vyskytne chyba, musí sa vykonať opakované zadanie príkazu. Prvé dva parametre v príkaze SCALE definujú ľavú a pravú hranicu na obrazovke, posledné dva parametre v príkaze SCALE dolnú a hornú hranicu obrazovky. Pri nesplnení uvedených podmienok sa vyskytne chyba. Všetky parametre príkazu môžu byť vyjadrené ľubovoľným aritmetickým výrazom.

Príklady:

```
SCALE 0,10,0,10          mierky obidvoch osí sú od 0 do 10
SCALE -30,20,-10,20      mierka osi x je 50 jednotiek,
                         mierka osi y je 30 jednotiek
SCALE A*(-1),A,B*(-1),B  mierka osi x je 2*A jednotiek,
                         mierka osi y je 2*B jednotiek
```

Po vykonaní príkazu SCALE sa kresliaca plocha obrazovky považuje za obdĺžnik, ktorého vrcholy majú súradnice (Xmin, Ymin), (Xmin,Ymax), (Xmax,Ymax), (Xmax,Ymin), počínajúc ľavým dolným rohom a pokračujúc v smere pohybu hodinových ručičiek. Ak sa kreslí bod so súradnicami, ktoré sú mimo rozsahu príkazu SCALE, potom sa bod kreslí na príslušnom okraji obrazovky. To znamená, že súradnica, ktorá prekračuje rozsah, definovaný príkazom SCALE, sa redukuje na príslušnú maximálnu alebo minimálnu hodnotu.

Príkaz SCALE platí dovtedy, kým nie je vykonaný iný príkaz SCALE.

Kreslenie osi x

Tvar príkazu je:

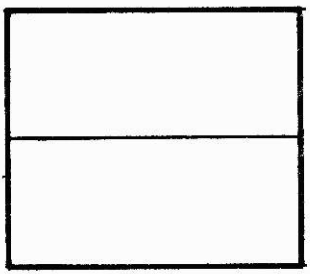
```
XAXIS výraz1, výraz2, výraz3
```

Príkaz XAXIS vykreslí na obrazovku x-ovú os v danom merítku, určenom príkazom SCALE. Bez ohľadu na to, či je grafický kríž priklopený alebo zdvihnutý, príkaz vykreslí priamku z bodu (výraz1, výraz3) do bodu (výraz2, výraz3), pričom výraz3 je y-ová súradnica začiatku a konca priamky. Po vykreslení priamky zostane grafický kríž na pozícii so súradnicami x=výraz2 a y=výraz3 v polohe zdvihnutý.

Ak je výraz1 < výraz2, kreslenie osi x prebieha v smere zľava doprava, v opačnom prípade sprava vľavo. V prípade, že výraz1 = výraz2, vykreslí sa bod so súradnicami (výraz1, výraz3).

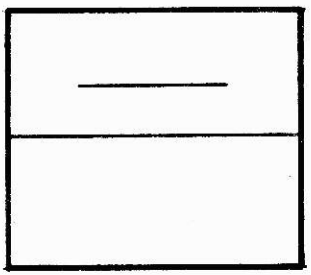
Príklad:

```
NEW!CR!
10 CLEAR!CR!
20 SCALE -10,10,-10,10!CR!
30 XAXIS -10,10,0!CR!
RUN!CR!
```



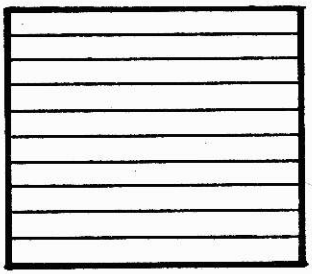
napíšte:

```
40 XAXIS -5,5,2!CR!
RUN!CR!
```



Príklad:

```
NEW
10 CLEAR
20 A=5
30 SCALE -A,A,-A,A
40 FOR K=-A TO A-1
50 XAXIS -A,A,K
60 NEXT K
RUN
```



Kreslenie osi y

Tvar príkazu je:

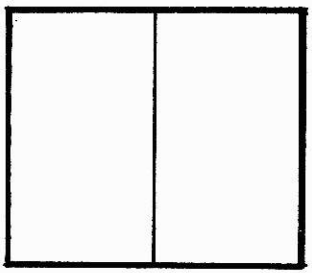
YAXIS výraz1,výraz2,výraz3

Príkaz YAXIS kreslí na obrazovku y-ovú os v mierke, zadanej v príkaze SCALE. Bez ohľadu na to, či je grafický kríž priklopený alebo zdvihnutý, príkaz vykreslí priamku z bodu (výraz3, výraz1) do bodu (výraz3, výraz2), pričom výraz3 je x-ová súradnica začiatku a konca priamky. Po vykreslení priamky zostane grafický kríž na pozíci so súradnicami x = výraz3 a y = výraz2 v polohe zdvihnutý.

Ak je výraz1 < výraz2, kreslenie osi y prebieha v smere zdola nahor, v opačnom prípade zhora nadol. V prípade, že výraz1 = výraz2, vykreslí sa bod so súradnicami (výraz3, výraz1).

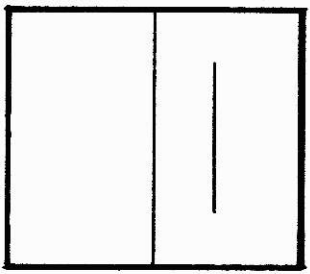
Príklad:

```
NEW!CR!
10 CLEAR!CR!
20 SCALE -10,10,-10,10!CR!
30 YAXIS -10,10,0!CR!
RUN!CR!
```



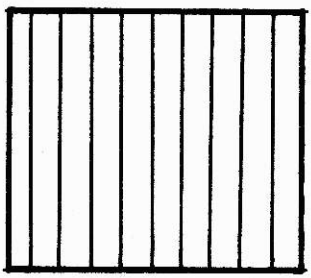
Napište:

```
40 YAXIS -5,5,2!CR!
RUN!CR!
```



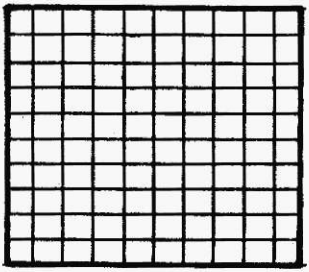
Příklad:

```
NEW
10 CLEAR
20 A=5
30 SCALE -A,A,-A,A
40 FOR K=-A TO A-1
60 YAXIS -A,A,K
70 NEXT K
RUN
```



Dopíšte:

```
50 XAXIS -A,A,K
RUN
```



Kreslenie priamky

Tvar príkazu je:

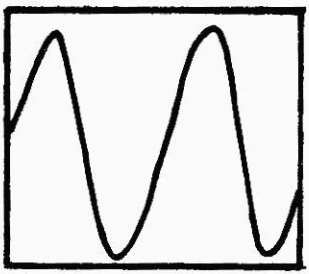
PLOT výraz1,výraz2

Príkaz PLOT vykreslí bod so súradnicami (výraz1,výraz2) alebo kreslí priamku z bodu, ktorý bol zobrazený posledným grafickým príkazom, do bodu so súradnicami (výraz1, výraz2). Ak bol grafický kríž pred vykonaním príkazu PLOT zdvihnutý, príkaz PLOT presunie grafický kríž do bodu so súradnicami (výraz1, výraz2), vytvorí bod na danej pozícii a grafický kríž zostane priklopený. Ak bol grafický kríž pred vykonaním príkazu PLOT priklopený, príkaz PLOT kreslí priamku z bodu, ktorý bol zobrazený posledným grafickým príkazom do bodu so súradnicami (výraz1, výraz2) a grafický kríž zostane priklopený v bode so súradnicami (výraz1, výraz2).

Príklad:

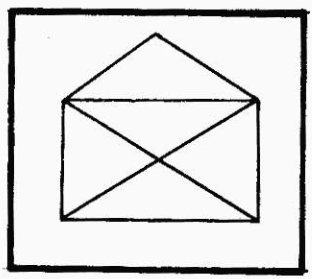
```
NEW
10 CLEAR
20 A=6.5
30 SCALE -A,A,-1.1,1.1
```

```
40 FOR K=-A TO A STEP 0.5
50 PLOT K,SIN(K)
60 NEXT K
RUN
```



Príklad:

```
NEW
10 CLEAR
20 SCALE 0,20,0,15
30 FOR I=1 TO 11
40 READ X,Y
50 PLOT X,Y
70 NEXT I
80 DATA 8,5,8,9,10,11,12,9,8,9
90 DATA 10,7,12,9,12,5,10,7,8,5
100 DATA 12,5
RUN
```



PENUP

Príkaz PENUP zdvihne grafický kríž v bode, ktorý bol zobrazený posledným grafickým príkazom.

Príkazy na presun grafického kríža:
Príkaz MOVE

Tvar príkazu je:
MOVE výraz1,výraz2

Príkaz MOVE presunie grafický kríž na pozíciu so súradnicami (výraz1, výraz2), pričom grafický kríž zostane v stave zdvihnutý. Príkazom možno presúvať grafický kríž bez kreslenia priamok na obrazovke. Začiatok kreslenia možno určiť v celej rovine, určenej príkazom SCALE, tzn. začiatok kreslenia určený súradnicami (výraz1, výraz2) sa určuje vzhľadom k mierke určenej posledným vykonávaným príkazom SCALE.

Príklad:
```
30 MOVE 3,5
60 MOVE 25,50
```

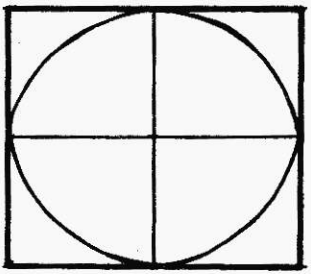
Príkaz DRAW

Tvar príkazu je:
DRAW výraz1,výraz2

Príkaz DRAW bez ohľadu na to, či je grafický kríž priklopený alebo zdvihnutý, kreslí priamku z pozície zobrazenej posledným grafickým príkazom do bodu so súradnicami (výraz1, výraz2). Súradnice sú určené vzhľadom k mierke určenej posledným príkazom SCALE.

Napíšte program:
```
5 CLEAR
```

```
10 SCALE -1,1,-1,1
20 XAXIS -1,1,0
30 YAXIS -1,1,0
40 DEG
50 FOR X=0 TO 360 STEP 5
60 DRAW SIN(X),COS(X)
70 NEXT X
RUN
```



IMOVE

Príkaz IMOVE zväčšuje schopnosť GBASIC-u presúvať grafický kríž na obrazovke v mierke, udanej posledným vykonaným príkazom SCALE.

Tvar príkazu je:

IMOVE výraz1,výraz2

Príkaz IMOVE presúva grafický kríž relatívne k pôvodnej pozícii grafického kríža, v ktorej sa nachádza grafický kríž po poslednom vykonanom grafickom príkaze. Ak sa grafický kríž pred vykonaním príkazu nachádzal v pozícii so súradnicami (výraz3, výraz4), vykonaním príkazu IMOVE sa grafický kríž presunie do bodu so súradnicami (výraz1+výraz3, výraz2+výraz4), pričom zostane zdvihnutý.

Príklady:

```
30 IMOVE 1,3
```

Ak pôvodná pozícia grafického kríža bola (x,y), príkaz v riadku 30 presunie kríž v smere osi x o jednu jednotku mierky vpravo, v smere osi y o tri jednotky mierky smerom hore, čiže do bodu so súradnicami (x+1,y+3).

```
40 IMOVE -6,-2
```

Ak pôvodná pozícia grafického kríža bola (x,y), príkaz v riadku 40 presunie kríž v smere osi x o šesť jednotiek mierky doľava, v smere osi y o dve jednotky mierky smerom dole, tzn. do bodu so súradnicami (x-6,y-2).

IDRAW

Príkaz IDRAW zväčšuje schopnosť interpretu kreslenia priamok v mierke udanej posledným vykonaným príkazom SCALE.

Tvar príkazu je:

```
IDRAW výraz1,výraz2
```

Príkaz IDRAW bez ohľadu na to, či je grafický kríž po poslednom vykonanom grafickom príkaze zdvihnutý alebo priklopený, kreslí priamku z pôvodnej pozície do novej pozície, pričom grafický kríž zostane priklopený. Pôvodná pozícia je pozícia, v ktorej sa nachádza grafický kríž po poslednom vykonanom grafickom príkaze. Ak sa grafický kríž pred vykonaním príkazu IDRAW nachádza na pozícii so súradnicami (výraz3,výraz4), vykonaním príkazu IDRAW sa vykreslí priamka z pozície (výraz3, výraz4) do pozície so súradnicami (výraz1+výraz3, výraz2+výraz4).

Všetky parametre v príkaze môžu byť vyjadrené ľubovoľným aritmetickým výrazom.

Príklad:

```
100 IDRAW 1,5
```

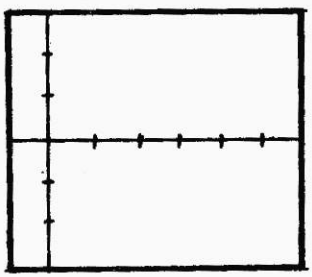
Ak pôvodná pozícia grafického kríža bola (x,y), príkaz v riadku 100 vykreslí priamku z bodu (x,y) do bodu so súradnicami (x+1,y+5) v mierke udanej príkazom SCALE.

110 IDRAW -5,2

Ak pôvodná pozícia grafického kríža bola (x,y), príkaz v riadku 110 vykreslí priamku z bodu (x,y) do bodu so súradnicami (x-5,y+2) v mierke udanej príkazom SCALE.

Napíšte program:

```
10 CLEAR
20 SCALE -2,30,-6,6
30 XAXIS -2,30,0
40 YAXIS -6,6,0
50 FOR X=0 TO 30 STEP 5
60 MOVE X,.2
70 IDRAW 0,-.4
80 NEXT X
90 FOR Y=-6 TO 6 STEP 2
100 MOVE -.5,Y
110 IDRAW 1,0
120 NEXT Y
RUN
```



10.3. Príkazy pre voľbu farby

Interpret GBASIC umožňuje pracovať aj s farbou. Pozadie predstavuje plochu obrazovky, na ktorú sa kreslí alebo vypisuje text. Jednotlivé farby majú nasledovné označenia:

0 - čierna
1 - modrá
2 - červená
3 - purpurová
4 - zelená
5 - bledomodrá
6 - žltá
7 - biela

INK výraz

Príkaz INK určuje, akou farbou sa bude kresliť alebo vypisovať text. Hodnota vyrazu musí byť z intervalu $\langle 0,7 \rangle$.

PAPER výraz

Príkaz PAPER určuje farbu pozadia. Hodnota ľubovoľného aritmetického výrazu musí byť z intervalu $\langle 0,7 \rangle$.

10.4. Príkaz pre vykreslenie motívu určeného reťazcom - BPLOT

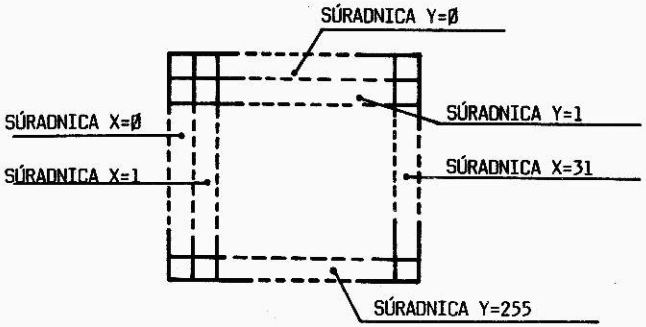
Príkaz BPLOT vykresľuje motív, pričom začiatok motívu je určený polohou tzv. znakového kurzora. Polohu znakového kurzora možeme ovládať len príkazom BMOVE.

Syntax: **BMOVE** X,Y

kde X,Y sú aritmetické výrazy

a X je po vyčíslení a orezaní desatinnej časti z intervalu ⟨0,31⟩
a Y je po vyčeíslení a orezaní desatinnej časti z intervalu ⟨0,255⟩

Pri použití príkazov BMOVE, BPLOT je pamäť rozdelená nasledovným spôsobom:



Pomocou príkazu BPLOT sa prepisuje obsah zadaného reťazca do pamäti video, pričom sa začína od pozície, kde je znakový kurzor a pokračuje sa smerom vpravo a dole, pričom sa postupne prepisujú byty obsahu reťazca do pamäti video. Tam, kde má byte obsahu reťazca bity nastavené na 1, vzniká bodka, kde sú nulové, zostáva obsah pamäti VIDEO bezo zmeny.

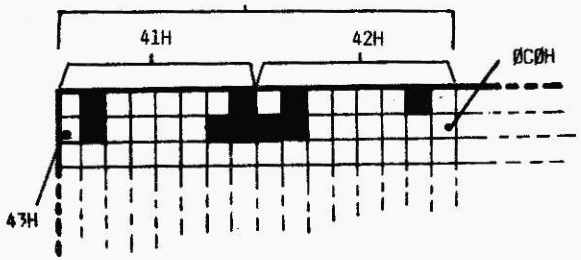
Syntax: BPLOT retvýraz,šírka

kde retvýraz je reťazcový výraz, ktorého výsledný reťazec sa bude byte po byte prepisovať do pamäti video
a šírka je číselný výraz, ktorého hodnota po vyčíslení a orezaní desatinnej časti, musí byť z intervalu 1,32 a určuje, koľko bytov z daného reťazca má byť prepísaných v jednom riadku pamäti video. Vzhľadom na to, že ASCII znaky majú 7-bitový kód, musíme, ak to je potrebné použiť pre nastavenie najvyššieho bitu, použiť funkciu CHRØ.

Príklad:
```
BMOVE 0,0
```

```
BPLOT "ABC"+CHR$(0C0H),2
```

do pamäti video budú prepísané postupne kódy 41H, 42H, 43H, 0C0H.



Príklad pre vykreslenie vodorovnej čiary v pozícii zadanej v premennej POZIC.

```
3 INPUT POZIC
5 T$=CHR$(0FFH)
10 FOR X=1 TO 31
20 T$=T$+CHR$(0FFH)
30 NEXT X
40 BMOVE 0,POZIC
50 BPLOT T$,32
```

Ak zvolená šírka vykreslenia podľa reťazca je väčšia ako je k dispozícii pozícii vpravo od polohy znakového kurzora, alebo
ak je dĺžka reťazca väčšia ako počet pozícií smerom dole, pričom berieme do úvahy aj zvolenú šírku vykresľovania,

potom časť reťazca (ktorá padne mimo obrazu, t.j. súradnice X>31 alebo Y>255) je ignorovaná.

Príklad:

```
10 A$="1234567890"
20 BMOVE 30,5
30 BPLOT A$,3
```

to znamená, že do pamäti video by mali byť prepisované hodnoty:
31H, 32H, 33H, 34H, 35H, 36H, 37H, 38H, 39H, 30H

V skutočnosti sú prepisované hodnoty:

na pozícii 30,5 hodnota 31H
na pozícii 31,5 hodnota 32H
hodnota 33H je ignorovaná
na pozícii 30,6 hodnota 34H
na pozícii 31,6 hodnota 35H
hodnota 36H je ignorovaná
na pozícii 30,7 hodnota 37H
na pozícii 31,7 hodnota 38H
hodnota 39H je ignorovaná
na pozícii 30,8 hodnota 30H

Pri práci s GBASICom si treba uvedomiť, že existujú 3 od seba navzájom nezávislé kurzory:

textový - ovládaný pomocou PRINT, ...
grafický - ovládaný pomocou MOVE, IMOVE, PLOT, DRAW, ..
znakový - BMOVE, BPLOT

11. STROJOVO ORIENTOVANÉ PRÍKAZY A FUNKCIE

11.1. PEEK

Tvar funkcie je:

PEEK (výraz)

Funkcia PEEK vráti obsah pamäťového miesta zadaného výrazom. Funkcia vždy vráti 8-bitovú hodnotu, tzn. celé číslo od 0 do 255. Výraz, určujúci adresu pamäťového miesta, musí byť z intervalu 0 až 65535. Výraz musí byť aritmetického typu. Ak je reťazcového typu, dôjde k chybe. Reálna hodnota výrazu sa pred vykonaním príkazu prevedie na celé číslo.

Príklad:

```
100 ADR1=PEEK(3000)
110 ADR2=PEEK(1000*PI)
120 X=PEEK(4000H)
```

11.2. POKE

Tvar príkazu je:

POKE výraz1, výraz2

Príkaz POKE vykonáva vloženie hodnoty určenej parametrom výraz2 na pamäťové miesto, ktorého adresa je zadaná parametrom výraz1. Parameter, určujúci adresu pamäťového miesta, musí byť z intervalu 0 až 65535. Príkaz POKE vkladá na pamäťové miesto 8-bitovú hodnotu, tzn. parameter výraz2 môže byť z intervalu 0 až 255, ináč dojde k chybe.

Obidva parametre musia byť aritmetického typu. Pri použití reťazcového typu dojde k chybe. Reálne hodnoty parametrov sa pred vykonaním prevedú na celé čísla.

Jedným príkazom POKE možno vkladať však i viac hodnot na pamäťové miesta, ktorých adresy narastajú o hodnotu 1.

Tvar príkazu je:

POKE výraz1, výraz2, výraz3, ..., výrazN

Pre parametre výraz1 až výrazN platia podmienky, uvedené pri príkaze POKE s dvoma parametrami. Príkaz POKE, ktorý má realizovať vkladanie viacerých hodnot, pracuje tak, že na pamäťové miesto s adresou výraz1 vloží hodnotu výraz2, na adresu výraz1+1 vloží hodnotu výraz3, na adresu výraz1+2 hodnotu výraz4 atď.

Príklad:
POKE 9900H,20,ASC("P"),40,80

Po vykonaní uvedeného príkladu budú na pamäťových miestach s adresami 9900H až 9903H nasledujúce hodnoty:

adresa pamäťového miesta	hodnota (dekadicky)
:	
9900H	20
9901H	80
9902H	40
9903H	80
:	

! UPOZORNENIE ! Príkaz POKE je veľmi nebezpečný príkaz, pretože nesprávnou voľbou adresy pamäťového miesta sa ľahko možu prepísať pracovné bunky interpretu G-BASIC v pamäti RAM (pozri mapu pamäti). Adresa pamäťového miesta, kde sa majú zapisovať hodnoty, sa musí starostlivo zvážiť.

11.3. OUT

Tvar príkazu je:

```
OUT výraz1, výraz2
```

Príkaz OUT vyšle na výstupný port, určený hodnotou výraz1, hodnotu parametra výraz2. Obidva parametre musia byť aritmetického typu, v opačnom prípade dôjde k chybe. Reálne hodnoty parametrov sa pred vykonaním príkazu prevedú na celočíselné. Hodnoty obidvoch parametrov musia byť z rozsahu 0 až 255.

V príkaze OUT musia byť určené obidva parametre. Počet takýchto dvojíc v jednom príkaze OUT je neobmedzený.

Príklad:

```
OUT 2,16
OUT PORT1,(A*A-2)/3,PORT2,(A*A-3)/2
```

11.4. INP

Tvar funkcie je:

```
INP (výraz)
```

Funkcia INP vráti hodnotu vstupného portu, určeného hodnotou výrazu. Parameter musí byť aritmetického typu, ak je reťazcového typu, dôjde k chybe. Reálna hodnota parametra sa prevedie na celočíselnú hodnotu. Hodnota parametra výraz musí byť z rozsahu 0 až 255.

Príklad:

```
PRINT INP(PORT1)
```

Funkcia INP môže mať aj tvar:

```
INP(výraz,maska)
```

V tomto prípade je s hodnotou vyčítanou z portu (jeho adresu dá výraz po vyhodnotení) prevedený logický súčin s hodnotou,

ktorú má maska (maska môže byť aritmetický výraz s hodnotou 0-255).

Príklad:
Program, ktorý je zapísaný v kapitole 11.10 v strojovom kóde a v assembleri, vyzerá v Basicu nasledovne:

```
1000 REM VYSTUPNY ZNAK V PREMENNEJ X
2000 IF INP(0C5H,10H)=0 THEN 2000
3000 IF X=18H THEN X=20H
4000 OUT 0C4H,255-X
5000 OUT 0C7H,1,0C7H,0
6000 RETURN
```

11.5. CALL

Pri vytváraní programov určitého zamerania môže byť niekedy výhodné uskutočniť časť programu rutinou, napísanou v assembleri. Týmto spôsobom sa môže urýchliť výpočet, optimálne využívať pamäť, alebo využívať už hotové rutiny, napísané v assembleri. Interpret G-BASIC umožňuje využitie rutín napísaných v assembleri funkciou CALL. Tvar funkcie je:

CALL(výraz)

Funkcia CALL umožňuje spojenie s programom napísaným v assembleri, ktorý je uložený od adresy určenej hodnotou parametra výraz. Hodnota parametra výraz musí byť z intervalu 0 až 65535. Reálna hodnota sa pred vykonaním funkcie prevedie na celočíselnú hodnotu.

V určitých prípadoch je nutné previesť určitú hodnotu do rutiny napísanej v assembleri, prípadne previesť hodnotu z rutiny do programu. Tento prenos sa prevádza cez dve dvojice registrov BC a HL. Prenos hodnoty do rutiny sa realizuje cez dvojicu registrov BC, prenos hodnoty do programu cez dvojicu registrov HL. Tvar funkcie je:

CALL (výraz1,výraz2)

Funkcia CALL napísaná v takomto tvare umožní spojenie s programom, napísaným v assembleri, ktorý je uložený od adresy určenej hodnotou parametra výraz1. Návrat do programu napísaného v G-BASICu sa vykoná po vykonaní assemblerovskej inštrukcie návratu (RET, RNZ,...). Hodnota parametra výraz2 sa prenesie cez dvojicu registrov BC do rutiny napísanej v assembleri. Hodnoty obidvoch parametrov musia byť z intervalu 0 až 65535. Reálne hodnoty parametrov sa pred vykonaním funkcie prevedú na celočíselné hodnoty. Parametre musia byť aritmetického typu, v opačnom prípade dojde k chybe.

Príklad:

```
10 POKE 9820H,CB 7H,79H,1FH,6FH,26H,0,0C9H
15 FOR X=0 TO 200
20 PRINT CALL(9820H,X)
30 NEXT X
```

Príkazom na riadku 10 zapíšeme od adresy 9820H program v strojovom kóde, ktorý posunie obsah C registra o jedno miesto vpravo (čiže ho vydelí dvoma) a výsledok presunie do registrov H a L.

Príkaz PRINT na riadku 20 vypisuje teda hodnoty premennej X vydelené dvoma.

11.6. Bitové funkcie BINAND, BINOR, BINNOT

Pri bitových funkciách sa hodnoty operandov vždy prevádzajú do 16 bitového binárneho tvaru.

Funkcia BINAND

Tvar funkcie:

BINAND(výraz1,výraz2)

Funkcia BINAND vykoná medzi oboma operandami logický súčin medzi odpovedajúcimi bitmi.
Výraz1, výraz2 sú aritmetické výrazy, ktorých hodnota po vyčíslení a orezaní desatinnej časti musí byť z intervalu 0 až 65535.

Funkcia BINOR

Tvar funkcie:

BINOR(výraz1,výraz2)

Funkcia BINOR vykoná medzi oboma operandami logický súčet medzi odpovedajúcimi bitmi. Pre výraz1, výraz2 platí to isté ako u funkcie BINAND.

Funkcia BINNOT

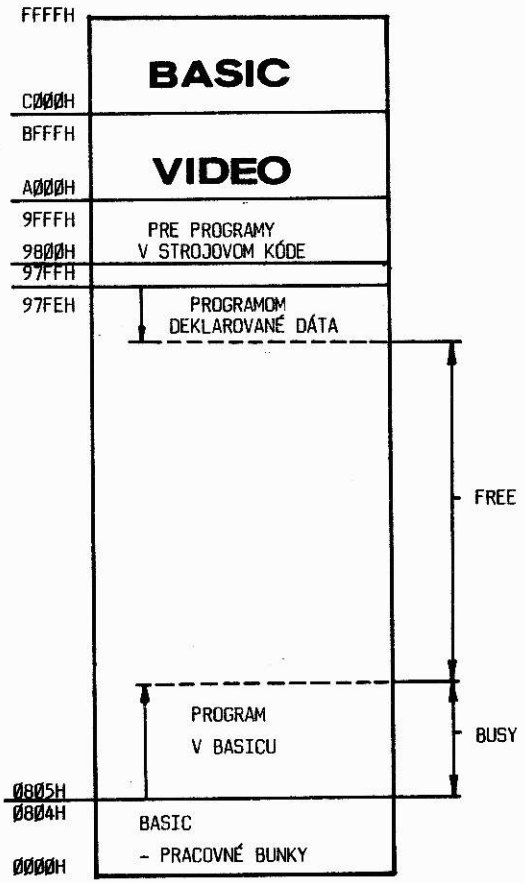
Tvar funkcie:

BINNOT(výraz)

Funkcia BINNOT vykoná s operandom negáciu, to znamená, že hodnotu všetkých bitov nastaví na opačnú.
Výraz je aritmetický výraz, ktorého hodnota po vyčíslení a orezaní desatinnej časti musí byť z intervalu 0 až 65535.

Príklady:	hodnota A
A=B INAND(8,4)	0
A=B INAND(7,1)	1
A=B INOR(6,1)	7
A=B INOR(15,1)	15
A=B INNOT(0)	65535
A=B INNOT(1)	65534

11.7. Rozdelenie pamäti mikropočítača, činnosť po zapnutí mikropočítača, funkcie BUSY, FREE



Pamäť od 0000H po BFFFH je pamäť typu RWM.
Pamäť od C000H po FFFFH je pamäť typu EPROM.

Po zapnutí mikropočítača, alebo stlačení klávies tomu ekvivalentných, BASIC pretestuje pamäť RAM a to od adresy 0000H po adresu 97FFH. To znamená, že pamäť od 9800H po 9FFFH nie je Basicom používaná a je výhodné umiestňovať do tejto oblasti programy, ktoré si užívateľ napíše v strojovom kóde.

V prípade, že pri testovaní pamäti RAM je zistená chyba, Basic vypíše správu RAM ERROR. Ak chyba nie je zistená, potom nie je vypísaná žiadna správa. V oboch prípadoch sa pokračuje skontrolovaním pamäti EPROM na kontrolný súčet. V prípade, že Basic vypíše správu ROM ERROR, došlo ku zmene obsahu pamäti EPROM. Kontrola pamäti EPROM na kontrolný súčet prebieha pred každým výpisom READY.

Po vykonaní testu pamäti RAM aj EPROM Basic vypíše READY a očakáva pokyny od operátora.

Ak sa objavil výpis RAM ERROR alebo ROM ERROR, potom nie je zaručená správna funkcia Basicu.

Funkcia FREE

Syntax: FREE

Funkcia FREE nás informuje o tom, koľko voľného miesta je pre program v Basicu a pre užívateľom deklarované dáta.

Funkcia BUSY

Syntax: BUSY

Funkcia BUSY nás informuje o tom, koľko miesta sme obsadili

programom v Basicu. Po zapnutí mikropočítača a po vykonaní príkazu NEW je hodnota tejto funkcie nulová.

Poznámka: Pamäťová bunka 97FFH má nasledujúci význam:
- v Basicu V5.6 a všetkých nasledujúcich verziách už patrí do oblasti pre programy v strojovom kóde
- v Basicu V5.5 používal Basic túto bunku ako pracovný byte a z toho vyplýva, že u Basicu V5.5 po použití príkazu MEMEND (pozri 11.8) nevznikne voľná súvislá oblasť, pretože bunku 97FFH používa Basic ako pracovný byte aj po použití príkazu MEMEND. Okrem tejto výnimky v Basicu V5.5 platí všetko, čo je v kapitole 11.8 uvedené.

11.8. Umiestňovanie užívateľských programov v pamäti

Pre užívateľské programy, napísané v strojovom kóde, je určená oblasť od adresy 9800H po adresu 9FFFH. V prípade, že táto oblasť je malá, možeme ju rozšíriť pomocou príkazu MEMEND.

Syntax: MEMEND
MEMEND výraz

Príkaz MEMEND je možné vykonať v priamom i príkazovom režime. Ak použijeme príkaz MEMEND bez parametra, potom tento príkaz iba zruší tabuľku premenných, čiže všetky premenné sú nedefinované. To znamená, že indexované premenné musí program znovu deklarovať príkazom DIM a neindexované číselné premenné nadobudnú hodnotu nula a reťazcové sú nastavené na prázdny reťazec.

Príkaz MEMEND výraz tak isto zruší tabuľku premenných a okrem toho prestaví koniec pamäti pre BASIC (pozri rozdelenie pamäti a je to adresa 97FEH) na požadovanú adresu. Túto požadovanú adresu určuje výraz po vyčíslení. Adresa musí byť

menšia ako 97FFH a musí byť väčšia ako adresa, kde končí program zapísaný v Basicu. Aby si operátor nemusel pamätať alebo vypočitavať uvedené adresy, preto sú v Basicu implementované nasledujúce funkcie:

funkcia EOMH

syntax: EOMH

táto funkcia vracia adresu pamäti (je to 97FEH), za ktorú sa nesmie prestaviť koniec pamäti pre Basic.

funkcia EOML

syntax: EOML

táto funkcia vracia adresu pamäti, na ktorú sme nastavili koniec oblasti pre Basic pomocou príkazu MEMEND. Po zapnutí systému je EOML=EOMH.

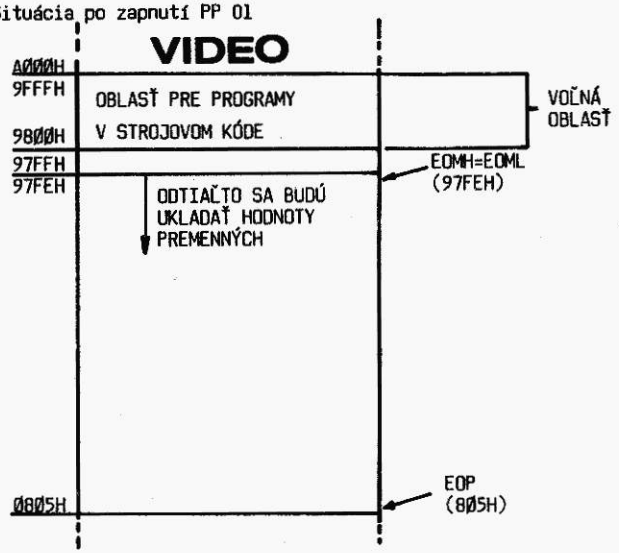
funkcia EOP

syntax: EOP

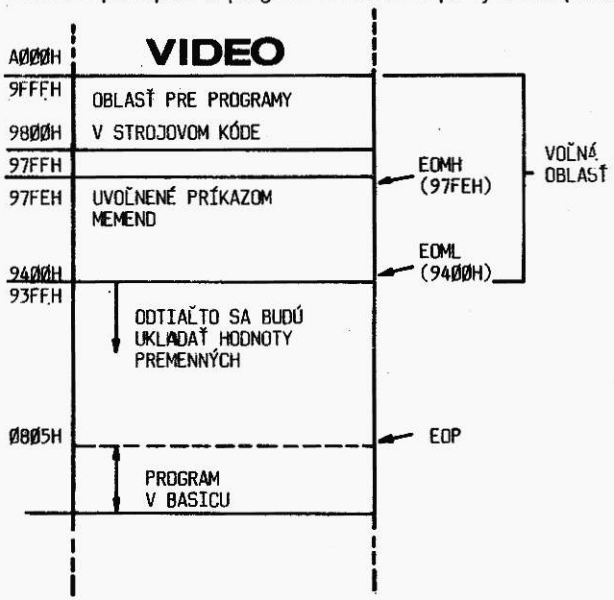
táto funkcia vracia adresu nasledujúcu po adrese, kde končí program napísaný v Basicu.

Pomocou príkazu MEMEND môžeme teda prestaviť koniec pamäti pre Basic v intervale ⟨EOP, EOMH⟩.
V prípade, že sa pokúsime prestaviť koniec pamäti mimo uvedeného intervalu, Basic nevykoná požadovanú činnosť a ohlási chybu č. 17.

Situácia po zapnutí PP 01



Situácia po napísaní programu v Basicu a po vykonaní príkazu MEMEND 9400H:



Čiže po vykonaní príkazu MEMEND 9400H je pamäť od 9400H po 9FFFH voľná. Basic do tejto oblasti nebude nič zapisovať ani z nej niečo čítať.

11.9. Podprogramy, ktoré môže používať program v strojovom kóde MHB 8080

Program v strojovom kóde napísaný užívateľom, nesmie používať pamäť od 0 po 804H v tom prípade, ak chce využívať podprogramy, ktoré sú súčasťou jazyka Basic.

Podprogramy, ktoré sú k dispozícii:

C003H - návrat do Basicu. Basic vypíše READY, program, ktorý sme predtým napísali v Basicu zostane zachovaný.

C00CH - je to tzv. tvrdý štart Basicu. Basic vykonáva tú istú činnosť, ako pri stlačení klávies ekvivalentných zapnutiu mikropočítača.

C016H - generuje priamku na displeji z bodu X1,Y1 do bodu X2,X2. Súradnice sa zadávajú na adresách:

X1	3CCH
Y1	3D5H
X2	3CDH
Y2	3D6H

Všetko musia byť 8 bitové hodnoty.

0000H - vypíše znak na TV, ASCII kód znaku musí byť v A registri.
Ak je adresa 516H = 0 - výpis len na TV
<>0 - výpis aj na tlačiareň

C05BH - vstup znaku z klávesnice
Ak je CY=1 - nič nie je stlačené

CY=0 - stlačená je klávesa na klávesnici,
jej kód je v A registri.

11.10. Možnosť rozšírenia BASICu o ďalšie drivery

Vprípade, že potrebujeme výstup z počítača na zariadenie, ktoré nemá medzistyk IRPR, môžeme to urobiť tak, že driver obsahujúci dané zariadenie napíšeme do pamäti, buď pomocou MONITORu alebo pomocou príkazu POKE a ďalej adresy 3, 4, 5 musíme prepísať inštrukciu skoku na daný driver. Driver musí končiť inštrukciou návratu a nesmie meniť žiaden register (okrem PSW). Z uvedeného vyplýva, že Basic pracuje cez adresu 3, ale iba vtedy, keď v príkaze LIST alebo PRINT uvedieme znak # alebo %. Ak je uvedený jeden z týchto znakov, potom výstup všetkých položiek uvedených v danom príkaze pracuje cez adresu 3, pričom kód daného znaku je v registri C.
Ak uvedieme znak # výstup príkazu LIST alebo PRINT, ide len cez adresu 3.
Ak uvedieme znak % výstup príkazu LIST alebo PRINT ide nielen cez adresu 3, ale aj na televízor.

Príklad:
PRINT 5*7

Na TV a súčasne aj cez driver (skok do ktorého je uvedený na adrese 3) je vypísaná hodnota 35.
Znaky # alebo % platia pre všetky položky len v danom príkaze LIST, PRINT.

Príklad:
Od adresy 9900H zapíšeme nasledujúci program pomocou inštrukcie POKE:

```
10 POKE 9900H,0IBH,0C5H,0E6H,10H,0CAH,0,99H,79H
20 POKE 9908H,0FEH,18H,0C2H,0FH,99H,3EH,20H,2FH
30 POKE 9910H,0D3H,0C4H,3EH,1,0D3H,0C7H,0AFH,0D3H,0C7H,0C9H
```

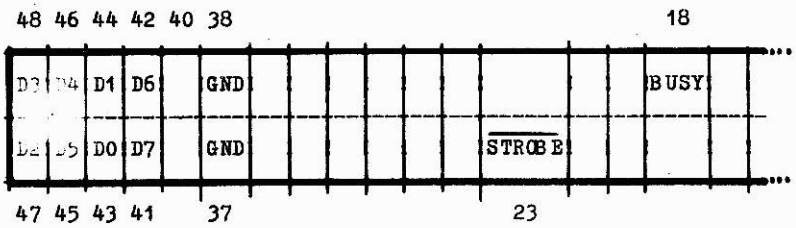
```
40 POKE 3,0C3H,0,99H
```

predchádzajúci program vyzerá v assembleri nasledovne:

```
9900
      IN   0C5H
9902  ANI  10H
9904  JZ   9900H
9907  MOV  A,C
9908  CPI  18H
990A  JNZ  990FH
990D  MVI  A,20H
990F  CMA
9910  OUT  0C4H
9912  MVI  A,1
9914  OUT  0C7H
9916  XRA  A
9917  OUT  0C7H
9919  RET
```

A teraz môžeme pomocou príkazu LIST**, PRINT** vypisovať aj na zariadenia s interfejsom Centronics, pričom tento interfejs vytvárame programovo pomocou podprogramu od adresy 9900H.

Rozloženie signálov na konektore IO pre pripojenie tlačiarne s medzistykom CENTRONICS



11.11. Príkaz MONIT

Tento príkaz možeme vykonať iba v priamom režime.

Syntax: MONIT

Pomocou tohto príkazu môžeme používať príkazy malého monitora. Monitor zobrazuje všetky výstupy a požaduje vstupy v hexadecimálnej sústave.

Po vykonaní príkazu MONIT sa objaví výpis:

MONITOR Vx.y

a výpisom dvojbodky monitor očakáva príkaz od operátora. Príkazy pre monitor sú jednoznakové a je to vždy jedno z písmen P, Z, S, N, M, Y. Pri zadaní iného znaku monitor vypíše otáznik a znovu výpisom dvojbodky očakáva vstup.

Príkaz P

Po stlačení príkazu P končí činnosť monitora a začne pracovať BASIC, ktorý výpisom READY ohlási pripravenosť k práci.

Príkaz Z

Pomocou tohoto príkazu možeme prezerať a modifikovať pamäť. Monitor požaduje výpisom OD: adresu, od ktorej chceme pracovať. Adresa sa zadáva ako štyri hexadecimálne číslice. Po zadaní týchto číslic monitor vypíše danú adresu a jej obsah a teraz:

- ak stlačíme klávesu medzera, zobrazí adresu o 1 vyššiu aj s obsahom
- ak stlačíme klávesu →| , zobrazí adresu o 1 nižšiu aj s obsahom

- ak stlačíme klávesu CR, potom končí tento príkaz a monitor očakáva nový príkaz
- možeme prepísať obsah danej adresy a to tým, že stlačíme postupne klávesy, ktoré reprezentujú obsah, na ktorý chceme adresu nastaviť. Obsah musia tvoriť dve hexadecimálne číslice. V prípade, že klávesa, ktorú sme stlačili nereprezentuje hexadecimálnu číslicu, je vypísaný otáznik a monitor očakáva nový príkaz výpisom dvojbodky. Ak obsah prepíšeme ako dve hexadecimálne číslice a vpravo od zadaných číslic sa objavia dve ďalšie hexadecimálne číslice, znamená to, že danú adresu sa nepodarilo prepísať. Príkaz Z totiž kontroluje aj hodnotu, ktorú zapísal do pamäti, či je skutočne v pamäti uložená.

Príkaz S

Pomocou príkazu S môžeme odštartovať program od adresy, ktorú príkaz S požaduje výpisom OD:. Adresu zadávame ako 4 hexadecimálne číslice. Ak sa z programu, ktorý štartujeme príkazom S chceme znovu vrátiť do monitora, potom program musí končiť inštrukciou návratu (RET, RNZ, RZ, ...).

Príkaz N

Pomocou príkazu N možeme nahrať danú oblasť pamäti na magnetofón.

Príkaz pomocou výpisov OD:, DO: vyžiada dolnú a hornú hranicu oblasti, ktorú treba nahrať na magnetofónovú pásku a výpisom K: požaduje zadanie čísla záznamu, pod ktorým bude oblasť nahratá. Číslo záznamu sa zadáva ako dve hexadecimálne číslice z intervalu 0 až 7F. Po zadaní čísla záznamu sa ihneď začne nahrávať na magnetofón, to znamená, že magnetofón už musí byť pripravený na nahrávanie.

Príkaz M

Pomocou príkazu M možeme snímať súbor z magnetofónu do pamäti. Príkaz vyžaduje výpisom K zadanie čísla záznamu a výpisom OD: adresu, od ktorej má záznam ukladať. To znamená, že záznam može byť prečítaný do inej oblasti, než z ktorej bol zapísaný. Ďalej je potrebné si uvedomiť, že príkaz M može prečítať len ten záznam, ktorý bol nahrávaný príkazom N, podobne príkazom KLOAD prečítame iba to, čo bolo nahrávané príkazom KSAVE a príkazom DLOAD prečítame len to, čo bolo nahrané príkazom DSAVE. To znamená, že napr. príkazom KSAVE a DSAVE vytvoríme záznamy, ktoré možu mať rovnaké číslo záznamu, ale pri snímaní príkazom KLOAD neprečítame záznam zapísaný príkazom DSAVE, aj keď má rovnaké číslo záznamu a opačne.

Význam výpisov BREAK a FILE IS NOT COMPLETE je rovnaký ako pri nahrávaní a snímaní príkazom KLOAD, KSAVE.

Príkaz Y

Príkaz Y má význam iba vtedy, ak máme ku PP01 pripojený diskový mechanizmus aj s diskovým radičom. Príkaz Y slúži na zatiahnutie operačného systému do pamäti a to tak, že z diskety založenej v diskovom mechanizme č.0 zosníma 18 sektorov z nultej stopy a uloží ich od adresy 3000H a odovzdá riadenie na adresu 3000H.

Príkaz Y pracuje nasledujúcim sposobom:

Po zadaní písmena Y monitor zistí stav diskového radiča. Ak diskový radič nie je prítomný, alebo ak je diskový mechanizmus č.0 nepripravený, monitor vypíše otáznik a očakáva nový príkaz. Ak je všetko v poriadku, potom monitor požiada o rekalibráciu mechanizmu č.0.

Monitor vypisuje písmeno I a stav operácie (vypisuje hodnotu výsledného bytu). Rekalibrácia prebieha dovtedy, kým neprebehne bez chyby (t.j., kým výsledný byte nie je 00).

Po úspešnej rekalibrácii monitor požiada o nahratie 18 sektorov z nultej stopy od adresy 3000H.

Monitor vypisuje písmeno R a stav operácie (vypisuje hodnotu výsledného bytu). Na čítanie nultej stopy je desať pokusov. Ak sa nepodarí načítať bez chyby. potom príkaz Y končí a monitor čaká na nový príkaz.

Ak sa podarí načítať bez chyby, potom je riadenie odovzdané na adresu 3000H a začne pracovať program zatiahnutý z diskety.

11.12. Práca s ROM modulom

V programovom jazyku GBASIC je implementovaný príkaz pre pripojenie ROM modulu. Príkaz ROM pracuje nasledovným spôsobom: pripojí prvé 4 Kbyte z ROM modulu na adresu 5000H a otestuje, či je ROM modul zasunutý, či je v ROM module zasunutá pamäť so správnym obsahom.

Ak nie je splnená aspoň jedna z uvedených podmienok, potom ohlási chybu 21, ináč odovzdá riadenie na adresu 5002H a obsah ROM modulu vykonáva ďalšiu činnosť.

Popis činnosti jednotlivých ROM modulov bude popísaný v príručkách dodávaných s ROM modulami.

Príloha č.1

Chyby v GBASICu

č.1 - zlá syntax príkazu
č.2 - pri aritmetických operáciach došlo k deleniu 0/0 alebo k pretečeniu
č.3 - hodnota mimo povolených medzí
č.4 - vstupný riadok je príliš dlhý
č.5 - má sa vykonať RETURN bez predchádzajúceho GOSUB alebo NEXT bez predchádzajúceho FOR.
č.6 - riadok s daným číslom neexistuje a nie je ani riadok s väčším číslom; v príkaze SETKEY je v úvedzovkách viac znakov ako 15
č.7 - v pamäti nie je miesto pre túto premennú;
v pamäti nie je miesto pre uloženie tohoto riadku
č.8 - zlé zátvorkovanie, chýba pravá zátvorka;
- v príkaze CONT - nemožno pokračovať príkazom CONT
č.9 - má byť uvedený identifikátor a nie je
č.10 - použitá indexovaná premenná má iné dimenzie, ako boli deklarované v príkaze DIM
č.11- v bežnom príkaze - dimenzovaná premenná nie je popísaná v DIM;
- v príkaze DIM - premenná už je definovaná
č.12 - index premennej je <= 0 alebo väčší ako 65535 (t.j. väčší ako bol popísaný v príkaze DIM)
č.13 - príkaz DATA je použitý v priamom režime
- príkaz INPUT je použitý v priamom režime
- príkaz DEF je použitý v priamom režime
č.14 - výsledný reťazec je dlhší ako 255 znakov
č.15 - identifikátor premennej je dlhší ako 15 znakov
č.16 - funkcia nebola definovaná príkazom DEF;
- príkaz DATA nie je v programe, alebo už sú všetky príkazy DATA prečítané a READ chce čítať
č.17 - pokus o prestavenie konca pamäti do zakázaných oblastí;
- v príkaze DEF - príkaz nie je jediný v riadku
- v príkaze DATA - príkaz nie je jediný v riadku

- v príkaze RESTORE - daný riadok neobsahuje príkaz DATA a ani riadky s vyšším číslom ho neobsahujú

č.18 - chyba v príkaze SCALE t.j. XMIN>=XMAX
alebo YMIN>=YMAX
alebo XMAX-XMIN>=INF
alebo YMAX-YMIN>=INF

č.19 - daný reťazec je prázdny;
- v príkaze ON GOTO, ON GOSUB je hodnota 1. parametra 0, alebo je málo prvkov zoznamu

č.20 - argument funkcie je neprípustný

č.21 - nie je zasunutá EPROM v ROM module, alebo ROM modul je prázdny alebo má chybný obsah

č.22 - zariadenie pripojené cez IRPR oznamuje signálom A0 chybu

Príloha č.2

Zoznam vyhradených symbolov

ABS
AND
ASC
ATAN
AT
AUTO +

BASE
BEEP
BINAND
BINOR
BINNOT
BMOVE
BPLOT
BUSY

CALL
CHR$
CLEAR
CONSOLE
CONT +
COS

DATA *
DEF *
DEG
DIM
DLOAD
DRAW
DSAVE

EDIT +
EOMH
EOML
EOP
EPS
ERR
EXP

FN
FOR
FRC
FREE

GO
GOSUB
GOTO

IDRAW
IF
IMOVE
INF
INK
INP
INPUT *

KEY
KLOAD
KSAVE

LEFT$
LEN
LET
LIST +
LN
LPRINTER

MEMEND
MID$
MONIT +
MOVE

NEXT
NEW +
NO
NOT

ON
OR
OUT

PAPER
PEEK
PENUP
PI
PLOT
POKE
PRINT

RAD
READ
REM
RESTORE
RETURN
RIGHT$
RND
ROM +
RUN +

SCALE
SETCHAR
SETKEY

SGN
SIN
SQR
STEP
STOP
STR$
SUB

TAB
TAN
THEN
TO
TRUNC

VAL

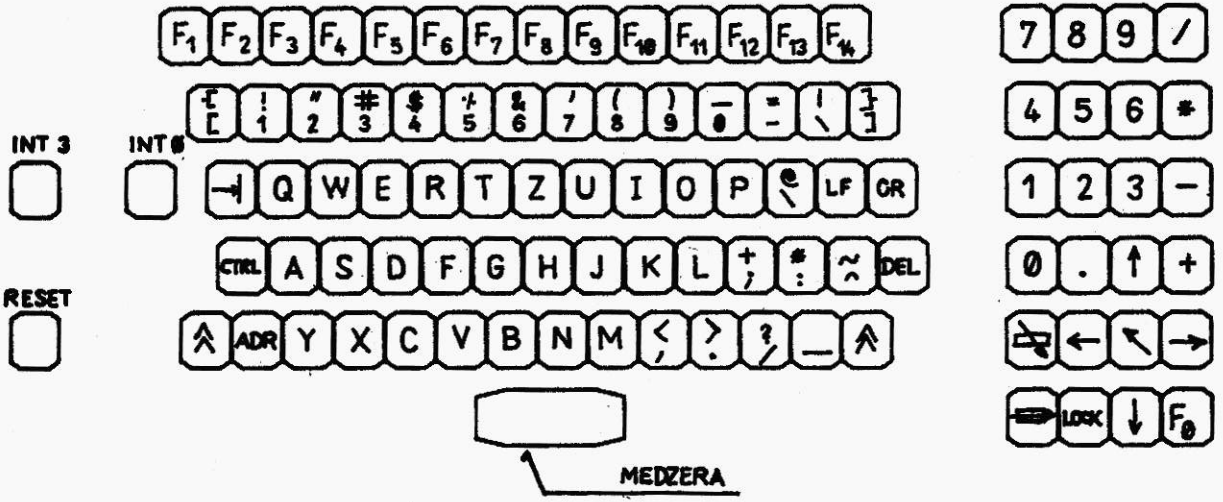
WAIT

XAXIS

YAXIS

* tento príkaz je možné použiť len v príkazovom režime
+ tento príkaz je možné použiť len v priamom režime

PRÍLOHA Č.3


F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 F13 F14
7 8 9 /
4 5 6 *
1 2 3 -
0 . +
INT 3
INT 6
RESET
Q W E R T Z U I O P LF CR
CTRL A S D F G H J K L DEL
ADR Y X C V B N M
LOCK
MEDZERA

Poznámky

Poznámky

Poznámky

Poznámky

Vytiskly Moravské tiskařské závody, národní podnik, Olomouc, provoz 21,
Ostrava 1, Novinářská 7

ZÁVODY
VÝPOČTOVEJ TECHNIKY
koncernový podnik
BANSKÁ BYSTRICA

Vydal útvar OTS