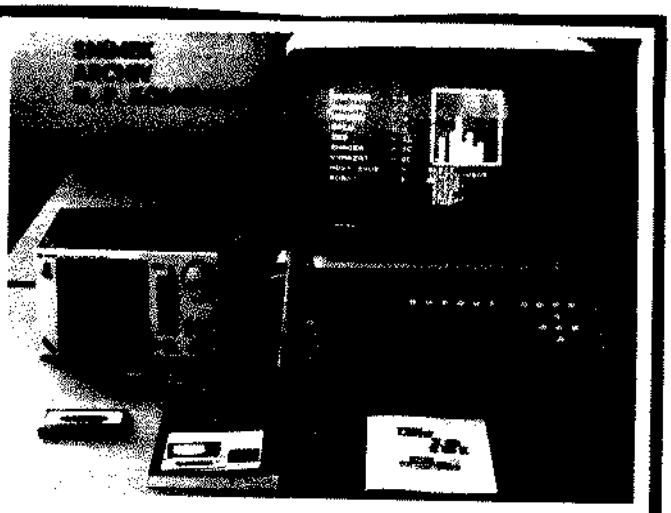
PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA



ÚVODEM

Seriál, který budeme na pokračování otiskovat, je určen všem, kteří se chtějí seznámit se základy programování v jazyku BASIC na počítači IQ 151, hlavně však žákům základních a středních škol.

Vychází z předpokladu, že většina žáků doposud neměla možnost seznámit se s programem napsaným v jazyku BASIC a nemá žádné konkrétní představy, jak takový program vypadá.

Je pravděpodobné, že žáci doposud neměli příležitost si samostatně sednout ke klávesnici počítače, a proto jsou následující řádky určeny všem začátečníkům při jejich prvních krůčcích poznávání možností výpočetní techniky.

V názvu je použito počátečních písmen Beginners' Allpurpose Symbolic Instruction Code — BASIC (čteme bejsik).

Jde o tzv. konverzační jazyk, což znamená, že uživatel pracuje s počítačem formou dialogu. Objeví-li počítač v programu syntaktickou chybu, sdělí to uživateli formou výpisu na obrazovku tak, aby bylo usnadněno její odstranění.

Zvládneme-li programovací jazyk, což není nic jiného než řada příkazů a povelů, pomocí kterých činnost počítače řídíme, a naučíme-li se z těchto příkazů sestavit potřebný program, dokázali jsme to hlavní — domluvit se s počítačem.

Za velmi krátkou dobu zjistíte, že to není tak složitý problém, jak se vám třeba ještě v tomto okamžiku jeví.

Pokud nemáte okamžitou možnost si sednout za klávesnici počítače IQ 151, vezměte si před sebe alespoň nákres klávesnice otištěný v tomto čísle časopisu Květy.

Je bezesporu prokázáno, že jedna hodina „hraní" si s počítačem nahradí několik hodin studia z učebnice, nezdržujeme se proto dlouhým povídáním a věnujeme se školnímu počítači IQ 151.

1. Základní seznámení s počítačem

1.1. Bezpečnost především

Počítač IQ 151 je elektrický spotřebič zařazený do I. izolační třídy a smí být připojen pouze do zásuvky s kolíkem propojeným na ochrannou soustavu.

V základní sestavě jsou do počítače zezadu zasunuty dva přídavné moduly BASIC 6 a VIDEO 32, které jsou připojeny pomocí mnohakolíkového konektoru /zástrčky/. Moduly z počítače nevyjímejte! Je-li to nutné, např. při výměně modulu za jiný /VIDEO 64 nebo GRAFIK/, provádějte vyjmutí či zasunutí modulu pouze při vypnutém počítači, jinak hrozí nebezpečí zničení počítače nebo modulu. Vždy se však předem dotkněte většího kovového předmětu /kovová skříň, radiátor ústředního topení/, abyste ze svého oděvu odstranili případný náboj statické elektřiny, který by mohl výbojem modul zničit. Ani potom se však kolíků konektoru raději nedotýkejte.

Na spodní straně počítače jsou pod krytem umístěny pojistky. Jejich případnou výměnu ponechejte odborné obsluze /učiteli/, která provede výměnu pojistek po vypnutí počítače a jeho odpojení od elektrovodné sítě.

Napájecí část, která je vestavěná do skříně počítače, se za provozu zahřívá. Ponechejte proto větrací otvory volné a nepokládejte na ně žádné papíry /s napsanými programy/apod.

Monitor /obrazovkový displej/ je zde zastoupen zcela běžným televizním přijímačem. Sami jej můžete pouze zapnout způsobem běžným u televizních přijímačů. Případnou úpravu jasu a kontrastu, nebo doladění vstupu televizoru na 10 - 12 televizní kanál, na jehož frekvenci náš počítač IQ 151 pracuje, přenechejte odbornému dozoru. Jakákoliv další manipulace s televizním přijímačem /mimo propojení s počítačem příslušným koaxiálním kabelem - při vypnutém televizoru i počítači/,

zvlášť odnímat zadní kryt a to i u přijímače odpojeného od sítě, není dovoleno.

Totéž se týká magnetofonu. Můžete jej propojit s počítačem příslušným propojovacím kabelem, vložit magnetofonovou kazetu a ovládat magnetofon běžnými tlačítky /nahrávání - přehrávání/.

Teprve, jsou-li všechny přístroje podle návodu k obsluze správně propojeny /počítač, televisor, magnetofon/, zasuneme přívodní šňůry všech přístrojů do zásuvek elektrovodné sítě a postupně zapneme: nejprve televizor a po rozjasnění obrazovky - počítač.

Po ukončení práce všechny přístroje vypneme a elektrické přívodní šňůry ze zásuvek vytáhneme.

1.2. Klávesnice

Je-li v počítači zasunut modul BASIC 6 a VIDEO 32, objeví se /po zapnutí televizoru a počítače/ v levém horním rohu obrazovky nápis

```
BASIC                          /čti bejsik rady -
READY                           bejsik je připraven/
■
```

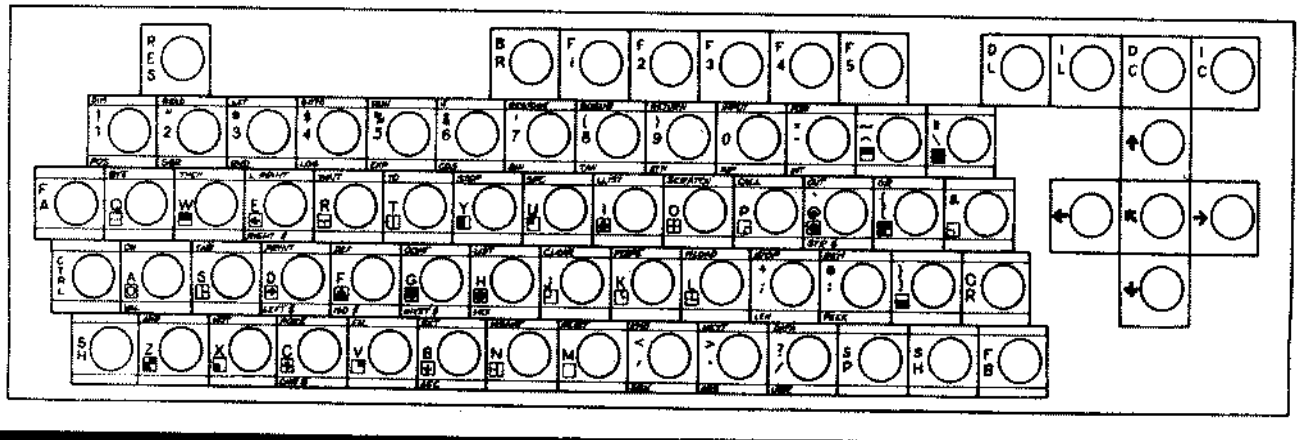
a blikající čtvereček, kterému se říká kurzor.

Podíváme-li se na klávesnici, zjistíme, že je velmi podobná klávesnici psacího stroje /pouze písmena Y a Z jsou vzájemně zaměněna/. Kdo umí alespoň částečně psát na psacím stroji, bude mít určitou výhodu.

Klávesnice má některá tlačítka barevně odlišená. Věnujme se nejprve těm černým.

Vlevo od černých tlačítek /přesně v jejich ose/ je napsáno velké písmeno, číslice, nebo jiný znak, který se zobrazí na obrazovce, když tlačítko krátce stiskneme.

(pokračování v příštím čísle)



PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

1. POKRAČOVÁNÍ

Počítač má tzv. membránovou klávesnici, jejíž stisk tlačítek je velmi malý /asi 0,3 mm/. Správné stisknutí je proto ještě ohlášeno akusticky tzv. „pípnutím" /BEEP - čti bíp/.

Některé elektrické psací stroje, přidržíme-li déle stisknutou klávesu, její úder automaticky opakují. Také náš počítač má tzv. repetici, a to u všech kláves. Přidržíme-li některé tlačítko /např. písmeno A/ déle, bude se jeho tisk - zobrazení opakovat.

Jistě jste si povšimli, že při stisku klávesy píše psací stroj malá písmena /malou abecedu/, zatímco počítač zobrazuje velkou abecedu.

Počítač totiž nezná malou abecedu; proto musí být všechny povely a příkazy jazyka BASIC psány velkou abecedou. Jsou počítače, které vůbec nedovedou malou abecedou psát; náš počítač je výjimkou, píše i malou abecedou, svůj zobrazená malá písmena jsou pro něj pouze „grafickým znakem" a nerozeznává je.

V první /k vám nejbližší/ řadě tlačítek jsou dvě šedá tlačítka /jedno vlevo, druhé vpravo/ označená jako SH /SHIFT - čti šift - přeřazovač/. Stiskneme-li a přidržíme-li jedno z těchto tlačítek a zároveň stiskneme jiné tlačítko s písmenem, zobrazí se na obrazovce znak malé abecedy.

U ostatních tlačítek se pomocí SH /šiftu/ zobrazí znak nakreslený vlevo nad osou tlačítka /nad základním znakem tlačítka, např. +, ?, apod./.

Píšeme-li na psacím stroji, dopadají jednotlivé typy písmen mezi tzv. křídelka, která nám ukazují, kam bude psán následující znak. Na obrazovce nám toto místo označuje kurzor.

Dopíšeme-li řádek až do konce, přejde kurzor automaticky na začátek následujícího řádku.

Pokud se vám podařilo napsat na obrazovku více než dva a půl řádku znaků, zablokoval se pravděpodobně kurzor a počítač odmítá zobrazovat další znaky. Nebojte se, nic se nestalo. Pouze jste zaplnili vstupní paměť /BAFR/ počítače a počítač čeká na odeslání zobrazených znaků do paměti.

Najděte vpravo na klávesnici šedé tlačítko s označením CR /CURSOR RETURN - čti kurzor ritern - kurzor zpět na začátek následujícího řádku/ a stiskněte je. Kurzor se uvolní, přejde na začátek následujícího řádku a na obrazovce se zobrazí sdělení

* Ø1 ERROR

Počítač nám tímto oznamuje, že napsanému sdělení neporozuměl.

I když některým chybovým hlášením prozatím neporozumíme, podívejme se na přílohu čís. 1 /viz str. 61 /, kde je popsán význam všech chybových hlášení, kterými nás bude počítač upozorňovat, že něčemu nerozumí, nebo něco nejde provést.

Tvar chybového hlášení bude např.

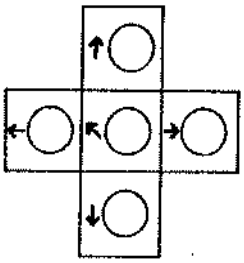
* Ø1 ERROR IN 2Ø

znamená to, že na řádku 2Ø v programu počítače je chyba označená v seznamu chybových hlášení pod číslem Ø1.

Jeden řádek na obrazovce obsahuje /je možno na něj napsat/ 32 znaků /písmen, číslic, mezer, znamének či grafických znaků/.

Počítač přísně rozlišuje písmeno O od číslice Ø /nula/ a také je na obrazovce různě zobrazuje. Nesmíme proto nikdy tyto dva znaky zaměnit! Také v textu této příručky bude nula psána vždy takto; Ø .

Ostatní znaky klávesnice prozatím vynecháme, zastavíme se pouze u pěti bílých tlačítek vpravo, označených šipkami:



těmito tlačítky ovládáme kurzor. Vyzkoušejte! Prostřední tlačítko odesílá kurzor do výchozí polohy v levém horním rohu obrazovky.

Ke vložení našeho prvního programu budeme potřebovat ještě jedno velmi důležité tlačítko /na psacím stroji bývá tou největší klávesou/ - mezerník; je to vpravo dole umístěné tlačítko označené SP /SPACE - čti spejs - mezera/.

Cokoliv jsme napsali na obrazovku, je pouze na obrazovce, počítač „o tom neví", v jeho paměti to není zaznamenáno.

Chceme-li, aby počítač příkaz napsaný na obrazovce provedl, nebo ho uložil do programové paměti, musíme mu to sdělit, tj. odeslat příkaz do paměti počítače stisknutím šedého tlačítka CR.

Prvním příkazem, s nímž se seznámíme, bude CLS /CLEAR SCREEN - čti klír skrín - smaž, vyčisti obrazovku/.

Napište na obrazovku tato tři písmena příkazu CLS a odešlete je do paměti počítače stisknutím tlačítka CR. Máte-li obrazovku „popsanou", stiskněte nejprve CR . Objeví se chybové hlášení a teprve nyní, když je kurzor na začátku nového řádku, napište příkaz CLS a odešlete. Obrazovka se okamžitě „smaže".

2. Základy programování v jazyku BASIC 6

2.1. Abeceda, čísla, znaménka

K tomu, abychom mohli psát programy máme na klávesnici počítače k dispozici tyto znaky:

A b e c e d a - 26 velkých písmen A až Z /anglické abecedy/.

Pro psaní příkazů, proměnných a pomocných slov používáme zásadně velkou abecedu. Při psaní různých vysvětlivek, sdělení a otázek pro informaci obsluhy počítače, můžeme z důvodu odlišení použít i malou abecedu.

Č í s l i c e a č í s l a - Ø až 9 /číslice/; nesmíme zaměnit Ø /nulu/ za písmeno velké O a obráceně!

Na obrazovku můžeme zapisovat jakákoliv /vícemístná/ čísla. Počítač však bude počítat pouze s osmi platnými číslicemi, z nichž před zobrazením výsledku poslední dvě místa zaokrouhlí, ale na obrazovce nezobrazí /tzv. skrytá číslice/. Na obrazovce se po zaokrouhlení zobrazí šest platných číslic /šestimístné číslo/.

Je-li však výsledkem vícemístná číselná hodnota, přejde počítač automaticky na semilogaritmické zobrazení s pevnou řádovou tečkou /desetinné znaménko/, umístěnou za první číslicí.

Vložená čísla zobrazí počítač tak, že před zápornými čísly zobrazí znak - /mínus/, před kladnými čísly znaménko + sice nezobrazuje, ale volné místo před číslem ponechává pro případnou změnu znaménka. Za každým číslem je umístěn jeden prázdný znak /jedno volné místo, kam nelze nic napsat/.

Vyzkoušejte následující řádky. Po napsání každého jednotlivého řádku jej odešlete z obrazovky do počítače stisknutím tlačítka CR . Zobrazí se výsledek, který máte pro kontrolu uveden v závorce. Po zaplnění obrazovky se automaticky posune text o jeden řádek nahoru, čímž se uvolní poslední /spodní/ řádek k napsání dalšího textu.

PRINT 1ØØØ /1ØØØ/

Číslo nemá více než šest míst, počítač proto nepřešel do semilogaritmického zobrazení.

Poznámka: příkaz PRINT znamená napiš /zobraz/ na obrazovce.

PRINT 1ØØØØØ /1E+Ø6/

což matematicky znamená 1 x 1Ø⁶, E - exponent,

PRINT 1Ø2Ø3Ø4Ø5Ø /1.Ø2Ø3E+Ø9/

všimněte si - nevýznamné nuly se nezobrazují. Pevnou řádovou tečku /desetinné znaménko/ umístí počítač vždy za první číslicí. Jako desetinného znaménka se užívá zásadně tečky!

PRINT Ø.ØØ37 /3.7E-Ø3/

PRINT Ø.5 /.5/

zobrazí se bez nuly. Pamatujte, že nevýznamné nuly počítač nezobrazuje a nemusíme je psát, a to ani před desetinnou tečkou.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

2. POKRAČOVÁNÍ

PRINT 56,35 /56 36/

čárka /nebo středník/ slouží jako oddělovač dvou čísel /konstant/. Vložené číslo /s omylem vloženou desetinnou čárkou/ zobrazí počítač jako dvě samostatná čísla.

Pomocná znaménka

. /tečka, desetinné znaménko/
, /čárka, oddělovač, oddělovač tisku/
; /středník, oddělovač, oddělovač tisku/
: /dvojtečka, oddělovač příkazů/
? /otazník/
! /vykřičník/
" /uvozovky, oddělovač textu/
( ) /kulaté závorky, nezaměňujte je za závorky [ ], { } /
& /znak „et"/
% /procento/
$ /string, značení řetězcových proměnných/
= /rovnítko - používé se v příkazu přiřazení/

Kulaté závorky používané u některých funkčních příkazů jazyka BASIC /bude uvedeno dále/, nelze nahradit závorkami [ ] nebo { } !

Aritmetické operátory

\+ /plus/ sečítání, znaménko kladných hodnot
\- /mínus/ odčítání, znaménko záporných hodnot
/ /zlomková čára/ dělení; nelze použít znaménka : /oddělovače/
* /hvězdička/ násobení; nelze použít tečky nebo x
^ /střížka/ umocňování

Relační operátory

= rovné se
< je menší než...
> je větší než...
<= je menší nebo rovno...
>= je větší nebo rovno...
<> nerovná se

Logické operátory

AND /logický součin/ a zároveň...
OR /logický součet/ nebo...
NOT /negace/ není pravda, že...

2.2. Co je to program a jeho tvorba

Chceme-li, aby počítač prováděl nějakou činnost, nebo řešil zadaný úkol, musíme mu to vhodnou formou sdělit. Program je vlastně přesný popis jednotlivých činností, které má počítač „krůček po krůčku" /měli bychom správně říci „řádek po řádku"/ vykonávat, od vložení základních údajů, až po vyřešení zadaného úkolu a zobrazení získaných výsledků.

Tvorbu programu můžeme rozdělit na tři samostatné části:

1. definujeme počítači hodnoty se kterými bude pracovat a určujeme způsob, jakým budou počítači zadány,
2. sdělíme počítači, jaké úkony má s vloženými konstantami provádět,
3. určíme počítači, jakou formou nám má získané výsledky početních či jiných úkonů zobrazit.

2.3. Programový řádek

Příkazy k činnosti sdělujeme počítači po jednotlivých /programových/ řádcích. Musíme si uvědomit, že programový řádek může obsahovat až 79 znaků, což zabere asi dva a půl řádku na obrazovce.

Na jeden řádek obrazovky se vejde pouze 32 znaků; na další /obrazovkový/ řádek přejde počítač autometicky.

Každý programový řádek musí začínat číslem řádku a končit stisknutím tlačítka CR .

Na jednom řádku může být i několik příkazů, oddělíme-li je od sebe oddělovačem : /dvojtečka, oddělovač příkazů/.

Číslem řádku musí být celé kladné číslo; maximálním číslem řádku je 63529.

V programu nemohou být dva řádky stejného čísla. Později napsaný řádek přepíše /přemaže/ již dříve napsaný řádek stejného čísla. Této možnosti používáme při opravách programu.

Program bude počítačem zpracován /řešen/ po jednotlivých řádcích podle vzestupných hodnot /čísel/ programových řádků, pokud nebude počítači příkazem uloženo přejít na jiný řádek.

Pokud za číslem řádku nic nenapíšeme, registruje počítač tento řádek jako prázdný /neexistující/ a do paměti jej nezařadí. Tímto způsobem můžeme dodatečně odstranit /vymazat/ již dříve napsané nepotřebné, nebo chybné řádky.

Při programování v jazyku BASIC je zvykem číslovat řádky např. 5 10 15..., nebo 10 20 30..., aby bylo možno při pozdější případné úpravě programu vkládat další doplňující programové řádky. Přečíslování řádků by bylo velice obtížné a u delších programů prakticky nemožné.

Opravované, nebo dodatečně vkládané řádky můžeme připisovat na konec již napsaného programu. Počítač si sám zařadí opravované nebo nově připsané řádky přesně tem, kam podle čísla řádku patří.

2.4. Struktura programu

Nyní již toho víme tolik, že můžeme přistoupit k sestavení prvního programu.

Napište na obrazovku:

```
10 DATA 3,8,15          /řádek odešlete stisknutím CR /
20 READ A,B,C           /řádek odešlete stisknutím CR /
```

␣ tímto znaménkem značíme v tisku místo, kam máme vložit mezeru - stisknout tlačítko SP . Vkládání mezer není pro činnost počítače nutné, program je však /pro nás/ přehlednější.

Tím jsme splnili první část tvorby programu a na prvním řádku programu /s číslem 10/ sdělujeme počítači, že budeme pracovat se třemi konstantami /čísly/, jejichž hodnoty /DATA/ jsou 3, 8, 15.

Na druhém řádku /s číslem 20/ přiřazujeme hodnoty uvedené za příkazem DATA proměnným uvedeným za příkazem READ /ěti řídí/, tedy proměnným označeným identifikátory A, B a C.

Řádek s příkazem DATA může být v programu zařazen kdekoliv /zpravidla na začátku nebo na konci programu, abychom v něm uvedená data mohli lehce obměňovat/; příkaz READ si tet DATA najde a přesně v tom pořadí, jak jsou uvedena, je přiřadí jednotlivým proměnným. V proměnné A bude tedy uložena konstanta 3, v B bude konstanta 5 a v C bude 15.

Za příkazem DATA i příkazem READ oddělujeme jednotlivé konstanty či proměnné zásadně oddělovačem , /čárkou/; za poslední vkládanou hodnotou čárku neděláme.

Zmýlíme-li se při vkládání /psaní/ programového řádku např. stisknutím jiného tlačítka nebo podob., vrátíme se kurzorem zpět na chybné místo /pomocí bílých tlačítek se šipkami/ a chybný znak nebo slovo přepíšeme správným.

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

3. POKRAČOVÁNÍ

Nyní budeme vkládat další programové řádky. Každý nově napsaný řádek nejprve překontrolujeme /případnou chybu odstraníme/ a teprve potom stisknutím [CR] odešleme do paměti počítače.

```
3Ø LET X=A+B+C
4Ø LET Y=X/3
5Ø LET Z=A^2
```

Nyní jsme splnili i druhou část tvorby programu a příkazujeme počítači, jaké úkony má s proměnnými provádět.

Přiřazovací příkaz má tvar LET jméno proměnné = /znaménko přiřazení, tedy nikoliv rovnítko/ a dále hodnota nebo výraz který proměnné přiřazujeme.

Řádek 3Ø bychom mohli číst takto: proměnné X bude přiřazena hodnota součtu proměnných A + B + C. /Do proměnné X bude uložena hodnota součtu hodnot přiřazených proměnným A, B a C/.

Řádek 4Ø už rozluštíte asi sami: proměnné Y bude přiřazena hodnota proměnné X dělená 3; tedy průměrná hodnota konstant uložených v proměnných A, B a C.

Na řádku 5Ø jsme použili nový operátor ^/střížku/, což je operátor pro mocninu neboli matematicky a^2 (A^2), a^5 (A^5) atp.

Ve třetí části programu sdělíme počítači, jakou formou a které ze získaných výsledků nám má zobrazit.

Napíšeme tedy:

```
6Ø PRINT A;B;C                    /použijeme oddělovače ;/
7Ø PRINT X,Y,Z                    /použijeme oddělovače ,/
8Ø END
```

Řádkem 6Ø žádáme počítač o zobrazení hodnot, s nimiž pracuje. Při použití oddělovače ; /středník/ budou hodnoty zobrazeny těsně za sebou a vynechánim nejnutnějších volných znaků /vysvětlení viz kapitolu 2.1./.

Pomocí řádku 7Ø zobrazí počítač výsledky tj. součet, průměrnou hodnotu vložených konstant a výsledek mocniny a^2. Použitím oddělovače , /čárka/ budou hodnoty /výsledky/ zobrazeny v tzv. tiskových zónách, které začínají na pozicích Ø, 14, 28, 42 a 56. Jsou tedy tiskové zóny rozděleny do několika řádků na obrazovce.

Rozdílné oddělovače na řádcích 6Ø a 7Ø jsme použili pouze z důvodu názornosti, abychom posoudili, jak který oddělovač upravuje tisk /zobrazení/ výsledků.

Poznámka: Tiskové zóny se používají tehdy, je-li k výstupu počítače připojena tiskárna, která tiskne 8Ø znaků na jeden řádek.

Na řádku 8Ø je příkaz END, příkaz k ukončení /zastavení/ činnosti počítače.

Tím je tvorba našeho prvního programu ukončena a program je uložen v paměti počítače. Abychom začínali z čistou obrazovkou, smažeme ji /CLS/.

Nyní program spustíme.

Napište na obrazovku příkaz RUN /čti ran - spusť program/, který po odeslání nejprve vynuluje případný obsah všech proměnných a spustí program počínaje nejnižším číslem řádku.

Program uložený v paměti počítače můžeme kdykoliv opětovně spustit příkazem RUN. Vypnutím počítače, nebo stisknutím červeného tlačítka [RES] /RESET - vymaž/ se program z paměti počítače vymaže.

Před vložením nového programu vždy dřívější program z paměti počítače vymažte.

Později se naučíme zaznamenat /přehrát/ program z paměti počítače na magnetofonovou kazetu k trvalému uschování.

2.5. Výpis programu

Náš program běží, opakuje však stále stejné hodnoty. Provedeme tedy malou úpravu programu, abychom mohli vkládat různé konstanty v průběhu programu přímo z klávesnice.

K provedení úpravy programu musíme vyvolat program uložený v paměti počítače zpět na obrazovku.

LIST /listing - výpis/ tento příkaz vypíše celý program z paměti počítače na obrazovku /samozřejmě po jeho odeslání tlačítkem [CR]/.

Program máme vypsán na obrazovce. Výpis byl ukončen slovem READY, což znamená, že počítač splnil daný povel /LIST/ a čeká na naše další přání.

Na místo, kde bliká kurzor, napište:

```
1Ø CLS
```

Naším novým řádkem 1Ø bude původní řádek přepsán. Například si tedy počítač po odstartování programu smaže obrazovku programově sám. Opravíme i další řádek, vložte:

```
2Ø INPUT A,B,C
```

Po spuštění programu se počítač na tomto řádku zastaví, vytiskne : /dvojtečku/ a počká na vložení tří hodnot z klávesnice.

Vložené konstanty uloží do uvedených proměnných A, B, C ve stejném pořadí.

Opravený program již známým způsobem spusťte /RUN a [CR]/. Obrazovka se sama smaže a vytiskne se : /dvojtečka/.

Vložte např. čísla 8, 1Ø, 17 a odešlete stisknutím [CR]. Čísel vložte přesně tolik, kolik je jich programovým řádkem 2Ø požadováno. Jednotlivá čísla oddělujeme čárkou, za poslední konstantou čárku nedáláme. Mezery můžeme vynechat.

Po odstartování program proběhne a na obrazovce se zobrazí žádané výsledky. Program můžete opětovně spustit příkazem RUN a vkládat různé hodnoty.

Pro toho, kdo nezná náš program, je však začátek označeny : /dvojtečkou/ poněkud nesrozumitelný. Provedeme proto další malou úpravu.

Program si znovu vyvolejte na obrazovku příkazem LIST. Zjistíte, že všechny řádky, které jsme upravovali, jsou správně zařazeny na svých místech.

Ještě jednou přepíšeme druhý řádek:

```
2Ø INPUT "vloz A,B,C";A,B,C
```

Pozor na oddělovače; mezi textovou částí vloženou do uvozovek a konstantami /proměnnými/ je, jako oddělovač textu, středník.

Cokoliv vložíme za příkaz INPUT nebo PRINT do uvozovek, vypíše se při běhu programu na obrazovku beze změn /a bez uvozovek/. Bývají to různá sdělení pro obsluhu počítače a pro tato sdělení můžeme používat i malou abecedu. Do původního řádku jsme pouze vložili určitý text, napsaný v uvozovkách, pro informaci obsluhy.

Řádek 6Ø můžeme odstranit, neboť vkládané hodnoty budou stále viditelné na obrazovce a jejich opětovný výpis je proto zbytečný.

```
6Ø        /a odešleme [CR], prázdný řádek/
```

Upravíme si také zobrazení získaných výsledků.

```
7Ø PRINT "soucet A+B+C=";X
8Ø PRINT "prumerna hodnota =";Y
9Ø PRINT "A^2 =";Z
1ØØ END
```

Dobře sledujte a pamatujte si, kde je nutno zařadit oddělovač a kde ne.

LIST - vyvolejte si opravený program na obrazovku a přesvědčte se, zda jsou opravy bez chyb. Je-li vše vpořádku, program spusťte.

Nyní již program vyhovuje po všech stránkách. Spusťte program a vkládejte různé konstanty /číselné hodnoty/, abyste si ověřili chod programu. Úpravou řádků 3Ø, 4Ø a 5Ø /a příp. vložením dalších/, můžeme programovat řadu různých matematických příkladů.

Spustíme znovu program a „omylem" vložíme místo požadovaných tří, pouze dvě konstanty.

Počítač vytiskne :: /dvě dvojtečky/ aby nás upozornil, že bylo vloženo méně hodnot než očekával, a že pokračuje vložení chybějící hodnoty.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

4. POKRAČOVÁNÍ

Vložíme-li hodnot více, nebo napíšeme-li za poslední vkládanou hodnotou , /čárku/, upozorní nás počítač sdělením
* EXTRA DATA IGNORED
tj. nadbytečné hodnoty ignoruji a převezme jen první tři vložená data.

Každou nadbytečnou čárku čte počítač jako by oddělovala vloženou Ø /nulu/.

Např. 15, 6, 5, čte jako 15 6 5 Ø
15, 6,, 5 čte jako 15 6 Ø 5

Až si dost „vyhrajete", nevypínejte počítač /z důvodu smazání programu/, naučíme se podle následujících řádků některé další možnosti oprav chybných programů.

Vše provádějte pomalu a s klidem.

2.6. Edice /úprava/ programu

Naučili jsme se již opravovat chybně napsaný text na obrazovce tím, že se na chybné místo vrátíme kurzorem a chybný znak přepíšeme.

Nyní si ukážeme jak postupovat, je-li zapotřebí něco z napsaného řádku vypustit /vymazat/, nebo naopak, potřebujeme-li doprostřed napsaného řádku něco delšího vložit.

Vyvolejte si náš program na obrazovku příkazem LIST.

Přesuňte kurzor pomocí tlačítka [↑] nahoru na řádek 2Ø, na kterém máme napsáno
2Ø INPUT "vloz A,B,C";A,B,C

Posuňte kurzor po řádku [→] těsně za příkazové slovo INPUT.

V pravém horním rohu klávesnice jsou dvě bílá tlačítka označena [DC] /DELET COLUMN - „srazit-odstranit" - vymazání znaku/ a [IC] /INSERT COLUMN - „rozhrnout" - místo pro vložení znaku/. Stiskněte několikrát tlačítko [DC], až celý vložený text /včetně uvozovek/ odstraníte.

Opravujeme-li, nebo upravujeme programový řádek, který byl již jednou odeslán do paměti počítače, musíme ho začít přejíždět kurzorem počínaje řádkovým číslem a po provedené úpravě řádek „přejet" kurzorem až do konce a odeslat.

Kdybychom odeslali řádek [CR] pouze z upravovaného místa, zařadil by počítač do své paměti jen to, co stojí vlevo od kurzoru /co již kurzor „přejel"/ a zbývající část řádku, stojící od kurzoru vpravo, by neregistroval.

Pokud jsme již /po předchozí úpravě - vymazání textu/ celý řádek kurzorem přejeli a řádek odeslali do paměti, najedeme na řádek 2Ø znovu a přesuneme kurzor ještě jednou za příkaz INPUT.

Stiskneme-li několikrát tlačítko [IC], posune se zbytek řádku / vpravo od kurzoru/ a do takto uvolněného místa můžeme vložit doplňující text, vynechaný znak apod.

Příliš daleko odsunutý text /v případě potřeby ho můžeme odsunout až na následující meziřádek/, lze přisunout zpět pomocí tlačítka [DC]. Přisunutí provádíme pozorně, abychom si omylem nevymazali také část přisunovaného textu.

Nezapomeňme, že na konci každého řádku /který již byl dříve odeslán do paměti počítače/, stojí znak CR, i když ho na obrazovce nevidíme. Můžeme se o tom přesvědčit.

Přejedeme-li programový řádek kurzorem až na konec řádku a najedeme na zde stojící /i když pro nás neviditelný/ znak CR, bude tento řádek /tímto neviditelným znakem CR/ opětovně odeslán do paměti počítače.

Potřebujeme-li pokračovat v psaní na tomto programovém řádku, musíme /neviditelný/ znak CR přepsat jiným znakem, nebo vložením mezery [SP].

2.7. Příkaz a povel

Začíná-li řádek, který píšeme na obrazovce číslem, považuje ho počítač za programový řádek a po odeslání jej uloží do programové paměti k pozdějšímu provedení.

Začíná-li řádek příkazem /příkazovým slovem/, považuje tento řádek počítač za povel, který má po odeslání řádku CR okamžitě vykonat.

Většina příkazových slov /PRINT, RUN apod./ může být použita jako příkaz /v programovém řádku/ i jako povel /k okamžitému provedení/.

Některé povely však v programu použít nelze a jsou určeny pouze k okamžitému provedení např. AUTO, MEM, MSAVE apod.

Všechny příkazy a povely budou dále popsány.

2.8. Proměnné - paměťová místa

Každý počítač má k dispozici určitý počet paměťových míst /v jazyku BASIC je nazýváme proměnné/, do kterých je možno uschovávat číselné konstanty nebo skupinu /řetězec - string/ nenumerických znaků /např. text/.

Každé proměnné může být přiřazena jedna číselná konstanta /číslo/ nebo skupina /řetězec/ znaků o délce nejvýše 48 znaků. Viz příkaz CLEAR na str. 67.

Obsah kterékoliv proměnné si můžeme kdykoliv zobrazit /vyvolat na obrazovku/ příslušným příkazem.

Přiřadíme-li obsazené proměnné /ve které je již něco uloženo - které již bylo něco přiřazeno/ nový obsah /konstantu nebo řetězec/, je původní obsah nenávratně přepsán obsahem novým.

V jazyku BASIC se pro označení /identifikaci/ jednotlivých proměnných používá jednoduchých kombinací písmen velké abecedy a případně i číslic.

Používá se tří druhu proměnných :

jednoduchých
indexovaných - jednorozměrných
- dvourozměrných
řetězcových

Nejpoužívanější jsou jednoduché proměnné, které je možno značit /identifikovat/ třemi způsoby :

a/ jedním písmenem velké abecedy, např.: A, B, C... až Z /máme k dispozici 26 proměnných/

b/ jedním písmenem a jednou číslicí, např.: A2, C9, BØ /k dispozici máme 26 x 1Ø = 26Ø proměnných/

c/ dvěma písmeny, např. AA, AB, AC... až ZZ /k dispozici máme 671 proměnných/

U p o z o r n ě n í !
Vzhledem k tomu, že jazyk BASIC používá některá dvoupísmenová příkazová a pomocná slova, nesmí být těchto slov použito jako identifikátorů /názvů proměnných/; jsou to ON, IF, TO, OR, PI.

Indexované proměnné nahrazují některá matematická značení jako a_1, a_2, $b_{4,2}$ apod. Identifikátory indexovaných jednorozměrových proměnných se však v jazyku BASIC píší poněkud odlišně, např. A(1), A(2), A(3), A(4), A(5) - jako označení pětiprvkového jednorozměrného číselného pole.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

5. POKRAČOVÁNÍ

Indexované dvourozměrné proměnné značíme např. B(1,1), B(1,2), B(2,1), B(2,2) - jako označení čtyřprvkového dvourozměrného pole B.

Nezaměňujte však nikdy označení /identifikátory/ proměnných za obecná čísla.
PRINT A*A nebude 2A, ale obsah proměnné A /např. 6/ násobený obsahem proměnné A, v tomto případě tedy 36.

Řetězcových proměnných používáme k uschování nenumerických proměnných. Jako identifikátorů používáme stejného značení proměnných jako v předešlých případech zakončených znakem $ /string - řetězec/.

Řetězec je skupina znaků přípustných v jazyku BASIC uzavřených do uvozovek.

Uzavírací uvozovky můžeme vynechat /viz příklad/, neobsahuje-li řetězec čárku, číselný znak, nebo mezeru /SP/.

Vyzkoušejte :

```
A$ = "TELEVIZOR          uvozovky vynechány! /odešli CR /
PRINT A$                                     /odešli CR /
B$ = " A MAGNETOFON"                         /odešli CR /
PRINT B$                                     /odešli CR /
PRINT A$;B$                                  /odešli CR /
```

2.9. Funkční operátory, matematické funkce a π

Abychom mohli sestavovat jednodušší lineární programy, naučíme se používat některé funkční operátory. Za každým operátorem se v závorce uvádí název proměnné, ve které je uložena příslušná hodnota, nebo přímo konstanta /číselná hodnota/, příp. matematický výraz.

Algebraické funkce

SQR(X) - $\sqrt{X}$, druhá odmocnina X /X musí být nezáporné/
Příklady: /každý příklad začínej příkazem PRINT/
SQR(A) - A je konstanta uložená v proměnné A
SQR(3.025E+07) = $\sqrt{30250000}$ = 5500
SQR(3 * 5 + 10) = 5

EXP(X) - e^x, e je tzv. základ přirozených logaritmů (e = 2,71828). Je to inverzní funkce vůči funkci LOG(X), platí tedy LOG(EXP(X)) = X
Příklady:
EXP(1) = 2.71828 = e
EXP(2) = e^2 = 7.38905

LOG(X) - přirozený logaritmus čísla X, pro X > 0
Příklady:
LOG(2,71828) = 1
LOG(1) = 0
LOG(30) = 1.47712

ABS(X) - absolutní hodnota |X|

Goniometrické funkce

SIN(X) - sin x, X vkládáme v radiánech
Příklady:
SIN(3,14159) = 0

Potřebujeme-li pracovat s argumenty goniometrických funkcí vyjádřených ve stupních, platí:
úhel v radiánech = $\frac{\text{úhel ve stupních} \cdot \pi}{180}$

Pro převod ze stupňů na radiány můžeme hodnotu ve stupních násobit číslem 0.0174533 ($= \frac{\pi}{180}$)
nebo dělit číslem 57.2958 ($= \frac{180}{\pi}$)

COS(X) - cos x, X vkládáme v radiánech
Příklady:
COS(6.28318 + 3.14159) = -1
cos 60° = COS(60 / 57.2958) = 0.5
použili jsme výše popsaného převodu ze stupňů na radiány

TAN(X) - tg x, X vkládáme v radiánech

ATN(X) - arctg x, X vkládáme v radiánech

Speciální funkce

SGN(X) - signum x je speciální funkce umožňující rozlišit, zda výrok X je kladný, záporný nebo nulový.
pro X > 0 je SGN(X) = 1
pro X < 0 je SGN(X) = -1
pro X = 0 je SGN(X) = 0

INT(X) - ponechá jen celočíselnou část čísla X, která není větší než X
Příklady:
PRINT INT(125.312) - výsledek 125
PRINT INT(-5.3) - výsledek -6 /je ponechána celočíselná část čísla X, která není větší než X /

Zaokrouhlování čísel na požadovaný počet desetinných míst
LET A = (INT(A * 10^D + 0.5)) / 10^D
přičemž A = obsah proměnné nebo přímo číselná hodnota, kterou zaokrouhlujeme, D = počet požadovaných desetinných míst nebo přímo konstanta /100, 1000 apod./, budeme-li požadovat u všech čísel zaokrouhlení na 2 nebo 3 desetinná místa.

RND(0) - generátor náhodných čísel. Příkaz generuje náhodná čísla v rozsahu od 0 /včetně/ až do 1 /vyjma/, max. hodnota je tedy 0.99999 /nikoliv však 1/.
Příklady:
10 PRINT INT(6 * RND(0) + 1)
20 WAIT(15)
30 GOTO 10
Poznámka: Program bude generovat náhodná čísla /1 až 6/ v rovnoměrném rozložení pravděpodobnosti /např. simulace hrací kostky/.
Na řádku 20 je příkaz WAIT /čti vejt - čekej/ a v závorce uvedené číslo představuje počet desetin sekundy čekacího času. Změnou tohoto parametru můžeme ovládat „čekací" dobu.
Příkaz GOTO /čti goutů - přejdi na/ přikazuje programu přechod na řádek s číslem uvedeným za příkazem /na řádek 10/.
Je zde tedy naprogramována nekonečná smyčka a program se sám nezastaví. Zastavit jej můžeme stisknutím tlačítka CTRL .
Stisknutím kteréhokoliv černého tlačítka se program znovu rozběhne. Stiskneme-li a přidržíme-li tlačítko CTRL a zároveň stiskneme tlačítko s písmenem C , ukončí se program s hlášením, na kterém řádku se zastavil, např.
BREAK IN 20 /čti brejk/

Generování náhodných čísel v rozsahu Nmin až Nmax.:
INT(A * RND(0)) + B
přičemž A = Nmax - Nmin + 1
B = Nmin

Generování náhodných čísel s požadovaným počtem desetinných míst:
(INT((A * RND(0)) * 10^D) / 10^D) + B
přičemž A = Nmax - Nmin
B = Nmin
D = žádaný počet desetinných míst

Parametr za příkazem /funkcí/ RND(0) je bezvýznamný a nemá vliv na generované čísla.
Potřebujeme-li však, např. pro „ladění" programu /viz část 3.4./, generování stále stejných čísel, uvedeme jako parametr záporné číslo např. RND(-1).

HEX(H) - převede hexadeciální číslo /šestnáctkové - v textu je uvádíme s písmenem H na posledním místě, tedy např. EC00H/ na jeho dekadickou hodnotu.
Předpokladem je, že hexadeciální /šestnáctkové/ číslo je tvořeno posloupností číslic a znaků: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
Příklad:
PRINT HEX(EC00) - dekadická hodnota je 60416

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

6. POKRAČOVÁNÍ

Ludolfovo číslo

PI - je uloženo v paměti počítače s přesností na pět desetinných míst 3.14159
Příklad:
PRINT PI

2.10. Definování uživatelských funkcí -

Při programování se může na různých místech programu několikrát opakovat složitý matematický výraz, který není obsažen v předešlém souboru funkcí.

Abychom tento složitý výraz nemuseli pokaždé opětovně do programu vkládat, můžeme jej předem definovat jako tzv. uživatelskou funkcí a takto uloženou v paměti ji kdykoliv na potřebném místě programu vyvolat.

Uživatelskou funkci definujeme příkazem DEF, ze kterým uvádíme označení funkce, z čehož dvě první písmena FN označující funkci jsou povinná a třetí je název jednoduché proměnné, např. A, X, B1, CC. Za označením funkce se v závorce uvádí jeden /ale i několik/ parametrů. Za přiřazovacím znakem = uvedeme příslušný výraz /uživatelskou funkci/.

Jako příklad si nadefinujeme funkci z kapitoly 2.9., pomocí které budeme zaokrouhlovat v programu čísla na dvě desetinná místa:

```
1Ø DEF FND(A) = (INT(A*1ØØ + .5)) /1ØØ
doprogramujeme
2Ø INPUT "Vloz cislo k zaokrouhleni"; A
3Ø PRINT FND(A)
4Ø GOTO 2Ø
```

Nyní můžeme vkládat různá čísla, např. 3.14159, 8.25843, 3Ø.2995 a sledovat způsob zaokrouhlování.

Pokud v programu kdekoliv vložíme nadefinovanou funkci, bude obsah proměnné nebo konstanta použitá jako argument této funkce zaokrouhlena na dvě desetinná místa.

Příklad: nejdříve stiskněte tlačítko [CR] a potom opravte a doplňte předcházející program

```
4Ø LET B = A*A
5Ø PRINT FND(B)
6Ø PRINT FND(6.38635)
7Ø GOTO 2Ø
```

Na tomtéž příkladu si ukážeme použití několika parametrů uváděných v závorce za označením uživatelské funkce.

Příklad:

```
1Ø DEF FNX(A,D)= (INT(A*1Ø^D + .5)) /1Ø^D
```

a v delším programu pak můžeme vkládat nejen hodnotu k zaokrouhlení, ale i stanovit počet míst, na které má být číslo zaokrouhleno.

např. 5Ø PRINT FNX (6.25741,3)

Příklad :

```
1Ø LET A=3.58672
2Ø DEF FND(A)=(INT(A*1Ø + .5)) /1Ø
3Ø PRINT A                          /před úpravou 3.58672/
4Ø A = FND(A)
5Ø PRINT A                          /po úpravě 3.6/
6Ø END
```

Podle způsobu stavby programu se může v průběhu programu změnit /upravit/ obsah proměnné; v našem případě se obsah proměnné A změnil z původních 3.58672 na 3.6

3. Sestavování obsáhlejších programů

3.1. Automatické číslování řádků

Při psaní delších programů začínají být čísla programových řádků tak velká, že jejich stálé opisování a přičítání kroků programátora zbytečně zdržuje.

Většinou číslujeme řádky s určitým pevně stanoveným krokem, a proto má počítač IQ 151 vestavěno automatické číslování programových řádků.

Napíšeme-li před zahájením programování povel AUTO, bude počítač automaticky /po odeslání [CR] / číslovat následující řádky s krokem po 1Ø.

AUTO 5,5 vložíme-li za povel parametry, započne řádkování na řádku odpovídajícímu prvnímu parametru s krokem odpovídajícím druhému parametru.

AUTO 1ØØ,2Ø - bude automaticky řádkovat po 2Ø, počínaje řádkem 1ØØ /první parametr udává řádek, na kterém řádkování začíná, druhý parametr udává krok/.

[CTRL] [(] - přidržíme-li stisknuté tlačítko [CTRL] a stiskneme tlačítko [(] /levá hranatá závorka/, zruší se povel AUTO - automatické číslování řádků bude zrušeno.

3.2. Funkční tlačítka

Některá tlačítka klávesnice mají v horním a dolním úzkém proužku uvedeny názvy nejčastěji používaných příkazů jazyka BASIC, či nejběžnější matematické funkce.

Vlevo na klávesnici je šedé tlačítko s označením [FA] /FUNKCE A - horní proužek/. Stiskneme-li /a přidržíme-li/ toto tlačítko a zároveň stiskneme černé tlačítko, které má v horním proužku napsáno příkazové slovo /např. PRINT, LIST, INPUT/, vypíše se tento příkaz automaticky na obrazovku.

Podobnou funkci má i šedé tlačítko [FB] /FUNKCE B - dolní proužek/, umístěné ve spodní řadě klávesnice vpravo. Vypíše příkazy či funkce uvedené v dolním proužku.

Obě tlačítka velmi zrychlují a usnadňují sestavování delších programů

POZOR ! Nezaměňte příkazy LIST a LLIST, a také PRINT a LPRINT. Příkazy s předsazeným písmenem L platí pro výpis programů nebo výsledků pomocí tiskárny, kterou je možno k počítači IQ 151 připojit jako přídavné zařízení.

3.3. Postup návrhu složitějšího programu

Nejprve si musíme přesně ujasnit, co vše má program dělat a navrhnout hrubé řešení problému.

Stanovíme, jakým způsobem budeme do počítače vkládat základní /počáteční/ hodnoty a jakou formou budeme požadovat zobrazení /dílčích nebo konečných/ výsledků.

Celý rozvržený program rozdělíme na posloupnost problémů a každou takto vzniklou část dělíme dále, až je počáteční rozsáhlý problém rozdělen na zcela jednoduché části - které již jdou lehce formulovat v programovacím jazyce.

Po této analýze začneme sestavovat podrobný návrh programu /po jednotlivých programových řádcích/ spojený s tzv. laděním programu.

3.4. Ladění programu

Dokončíme-li při sestavování programu nějakou logicky uzavřenou část, ihned se přesvědčíme, zda pracuje tak, jak jsme předpokládali; ověříme si, zda jsme se nedopustili nějaké chyby. Tomuto postupu při odstraňování chyb se říká „ladění programu".

Okamžité ověřování právě dohotovených částí programu má mnoho výhod:

1/ Právě dokončenou část programu máme ještě v paměti, pamatujeme si její činnost a náhodnou chybu snadněji najdeme.

2/ Pokud jsme provedli pečlivou kontrolu a odstranění případných chyb v předcházejících částech programu a program přesto nepracuje /nefunguje/, můžeme předpokládat, že náhodná chyba může být jen v právě dokončené části programu.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

7. POKRAČOVÁNÍ

3/ Přistoupíme-li k odstraňování chyb až po úplném dohotovení celého programu, může se stát, že po odstranění chyby, která se nalézala někde na začátku programu, se podmínky pro další průběh programu natolik změní, že bude nutno zbývající část programu zcela nebo zčásti přepracovat.

Časem poznáte, že riskovat se nevyplácí a naučíte se ihned po napsání každou ucelenější část programu přezkoušet.

Každý začínající programátor dojde velmi brzy ke dvěma zjištěním :

- do každého programu se vloudí alespoň jedná chyba,
- ty nejjednodušší chyby se nejhůře hledají.

3.5. Vývojový diagram

Tvoříme-li program, sestavujeme ho spravidla nejprve na papíře. Jedná-li se o složitější program, pomáhají si někteří programátoři tzv. vývojovým diagramem, nebo kopenogramem.

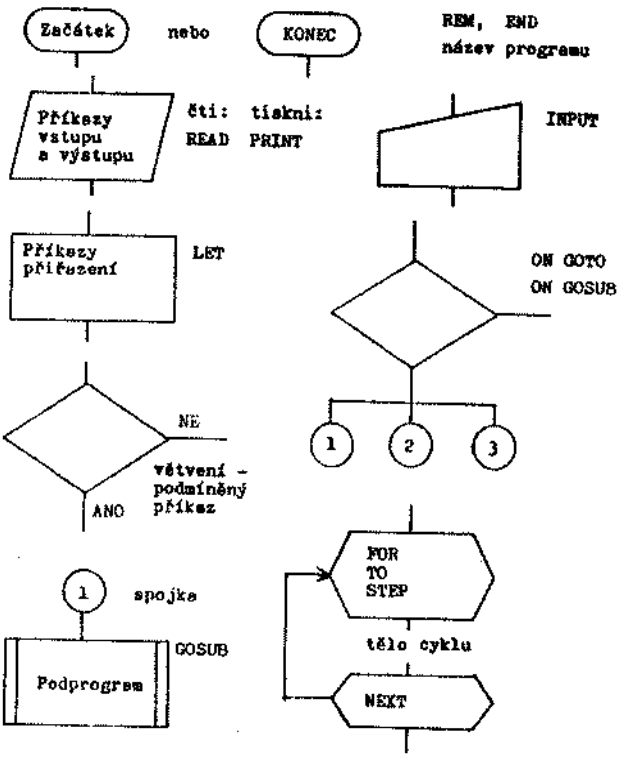
Celý problém přípravy programu by se dal shrnout takto:

- Jednoduché /lineární/ programy sestavujeme přímo na obrazovce počítače, nebo /což je častějším jevem/ si program nejdříve napíšeme na list papíru.

- Při středně složitých programech /větvených, cyklických/ je vhodné vypracovat vývojový diagram a to spravidla na co největším archu papíru /z důvodu přehlednosti/.

- U velmi složitých programů se vývojový diagram stává spletitým a nepřehledným. Někteří programátoři používají pro větší názornost kopenogram, tj. píší určité celky programu na různobarevné papíry /nebo je uzavírají do barevných rámečků/. Toto barevné rozlišení přispívá ke snadnému hledání jednotlivých na sebe navazujících částí programu.

Pokud nejsme profesionální programátoři, používejme pouze základní značky grafického znázornění vývojových diagramů. Stačí, omezíme-li se na tyto základní značky :



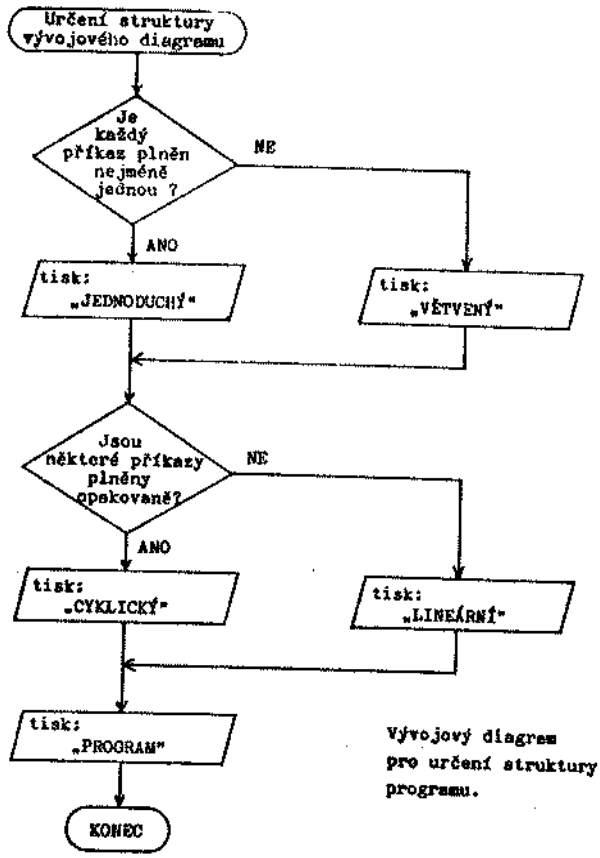
Podle vnitřní struktury dělíme programy na:

- jednoduché lineární
- větvené lineární
- jednoduché cyklické
- větvené cyklické

Na následující stránce je uveden pomocný vývojový diagram, pomocí kterého můžeme lehce určit strukturu sestavovaného programu.

Projdeme-li uvedeným diagramem, zjistíme vnitřní strukturu našeho programu. Prozatím byly všechny naše programy jednoduché, lineární.

Vývojový diagram „čteme" vždy shora dolů a to i v případě, že průběh není vyznačen šipkami.



Vývojový diagram pro určení struktury programu.

3.6. Zadání programu „ampule"

Přistoupíme k sestavení složitějšího programu, na němž se naučíme používat rozhodovacího příkazu IF ... THEN ... /čti if dzen/.

Zadání: Laboratoř vyrobí za určitý časový úsek 3Ø ampulí vzácného léku. Ampule musí obsahovat minimálně 4Ø a maximálně 5Ø mg léku. Automatické váhy hlásí počítači pořadová čísla ampulí 1 až 3Ø a jejich hmotnost. Počítač má sestavit tabulku se čtyřmi sloupci:

AMPULE POD V TOLER. NAD

Ve sloupci AMPULE má počítač vytisknout pořadové číslo ampule a její hmotnost vytisknout ve sloupci POD /nedosahuje li hranice tolerance/, v TOLER. /je-li hmotnost ampule správná/, nebo ve sloupci NAD /má-li ampule hmotnost vyšší než je dovoleno/.

Celou stránku 34 zabírá vývojový diagram programu „ampule", do kterého /po pravé straně jednotlivých grafických symbolů/ byly připsány i čísla řádků sestaveného programu, aby bylo usnadněno porovnání jednotlivých částí programu s vývojovým diagramem.

Ve výpisu programu jsou jednotlivé ucelené části odděleny vodorovnou dělící čárou, jejíž používání rovněž usnadňuje orientaci ve složitých výpisech programů.

```
1Ø DATA 37,41,45,51,5Ø,49,4Ø,55,47,44
2Ø DATA 42,39,48,44,44,46,52,43,42,5Ø
3Ø DATA 41,4Ø,42,47,48,38,45,5Ø,5Ø,51
```

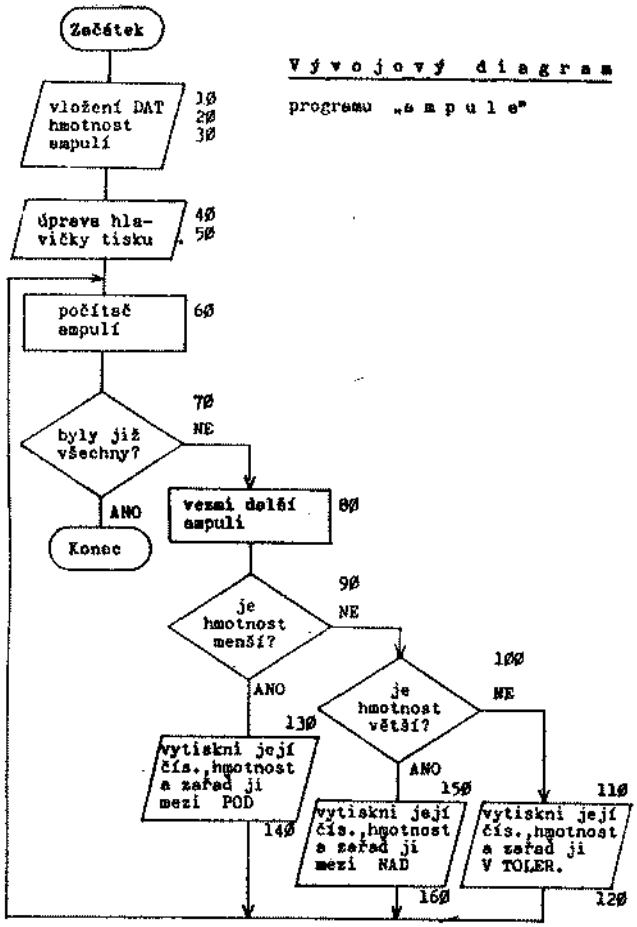
(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

8. POKRAČOVÁNÍ

```
4Ø PRINT "AMPULE   POD     V TOLER.    NAD"
5Ø PRINT "-------------------------------"
6Ø LET C=C+1
7Ø IF C=31 THEN 17Ø
```



Vývojový diagram programu „a m p u l e"

```
8Ø READ A
9Ø IF A<4Ø THEN 13Ø
1ØØ IF A>5Ø THEN 15Ø
11Ø PRINT TAB(2) C; TAB(19) A
12Ø GOTO 6Ø
13Ø PRINT TAB(2) C; TAB(9) A
14Ø GOTO 6Ø
15Ø PRINT TAB(2) C; TAB(27) A
16Ø GOTO 6Ø
17Ø END
```

Řádky 1Ø až 3Ø nám nebudou dělat potíže; za poslední konstantou na řádku nesmíme udělat čárku /oddělovač/.

Na řádku 4Ø nejsou symbolem ␣ vyznačeny počty vložených mezer [SP] , spočítejte je podle symbolů podtržení na řádku 5Ø. Jako znaku podtržení je na řádku 5Ø použito posledního černého tlačítka ve třetí řadě klávesnice vpravo.

Do proměnné C bude program ukládat /připočítávat/ čísla jednotlivých ampulí od 1 do 3Ø /viz řádek 6Ø/.

Řádek 7Ø obsahuje nový /rozhodovací/ příkaz. Řádek bychom mohli číst takto:

IF /je-li/ C=31 THEN /potom/ přejdi na programový řádek 17Ø. Pokud proměnná C neobsahuje konstantu 31 /počet ampulí/, přejde program na následující řádek a příkaz uvedený za THEN se nevykoná.

> Pamatujme ! Je-li výraz stojící za IF pravdivý, potom se příkaz, nalézající se za THEN, vykoná. Není-li výraz za IF pravdivý, přejde program ihned na další programový řádek.

Řádek 8Ø přečte příkazem READ při každém průběhu programem jednu další konstantu /hodnotu/ z DATA a uloží ji do proměnné A.

Na řádcích 9Ø a 1ØØ jsou opět podmíněné příkazy, které podle toho, zda obsah proměnné A bude <4Ø nebo >5Ø, odešle program k pokračování na příslušné programové řádky.

Řádek 11Ø obsahuje tzv. tabulátor. Znamená to, že PRINT TAB(2) C zobrazí obsah proměnné C /číslo ampule/ počínaje 2. sloupcem /pozicí/ na řádku. Víme, že sloupce /pozice/ na řádku jsou značeny Ø až 31, zobrazí /vytiskne/ se tedy konstanta uložená v proměnné C na 4 pozici od levého kraje obrazovky /musíme počítat i s „nezobrazeným" kladným znaménkem/.

Příkaz GOTO na řádku 12Ø znamená přejdi na programový řádek, jehož číslo je uvedeno za příkazem. Program se tedy přesune /vrátí/ na řádek 6Ø, zvětší hodnotu proměnné C o jednu /připočte další ampuli/ a bude se sem vracet tak dlouho, dokud příkaz READ nevyčerpá celou zásobu DAT a proměnná C nenabude hodnoty 31.

V tom okamžiku přejde program na poslední řádek s číslem 17Ø, kde bude příkazem END ukončen.

Vložme předešlý program pečlivě do paměti počítače a dříve než ho spustíme /RUN/, přečtěte si pozorně následující řádky.

Tisk /zobrazení/ výsledků programu, obsahující hmotnosti jednotlivých ampulí, je mnohem delší než počet řádku na obrazovce. Po zaplnění obrazovky prvními údaji, počne obsah „rolovat" směrem vzhůru. Zastavit „rolování" /pozastavit běh programu/ můžeme stisknutím tlačítka [CTRL] ; po stisknutí kteréhokoliv černého tlačítka běh programu bude pokračovat. Zastavováním programu můžeme kontrolovat správnost zápisu hmotnosti ampulí do příslušných sloupců.

Stejným způsobem můžeme zastavovat i výpis /LIST/ delších programů.

3.7. Priorita operátorů

Při programování příkladů na úrovni středních škol se mohou vyskytovat i poměrně velmi složité kombinace funkčních, aritmetických i logických výrazů. Podobně jako platí přísné pořadí početních úkonů v matematice, platí i v jazyku BASIC pořadí vyhodnocování jednotlivých operátorů.

Vložený výraz je počítačem vyhodnocován zleva doprava, přičemž priorita operací je dána níže uvedeným pořadím:

1. ABS, COS, LOG, SIN etd. nejdříve jsou vyhodnoceny funkce,
2. A^2 dále jsou vyhodnoceny mocniny
3. -X unární mínus - vytvoření opačného čísla k číslu X,
4. * , / násobení, dělení /stejná priorita/,
5. + , - sečítání, odčítání /stejná priorita/,
6. <, <=, =, >=, >, <> relační operátory,
7. NOT, AND, OR logické operátory /priorita v pořadí tak, jak jsou psány/.

Výraz je počítačem zpracovávám zleva doprava, dokud se nedospěje k první levé závorce. K ní je vyhledána odpovídající pravá závorka a potom je vyhodnocen výraz uvnitř těchto závorek. Postup se stále opakuje /směrem zleva doprava/, až po vyhodnocení posledního výrazu /s poslední pravou závorkou/.

Po vyhodnocení výrazů v závorkách se provádějí operace s vyšší prioritou. Operace se stejnou prioritou se vyhodnocují vždy zleva doprava.

Potřebujeme-li změnit pořadí řešení výroku, použijeme potřebný počet úplných párů závorek.

Záporné znaménko před konstantou /nebo proměnnou/, kterou umocňujeme, chápe počítač jako znaménko pro celý výraz, tedy -5^2 je chápáno jako -(5^2) = -25 a nikoliv jako (-5)^2 = +25 .

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

9. POKRAČOVÁNÍ

Ke každé otevírací /levé/ závorce musíme mít příslušnou uzavírací /pravou/ závorku. Nejsme-li si jisti prioritou některého operátoru, pomozme si použitím závorek.

Nadbytečné, ale správně umístěné závorky nemohou uškodit.

3.8. Programový cyklus

Programový cyklus umožňuje několikanásobné zopakování určitých příkazů.

Příkaz cyklu v jazyku BASIC má tvar :

```
FOR ... TO ... STEP ...  -  ... NEXT
```

Sestavte následující program pro výpis výsledků malé násobilky podle vloženého základu Z.

```
1Ø CLS
2Ø INPUT "Vloz zaklad (1 az 1Ø) "; Z
3Ø FOR I=1 TO 1Ø STEP 1                    /hlavička cyklu/
4Ø PRINT I,I*Z                             /tělo cyklu - výpočet/
5Ø NEXT I                                  /testování - konec cyklu/
6Ø GOTO 2Ø
```

Na řádku 3Ø je tzv. hlavička cyklu, která uvádí, že v prvním průběhu programu bude do řídící proměnné cyklu I vložena hodnota 1. Obsah této proměnné bude při každém průchodu cyklem zvyšován o hodnotu kroku, uvedenou za příkazovým slovem STEP, a to až do hodnoty uvedené za pomocným slovem TO. Pokud krok cyklu bude 1 /jedna - jako v našem případě/, můžeme část STEP 1 vynechat.

Řádek 4Ø představuje tělo cyklu, které se bude při každém průchodu programu cyklem měnit podle okamžitého obsahu použitých proměnných. Tělo cyklu může obsahovat i celou řadu programových řádků.

Příkaz NEXT I na řádku 5Ø, představující konec cyklu, testuje, zda hodnota řídící proměnné již dosáhla konečné hodnoty stanovené /uvedené/ za slovem TO; v tom případě program pokračuje následujícím řádkem. Nebylo-li ještě konečné hodnoty dosaženo, vrátí se program na řádek 3Ø, obsah řídící proměnné se zvýší o další krok a opakují se neprogramované příkazy těla cyklu.

Za příkazem NEXT /konec cyklu/ uvádíme jméno řídící proměnné. Jazyk BASIC - 6 dovoluje jméno řídící proměnné v příkazu NEXT vynechat.

Vyzkoušejte :

```
1Ø DATA LEDEN, UNOR, BREZEN, DUBEN, KVETEN, CERVEN
2Ø DATA CERVENEC, SRPEN, ZARI, RIJEN, LISTOPAD, PROSINEC
3Ø DIM M$(12)
4Ø FOR I=1 TO 12
5Ø READ M$(I)
6Ø NEXT I
7Ø INPUT "Vloz cislo mesice"; X
8Ø PRINT M$(X)
9Ø PRINT
1ØØ GOTO 7Ø
```

Nepolekejte se skutečnosti, že řádek 1Ø /a také řádek 2Ø/ se nevejde na jeden řádek obrazovky a pište klidně dál; neobávejte se i případného nesmyslného /náhodného/ rozdělení textu do dvou /obrazovkových/ řádků.

Použijeme-li v programu indexovanou proměnnou s více než třemi indexy, nebo je-li některý index větší než 1Ø, musíme pro jejich uložení rezervovat /dimenzovat/ v paměti potřebné místo. Příkaz DIM na řádku 3Ø nám rezervuje místo pro uložení dvanácti indexovaných řetězcových proměnných /1 až 12/ s označením M$(1) až M$(12).

Jediným příkazem DIM je možno dimenzovat i více proměnných, např.:

```
1Ø DIM M$(12), A(1ØØ) , BS(15,1Ø)
```

Řádek dimenzuje dvanáct paměťových míst pro řetězcovou /indexovanou/ proměnnou M$, sto míst pro indexovanou proměnnou A a stopadesát míst pro dvourozměrnou indexovanou proměnnou BS /15 . 1Ø = 15Ø/.

V případě potřeby můžeme cykly vkládat do sebe, nikoliv však přes sebe /křížení/.

Dovolené řešení cyklů

```
FOR A=1 TO 5
 -
 -
 FOR B=1 TO 1Ø
  -
  -
 NEXT B
 -
 -
 FOR C=1 TO 1Ø
  -
  -
 NEXT C
 -
 -
NEXT A
```

Nedovolené řazení

```
FOR A=1 TO 1Ø
 -
 -
 FOR B=1 TO 1Ø
  -
  -
 NEXT A
 -
 -
NEXT B
```

3.9. Nahrávání programu na magnetofonovou kazetu

Program uložený v paměti počítače je po vypnutí počítače síťovým vypínačem nenávratně ztracen. Sestavujeme-li delší program, je vhodné každou dokončenou část přehrát ihned na magnetofonovou kazetu, a to případně ještě před konečnou úpravou či laděním programu. Ušetříme si tím zdlouhavé opětovné vkládání programu, dojde-li k náhodnému výpadku elektrické energie. Je daleko rychlejší ještě jednou opravit některé chyby, než znovu vkládat třeba i 1ØØØ řádků náhodně smazaného programu. Jednu magnetofonovou kazetu si ponechejme pouze pro záznamy „rozpracovaných" programů. Tento pracovní záznam smažeme teprve tehdy, máme-li konečnou verzi bezpečně nahranou /a překontrolovanou přehráním/ na definitivní kazetě.

Kazety si zřetelně označte a vařte si záznamy, co je na které kazetě uloženo. U kazet s konečným /upraveným a odladěným/ programem odstraníme bezpečnostní výstupky, které zabrání náhodnému stisknutí nahrávacího tlačítka.

Budeme-li později na tuto kazetu nahrávat nový /nebo další/ program, přelepíme vylomený /bezpečnostní/ otvor isolepou; jinak automatika magnetofonu nedovolí nahrání nového programu.

Používáme-li magnetofon K-10 TESLA, vytáhneme z něho nahrávací kabel, neboť jeho zasunutím do zdířky magnetofonu se automaticky vypíná vestavěný mikrofon.

Nyní nahrajeme hlasové záhlaví, název programu a případné další pokyny pro obsluhu programu. Je-li záhlaví příliš krátké, raději je několikrát opakujeme, neboť krátké záhlaví se uprostřed pásku /na němž máme nahranou řadu různých programů/ obtížně hledá. Běžné kazety jsou pro záznam programů zbytečně dlouhé, proto někteří programátoři vkládají mezi jednotlivé programy před nahráním záhlaví asi 60 sekund hudby. Tato hudební vložka velmi usnadňuje hledání začátků jednotlivých nahraných programů.

Těsně za nahrávkou záhlaví magnetofon zastavíme. Obnovíme propojení magnetofonu s počítačem příslušným kabelem. Na obrazovku počítače napíšeme povel MSAVE /MEMORY SAVE, čti sejf - slengově: ulož co je v paměti do „sejfu"/. Nyní spustíme nahrávání magnetofonu /chod vpřed + červené nahrávací tlačítko/, ozve se tzv. „pilotní" tón /pískání/, který necháme asi 10 sekund znít. Teprve po těchto 10 sekundách stiskneme na klávesnici počítače tlačítko CR a odešleme připravený příkaz MSAVE.

Střídavé „vrčení" a „pilotní tón" nám oznamuje, že jsou nahrávány jednotlivé řádky programu. Příslušným knoflíkem na magnetofonu snížíme hlasitost na nejmenší možnou míru /zvuk při nahrávání programu není lahodný/.

Jakmile se na obrazovce objeví slovo READY a kurzor, nahrávání skončilo a celý program máme uložen na magnetofonové kazetě. Magnetofon zastavte.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

10. POKRAČOVÁNÍ

3.10. Přehrávání programu z magnetofonové kazety

Předpokládáme, že máme propojen počítač s magnetofonem příslušným nahrávacím kabelem. Je dobré počítač na několik sekund vypnout síťovým vypínačem, abychom vymazali případný obsah jeho paměti.

Na obrazovku napíšeme povel MLOAD /MEMORY LOAD, čti loud - slengově „vyloudit" program/.

Do magnetofonu založíme kazetu s programem a spustíme přehrávání. Poslechneme si hudbu /pokud ji máme nahranou/, celé hlasové záhlaví /název programu včetně příp. pokynů/ a jakmile se ozve „pilotní tón" /víme, že trvá asi 10 sekund/, odešleme napsaný povel MLOAD stisknutím tlačítka [CR].

Po ukončení „pilotního tónu" se počnou jednotlivé řádky programu přehrávat do paměti počítače, což můžeme opticky sledovat na obrazovce. Později se naučíme sledovat celý nahrávený program a kontrolovat, zda nejde o chybnou nahrávku. Po ukončení nahrávky /zobrazí se READY a kurzor/ magnetofon zastavíme a můžeme nahraný program spustit. Také při přehrávání snížíme hlasitost příslušným regulátorem.

> Důležité upozornění !
>
> Může se stát, že po 10 až 50 hodinách provozu magnetofonu /podle jakosti používaných kazet/ se začnou objevovat v přehrávaném programu chyby. Některá písmena velké abecedy se změní na malá či jiné znaky. Tyto příznaky signalizují, že štěrbina magnetofonové hlavy je zanesena otěrem z pásku.
>
> V příslušenství k magnetofonu je tyčinka z umělé hmoty opatřená na koncích proužkem filcu. Na filc kápneme 2 - 3 kapky lihu a tímto přípravkem štěrbinu magnetofonové hlavy očistíme. Očistíme také přítlačnou /pogumovanou/ kladku a unášecí hřídel. Poruchu je tímto zásahem spravidla beze zbytku odstraněna.

V prodeji jsou občas k dostání tzv. čisticí kazety se zvláštním krátkým páskem, napuštěným čisticím prostředkem. Přehráním této kazety se hlava magnetofonu očistí.

4. Grafika na IQ 151

4.1. Psaní háčků a čárek

U velké většiny počítačů nemáme prozatím možnost psát českou abecedou tj. používat háčků a čárek. Zpravidla nám to nevadí a rychle si zvykneme číst vložené vysvětlující texty i bez háčků.

Vyhýbejme se však textům, které by připouštěly možnost dvojího významu, např.

ZADEJ HODNOTU X

Není jasno, máme-li hodnotu X zadat - vložit, nebo žádat od počítače /aby nám ji sdělil/.

Nemožnost psát správně český vadí nejvíce při psaní vlastních jmen osob, kdy může dojít k záměně osob, nebo různým zkomolením.

Při zobrazování jmen osob nebo nadpisů si můžeme vypomoci malým trikem.

Vyzkoušejte :

Na začátek řádku napište "/uvozovky/. Celý řádek přejeďte kurzorem, až se kursor objeví na začátku následujícího /obrazovkového/ řádku. Stiskněte jedenkrát [SP], aby kurzor nestál pod uvozovkami. Nyní napište jméno

NADEZDA SEMIROVA"

a za jménem umístěte uzavírací uvozovky. Vraťte se pomocí tlačítka [←] kurzorem zpět, až nad jméno NADEZDA, resp. přesně nad písmeno E, stiskněte a přidržte [SH] a napište malou abecedou dvě písmena v. Popojeďte kurzorem nad příjmení a stejným způsobem doplňte „háček" nad písmenem S /a druhým písmenem R/. Jako čárky nad I a Á použijte apostrof /viz horní řadu klávesnice, tlačítko číslice 7/.

Nyní se vraťte zpět až na počátek řádku /na uvozovky/ a pomocí tlačítka [IC] odtlačte řádek tak daleko, abyste před uvozovky mohli napsat číslo řádku a příkaz PRINT.

```
10 PRINT "
```

Po napsání čísla řádku a příkazu, přejeďte kurzorem celý řádek až za poslední /uzavírací/ uvozovky a takto dohotovený řádek odešlete [CR].

Smažete-li nyní obrazovku a odstartujete náš jednořádkový program příkazem RUN, zjistíte, že IQ 151 dokáže psát i háčky a čárky.

Samozřejmě, že jméno nebo nadpis můžeme umístit kdekoliv na řádku obrazovky, např. uprostřed.

4.2. Přepnutí do grafického režimu

Při sestavování demonstračních programů můžeme pro zvýšení názornosti používat i řadu grafických znaků, jejichž symboly jsou nakresleny na klávesnici počítače.

Do grafického režimu můžeme přejít dvěma způsoby:

1. Stisknutím a přidržením tlačítka [CTRL] a stisknutím tlačítka s písmenem [O] přepneme klávesnici do grafického režimu. Nyní můžeme pomocí jednotlivých tlačítek označených grafickými symboly zobrazovat tyto znaky na obrazovce.

Při kreslení různých grafických obrazců se může stát, že v určitém případě nebude možno posunout kurzor ani tlačítkem [→], ani tlačítkem [SP] - mezerníkem. V tom případě použijte tzv. prázdného grafického znaku umístěného na tlačítku písmene M.

Budeme-li kurzorem znovu /např. při opravě/ přejíždět programový řádek ve kterém je použito grafických znaků, zastaví se kursor na těchto znacích a bude vyžadovat opětovné přepnutí do grafického režimu.

Návrat z grafického do normálního režimu /textového/ provedeme tlačítky [CTRL] a [N] /viz přílohu čís.2 /.

2. Přepnutí do grafického režimu můžeme zajistit i programově vložením funkce CHR$ /CHARAKTER STRING/ a v závorce uvedeným číslem požadovaného přepnutí /viz přílohu čís.2 /.

Vyzkoušejte:

```
10 PRINT CHR$(15)"GOOGO"
```

zobrazí na obrazovce /po odstartování programu/ pět srdcových znaků. Přepnutí programového řádku do grafického režimu platí pouze do konce tohoto řádku, potom se počítač automaticky vrátí do normálního /textového/ režimu.

Přepnutí zpět do normálního režimu ještě v tomtéž řádku můžeme v případě potřeby naprogramovat vložením funkce CHR$(14) - viz přílohu čís.2 .

Vyzkoušejte si sami některé příklady.

4.3. Inverzní zobrazení

Jak v normálním /textovém/ režimu, tak i v režimu grafickém můžeme stisknutím tlačítka [CTRL] a [S] přepnout zobrazování do inverzního režimu, ve kterém budou písmena abecedy i grafické znaky zobrazovány inverzně tj. černé znaky na bílém poli.

Tohoto způsobu používáme pro zvýraznění některých částí textu /nadpisů apod./, nebo z důvodu obměny grafických znaků.

Přepnutí do inverzního zobrazení lze provést i programově /viz přílohu čís. /.

Vyzkoušejte:

```
10 PRINT CHR$(19)"  IQ 151  "
```

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

11. POKRAČOVÁNÍ

4.4. Zobrazení znaku na libovolné pozici obrazovky

Máme-li v počítači zasunut modul VIDEO 32, pracuje počítač IQ 151 se 32 řádky na obrazovce. Při normálním /textovém/ režimu z důvodu lepší čitelnosti textu používá pouze každý druhý řádek. Toto vylepšení textového režimu nám však vadí při zobrazování grafických znaků navazujících na sebe ve svislém směru. Při „kreslení" obrazců pomocí grafických znaků potřebujeme rovněž přesunovat začátek zobrazení na různá místa obrazovky. To vše nám umožní rozšířený příkaz

```
PRINT&r,s                                   /& čteme et/
```

kde r znamená číslo řádku obrazovky v rozmezí ⟨0;30⟩ a s číslo sloupce obrazovky v rozmezí ⟨0;31⟩.

Vyzkoušejte:

```
PRINT&15,20;"ABC"
```

zobrazí na 15 řádku počínaje sloupcem 20 písmena ABC.

```
PRINT&25,10;123                    /musíme použít oddělovače/
```

zobrazí na 25 řádku počínaje sloupcem 10 číslo /konstantu/ 123.

> Při použití rozšířeného příkazu PRINT&r,s vkládáme prvky tisku /písmena a grafické znaky/ do uvozovek, konstanty /číselné hodnoty/ oddělujeme od příkazu středníkem.

Příkaz můžeme dále kombinovat s přepínáním do grafiky nebo do inverse, např.:

```
10 CLS
20 FOR I=0 TO 25
30 PRINT CHR$(15)&I,15;"T"
40 NEXT I
50 END
```

Program zobrazí na obrazovce svislou osu ve sloupci 15.

Další uvedený program zobrazí svislou i vodorovnou osu včetně překřížení.

```
10 CLS
20 FOR I=0 TO 29
30 PRINT CHR$(15)&I,15;"T":NEXT I
40 FOR I=0 TO 14
50 PRINT CHR$(15)&14,I;"Q";:NEXT I
60 PRINT CHR$(15)&14,15;"O"
70 FOR I=16 TO 30
80 PRINT CHR$(15)&14,I;"Q";:NEXT I
90 END
```

Pokuste se uvědomit, proč musí být na řádku 50 /a také 80/ středník. Pokud si nevzpomenete, zkuste středník odstranit a potom program odstartovat. ↓

Podprogram „rámeček"

Další část /do konce kapitoly/ si pouze přečtěte. Její konkrétní použití ponecháme až do prostudování kapitoly 5.

Tam bude také vysvětleno, proč v tomto podprogramu je použito tak vysokých čísel programových řádků.

```
10000 REM * RAMECEK *
10010 FOR I=S1 TO S2 : PRINT CHR$(15)& R1,I;"Q";& R2,I;"Q";:
      NEXT I
10020 FOR I=R1 TO R2 : PRINT CHR$(15)&I,S1;"T";& I,S2;"T" :
      NEXT I
10030 PRINT CHR$(15)&R1,S1;"P";& R1,S2;"—";& R2,S1;"K";
      & R2,S2;"J"
10040 RETURN
```

Těchto pět řádků tvoří universální podprogram, který umí zobrazit čtverec či obdélník na kterémkoliv /předem naprogramovaném/ místě obrazovky.

Podobných universálních podprogramů budeme mít během času celou řadu a nahrajeme-li si je na kazetu, budeme mít možnost je používat v různých dalších programech, aniž bychom je museli znovu zdlouhavě do programu vkládat.

Vidíme zde nový tzv. nevýkonný příkaz REM /REMARK - poznámka/, po jehož přečtení přechází počítač ihned na další programový řádek a vše, co je dále na řádku uvedeno, ignoruje. Tohoto příkazu používáme ke psaní různých poznámek, upozornění či názvu programů nebo jejich částí - tedy vše, co určeno pouze programátorovi. Text je zobrazován pouze při výpisu programu /LIST/.

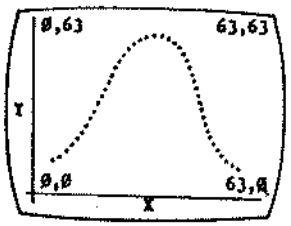
Poslední řádek programu obsahuje příkaz RETURN, jehož význam bude vysvětlen v kapitole 5.

Na řádku 10030 jsou programově zakódovány všechny čtyři grafické znaky rohů ⌜ ⌝ ⌞ ⌟ /P, ¯, K, J - porovnejte s klávesnicí/.

4.5. Semigrafika

Pro bodové znázornění grafických obrazců /křivky, sinusoidy, paraboly apod./ má počítač IQ 151 vestavěnou „semigrafiku", kterou je možno na obrazovce zobrazit 64 krát 64 bodů v souřadnicích X ∈ ⟨0;63⟩ , Y ∈ ⟨0;63⟩ .

Pamatujme, že bod X = 0, Y = 0 je na obrazovce umístěn vlevo dole /viz obrázek/.



Příkaz PLOT X,Y zobrazí na obrazovce bod ■ na místě daném vloženými souřadnicemi X a Y.

Příkaz UNPLOT X,Y vymaže, odstraní bod □ zobrazený na souřadnicích X, Y.

Program zobrazí čtyři body /v každém rohu obrazovky jeden/ na daných souřadnicích.

Vyzkoušejte:

```
10 PLOT 10,50 : PLOT 50,50
20 PLOT 10,10 : PLOT 50,10
30 END
```

U tohoto programu jsme použili možnost umístění více příkazu na jednom řádku - příkazy oddělujeme oddělovačem : /dvojtečka/.

Následující delší program nám umožní posoudit bodové /semigrafické/ znázornění sinusoidy. Vpravo v závorkách jsou uváděny vysvětlující nebo doplňující poznámky.

```
10 CLS : PRINT
20 PRINT CHR$(19)TAB(8);" S I N U S O I D A "    /proložené
30 PRINT                                          inversně/
40 PRINT "Vloz delku PERIODY"
50 PRINT "sinusoidy ⟨20;60⟩"          /v rozmezí od 20 do 60/
60 PRINT "a AMPLITUDU - rozkmit"
70 PRINT "sinusoidy ⟨20;60⟩"
80 INPUT "PERIODA, AMPLITUDA"; A,C        /vlož např. 40,40/
90 CLS
100 FOR I=0 TO 31                          /zobrazení osy X
110 PRINT CHR$(15)&16,I;"Q"                 uprostřed
120 NEXT I                                  obrazovky/
130 LET A=PI/(A/2)
140 LET C=C/2
150 FOR X=0 TO 62
160 LET B=SIN(B)                          /Zobrazení sinusoidy
170 LET B=B+A                              semigrafikou -
180 LET Y=INT(B*C+31.5)                    bodově/
190 PLOT X,Y
200 NEXT X
210 IF INKEY$ = "" THEN 210               /čekací smyčka -
220 GOTO 10                                stiskni kterékoliv
                                           černé tlačítko !/
```

Program nemusí mít koncový příkaz END, neboť se /po stisknutí kteréhokoliv černého tlačítka/ vrací zpět na počáteční řádek /řádky 210 a 220/.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

12. POKRAČOVÁNÍ

Je-li zapotřebí vložit v průběhu programu jeden znak z klávesnice /bez stisknutí tlačítka RETURN/, vkládáme jej příkazem INKEY$.

Vyzkoušejte:

```
1Ø CLS                              /Stiskněte a přidržte
2Ø A$ = INKEY$                       kteroukoliv klávesu.
3Ø IF INKEY$ = "" THEN 3Ø           Program zastavte
4Ø PRINT A$;                        červeným tlačítkem
5Ø GOTO 2Ø                          [RES] /
```

Jednotlivé znaky, se kterými může počítač pracovat, jsou dány mezinárodně užívaným kódem ASCII.

Největší možný počet kombinací je $2^8 = 256$. Prvních 32 znaků /Ø až 32/ jsou řídicí a ovládací kódy, další znázorňují písmena, číslice, znaménka a znaky, kterými počítač disponuje. Kompletní tabulka kódu ASCII /čti aski/ je uvedena např. v publikaci Programování počítače IQ 151 v jazyku BASIC /vydalo KOMENIUM n.p. 1984/.

Řádek 3Ø našeho programu /čekací smyčka/ čeká na stisknutí kteréhokoliv černého tlačítka, program se stále vrací na řádek 3Ø.

Stiskneme-li kterékoliv tlačítko, uloží příkaz INKEY$ do řetězcové proměnné jeho číselný kód /podle ASCII - tabulka standardních kódů/ a řádka 4Ø jej zobrazí.

Příkazu INKEY$ můžeme použít v průběhu programu, potřebujeme-li dodatečně /např. podle získaných výsledků, nebo v různých zábavných hrách/ rozhodnout o jeho dalším pokračování. Je tedy možno použít příkazu INKEY$ jako přepínače.

Vyzkoušejte:

```
-
-
-
5Ø PRINT "Mam pokracovat v programu ?"
6Ø PRINT SPC(1Ø)"- stiskni tlačítko A"
7Ø PRINT "Mam opakovat znovu od zacatku ?"
8Ø PRINT SPC(1Ø)"- stiskni tlacitko Z"
9Ø PRINT "Mam program ukoncit ?"
1ØØ PRINT SPC(1Ø)"- stiskni tlacitko K"
11Ø IF INKEY$="A" THEN 15Ø              /bude pokračovat /
12Ø IF INKEY$="Z" THEN RUN              /znovu od začátku/
13Ø IF INKEY$="K" THEN 52Ø                        /ukončit/
14Ø IF INKEY$="" THEN 11Ø                 /čekací smyčka/
15Ø       pokračování
-         programu                                  %
-
52Ø END
```

Příkaz SPC(1Ø) zobrazí deset mezer /spejs/, totéž jako bychom desetkrát po sobě stiskli tlačítko [SP].

"" je tzv. prázdný znak - žádné tlačítko nebylo stisknuto.

5. Příkazy skoků, přepínače, podprogramy

V předcházejícím programu vidíme, že je někdy nutné porušit čísly udávanou posloupnost programových řádků.

Mimo jednoduchého nepodmíněného příkazu GOTO /jdi-skoč/, použitého v kapitole 3.6, máme možnost použít i příkazu skoku do podprogramu se zpětným návratem /po vykonání operace uložené v podprogramu/.

Vyzkoušejte:

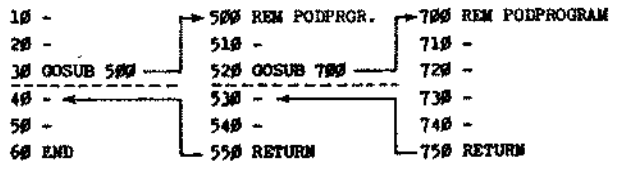
```
1Ø CLS
2Ø R1=4:R2=16:S1=2:S2=13
3Ø GOSUB 1ØØØØ
4Ø R1=1:R2=27:S1=19:S2=3Ø
5Ø GOSUB 1ØØØØ
6Ø WAIT(2Ø)
7Ø R1=9:R2=19:S1=9:S2=25
8Ø GOSUB 1ØØØØ
9Ø END
```

a nyní připojte podprogram „rámeček" z kapitoly 4.4.

Na řádcích 2Ø, 4Ø a 7Ø jsou uloženy rohové body jednotlivých obrazců. Na řádcích 3Ø, 5Ø a 8Ø je příkaz GOSUB /čti gousab - jdi, skoč do podprogramu uvedeného na řádku 1ØØØØ/. Počítač se v průběhu zpracovávání programu celkem třikrát vrací k podprogramu /nakreslení-zobrazení obrazce/, pokaždé však s jinými počátečními parametry. Počítač si „poznačí" tzv. návratovou adresu, umístění příkazu GOSUB, kterým byl odeslán do podprogramu, a jakmile při provádění podprogramu dojde až k příkazu RETURN /čti ritern - návrat/, vrátí se zpět do hlavního programu na nejbližší další příkaz /nebo programový řádek/ se příkazem GOSUB, jehož návratovou adresu si zapamatoval.

V případě potřeby můžeme vkládat další podprogramy do jiných podprogramů, vždy však musí jít o nepřerušenou posloupnost příkazů.

Příklad:



Podmíněný skok

Příkazy ON ... GOTO a ON ... GOSUB umožňují několikanásobně větvit program podle okamžitých výsledků nebo potřeb.

Příklad:

```
1ØØ ON A GOTO 25Ø, 78Ø, 1Ø5Ø, 15ØØ
```

Příkaz nahrazuje několik příkazů ke skokům na různé programové řádky podle okamžitého obsahu proměnné A /můžeme samozřejmě použít jakoukoliv jednoduchou proměnnou/.

Bude-li při zpracovávání řádku 1ØØ v proměnné A uložena hodnota 1, přejde program na řádek 25Ø, při hodnotě 2 na řádek 78Ø atd. Bude-li obsah proměnné A nulový nebo větší než počet čísel řádků uvedených za GOTO, bude program pokračovat na nejbližší vyšším řádku /např. 11Ø/. Při záporném čísle v proměnné A ohlásí počítač chybu.

Číslo řádku uvedené za příkazem GOTO musí v programu existovat, jinak bude hlášena chyba. Obsahem proměnné nemusí být celočíselná konstanta, ale jakékoliv číslo v rozsahu $2 \leq X < 3$, např. 2,285 /bude přepnuto jako by byla vložena konstanta 2/.

Podle stejných pravidel může být i příkaz GOSUB opatřen přepínačem ON, např.

```
2ØØ ON X GOSUB 1ØØØ, 2ØØØ
```

samozřejmě musí podprogramy končit příkazem RETURN a program se vrátí na nejbližší číslo řádku za řádkem 2ØØ.

6. Práce s řetězci /string/

Zcela neobvyklou prací na počítači je používání tzv. řetězcových proměnných /string/. Obsahem řetězcových proměnných mohou být čísla, slova psaná malou i velkou abecedou, znaménka /jako znaky/ i grafické znaky či symboly.

Maximální délka řetězce může být 255 znaků zpravidla uzavřených do uvozovek, které však nejsou součástí řetězce. Neobsahuje-li řetězec čárku /,/, číselný znak nebo mezeru, můžeme uvozovky vynechat. Z důvodu zabránění vzniku zbytečných chyb se však zpravidla používá uvozovek i tam, kde to není bezpodmínečně nutné.

Identifikátory řetězcových proměnných musí být doplněny znakem $ /čti string/, např. A$, B6$, C$(1), D$(2,2)

V jednom programu se mohou používat jednoduché i řetězcové proměnné /případně i indexované proměnné/, mající stejné označení jako např. A$ i A, B1$ i B1, C$(2) i C(2) apod.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

13. POKRAČOVÁNÍ

Příkazy pro práci s řetězcovými proměnnými:

LEN(R$) - zobrazí počet znaků řetězce R$

LEN(A$+B$) - příkaz může mít i složitější výrazy

LEFT$(R$,X) - /čti left string - zleva/ zobrazí X znaků řetězce R$ zleva

RIGHT$(A$,X)- zobrazí posledních X znaků /zprava/ řetězce A$

MID$(B$,X) - řetězec vytvořený ze znaků řetězce B$ počínaje X-tým znakem

MID$(C$,X,Y)- řetězec vytvořený z Y znaků řetězce C$, počínaje X-tým znakem

VAL(C) - převede řetězec C zobrazující číslo na numerickou hodnotu. Řetězec smí obsahovat pouze číslice, desetinnou tečku, znaménka + nebo - a E jako exponent.

STR$(C) - je inverzní funkcí k příkazu VAL(C), zobrazí řetězec, který je znakovým převodem číselného výrazu.

Řetězcové funkce nám umožňují:

- spojovat dva nebo několik znakových řetězců,
- zjišťovat délku řetězce nebo řetězců,
- vybírat části řetězce a jejich skládáním tvořit řetězce nové,
- převádět řetězce na číselný tvar,
- převádět čísla /konstanty/ na řetězcový tvar.

Příklad:

```
1Ø INPUT A$                                   /vlož KOLOSEUM/
2Ø PRINT LEN(A$)
3Ø PRINT LEFT$(A$,4)
4Ø PRINT RIGHT$(A$,2)
5Ø PRINT "ROZ";RIGHT$(A$,2)
6Ø PRINT MID$(A$,3,3)
7Ø END
```

Musíme si uvědomit, že s číslem vloženým jako řetězec pracuje počítač jako s řetězcem a nikoliv jako s konstantou.

Příklad:

```
1Ø LET A$="5Ø"
2Ø PRINT A$+A$
3Ø LET B=VAL(A$)
4Ø PRINT B+B
5Ø LET B$=STR$(B)
6Ø PRINT B$
7Ø END
```

Oba programy nám velmi názorně předvedou práci s řetězcovými proměnnými.

7. Práce se strojovým kódem počítače

Úkolem této příručky není naučit vás pracovat se strojovým kódem počítače. Přesto se však nemůžeme zcela vyhnout některým zásahům do strojového kódu.

Úpravy strojového kódu provádíme po dokonale úvaze a několikanásobné kontrole vkládených hodnot i adres, na které upravené hodnoty ukládáme.

Vložíme-li do programu omylem /syntaktickou/ chybu, upozorní nás na ni překládač jazyka BASIC, nebo program prostě nepracuje tak, jak jsme předpokládali.

Vložíme-li při práci s monitorem nesprávný údaj, nebo správný údaj na nesprávnou adresu, je nebezpečí havárie celého programu, který se může vymknout naší kontrole a jedinou možností /jak zastavit nekontrolovatelný běh počítače/ je počítač vypnout.

7.1. Změna řádkování

V kapitole 4.4. jsme konstatovali, že ačkoliv počítač pracuje s 32 řádky na obrazovce, vynechává z důvodu lepší čitelnosti textu každý druhý řádek. Při „kreslení" různých grafických znaků nám však toto vynechávání řádků vadí.

Vložme do paměti počítače tento program:

```
1Ø PRINT "┌─ ─ ─┐"                          /grafické znaky/
2Ø PRINT "│     ●│"
3Ø PRINT "│   ●  │"                        /při vkládání grafických
4Ø PRINT "│ ●    │"                          znaků nutno přepínat do
5Ø PRINT "└─ ─ ─┘"                          grafického režimu a zpět/
6Ø END
```

Spustíme-li program /RUN/, zjistíme, že se naprogramovaná „hrací kostka" zobrazila s mezerami /roztrhaná do jednotlivých řádků/.

Ponechejme program v paměti počítače a na obrazovku napišme:

```
PRINT PEEK(2Ø)                                a odešleme [CR]
```

Na obrazovce se vypíše obsah paměťového místa s adresou 2Ø; normálně to bývá hodnota 2, což znamená, že počítač „píše" na každý druhý řádek.

Na tuto adresu /2Ø/ vložíme hodnotu 1, aby se zobrazoval každý řádek. Napíšeme na obrazovku:

```
POKE 2Ø,1                                     a odešleme [CR]
```

Příkazem POKE vkládáme na adresu 2Ø požadovanou hodnotu řádkování /v tomto případě 1/.

Spustíme-li nyní program /RUN/, kostka se zobrazí bez dříve nadbytečných /prázdných/ řádků.

Vyvolejme si vložený program na obrazovku /LIST/. Zjistíme, že výpis je proveden hustým řádkováním. Upravme řádkování ještě jednou; vložme

```
POKE 2Ø,5 : LIST                  /dva příkazy na jednom řádku/
```

Vložený program se znovu vypíše na obrazovku, tentokrát s řádkováním po 5 řádcích.

Nakonec vrátíme na adresu 2Ø původní hodnotu 2, nebo počítač na okamžik vypneme, čímž se program monitoru /na původní hodnotu/ upraví sám.

7.2. Změna polarity magnetofonové nahrávky programu

K počítači IQ 151 jsou dodávány dva druhy magnetofonu, buď TESLA K-10 nebo M 710.

Dokud používáme k nahrávání i přehrávání pouze jeden z těchto magnetofonů, je vše vpořádku. Pokusíme-li se však přehrát program nahraný na jednom z těchto magnetofonů na magnetofonu druhém, zjistíme, že to nejde. Výstup z jednoho typu magnetofonu je v opačné polaritě k výstupu ze druhého magnetofonu.

Je sice možné sestavit celkem jednoduché zařízení, které by provedlo změnu polarity, pomocí si však můžeme daleko jednodušším způsobem a to změnou kódu v monitoru počítače. Vložíme:

```
PRINT PEEK(27)                                a odešleme [CR]
```

Zjistíme, že na adrese 27 je normálně vložena hodnota 86; změnou této hodnoty

```
POKE 27,214
```

na hodnotu 214 se změní vstupní polarita počítače a umožní nám přehrát magnetofonovou kazetu /nahraný program/ nahranou na magnetofonu s opačnou polaritou.

Rovněž na tuto adresu vložíme nakonec původní hodnotu /86/, nebo si ji upraví počítač sám po vypnutí síťovým vypínačem /po ukončení práce/.

Pro upřesnění uvádíme, že program nahraný na jakémkoliv typu magnetofonu lze na tomtéž přístroji zpětně přehrát; bez jakýchkoliv zásahů do monitoru.

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

14. POKRAČOVÁNÍ

7.3. Generování tónů na IQ 151

Nakonec se seznámíme ještě s jednou možností IQ 151 - generováním hudebních tónů. Výška tónu je uložena na adrese 24, délka tónu na adrese 23. Příkaz CALL HEX(F973) vyvolá podprogram monitoru; ozve se tón o výšce a délce odpovídající hodnotám uloženým na výše uvedených adresách.

Vložte program, ke kterému je vpravo v závorkách připojeno vysvětlení.

```
1Ø LET P=5                          /délka mezer - pauzy/
15 IF A=2 THEN 17Ø
2Ø READ N                           /celkový počet tónů/
3Ø FOR I=1 TO N                     /hlavička smyčky/
4Ø READ V,D                         /výběr tónu, výška, délka/
5Ø POKE 23,D                        /uložení délky tónu/
6Ø POKE 24,V                        /uložení výšky tónu/
7Ø CALL HEX(F973)                   /volání podprogramu/
8Ø WAIT (P)                         /pauza/
9Ø NEXT I                           /konec smyčky /
1ØØ ON A GOTO 14Ø                   /přepínač/
11Ø RESTORE                         /obnovení čtení dat/
12Ø LET A=A+1                       /čítač přepínače/
13Ø GOTO 15                         /opakování melodie/
14Ø LET P=1Ø                        /úprava délky pauz/
15Ø LET A=A+1                       /čítač přepínače/
16Ø GOTO 15                         /opakování melodie/
17Ø END
2ØØ DATA 16                         /počet tónů/
21Ø DATA 4Ø,52,4Ø,46,4Ø,41,4Ø,39,4Ø,34,4Ø,3Ø,4Ø,27,4Ø,26
22Ø DATA 8Ø,26,8Ø,27,8Ø,3Ø,8Ø,34,8Ø,39,8Ø,41,8Ø,46,8Ø,52
```

Při vkládání programu se pokuste rozebrat, jaká je funkce každého řádku a kudy bude program probíhat.

8. P ř í l o h y

Abychom při sestavování programů nemuseli neustále listovat v této příručce, přinášíme na několika přílohách přehled všech hlavních pokynů potřebných pro programování.

Doporučujeme přepsat přílohy 1, 2 a 3 na formát A4 a vložit do průhledných obalů z umělé hmoty, abyste je měli trvale u počítače „po ruce".

V posledním přehledu je uvedena většina nejpoužívanějších příkazů, povelů a pomocných slov jazyka BASIC - 6 i s ukázkami použití a stručným výkladem činnosti.

Později /až zvládnete základy jazyka BASIC/ se budete vracet pouze k těmto přílohám, abyste si osvěžili paměť a přesvědčili se o správnosti vkládaného programu.

Příloha 3 tvoří pomocný rastr pro rozvržení grafických obrazců na obrazovce. Přiložíme-li na tuto přílohu průsvitný papír, můžeme na něm sestavit potřebnou „grafiku" i s odečtením všech potřebných bodů obrazovky, na kterých má být „grafika" zobrazená.

Závěrem Vám přejeme, aby se počítač IQ 151 stal nejen Vaším trvalým pomocníkem při řešení všech možných problémů, ale i dobrým společníkem při různých počítačových hrách.

Příloha 1

Seznam chybových hlášení IQ 151

00 K příkazu NEXT chybí příkaz FOR
01 nesrozumitelný příkaz, nelze vykonat
02 byl použit příkaz RETURN bez příkazu GOSUB
03 Nedovolený příkaz/povel v neočíslovaném řádku
04 Použitý parametr je větší než 32 767
05 Číselné přeplnění
06 Paměť pro program nestačí, zaplnění paměti
07 Odkaz na neexistující číslo řádku
08 Překročení dovolené nebo nadeklarované velikosti indexu
09 Opakovaná deklarace téhož pole, dimenzováno dvakrát
10 Dělení nulou
11 Příkazu INPUT nebo DEF FN. použito v neočíslovaném řádku
12 Nedovolená operace s řetězcem
13 Přeplnění oblasti CLEAR nebo USR
14 Řetězec je delší než 255 znaků
15
16 Nelze pokračovat příkazem CONT
17 Pro použitou operaci chybí příkaz DEF FN.
18 Parametr je větší než 65 635
19 Překročen parametr v příkazu PLOT nebo PRINT*
20 Pokus o zrušení neexistujícího pole
21 Identifikátor nezačíná písmenem
22 Překročení počtu parametrů definované operace. Počet skutečných parametrů definované operace není roven počtu formálních parametrů.

AUTO automatické číslování řádků po 1Ø
AUTO 5,5 automatické číslování řádků po 5, počínaje řádkem 5
AUTO 1ØØ,1Ø automatické číslování řádku po 1Ø, počínaje řádkem číslo 1ØØ
CTRL [C] zrušení povelu AUTO
LIST vypsání vloženého programu na obrazovku
LIST 5ØØ vypsání vloženého programu počínaje řádkem 5ØØ

Příloha 2

Seznam řídících znaků volitelných klávesou CTRL

CTRL - zastavení výpisu programu nebo běhu programu
CTRL A - zrušení napsané posloupnosti znaků
CTRL B - zablokování klávesnice; opětovným vložením se zruší
CTRL C - v zastaveném výpisu nebo v zastaveném běhu programu způsobí přerušení s hlášením, na kterém řádku došlo k přerušení BREAK IN číslo řádku
CTRL G - pípnutí
CTRL N - přepnutí z grafického do normálního režimu
CTRL O - přepnutí do grafického režimu
CTRL R - přepnutí z inverzního do normálního režimu
CTRL S - přepnutí do inverzního režimu
CTRL I - zrušení povelu AUTO, automatického řádkování

Seznam řídících znaků použitelných v programu

PRINT CHR$(X)

7 - pípnutí
8 - posun kurzoru vlevo
9 - tabelátor, posunuje kurzor po osmi znacích
12 - přesun kurzoru na nultý řádek a nultou pozici
13 - ukončení řádku, zrušení grafického a inverzního režimu
14 - přepnutí z grafického do normálního režimu
15 - přepnutí do grafického režimu
18 - přepnutí z inverzního do normálního režimu
19 - přepnutí do inverzního režimu
24 - posun kurzoru vpravo
25 - posun kurzoru nahoru
26 - posun kurzoru dolů
28 - vsunutí znaku v délce tiskového řádku
29 - zrušení znaku v délce tiskového řádku
31 - mazání obrazovky

(pokračování v příštím čísle)

PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA

15. POKRAČOVÁNÍ

P Ř E H L E D

povelů, příkazů a funkcí jazyka BASIC 6

P o v e l y

AUTO	AUTO AUTO 5,5 AUTO 1ØØ,1Ø CTRL [	automatické číslování programových řádků po 1Ø totéž po 5 od řádku 5 totéž od řádku 1ØØ po 1Ø zrušení povelu AUTO
CONT		pokračování programu nebo výpisu zastaveného příkazem STOP nebo END
LIST	LIST LIST 5ØØ	vypíše vložený program na obrazovku totéž od řádku 5ØØ
MEM		zobrazí počet zbývajících volných byte /bajtů/ paměti
MLOAD		přepíše mgf. záznam do paměti počítače
MSAVE		přepíše program z paměti počítače na mgf. kazetu

P ř í k a z y

CLS		smaže - vyčistí obrazovku
DATA READ RESTORE	DATA 15,8,STO READ A,B,C$ RESTORE RESTORE 15ØØ	vkládání dat: číselných i řetězcových čtení dat z příkazu DATA a ukládání do proměnných příkaz READ započne číst DATA opět od počátku totéž, ale počínaje DATY na čísle řádku 15ØØ
END		ukončí program
DIM	DIM A(3Ø) DIM B(3Ø,2Ø) DIM C(3Ø),D(5Ø)	dimenzuje prostor pro uložení 3Ø indexovaných proměnných jednorozměrného pole dimenzuje prostor pro dvourozměrné pole možno dimenzovat najednou více polí
FREE	FREE A(3Ø)	zruší dimenzování pole
DEF FN. FN.	DEF FNA(cokoliv) FNA(X)	lze definovat jakýkoliv matematický výraz vyvolání definovaného výrazu
FOR TO STEP NEXT	FOR I=Ø TO 2Ø STEP 5 - tělo cyklu - - NEXT I	programový cyklus od I=Ø až do I=2Ø po krocích 5; je-li krok 1, může se STEP 1 vynechat ukončení cyklu ; označení I možno vynechat
GOTO	GOTO 15ØØ	přejdi na řádek...
GOSUB RETURN	GOSUB 2ØØØ	přejdi do podprogramu na řádek... ukončení podprogramu; program se vrátí na nejbližší vyšší číslo řádku se GOSUB
IF THEN	IF A=Ø THEN 5ØØ	je-li výrok pravdivý, potom přejdi na řádek..., nebo vykonej... možno použít operátorů =, >, <, <>, >=, <=
INKEY$		řetězec nabývá hodnoty právě stisknutého tlačítka
INPUT	INPUT A,B,C$	zastaví program a čeká na vložení hodnot z klávesnice
LET	LET A=58	proměnné A přiřadí hodnotu nebo výraz
ON GOTO ON GOSUB	ON A GOTO 1ØØ,5ØØ ON A GOSUB 1ØØ,5ØØ	je-li v proměnné A hodnota 1, přejde program na řádek 1ØØ, je-li v A hodnota 2 přejde na řádek 5ØØ; je-li hodnota vyšší nebo Ø, bude pokračovat nejbližším dalším řádkem. Při záporném obsahu A bude hlášena chyba
PRINT	PRINT A;B$ PRINT"Ahoj !" PRINT&15,1Ø;"A"	vypíše okamžitý obsah proměnných... vypíše beze změny vše co je v uvozovkách zobrazí na řádku 15, sloupci 1Ø písmeno A
REM		poznámka, nezobrazuje se
RUN	RUN RUN 15ØØ	vynuluje proměnné a spustí program od nejnižšího čísla řádku spustí program od čísla řádku 15ØØ
STOP		zastaví program s výpisem na kterém řádku
WAIT	WAIT(5Ø)	čekej; bude pokračovat po uplynutí v závorce uvedeného počtu desetin sekundy

P ř í k a z y pro práci s řetězcy - string

LEN	LEN(R$) LEN(A$+B$)	zobrazí počet znaků v řetězci může mít i složenou funkci
LEFT$	LEFT$(A$,3)	zobrazí první 3 znaky zleva řetězce A$
RIGHT$	RIGHT$(B$,4)	zobrazí poslední 4 znaky řetězce B$
MID$	MID$(R$,3) MID$(R$,3,5)	řetězec vytvořený ze znaků řetězce R$, počínaje 3 znakem řetězec vytvořený z 5 znaků řetězce R$, počínaje 3 znakem
VAL	VAL(C$)	převedení řetězce C$ zobrazujícího číslo na numerickou hodnotu
STR$	STR$(C)	inverzní funkce k VAL zobrazí řetězec, který je znakovým převodem číselného výrazu

F u n k c e

ABS	ABS(X)	zobrazí absolutní hodnotu|X|
ASC	ASC(Z$)	dekadické číslo odpovídající v kódu ASCII vloženému znaku nebo prvnímu znaku řetězce
ATN	ATN(X)	arctg X; výsledek v radiánech
COS	COS(X)	cos X; pro X v radiánech
EXP	EXP(X)	e^X
HEX	HEX(H)	dekadické vyjádření hexadecimálního čísla H
CHR$	CHR$ CHR$(15)	znak odpovídající v kódu ASCII číslu X programový přepínač
INT	INT(X)	největší celé číslo, které není větší než X
LOG	LOG(X)	ln X, pro X>Ø
RND	RND(Ø)	generátor pseudonáhodných čísel <Ø ; 1>
SIN	SIN(X)	sin X, pro X v radiánech
SQR	SQR(X)	$\sqrt[4]{X}$, pro X nezáporné
TAN	TAN(X)	tg X,pro X v radiánech
TAB	TAB(X)	tabulátor, bude tisknout až od sloupce X

Relační operátory a nestandardní příkazy

NOT	NOT X	negace
AND	X AND Y	logický součin
OR	X OR Y	logický součet
PI		Ludolfovo číslo 3.14159
PLOT	PLOT 1Ø,2Ø	zobrazí na zadaných souřadnicích ■
UNPLOT	UNPLOT 1Ø,2Ø	vymaže ze zadaných souřadnic zobrazený ■
SCRATCH		vymaže vložený program

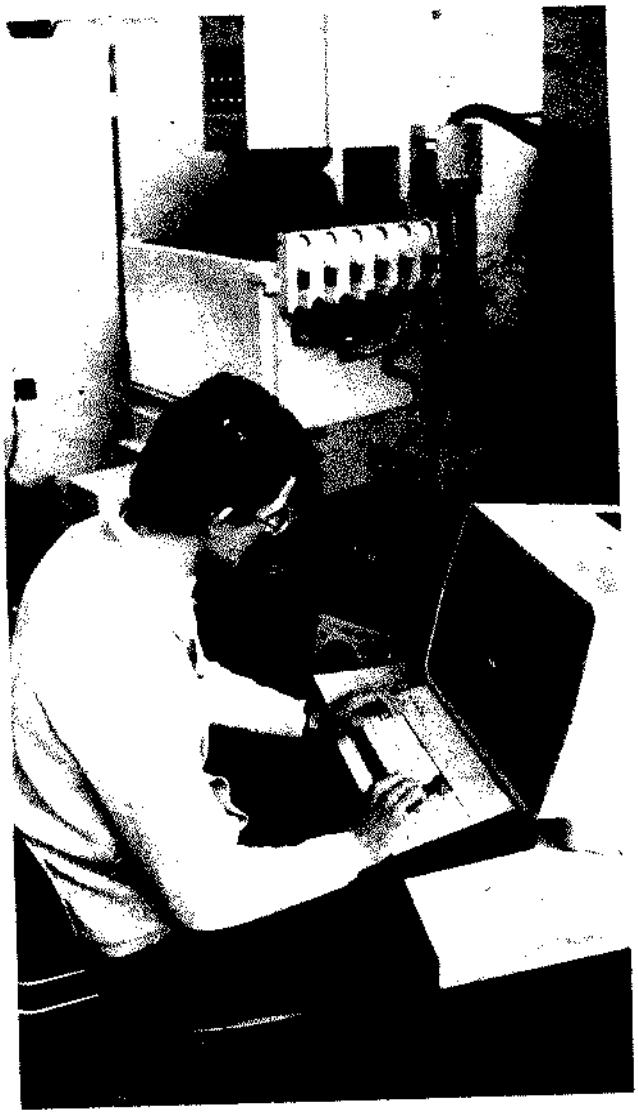
Některé příkazy pro práci s monitorem

CALL	CALL(...)	vyvolání strojového programu
PEEK POKE	PEEK (2Ø) POKE 2Ø,1	určí-přečte obsah paměťové bunky na adrese... slouží pro zápis do dané paměťové bunky
CLEAR		nastevuje hodnotu všech číselných proměnných na Ø, nastevuje řetězcové proměnné na prázdný řetězec, ruší všechny předcházející deklarace a obnovuje DATA
	CLEAR 5ØØ CLEAR 5ØØ,1ØØØ	rezervuje 5ØØ paměťových míst pro ukládání řetězců /oblast CLEAR/; jinak jen 48 míst rezervuje navíc 1ØØØ paměťových míst pro uložení programu ve strojovém jazyce /oblast USR/

KONEC

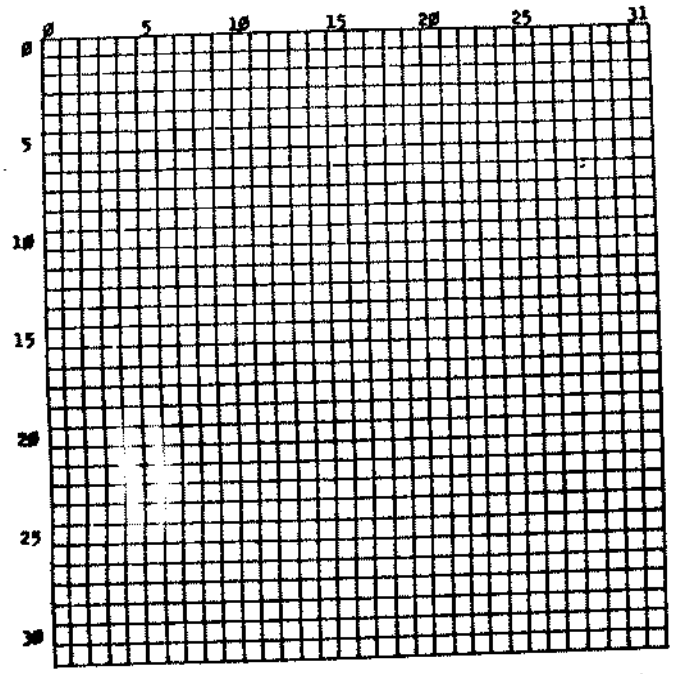
PROGRAMOVÁNÍ V JAZYKU basic

ING. ARCH. ZDENĚK JEDLIČKA



Ve čtrnáctém pokračování seriálu Programování v jazyku BASIC se autor odvolával na přílohu číslo 3, pomůcku pro rozvržení grafiky na obrazovce. Přílohu uveřejňujeme dodatečně a čtenářům, kteří ji postrádali, se upřímně omlouváme.

Příloha 3



PRINT# R,S tisk na libovolnou pozici /R = řádek ⟨0;30⟩ /
/S = sloupec ⟨0;31⟩/

PRINT# 15,12;"*" zobrazí * na 15 řádku, 12 sloupci
PRINT CHR$(15)# 2,10;"O" přepnutí do grafiky, která zůstává v činnosti až do konce řádku; zobrazí na 2 řádku, 10 sloupci grafický znak ♥.
Délka řádku na obrazovce je 32 znaků; značení 0 až 31
Délka počítačového řádku je 80 znaků; 0 až 79, poslední je C.