Programy pro počítač IQ 151

Stanislav Vejmola - Karel Výrut

Programy pro počítač IQ 151

STÁTNÍ
PEDAGOGICKÉ
NAKLADATELSTVÍ

Ing. Stanislav Vejmola, CSc.
Ing. Karel Výrut, CSc.

Programy pro počítač IQ 151

Státní
pedagogické
nakladatelství
Praha 1987

Značka pro textovou proměnnou má v první části publikace tvar $,
v programech druhé části má z technických důvodů tvar $

Jednotlivé části zpracovali:
Ing. Stanislav Vejmola, CSc., část druhá
Ing. Karel Výrut, CSc., část první

Lektorovaly Ing. Zdeňka Němcová a Ing. Jaroslava Pytlíčková

1. vydání

PŘEDMLUVA

Předkládaná knížka obsahuje část BASIC IQ 151 (autor Ing. Výrut, CSc.) a sbírku úloh v jazyce BASIC IQ 151 (autor Ing. Vejmola, CSc). Vznikla na základě aktuální potřeby škol i široké veřejnosti mít k dispozici přehlednou pomůcku a současně i vzory programů pro prvotní seznámení s počítačem.

První část příručky obsahuje stručný popis počítače IQ 151, popis práce s vnější magnetickou pamětí a se souřadnicovým zapisovačem. Těžiště této části textu však spočívá ve výkladu programovacího jazyka BASIC IQ.

Ve druhé části učebního textu jsou uvedeny hotové úlohy a jejich programová realizace na počítači IQ 151.

Uvedené úlohy nebo jejich části mohou bezprostředně sloužit samy o sobě, nebo při programování jiných programů.

Část předkládaných programů jsou hry. I ty mohou být dobrou pomůckou k programování některých situací, které se pak mohou vyskytovat i v jiných programech. Samostatnou otázkou je analýza her na počítačích, která se často vyskytuje i v seriózních publikacích např. o umělé inteligenci, kde zjednodušená pravidla hry poskytují možnost přesnějšího rozboru.

Čtenářům a hlavně začínajícím programátorům především z řad studentů a dětí přejeme pěkné chvilky nad počítačem - k čemuž však někdy mohou vést také předcházející trudné chvilky nad programem. To je však cesta vedoucí k úspěchu.

A u t o ř i

OBSAH

PŘÍLOHY

ČÁST PRVNÍ

1 Počítač IQ 151

Počítač IQ 151 je osobní počítač tuzemské výroby; je určen především pro technické výpočty, méně vhodný je pro zpracování ekonomických úloh, především úloh s rozsáhlými soubory dat.

Konfiguraci počítače IQ 151 obvykle tvoří:

(1) počítač (procesor)
(2) klávesnice
(3) obrazovka
(4) vnější paměť - magnetofon (například kazetový magnetofon TESLA M710A, K-10 apod.)

K počítači lze připojit další výstupní jednotky, například kreslicí zařízení MINIGRAF tiskárny CONSUL apod.

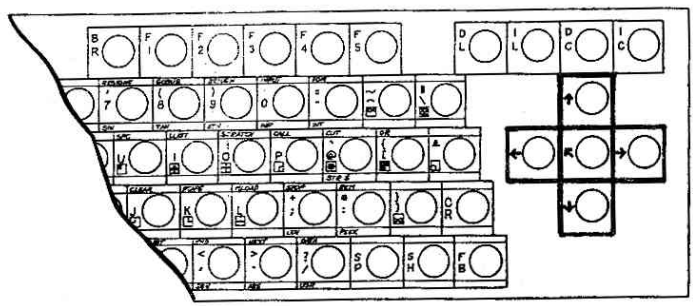
1.1 Procesor

IQ 151 pracuje s osmibitovým mikroprocesorem TESLA MHB 8080. Je vybaven pamětí typu RAM (Random Access Memory) pro uložení programu a dat uživatele a pevnou pamětí typu ROM (Read Only Memory), resp. EPROM, v níž je uložen například interpretační překladač jazyku BASIC a MONITOR - řídicí program systému. Standardní kapacita paměti RAM je 32 Kbyte. Součástí procesoru jsou též moduly pro práci s běžným televizním monitorem - VIDEO 32, zobrazující 32 x 32 znaků na ploše obrazovky, modul pro práci se souřadnicovým zapisovačem apod.

1.2 Klávesnice

Standardním vstupním zařízením IQ 151 je klávesnice; obsahuje celkem 71 barevně odlišených kláves:

(1) Klávesy pro pohyb kurzoru (bílé) v pravé části klávesnice; jsou označeny šipkami vyjadřujícími možné pohyby kurzoru po obrazovce.



Obr. 1 Klávesy pro pohyb kurzoru

Kurzor, česky bychom řekli "ukazatel polohy na obrazovce" [1], určuje aktivní bod na obrazovce; znak vkládaný z klávesnice se zobrazí vždy do místa nastavení kurzoru.

Nastavení kurzoru po uvedení počítače do provozu :

```
BASIC

READY

_
```

Poznámka

V dalším textu vyjadřujeme pozici kurzoru znakem podtržení. Například PRIN_ (kurzor je na pozici za znakem N).

(2) Klávesy pro zápis znaků (černé)

Jednou klávesou lze na obrazovce vygenerovat až 5 různých symbolů (znaků, slov, grafických znaků). Například první klávesou na klávesnici lze zapsat číslici "1", znak "!", slova "DIM" a "POS". K volbě požadovaného znaku proto používáme i funkčních kláves. Platí:

---

1 V případě IQ 151 je kurzor zobrazen jako čtverec, velikosti odpovídající jednomu znaku.

(a) pouhým stiskem klávesy získáme základní znak (např. velké písmeno, číslici);

(b) malé písmeno nebo znak v legendě nad základním znakem získáme současným stiskem klávesy základního znaku a klávesy SH (SHIFT);

(c) slovo uvedené v horním rámečku klávesnice (klíčové slovo) zapíšeme stiskem příslušné klávesy a klávesy FA;

Příklad :

FA + W = THEN
FA + D = PRINT

(d) slovo v dolním rámečku (operátor) zapíšeme pomocí klávesy FB.

Příklad

FB + 5 = EXP

(e) grafický režim zobrazení

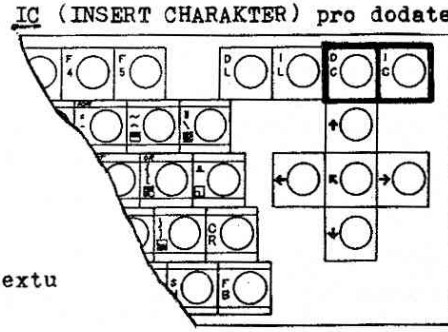
Pro zápis grafických znaků je třeba nejdříve zvolit grafický režim současným stiskem kláves:

CTRL a písmene O.

Grafický znak pak zobrazíme pouhým stiskem příslušné klávesy. Grafický režim zobrazení zrušíme současným stiskem kláves:

CTRL a písmene N.

(3) Klávesy pro opravu vloženého textu jsou v pravé horní části klávesnice; jsou označeny DC (DELETE CHARAKTER) pro vymazání znaků a IC (INSERT CHARAKTER) pro dodatečné vložení znaků.



Obr. 2
Klávesy pro opravu textu

Příklady

(a) chybný text: 10 PRINNT správně má být: 10 PRINT
oprava: 10 PRINNT DC PRINT

(b) chybný text: 10 PRNT správně má být: 10 PRINT
oprava: 10 PRNT IC PR NT I PRINT

(4) Klávesa BR (BREAK) slouží pro přechod z režimu zápisu programu do režimu práce MONITORu; pro běžnou práci s počítačem IQ však není nezbytné se MONITORem zabývat.

(5) Klávesy kláves F1, F2, F3, F4, F5, IL a DL mají význam pouze pro případy použití v aplikačních programech, resp. při práci s podprogramy.

(6) Klávesa SP (SPACES) má funkci mezerníku.

(7) Šedé klávesy (Pozor, barevný odstín mezi bílou a šedou je na klávesnici IQ většinou jen nepatrný.)

(a) O funkci a významu kláves FA, FB, SH, CTRL jsme se zmínili v odstavci (1).

(b) klávesa CR (CURSOR RETURN) má několik funkcí:

(b_1) uloží zapsané znaky z klávesnice do paměti.

(b_2) posunuje kurzor do prvního sloupce následujícího řádku,

(b_3) je pokynem k realizaci (k výkonu) zapsaných instrukcí; z toho vyplývá i její funkce v tzv. přímém výpočetním režimu, v němž využíváme počítače jako kalkulačku.

Příklady

(a) PRINT 3.1415 * 80.50 CR
252.891

(b) PRINT (56.60 + 145.90) * (12.70 / 0.9) CR
2857.5

(c) PRINT (58.5 * (-0.69)) CR
-40.365

(8) Klávesa RES (RESET) - červená klávesa; způsobí smazání znaků na obrazovce a obsahu vnitřní paměti.

1.3 Vnější paměti

Počítač IQ 151 disponuje pouze magnetopáskovou vnější pamětí. Připojit lze jakýkoliv kazetový magnetofon [1], například magnetofon TESLA M710A. Je výhodné pracovat s magnetofonem, který je vybaven zesilovačem; můžeme pak záznam dat doplnit záznamem zvuku, což má význam pro identifikaci programů uložených na magnetické pásce v kazetě.

1.4 Další součásti konfigurace IQ 151

Počítač IQ je v současné době možné doplnit o grafický modul GRAFIK umožňující práci v grafickém režimu, v rozsahu 512 x 325 bodů na obrazovce, dále o tiskárny, například CONSUL 2112 nebo 2113, též o souřadnicový zapisovač MINIGRAF 0507 (Aritma), o modul SESTYK pro připojení dálnopisu apod.

Pro zpracování většiny úloh jsou nezbytné také tiskárny, případně lze použít jako tiskárny kreslicí zařízení MINIGRAF.

Vnitřní paměť typu RAM lze rozšířit až na 64 Kbyte; v tomto případě je pak možné pracovat s operačním systémem MIKROS.

K počítači IQ 151 lze připojit také speciální grafické jednotky pro formáty A3 a A4, které dodávají Laboratorní přístroje, k. p.

---

[1] Omezením v tomto směru jsou pouze nevhodné konektory některých magnetofonů.

2 Stavba jazyka BASIC

Programovací jazyk BASIC (Beginner´s All-purpose Symbolic Instruction Code) je jednoduchý algoritmický jazyk, který se stal základním programovacím prostředkem pro práci s minipočítači a později i s mikropočítači, k nimž můžeme řadit i počítač IQ 151.

BASIC je dokonale vybaven pro práci v dialogovém režimu, čímž zpřístupňuje počítač i těm uživatelům, kteří nejsou specialisty v oblasti výpočetní techniky.

Na druhé straně je třeba upozornit na skutečnost, že jazyk BASIC je určený především pro programování úloh z oblasti vědeckotechnických výpočtů a matematiky, je vhodný i pro zpracování ekonometrických a statistických úloh, ale již méně vyhovuje pro programování ekonomických úloh, vyžadujících práci s rozsáhlými soubory dat a majících požadavky na kvalitní velkokapacitní paměti. Tato nevýhoda je v případě počítače IQ 151 zvláště výrazná, neboť programovací, ale ani technické prostředky tohoto počítače, neposkytují uživateli takové možnosti, jak je tomu například v případě verzí jazyka BASIC pro mikropočítače typu IBM PC.

2.1 Znaky

Základním prvkem programovacího jazyka jsou znaky. Množina znaků, které lze používat pro zápis programů, tvoří abecedu jazyka.

V jazyce BASIC, jako v každém programovacím jazyce, můžeme používat pouze znaků anglické (latinské) abecedy, číslic, operátorů (např.: +, /, - , = apod.), interpunkčních znamének a speciálních znaků, které mají význam pouze pro zápis programů.

Na obrazovce můžeme vygenerovat znaky pomocí

(a) příslušných kláves na klávesnici počítače
(b) kódu ASCII [1] (hexadecimálního kódu).

Například znak % se může zapsat jednak příslušnou klávesou, jednak instrukcí PRINT CHR$(37).

Obecně platí, že pomocí ASCII kódů lze případně generovat i takové znaky, které nejsou na klávesnici počítače (viz Příloha 4).

2.2 Klíčová (standardní) slova jazyka

Klíčová slova používáme pro stavbu příkazů a pro vyvolání funkcí; mají jednoznačně určený význam a tvoří slovní zásobu jazyka BASIC. Například:

EXP, INPUT, GOTO, RUN, IF, CHR$ apod.

K zápisu klíčových slov lze v jazyce BASIC počítače IQ 151 používat pouze velkých písmen.

Každý příkaz v jazyce BASIC začíná klíčovým slovem; úplný seznam klíčových slov uvádíme v příloze 3.

2.3 Instrukce

Instrukce je předpis pro vykonání jedné operace počítače. Instrukce jsou jednak výkonné - příkazy, které realizují určitý prvek algoritmu, jednak nevýkonné - popisy, mající charakter pomocných informací pro výkonné instrukce (například popisy proměnných, popisy tiskových formátů atd.).

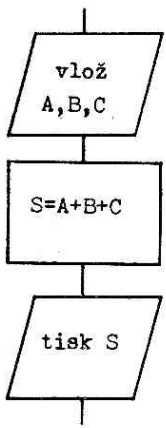
Zvláštním druhem instrukcí jsou povely. Povel sdělujeme počítači stiskem příslušné klávesy (např. CR, RESET apod.) nebo instrukcí bez uvedení čísla řádku. Na rozdíl od příkazů,

[1] American Standard Code for Information Interchange. Tabulku kódu ASCII uvádíme v příloze.

které se vykonávají až po instrukci RUN, se povel vykonává okamžitě.

Příkazy tvoříme pomocí klíčových slov, případně operátorů vyjadřujících druh operace, a pomocí operandů, specifikujících údaje, s nimiž se má příkaz vykonat.

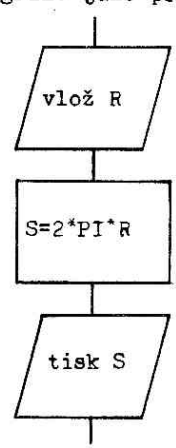
Příklad (součet tří čísel)




```
10 INPUT A, B, C    příkaz vstupu
20 S = A + B + C    aritmetický příkaz
30 PRINT S          příkaz pro zobrazení výsledku
```

Většinu instrukcí lze použít buď jako příkaz, nebo i jako povel. Jsou však povely, například AUTO, MSAVE a další, které nelze použít v programu jako příkaz.

Příklad



(a) PRINT jako povel

```
PRINT 2 * PI * 6
```

(b) PRINT jako příkaz

```
10 INPUT R
20 S = 2 * PI * R
30 PRINT S
```

(PI je konstanta (funkce), vyjadřující Ludolfovo číslo.)

2.3.1 Programové řádky

Příkazy a popisy zapisujeme do programových řádků. V případě IQ 151 může programový řádek obsahovat až 79 znaků, což je dva a půl řádku na obrazovce. Přechod z jednoho řádku obrazovky na nový řádek v rozsahu jednoho programového řádku zajišťuje systém automaticky. Programový řádek v programech v jazyce BASIC plní i funkci návěstí.

Programový řádek tvoří:

(a) číslo řádku (celé kladné číslo 1 - 65529);
(b) text programového řádku.

číslo řádku (text (:text ...))

Na jednom programovém řádku může být několik příkazů oddělených dvojtečkou (:).

Příklad (výpočet třetí mocniny proměnné A)

```
10 INPUT A: PRINT A*A*A
```

Jsou-li v programu dva nebo více řádků se stejnými čísly, počítač akceptuje poslední zápis (programové řádky se stejnými čísly se tedy v paměti "přepisují"). Tímto způsobem lze také opravovat (nahrazovat) chybné programové řádky v paměti řádky novými.

Příklad

```
10 INPUT A, B, C
20 T = A + B + C/2
20 T = A + B/2 + C/4
20 T = A + B/C
```

v paměti je uložen program:

```
10 INPUT A, B, C
20 T = A + B/C
```

Zvláštním případem je oprava (přesněji zrušení, vymazání) řádku zápisem příslušného čísla řádku, ale bez jakéhokoliv textu.

Příklad

```
10 INPUT A, B
20 PRINT "Podíl dvou čísel"
30 PRINT A / B
20
```

v paměti je uložen program:

```
10 INPUT A, B
30 PRINT A / B
```

Vkládání každého programového řádku ukončujeme klávesou CR; teprve potom je text programového řádku uložen z vyrovnávací paměti obrazovky do vnitřní paměti procesoru a je systémem akceptován.

Program je realizován ve vzestupném pořadí čísel řádků. Proto lze do programu kdekoliv vkládat čísla nových řádků. Z tohoto důvodu volíme raději číslování s krokem větším než 1, obvykle 10. Například:

```
10 INPUT A, B, C
20 D = B * B - 4 * A * C
30 X1 = (-B + D  0.5) / 2   A
15 PRINT "Vypocet diskriminantu"
11 IF A = 0 THEN PRINT "Chyba"
```

Program se realizuje v pořadí čísel řádků: 10,11,15,20, 30; systém sám pro výpočet uspořádá řádky vzestupně podle jejich čísel.

Automatické číslování programových řádků lze zajistit povelem:

AUTO.

Uvedeme-li povel AUTO bez parametru, platí implicitně řádkování s krokem 10. Krok lze volit i explicitně, například povel:

AUTO 5

zajistí automatické číslování řádků s krokem 5, tj. řádky se číslují 5, 10, 15 atd.

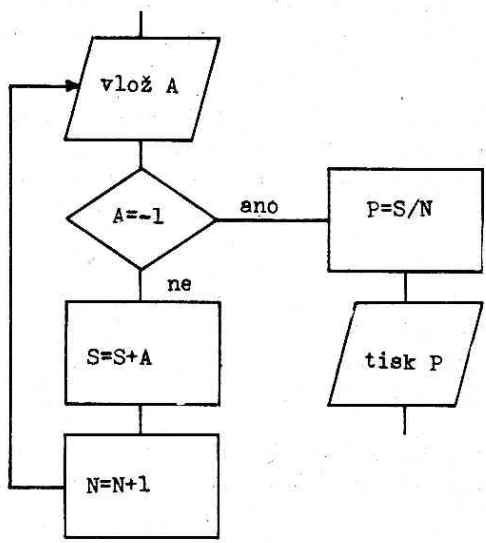
Lze též uvést, od kterého čísla řádku funkci AUTO požadujeme. Například

AUTO 100,20

(řádky se automaticky číslují počínaje řádkem 100 a s krokem 20).

Číslo řádku slouží také jako návěstí, na které se lze odvolat například v příkazech GOTO, IF a v dalších. V následujícím příkladu jde o návěstí v příkazech (20) a (50).

Příklad (aritmetický průměr souboru hodnot; vkládání hodnot končí vložením hodnoty -1, tzv. koncového znaku souboru)




```
10 INPUT A
20 IF A = -1 THEN 60
30 S = S + A
40 N = N + 1
50 GOTO 10
60 P = S/N
70 PRINT P
```

Zrušení automatického číslování zajistíme stiskem kláves:

CTRL +

Poznámka

V paměti počítače IQ lze uložit více programů, jsou-li ukončeny příkazem STOP či END; tato podmínka nemusí být ovšem splněna u posledního programu. Ani v těchto případech ovšem nelze v různých programech použít stejných čísel řádku.

2.4 Sestavení programu - příkazy LIST, CLS a RUN

(1) Již jsme se zmínili, že programový řádek ukládáme do paměti klávesou CR.
Text řádku, tak jak jsme jej zapsali z klávesnice a jak je zobrazen na obrazovce, je dočasně uložen ve vyrovnávací paměti klávesnice; teprve po stisku klávesy CR je uložen do vnitřní paměti procesoru.

(2) Zpráva systému

READY

sděluje, že počítač čeká na další instrukce.

(3) Mezery (klávesa SP) v textu programu nemají syntaktický význam; činí pouze text programu přehlednější.

Příklad

10 INPUT A, B ale i 10INPUTA,B

(4) Chyby ("překlepy") v textu programu opravíme klávesami DC, IC a klávesami pro pohyb kurzoru (viz též kapitola 2.2.2). Opravujeme-li text programového řádku, který je již v paměti, postupujeme takto:

- vyvoláme text řádku na obrazovku příkazem LIST;
- nastavíme kurzor na číslo příslušného programového řádku;
- provedeme opravu (pomocí kláves IC, DL apod.);
- kurzorem přejdeme na konec řádku a stiskneme klávesu CR.

Pozor!

Podmínkou proto, aby systém akceptoval text celého řádku je, abychom kurzorem skutečně přešli všechny znaky řádku

a teprve potom řádek uložili do paměti klávesou CR (zde je podstatný rozdíl mezi prací s počítačem IQ a s počítači typu IBM PC).

(5) Výpis textu programu z paměti na obrazovku volíme příkazem:

LIST

(LIST - listing, výpis)

Příkaz LIST vypíše na obrazovce program od prvního programového řádku.
Požadujeme-li však pouze výpis části programu, například od řádku 100 výše, zapíšeme:

LIST 100

Příkaz LIST /číslo řádku/ tedy nevypíše pouze specifikovaný řádek, ale celý zbytek textu programu, počínaje specifikovaným řádkem.

(6) Příkaz CLS

Příkaz CLS způsobí smazání všeho textu z obrazovky; je vhodné jej zařadit do programu tak, aby výsledky úlohy byly zobrazovány vždy na čisté pole obrazovky.

Příklad (tabelace vkládaných hodnot, jejich druhých a třetích mocnin):

```
 5 CLS
10 INPUT A
20 PRINT A; A*A; A*A*A
30 GOTO 10
RUN
```

(po povelu RUN se smaže text programu z obrazovky).

(7) Pokyn k realizaci programu, který je uložen ve vnitřní paměti, sdělujeme počítači povelem:

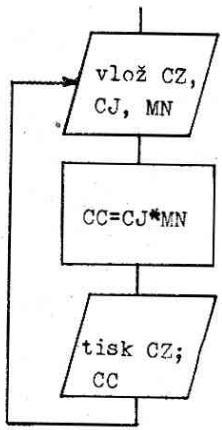
RUN

Je-li v programu uveden příkaz vstupu (například INPUT), požaduje systém po povelu RUN vložení vstupních dat tím, že na obrazovce zobrazí znak dvojtečky (:).

Vkládání vstupních dat ukončíme, podobně jako vkládání

programových řádků, klávesou CR.

Příklad výpočet ceny celkem (CC) zboží CZ (číslo zboží) z údajů o ceně za jednotku (CJ) a o množství zboží v jednotkách (MN)




```
10 INPUT CZ, CJ, MN
20 CC = CJ * MN
30 PRINT CZ; CC
40 GOTO 10
RUN

: 1,5,10          vstup
 1  50           výstup
: 2,5.50,50.00    vstup
 2  275          výstup
:                 atd.
```

Program uložený v paměti lze příkazem RUN realizovat kdykoliv, opakovaně, s použitím různých vstupních hodnot.

Dopustíme-li se při vkládání vstupních dat chyby tím, že vložíme méně hodnot než udává seznam proměnných u příkazu vstupu (INPUT), reaguje systém znakem

::

Chybu odstraníme dodatečným vložením chybějících hodnot. Naopak na vložení většího počtu vstupních hodnot reaguje systém zprávou

* EXTRA DATA IGNORED

a nadbytečné položky skutečně ignoruje.

Příkaz RUN současně vynuluje obsahy všech proměnných v paměti a program realizuje vždy od nejnižšího čísla programového řádku.

Chceme-li však realizovat program od některého z řádků

s vyšším programovým číslem, musíme číslo řadku uvést v příkazu RUN:

RUN číslo řádku

Příklad

.
.

RUN 70 (program se realizuje od řádku 70; v tomto případě se ovšem případné obsahy proměnných používaných do řádku 70 nenulují, což může být, především v případech ladění programu, výhodné a smyslem takovéto realizace výpočtu)

Poznámka
Příkazem RUN současně přiřadíme všem abecedněčíselným proměnným prázdný řetězec (mezery).

2.4.1 Překlad programu

Algoritmus zapsaný instrukcemi programovacího jazyka tvoří tzv. zdrojový program (Source Program). Počítač takovýto program nejprve analyzuje, dekóduje a upraví ("přeloží") do tvaru, který je již vhodný pro zpracování. U každé instrukce programu provádí kontrolu, zda jsme dodrželi slovní zásobu jazyka a zda je instrukce i syntakticky správná.

Syntaxi programovacího jazyka tvoří pravidla pro tvorbu lexikálních prvků jazyka (čísel, abecedněčíselných konstant [1], identifikátorů, klíčových slov, operátorů a oddělovačů), příkazů a programových řádků.

Překlad je úspěšný pouze tehdy, pokud programátor dodržel všechna pravidla pro zápis programu. V případě chyby sdělí počítač, přesněji řečeno zvláštní program počítače - překladač, zprávu o druhu chyby, a to číslem (kódem), případně místem výskytu (číslem řádku).

[1] Užívanými pojmy jsou též alfanumerická konstanta textová položka, řetězec znaků.

Příklad

```
.
.
50 PRNT A
.
RUN
* 01 ERROR IN 50
```

Ani nejdokonalejší počítač však neodhalí naše prohřešky proti sémantice textu programu a nepozná chyby v algoritmu řešení úlohy.

Překlad programu spočívá především v transformaci symbolů operací (INPUT, PRINT, GOTO apod.) do číselného kódu.

Další etapa analýzy (interpretace) programu nastává až po pokynu k výpočtu příkazem

RUN

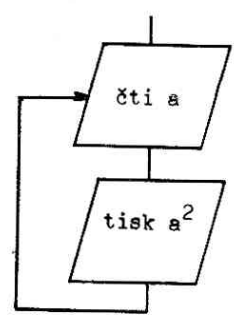
Může se stát, že v programu, v němž se nevyskytly syntaktické chyby, odhalí počítač dodatečně chyby vyplývající z nesprávných deklarací polí, chybného ukončení příkazu cyklu apod.

Při výpočtu se konečně mohou projevit i chyby logické, způsobené chybným algoritmem řešení.

Proces postupného odstraňování všech druhů případných chyb nazýváme laděním programu.

Příklad

Úkolem je sestavit a odladit program pro výpočet druhých mocnin hodnot vkládaných z klávesnice.



```
10 INPUD A
20 PRINT A*A          * symbol násobení
30 GOTO 10
RUN                   pokyn k výpočtu
```

Reakce počítače:

```
*01 ERROR IN 10
```

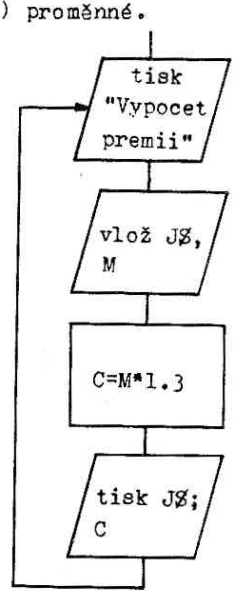
(01 kód chyby, IN 10 = "v řádku č. 10"; v tomto případě se jedná o syntaktickou chybu: správně má být INPUT).

Seznam chyb a jejich kódů je uveden v příloze 1.

2.5 Konstanty a proměnné

Údaje, které chceme programem zpracovávat, jsou v programu zapsány buď přímo svojí skutečnou hodnotou, nebo pouze symbolicky jménem (identifikátorem). Podle toho rozlišujeme v programech:

(a) konstanty,
(b) proměnné.



Příklad (výpočet zvýšení mzdy o 30 %; J$-jméno, M-mzda, C-čistá mzda)

```
 5 PRINT "Vypocet premii"
10 INPUT J$, M
20 C = M * 1.3
30 PRINT J$;C
40 GOTO 10
```

V našem programu jsme použili:

(a) proměnné: JØ, M, C, jejichž hodnota se mění po každé realizaci příkazu INPUT, v závislosti na hodnotách vložených z klávesnice;

(b) konstanty: "Vypocet premii", 1.3, jejichž hodnotu příkaz vstupu neovlivní.

Je zřejmé, že na rozdíl od konstant se hodnoty proměnných mění, modifikují, v našem příkladu například vložením konkrétní hodnoty v příkazu INPUT. Jinými slovy: konstanta je konkrétní číselná nebo abecedněčíselná hodnota uvedená přímo v programu, kdežto proměnným přiřazujeme hodnoty až v průběhu plnění programu.

Z našeho příkladu je také zřejmé, že modifikace hodnot proměnných je spojena s postupným přepisováním obsahu paměťových míst (JØ,M,C) v každém kroku programu (po každém provedení příkazu INPUT).

Příklad

```
10 INPUT JØ,M
.
.
40 GOTO 10
```

krok	vložíme	obsah paměti	
		JØ	M
1.	NOVAK, 2600	NOVAK	2600
2.	HORAK, 5000	HORAK	5000
3.	HUDEC, 2100	HUDEC	2100
4.	CHVALA,3000	CHVALA	3000
atd.			

Při zápisu programu je nutno rozlišovat konstanty a proměnné číselné (numerické), v našem příkladu jsou to proměnné M a C a konstanty a proměnné abecedněčíselné (alfanumerické, textové); v našem případě jde o alfanumerickou konstantu "Vypocet premii" a alfanumerickou proměnnou JØ.

veličiny	konstanty	proměnné
numerické	1.3	M, C
alfanumerické	"Vypocet premii"	JŠ

2.5.1 Číselné konstanty (čísla)

Pro zápis numerických konstant lze použít pouze těchto znaků:

1,2,3,4,5,6,7,8,9,0,+,-,.,E,

V programech zapisujeme číselné konstanty jednak způsobem běžným pro zápis čísel, jednak exponenciální (semilogaritmickou) formou.

V prvním případě, při přímém zápisu hodnoty konstanty, používáme pouze číslice, znaménka (znaménko + není třeba uvádět) a desetinné tečky (místo desetinné čárky, kterou v jazyce BASIC IQ nelze používat). V případě počítače IQ 151 (8mi bitový mikroprocesor) lze tímto způsobem vyjádřit maximálně šestimístná čísla.

V zápisech exponenciální formou vyjadřujeme konstanty pomocí mantisy, znaku E a exponentu :

mEe

m je číselná konstanta v přímém tvaru;

E je znak, který vyjadřuje, že se jedná o exponenciální tvar zápisu čísla;

e je exponent, celé číslo udávající mocninu deseti, kterou se vynásobí hodnota m.

Příklady

zápis		hodnota
1E1	10^1	10
1E2	10^2	100
-1E1	-10^1	-10
-1E4	-10^4	-10000
1E-1	10^{-1}	0.1
1E-02	10^{-2}	0.01
-1E-04	-10^{-4}	- 0.0001
4E1	4.10^1	40
1550E-5	1550.10^{-5}	0.0155

V exponenciálním tvaru sděluje počítač často také výsledky výpočtu:

Příklad

```
PRINT 123456.78 * 9000    CR
 1.11111E+09
```

Nutnost vyjadřovat vstupní či výstupní hodnoty v exponenciálním tvaru je dána omezenou možností zobrazení rozsahu hodnot v přímém tvaru. Tato skutečnost je v případě osmibitového počítače zvlášť výrazná (viz omezení možnosti zápisu čísla v běžném tvaru).

Příklady

(a)
```
PRINT 123456    CR
 123456
```

(b)
```
PRINT 1234567    CR
 1.23457E+06
```
(výsledek má více než 6 míst, proto jej systém zobrazí v exponenciálním tvaru)

(c)
```
PRINT 0.5    CR
 .5
```
(všimněte si, že počítač nezobrazuje nevýznamné nuly v konstantách; v programu je také nemusíme zapisovat)

(d) PRINT 0.0037; .6 CR
3.7E-03 .6

(e) PRINT 1020304050 CR
1.0203E+09

(f) PRINT 6666665E5 CR
6.66667E+11

Hodnota exponentu nesmí překročit 38.

Poznámky
(1) Je přípustné vynechat 0 před desetinnou tečkou. Lze tedy například zapsat:
0.120, ale i .120
(2) Před výstupem se mantisa výsledků zaokrouhluje na 6 číslic; výstup nevýznamných nul se potlačí.
(3) S hodnotami v intervalu:
-0.9403954E-38; 0.9403954E-38
pracuje IQ jako s hodnotou 0.
(4) Ve všech případech může být číselná konstanta nahrazena číselným výrazem; výjimkou jsou konstanty v příkazech INPUT a DATA, GOTO a GOSUB, ON GOTO a ON GOSUB a konstanty v příkazech RUN a LIST.

Příklad

```
10 WAIT (50)
20 WAIT (A+T1)
```

2.5.2 Abecedněčíselné konstanty

Abecedněčíselná konstanta (řetězec znaků, textová konstanta) je skupina jakýchkoliv znaků z množiny znaků počítače uzavřená v uvozovkách.

Příklad

```
"VYPOCETNI CENTRUM VYSOKE SKOLY EKONOMICKE"
"ABC"
"123"
"
"              " (řetězec mezer)
```

Abecedněčíselnou proměnnou nemůžeme používat v aritmetických operacích.

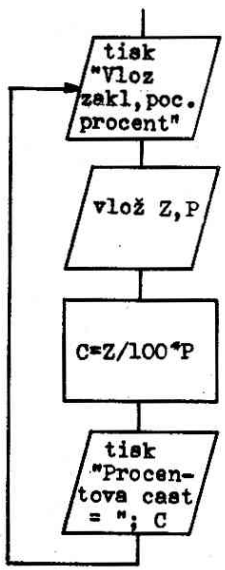
Příklad (tisk textu)

```
10 PRINT "SES Praha 2"
RUN
```

výstup:

```
SES PRAHA 2
```

Příklad (výpočet procentové části)




```
10 PRINT "VLOZ ZAKLAD,
POCET PROCENT"
20 INPUT Z,P
30 C = Z/100 * P
40 PRINT "PROCENTOVA
CAST =";C
50 GOTO 10
RUN
VLOZ ZAKLAD, POCET PROCENT
: 1000,40   CR
PROCENTOVA CAST = 400
VLOZ ZAKLAD, POCET PROCENT
.
.
.
.
```

Maximální délka abecedněčíselné proměnné je 48 znaků.

2.5.3 Abecedněčíselné a číselné proměnné

Na rozdíl od konstant jsou proměnné v programu zapsány nikoliv skutečnou hodnotou, nýbrž pouze symbolem a svoji hodnotu během realizace programu mohou měnit.

Pro tvorbu symbolů proměnných (jmen proměnných, identifikátorů proměnných) platí následující pravidla:

(1) Jména číselných proměnných tvoříme:

(a) jedním velkým písmenem latinské abecedy, například:

A, B, C, ... X, Y, Z

(b) jedním velkým písmenem latinské abecedy a jednou číslicí, například:

A1, A2, ... A9, A0, B1 ... Z9, Z0

(c) dvěma písmeny latinské abecedy, například:

AA, AB, AC ... ZX, ZY, ZZ

s výjimkou kombinací, které jsou totožné s dvoupísmenovými klíčovými slovy:

ON, IF, GO, TO, PI a OR

(2) Jména abecedněčíselných proměnných tvoříme podobně jako v případech (1), avšak připojujeme znak $.

Například

A$, AW$, Z9$

Pro označení proměnných lze sice použít i identifikátorů s větším počtem znaků, například:

ALFA, CENA

apod., počítač však třetí a další znaky již nerozlišuje. Uloží tedy v paměti proměnnou BETA1 do téhož místa jako BETA2 apod.

(3) Ve všech případech může být abecedněčíselná konstanta nahrazena abecedněčíselným výrazem. Výjimkou jsou pouze řetězce znaků v příkazech INPUT a DATA.

Příklad

```
10 PRINT LEN(A$)
20 PRINT LEN("Praha")
30 PRINT LEN(B$ + C$)
```

(4) Pozor na chybné zápisy identifikátorů:

1A identifikátor nezačíná písmenem,
A*1 obsahuje nedovolený znak,
ON1 obsahuje klíčové slovo ON.

2.5.4 Indexované proměnné

Číselné i abecedněčíselné proměnné jsou jednak jednoduché (skalární), jednak indexované, tj. jsou prvky jednorozměrných (vektory), dvojrozměrných (matice) nebo trojrozměrných polí.

Pole jsou uspořádané množiny prvků stejného typu, tj. číselných nebo abecedněčíselných.

Například prvky vektoru:

$$x = (x_1, x_2, \ldots, x_i, \ldots, x_n)$$

jsou označeny stejnými identifikátory (x), ale liší se svými indexy. V jazyce BASIC zapisujeme indexy do závorek.

Například prvky pětiprvkového pole A označíme:

A(0), A(1), A(2), A(3) a A(4)

Poznámka
Například identifikátor A2 představuje jednoduchou proměnnou zatímco A(2) představuje prvek pole A.

Dvojrozměrné pole můžeme považovat za matici (tabulku), v níž je každý prvek pole charakterizován číslem řádku a číslem sloupce. Například dvojrozměrné pole M() obsahuje prvky M_{ij}, tj. každý prvek M je určen svým řádkovým indexem - i - a indexem sloupcovým - j.

Pozor! V jazyce BASIC IQ začíná číslování indexů od nuly.[1]

Například pole M(3,2) je uloženo v paměti takto:

sloupce řádky	1.	2.	3.
1.	M(0,0)	M(0,1)	M(0,2)
2.	M(1,0)	M(1,1)	M(1,2)
3.	M(2,0)	M(2,1)	M(2,2)
4.	M(3,0)	M(3,1)	M(3,2)

[1] Například v BASICu IBM PC lze hodnotu indexu prvního prvku 1 nebo 0 volit.

Příklad

(a) (čtení trojrozměrného pole A_{ijk}, kde i=0,1; j=0,1; k=0,1,2):

```
10 INPUT A(0,0,0), A(0,0,1), A(0,0,2), A(0,1,0),
A(0,1,1),A(0,1,2), A(1,0,0), A(1,0,1), A(1,0,2),
A(1,1,0), A (1,1,1), A(1,1,2)
```

(b) (součet prvků pro i = 1)

```
20 S=A(1,0,0)+A(1,0,1)+A(1,0,2)+A(1,1,0)+A(1,1,1)+
+A(1,1,2)
```

Poznámky

(1) Indexem může být číselný výraz. Například:

```
PRINT A(I+1).
```

(2) Hodnota indexu prvku jednorozměrného pole musí být v intervalu:

0 - 65535.

2.6 Zobrazení dat v paměti

Víme již, že údaje, s nimiž počítač pracuje, musí být uloženy v paměti.

Základní jednotkou struktury dat v paměti je bit.

Bit je dvojková číslice, proto může nabývat pouze hodnoty 0, nebo 1.

Skupina osmi bitů tvoří byte ("bajt"). V 1 byte můžeme zobrazit 2^8 kombinací, tedy čísla 0 - 255 v binární formě. Pro vyjádření čísel větších rozsahů využíváme spojení několika bytů a zobrazení v pohyblivé řádové čárce (viz zobrazení čísel v exponenciálním tvaru). Rozsah zobrazovaných hodnot je v tomto případě dán hodnotou exponentu, v případě IQ 151 max. 38.

Abecedněčíselné znaky se převádějí do číselného kódu (viz tabulka kódu ASCII v příloze). Například položka PRAHA bude uložena:

80 82 65 72 65

Obsah bytů se obvykle vyjadřuje dvěma číslicemi šestnáct-

kové (hexadecimální) soustavy. Číslice šestnáctkové soustavy mají desítkové hodnoty 0 - 15, s označením 0 až 9 pro šestnáctkové číslice 0 - 9 a A pro 10, B pro 11, C pro 12, D pro 13, E pro 14 a F pro 15.

2.7 Popis proměnných - příkazy DIM, FREE a CLEAR

Potřebné místo v paměti pro proměnné, jichž používáme v programu, vyhradíme jejich popisem neboli deklarací. V jazyce BASIC platí implicitní popisy, tzn. že potřebné místo v paměti se vyhradí automaticky při prvním výskytu proměnné v programu.

Implicitní popis se však nevztahuje na indexované proměnné (pole), je-li počet indexu větší než 3 (více jak trojrozměrné pole) nebo je-li počet prvků pole větší než 10.

K popisu polí používáme příkaz DIM (Dimension):

DIM identifikátor (rozsah pole), ...

Rozsah pole je vždy celé číslo, které udává horní mez indexu pro dané pole. Počet specifikací rozsahu pole (1,2,3 atd.) udává počet rozměrů pole (vektor, matice, trojrozměrné pole apod.).

Příklad

Příkazem 10 DIM A(10), B(3,3), C$(0), D(2,2,2,2) deklarujeme:
jedenáctiprvkové pole A (A(0), A(1), ... A(10));
šestnáctiprvkové pole B (B(0,0), B(0,1) ... B(3,3));
jednoprvkové abecedněčíselné pole C$ (C$(0)) a čtyřrozměrné pole D.

Popis B(3,3) například deklaruje čtvercovou matici s horní mezí řádkového indexu 3 a horní mezí sloupcového indexu 3; matice má tedy 4 řádky a 4 sloupce.

Deklarované pole v paměti můžeme zrušit příkazem

FREE.

Příklad

```
10 DIM A(20), B(5,5), C(15)
.
.
70 FREE A      Deklarace pole A se zruší a rezervované
               místo v paměti se uvolní.
```

Veškeré deklarace v paměti ruší také příkaz CLEAR.

Příkaz CLEAR ruší (nuluje) hodnoty všech číselných proměnných a abecedněčíselným proměnným přiřazuje prázdné řetězce (mezery).

Příkazem CLEAR

CLEAR číselná konstanta

deklarujeme v paměti místo pro abecedněčíselné proměnné; rozsah udává číselná konstanta. Implicitní deklarace pro abecedněčíselné proměnné = 48 znaků (paměťových míst).

Příkaz DIM nelze použít v přímém výpočetním režimu, tj. bez čísla programového řádku.

2.8 Aritmetické operace

Aritmetické výrazy tvoříme z číselných proměnných a konstant, závorek a aritmetických operátorů:

+	sčítání
-	odčítání
*	násobení
/	dělení
^	umocňování

Příklady zápisů aritmetických výrazů

	Matematický zápis	Zápis v jazyce BASIC
(a)	$a + b$	A + B
(b)	$a(b + c)$	A * (B+C)
(c)	$\frac{5ax + k}{2bx + k_1}$	(5*A*X+K)/(2*B*X+K(1))
(d)	$a^2 + b^2$	a ^ 2 + b ^ 2

(e) $((x+1)^2 + r)^3$ ((X+1) ^ 2 + R) 3

(f) $x + \frac{a}{b}$ X + A/B

(g) $(a^b)^c$ A ^ B ^C

(h) $\ln^2 x$ LOG(X) ^ 2

(i) $\sqrt[3]{x}$ x ^ (1/3)

Chybně zapsané výrazy		Oprava
-X * -2	dva aritmetické operátory	X * (-2)
A(B+C)	chybí operátor násobení	A * (B+C)
(I + J/K	chybí pravá závorka	(I + J)/K

2.8.1 Pořadí vyhodnocování aritmetických výrazů

Vyhodnocením aritmetického výrazu získáme číselnou hodnotu. Aritmetické výrazy se vyhodnocují zleva doprava, v pořadí:

- umocňování
- násobení a dělení
- sčítání a odčítání

Příklad

Výraz -X + Y - W/T * S Q + 50.5 se vyhodnotí postupně:

a = S Q
b = W / T
c = b * a
d = -X
e = d + Y
f = e - c
g = f + 50.5

Standardní postup vyhodnocování výrazu lze ovlivnit použitím závorek; výraz v závorce se provede jako první, resp. výrazy v závorkách se realizují v pořadí od vnitřních závorek, je-li použito kombinace závorek.

Příklady

Matematický zápis	Zápis v jazyce BASIC
(a) $\frac{a}{bc}$	A/B/C nebo A/(B*C)
(b) $-alfa^{beta}$	-AL BT
(c) $\frac{-5.1 + a_i}{b_i}$	(-5.1 + A(I))/B(j)
(d) $a^2 - b^2$	A 2 - B 2
(e) $\frac{a + b}{c + d}$	(A + B)/(C + D)
(f) $a + \frac{b}{c + d}$	A + B / (C + D)

3 Standardní funkce

Standardní funkce jsou součástí systému; nahrazují programování některých často používaných matematických funkcí a operací.

Kromě standardních funkcí můžeme používat i funkcí, které si sami naprogramujeme; těmto funkcím říkáme podprogramy a věnujeme se jim v příslušné kapitole.

Obecný tvar standardních funkcí:

jméno funkce (argument funkce)

kde: jméno funkce je klíčové slovo jazyka BASIC;
argumentem funkce může být konstanta, proměnná i výraz. Argument uvádíme vždy v kulatých závorkách.

Je-li jméno funkce zakončeno znakem $, jedná se o abecedněčíselnou funkci a jejím argumentem může být pouze abecedněčíselná položka. Existuje však funkce bez argumentu (PI).

Standardní funkce lze používat i v přímém výpočetním režimu. Například:

```
PRINT SIN(90)
PRINT SQR(180)
```

Přehled standardních funkcí

Funkce	Význam	Příklad
ABS(X)	absolutní hodnota výrazu X	PRINT ABS(A-B*C)
ASC(a)	dekadický ekvivalent prvního znaku řetězce a v kódu ASCII	10 A$="PRAHA" 20 PRINT ASC(A$) výstup: 80 (neboť P v ASCII==80)

Funkce	Význam	Příklad
CHR$(X)	znak odpovídající v kódu ASCII číslu X	PRINT CHR$(100) výstup: d (viz tabulka kódu ASCII)
	Poznámka O významu této funkce se zmiňujeme v kapitole Grafické možnosti počítače IQ 151.	
EXP(X)	exponenciální funkce - e^x kde: e je základ přirozených logaritmů	80 INPUT W 90 R = EXP(W)
FNi(X)	definice funkce; i je identifikátor jednoduché číselné proměnné	
	Poznámka Významu a použití této funkce je věnována kapitola Podprogramy.	
HEX(X)	dekadický ekvivalent hexadecimálního čísla x	PRINT HEX(80) výstup: 128
	Operandem funkce HEX musí být hexadecimální číslo v intervalu (0 - FFFF). Použití funkce HEX se nevyhneme při práci se strojovým kódem počítače (viz příkazy POKE, CALL apod.).	
INT(X)	celočíselná dolní hodnota výrazu x	INT(3.6)=3 INT(-5.2)=-6 INT(-1.9)=-2
LOG(X)	přirozený logaritmus x (základ logaritmu je e = 2.71828...)	(a) 100 PRINT LOG(A*B/C) (b) PRINT LOG(10) výstup: 2.30259
	Poznámka Počítač IQ 151 nemá funkci dekadického ani dvojkového logaritmu LOG10, LOG2.	

Funkce	Význam	Příklad
RND(0)	generuje náhodná čísla v rozsahu 0 - 0.99999	10 PRINT INT(6*RND(0)+ +1) 20 WAIT(15) 30 GOTO 10 (generování náhodných čísel v rovnoměrném rozložení pravděpodobnosti)
SGN(X)	SGN(X)=1 pro x > 0, -1 pro x < 0, 0 pro x = 0	80 INPUT X,Y,Z 90 PRINT SGN(SQR(X/Y *Z))
SQR(X)	druhá odmocnina z x	PRINT SQR(258) výstup: 16.0624
PI	přibližná hodnota Ludolfova čísla	50 INPUT R 60 P = PI*R*R 70 P1= 2*PI*R
WAIT(s)	Mezi funkce můžeme zařadit i WAIT() definující čekací dobu v desetinnách sekundy (s^{-1}), např. při výstupu výsledků na obrazovku.	

3.1 Goniometrické funkce

Počítač IQ 151 nemá možnost volby stupňů, grádů či radiánu pro argumenty goniometrických funkcí a pracuje pouze s úhly vyjádřenými v obloukové míře, v radiánech. Podobně funkční hodnoty cyklometrických funkcí (ATN) jsou vždy uvedeny v obloukové míře.

Připomínáme proto, že platí vztah

$$\text{úhel v radiánech} = \frac{\text{úhel ve stupních} * PI}{180}$$

a pro převod ze stupňů na radiány můžeme danou hodnotu ve stupních násobit konstantou 0.0174533 (PI/2) nebo dělit konstantou 57.2958 (180/PI).

Funkce	Význam	Příklad
ATN(X)	arctg x v radiánech	PRINT ATN(90) výstup: 1.55969
COS(X)	cos x pro x v obloukové míře (v radiánech)	PRINT COS(90) výstup: -.448069
SIN(X)	sin x, pro x v radiánech	PRINT SIN(90) výstup: .893999
TAN(X)	tg x, pro x v radiánech	

Příklad (výpočet COS 60^{0})

```
10 H = COS(60 / 57.2958)
20 PRINT "cos 60 ="; H
RUN
cos 60 = .5
```

3.2 Alfanumerické funkce

Alfanumerické funkce využíváme pro práci s řetězci znaků; umožňují především spojování řetězců, zjištění počtů znaků v řetězci, vytvoření podmnožiny znaků v řetězci a konverze abecedněčíselných a číselných položek.

Funkce	Význam	Příklad
LEFT$(a,b)	podmnožina prvních b znaků řetězce a	10 A$="BRNO" 20 PRINT LEFT$(A$,2) výstup: BR
MID$(a,b)	podmnožina znaků řetězce a, počínaje b-tým znakem	10 A$="PARDUBICE" 20 PRINT MID$(A$,4) výstup: DUBICE
MID$(a,b,c)	podmnožina c znaků řetězce a, počínaje b-tým znakem	10 A$="PROSETIN" 20 PRINT MID$(AS,4,3) výstup: SET

Funkce	Význam	Příklad
RIGHT$(a,b)	podmnožina posledních b znaků řetězce a	10 A$="CHRUDIM" 20 PRINT RIGHT$(A$,4) výstup: UDIM
LEFT$(A$,X)	vymezí x znaků A$ zleva	10 U$="KOLIN" 15 PRINT LEFT$(U$,3) výstup: KOL
LEN(A$)	zobrazí počet znaků proměnné A$	10 D$="UHERSKY OSTROH" 20 PRINT LEN(D$) výstup: 14 10 Q$="VYSOKA" 20 W$="SKOLA" 30 E$="EKONOMICKA" 40 PRINT LEN(Q$+W$+ +E$) výstup: 21
VAL(a)	číslice, vyskytující se v řetězci a před 1. nečíselným znakem	10 A$="100 00 PRAHA 10" 20 PRINT VAL(A$) výstup: 10000 10 A$="123.45 + 67.1" 20 PRINT VAL(A$) výstup: 123.45
	Poznámka Desetinná tečka je zahrnuta mezi číselné znaky, znaménka (v tomto případě +) nikoli.	
STR$(X)	konverze číselného výrazu do abecedněčíselné proměnné	10 A=12345 20 A$=STR$(A) 30 PRINT A$ výstup: 12345
	Poznámka Funkce STR$ je inverzní k funkci VAL; s výhodou ji využijeme v případech, kdy potřebujeme	

Funkce	Význam	Příklad
	pracovat s číselným výrazem v abecedněčíselných operacích.	
INKEY$	přiřadí znak vygenerovaný právě stisknutou klávesou do proměnné funkce	10 A$ = INKEY$ 20 IF INKEY$=" "THEN 90

4 Příkazy jazyka BASIC

4.1 Přiřazovací příkaz LET

Přiřazovací příkaz vyhodnotí výraz a přiřadí jeho hodnotu proměnné

LET proměnná = výraz,

kde: proměnná je jméno číselné nebo abecedněčíselné proměnné; abecedněčíselným proměnným lze přiřadit pouze abecedněčíselné výrazy, číselným proměnným pouze číselné výrazy; výraz může být konstanta, proměnná, výraz

Poznámky
(1) Slovo LET není povinné.
(2) Pozor! Přiřazovací příkaz není rovnicí. Znaménko = v přiřazovacím příkazu proto nemá funkci operátoru "rovná se", ale oddělovače, resp. funkci přiřazení. Proto zápis

A = 3

čteme "A přiřaď 3" a nepozastavujeme se nad zápisy typu

S = S + A

ani v případech, kdy A je zřejmě různé od 0.
(3) V BASICu IQ nelze používat vícenásobné přiřazení; například zápis A=B=1 je syntakticky nesprávný.

Příklady

```
 5 INPUT C, D
10 LET A = C + D
20 LET A$ = "ALFA"
30 S = S + 1
40 Q = 1
50 B$ = HEX(0000)
60 T = 2.5 + X/2 + SIN(X - 0.5)
```

Příklad

```
10 A$ = "VYSOKA"
20 B$ = "SKOLA EKONOMICKA "
30 C$ = "PRAHA"
40 PRINT A$;B$;C$
RUN
```

VYSOKA SKOLA EKONOMICKA PRAHA

4.2 Příkazy vstupu

Proměnným lze přiřadit hodnoty:

(a) příkazem LET,
(b) příkazy vstupu: READ (s částí DATA, popř. RESTORE a INPUT.

4.2.1 Příkaz READ s částí DATA a RESTORE

Hodnoty proměnných lze uložit přímo do programu:

DATA h_1, h_2, ..., h_n

Takto uložené hodnoty přiřazujeme proměnným příkazem:

READ p_1, p_2, ..., p_n

Příklad

```
10 READ A,B,C,D$
20 ...
.
.
.
90 DATA 4, 5.5, 90, Plocha
```

Proměnným se přiřadí hodnoty v pořadí, jež odpovídá seznamům proměnných v části READ a seznamu hodnot v části DATA: A=4; B=5.5; C=90; D$="Plocha".

Použití části RESTORE způsobí opakované přiřazování hodnot uvedených v části DATA.

Příklad

```
10 REAL A,B,C,D,E.F,G,H

50 RESTORE
```

```
60 READ W,T,U,Z
.
.
110 DATA 1,0,9,2,3,8,7,5
```

Proměnným se přiřadí hodnoty: A=1; B=0, C=9; D=2, E=3; F=8; G=7; H=5 a dále W=1; T=0; U=9; Z=2.

Neobsahuje-li část DATA dostatečný počet hodnot, hlásí systém chybu číslo 03:

"Příkaz READ nenašel data".

Příklad

```
10 READ A1,A2,A3,A4
20 PRINT A1;A2;A3;A4
30 RESTORE
40 READ B1,B2,B3,B4,B5,B6,B7,B8
50 DATA 0,1,0,2,0,3,0,4
60 PRINT B1;B2;B3;B4;B5;B6;B7;B8
RUN
```

výstup

```
0  1  0  2
0  1  0  2  0  3  0  4
```

Část DATA může být v programu umístěna kdekoliv; je-li uvedena část RESTORE, musí předcházet příslušný příkaz READ.

4.2.2 Příkaz INPUT

Příkazem INPUT přiřazujeme proměnným hodnoty vkládáním z klávesnice, v průběhu výpočtu. Na rozdíl od příkazu READ nejsou tedy vstupní hodnoty uvedeny přímo v programu. Hodnoty vkládaných proměnných oddělujeme čárkou a vkládání ukončujeme klávesou CR.

Příklad

```
10 INPUT Q, P
20 V = Q / P
30 PRINT V
RUN

:                    dotaz systému na vstupní hodnoty
                     (výzva)
51900, 12  CR        vložení hodnot z klávesnice (odpověď
                     na výzvu)
 4325                výpočet a zobrazení výsledku
```

Příklad

```
10 INPUT A,B,C$,D$
.
.
RUN

: 1,1986,NOVAK JAN, PRAHA KBELY        zápis z klávesnice
```

Hodnoty vkládáme v pořadí seznamu proměnných v příkazu INPUT. Typ vkládané hodnoty proměnné musí odpovídat typu proměnné, jíž je hodnota přiřazována - číslo číselné proměnné, řetězec znaků abecedněčíselné proměnné. Prohřešek proti tomuto pravidlu oznámí systém zprávou

* INPUT ERROR

a zopakuje výzvu na vložení vstupních dat (:).

Příklad

```
10 INPUT A, A$
20 PRINT A$;A
RUN
: 200, NOVAK
NOVAK  200
```

ale:

```
RUN
: NOVAK, 200
  INPUT ERROR (chyba vstupních dat)
:              (počítač očekává opravu vstupních dat)
```

Vysvětlující řetězec v příkazu INPUT

V příkazu INPUT lze použít řetězec, jimž uživateli programu sdělíme, jaké údaje jsou příkazem požadovány:

```
INPUT "řetězec znaků"; proměnné
```

Příklad

```
.
.
.
50 INPUT "Vloz pocet radku,pocet
sloupcu matice"; M,N
.
.
.
RUN

výstup:
Vloz pocet radku,pocet sloupcu matice:
```

Poznámky

(1) Hodnoty abecedněčíselných proměnných nemusí být vkládány v uvozovkách; pak ale budou ignorovány případné mezery před prvním nebo za posledním znakem:

Příklad

```
10 INPUT A$,B$
20 PRINT A$
30 PRINT B$
RUN

: PRAHA, "  PRAHA"
PRAHA
  PRAHA
```

(2) Řetězec znaků obsahující čárku musí být vždy zapsán v uvozovkách, neboť čárka plní funkci oddělovače.

Příklad

```
60 INPUT "Jmeno, adresa"; J$,A$
.
.
RUN

Jmeno, adresa: NOVAK, "Praha 4, Leva 5"
```

(3) Vložíme-li na výzvu systému méně hodnot, než je v seznamu příkazu INPUT, opakuje počítač výzvu:

::

a čeká na dodatečné vložení hodnot. Hodnoty vložené nad požadovaný počet systém ignoruje.

(4) Vkládání dat lze přerušit klávesou CR; pokračovat můžeme po příkazu CONT.

(5) Stiskneme-li na výzvu systému (:) klávesu CR bez vložení hodnot, přiřadí se odpovídajícím číselným proměnným hodnota 0, abecedněčíselným proměnným hodnota prázdného řetězce (mezery).

(6) Je zřejmé, že příkaz INPUT nelze použít v přímém výpočetním režimu, tedy bez čísla řádku.

4.2.3 Příkaz výstupu PRINT

K zobrazení výsledků na obrazovce slouží příkaz PRINT:

PRINT p_1 oddělovač p_2 oddělovač ...

kde: p_i jsou prvky seznamu příkazu; prvkem seznamu může být: konstanta, jméno proměnné, výraz, jméno funkce

oddělovačem jsou:

, (čárka)

; (středník)

" (uvozovky)

TAB.

Poznámky

(1) Při použití středníku jako oddělovače jsou údaje odděleny jednou mezerou.
Při použití čárky se každý údaj tiskne na začátku zóny;

zóny mají délku 14 znaků (sloupců), pouze poslední, 6. zóna, má délku 9 znaků.
Zóny začínají v sloupcích 0,14,20,38,42 a 56; třetí, čtvrtá a pátá zóna přechází na nový řádek obrazovky.

(2) Za každým číslem se zobrazí mezera. Mezera se zobrazí i za číslem, jehož znaky končí v 31. sloupci; v tomto případě bude mezera zobrazena v 1. sloupci (v 0. pozici) následujícího řádku.

(3) Místo znaménka + se zobrazí mezera.

Příklady

(a)
```
10 PRINT A;B;C;D;E
20 PRINT A,B,C,D,E
RUN
```

výstup:

```
 0  0  0  0  0
```

tj. vzdy 1 mezera mezi prvky a 1 místo na znaménko (+ se nezobrazuje)

```
 0            0            0
        0            0
```

(Hodnota 0 se tiskne proto, že proměnným A,B,C,D,E jsme nepřiřadili žádnou hodnotu a hodnotu 0 mají implicitně.)

(b)
```
10 A1=10: A2=-10: A3=1E2: A$="PRAHA"
20 PRINT A1;A2;A3;A$
RUN
```

výstup:

```
 10 -10  100 PRAHA
```

(4) Na jeden řádek obrazovky IQ lze zobrazit 32 znaků.
Tiskový řádek však obsahuje celkem 80 znaků v 6 zónách.
Tiskové pozice číslujeme 0 - 31. Využitelná kapacita tiskového řádku je pouze 79 znaků (80. znakem je vždy znak CR, který se ovšem nezobrazuje).
Nevejde-li se číslo do 6. zóny, zobrazí se zbytek tvaru čísla na další řádek (číslo se "roztrhne"); podobně například 3. zóna má 4 znaky na jednom řádku a zbývajících 10 znaků na řádku následujícím, 5. zóna je rozdělena na 8 a 6 znaků.

(5) Použití oddělovače TAB slouží pro tabulkovou úpravu výstupu, viz odstavec (11).

(6) Uvozovky používáme pro výstup textu (abecedněčíselných konstant, řetězců znaků).

Příklad

(a)
```
10 PRINT "Vystupni sestava 01"
RUN
```
výstup:
```
Vystupni sestava 01
```

(b)
```
10 INPUT C
20 INPUT M
30 CC = C*M
40 PRINT "Cena celkem =", CC; "Kcs"
RUN
```
výstup:
```
Cena celkem =  600 Kcs
```

(7) Prvkem příkazu PRINT může být i aritmetický výraz.

Příklad
```
10 INPUT A,B,C
20 PRINT (A+B) / (C+1)
```

(8) Dalším prvkem příkazu PRINT může být definice počtu mezer pomocí funkce SPC (SPACES):

SPC (pocet mezer)

Počet mezer může být uveden jak konstantou, tak proměnnou.

Příklady

(a)
```
10 A=1
20 PRINT SPC(10);A
RUN
          1
```

(b)
```
10 INPUT J$,A$,P
20 PRINT J$; SPC(2); A$; SPC(5); P
RUN
: ADAM,PRAHA,1000   CR
```
výstup:
```
ADAM  PRAHA     1000
```

(c)
```
10 A=10
20 PRINT "A";SPC(A);"B"
RUN
```
výstup:
```
A              B
```

(9) Příkaz PRINT používáme pro výstup výsledků v přímém výpočetním režimu:

Příklady

(a) PRINT 20*2.25 CR
45

(b) PRINT HEX(10)
16

(c) PRINT SIN(90)+1
1.894 (hodnota argumentu v radiánech)

(10) Výstup čísel v exponenciálním tvaru:

Příklad

```
10 FOR I = 1 TO 10
20 PRINT 10 ^ I
30 NEXT I
RUN
10
100
1000
10000
100000
1E+06
1E+07
1E+08
1E+09
1E+10
```

(11) Každé nové použití příkazu PRINT způsobí přechod na nový řádek obrazovky:

Příklad

```
10 PRINT "VYPOCETNI"
20 PRINT "CENTRUM"
RUN
```

výstup:

```
VYPOCETNI
CENTRUM
```

Výstup nového řádku lze však potlačit zápisem oddělovače na konci seznamu prvků příkazu PRINT:

Příklad

```
10 PRINT "A"
20 PRINT "B";
30 PRINT "C"
RUN
```

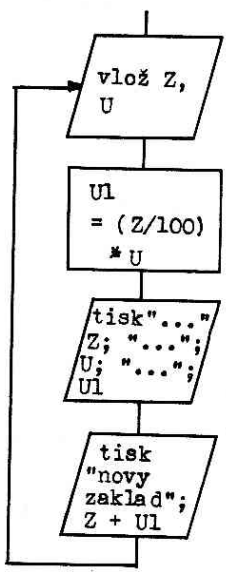
výstup:

```
A
BC
```

(12) Společný výstup textu a proměnných:

Příklady

(a) výpočet úroku a přičtení úroku k základu
(Z=základ; U=úroková míra)




```
10 INPUT Z,U
20 U1=(Z/100)*U
30 PRINT "Zaklad"; Z;
"urokova mira"; U; "urok" ;U1
40 PRINT "novy zaklad"; Z+U1
50 GOTO 10
RUN
 : 1000,4
```

výstup:

```
Zaklad 1000 urokova mira 4
urok 40
novy zaklad 1040
```

(b)

```
PRINT V "TISK" V "TISK" V
PRINT V;"TISK";V;"TISK";V
```

výstup:

```
 0 TISK 0 TISK 0
 0 TISK 0 TISK 0
```

Je-li oddělení proměnných a textu evidentní, nemusíme používat oddělovačů.

(13) Použití tabelační funkce (tabulátoru) TAB usnadní nastavení kursoru na požadované místo na obrazovce; jedná se o obdobu tabelační funkce psacího stroje nebo textových procesorů. Hodnota INT(TAB) musí být v intervalu 0 - 255.

Příklady

(a)
```
10 PRINT TAB(3);"A"
RUN

výstup:
   A
```

(b)
```
10 FOR I=1 TO 5
20 PRINT TAB(I*2);I*I
30 NEXT I
RUN

výstup:
 1
   4
     9
      16
        25
```

Funkcí TAB však nelze kursor na řádku posunout vlevo, tzn. vracet. V takovém případě systém funkci TAB ignoruje.

Příklad

```
10 PRINT TAB(10);"A"; TAB(5);"B"
RUN

výstup:
      AB
```

(14) Příkazem

PRINT řádek, sloupec

lze zobrazit prvky příkazu PRINT na libovolné pole obrazovky, specifikované parametry řádek v rozsahu (0-30) a sloupec v rozsahu (0-31).

Příkaz PRINT 0,0 umístí kurzor do levého horního rohu obrazovky, příkaz PRINT 0,31 do pravého horního rohu. Takto lze též zobrazit znaky na již obsazená pole (přepisovat znaky)

4.3 Příkazy skoku a větvení programu

Řádky programu se vykonávají postupně, ve vzestupném pořadí podle čísel řádků. Chceme-li tuto sekvenci porušit, můžeme použít příkazu skoku GOTO, podmíněného příkazu IF nebo přepínače ON.

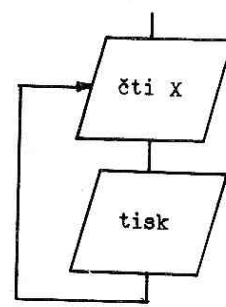
4.3.1 Příkaz skoku GOTO

Příkazem GOTO (nepodmíněným příkazem skoku) předáváme řízení programu na programový řádek určený návěstím:

```
GOTO        číslo řádku
GO TO
```

Příklad (tabelace x, x^2, x^3, 1/x)

```
10 INPUT X
20 PRINT X; X*X; X*X*X; 1/X
30 GO TO 10
RUN
```



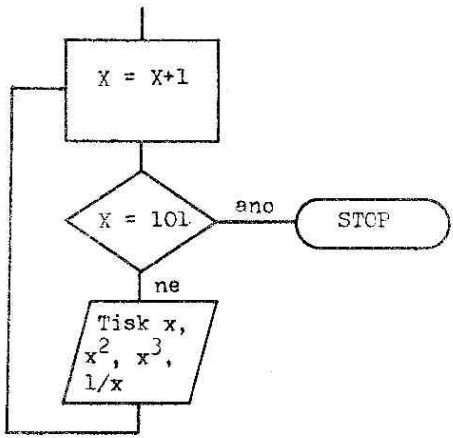
výstup:

```
: 2
 2  4  8    .5
: 5
 5 25   125   .2
.
.
atd.
```

Program se plní v pořadí řádků 10, 20, 30, 10, 20 ...

V tomto případě jde o tzv. neřízený nekonečný cyklus příkazů; plnění příkazů 10 až 30 se provádí donekonečna. Avšak vzhledem k tomu, že tisk výsledku je programován v každém kroku, lze ukončení výpočtu provést ze strany uživatele kdykoliv, v kterémkoliv kroku postupu, pouhým ukončením vkládání vstupních hodnot. To platí ovšem pouze v případě interaktivní práce s počítačem.

Příklad (tabelace x, x^2, x^3, 1/x, pro x od 1 do 100)




```
10 X=X+1
20 IF X= 101 THEN GOTO 90
(o příkazu IF viz další
odstavec)
30 PRINT X;X^2;X^3;1/X
40 GOTO 10
90 STOP
```

Příklad ilustruje řízený průběh cyklu.

Ukončení příkazu cyklu je podmíněno splněním podmínky v příkazu (20). Podrobněji viz následující odstavec.

4.3.2 Podmíněné příkazy

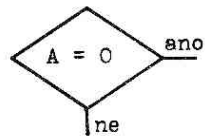
(a) Podmíněný výraz (podmínka)

V podmínce porovnáváme operandy, tj. proměnné, konstanty nebo aritmetické výrazy, pomocí relačních operátorů.

Funkce relačních operátorů

relace	zápis v BASICu
a = b	A = B
a > b	A > B
a < b	A < B
a ≧ b	A > = B
a ≦ b	A < = B
a ≠ b	A < > B

Při vyhodnocení je podmíněný výraz buď pravdivý, podmínka platí (ano), nebo je nepravdivý, podmínka neplatí (ne). Například:



Číselné porovnání (numerické) se provádí aritmeticky; určuje se a porovnává aritmetická hodnota číselných operandů:

A > B (je-li například A = 3, B = 2, je výraz pravdivý)

Abecedněčíselné porovnání abecedněčíselných operandů se provádí porovnáním kódových hodnot znaků (hexadecimálně), znak po znaku, zleva doprava.

A$ > B$ (je-li například A$ = "BRNO" a B$ = "PRAHA", je výraz nepravdivý, neboť B v kódu ASCII je 66, zatímco P je 80)

Poznámky

(1) Při porovnání se neuvažují případné mezery za posledním znakem alfanumerické položky.
Například položky:

"VSE" = "VSE "

(2) Při porovnání abecedněčíselných operandů nestejné délky se kratší operand doplní mezerami zprava. Kód mezery = = 32. Například při porovnávání položek:

"ABC" a "ABCD"

se porovnávají postupně znaky: A s A, B s B, C s C a D s mezerou; vzhledem k nízké hodnotě kódu mezery (D=68) platí: ABCD je větší než ABC.

(b) Složený podmíněný výraz; logické operátory

Pro vyjádření složitějších podmínek používáme logické operace konstruované pomocí logických operátorů:

AND logický součin, konjunkce ("a"),
OR logický součet, disjunkce ("nebo"),
NOT negace.

Logické operátory AND a OR se vztahují ke dvěma operandům; operand NOT je unární, vztahuje se tedy pouze k jednomu operandu (A AND B, A OR B, NOT A, NOT B apod.).

Logická hodnota vznikne vyhodnocením složené logické operace:

Operace	Hodnota výrazu			
A	ano	ano	ne	ne
B	ano	ne	ano	ne
negace A	ne	ne	ano	ano
konjunkce A AND B	ano	ne	ne	ne
disjunkce A OR B	ano	ano	ano	ne

Příklad (vyhodnocení logických operací)

Logická operace	Hodnoty	Vyhodnocení operace
NOT X = Y	X = 6, Y = 5	ano
A = B AND C = 6	A = 1, B = 1, C = 6	ano
	A = 1, B = 2, C = 6	ne
	A = 1, B = 1, C = 6.1	ne
A = B OR C = B	A = 1, B = 1, C = 1	ano
	A = 1, B = 1, C = 2	ano
	A = 1, B = 2, C = 2	ano
	A = 1, B = 2, C = 1	ne
X(1)=X(2) AND X(3)=0	X(1)=1, X(2)=0, X(3)=0	ne
A$=B$ OR C$="PRAHA"	A$="A", B$="AS",C$="Praha"	ne

Poznámka
"Výlučné nebo" - XOR - lze vyjádřit jako:
AND NOT.

Příklad

M = N AND NOT N = 0	složená podmínka platí, platí-
(M = N XOR N = 0)	-li jedna, a jen jedna z pod-
	mínek M=N nebo N=0

(c) Podmíněný příkaz IF

V jazyce BASIC IQ lze použít pouze neúplný podmíněný příkaz. Lze však použít složený podmíněný příkaz.

Příkaz IF má tvary:

(1) IF podmínka THEN číslo řádku

(2) IF podmínka THEN příkaz: příkaz: ... : příkaz

Příkazem IF lze programovat podmíněné skoky v běžné sekvenci plnění programových řádků:

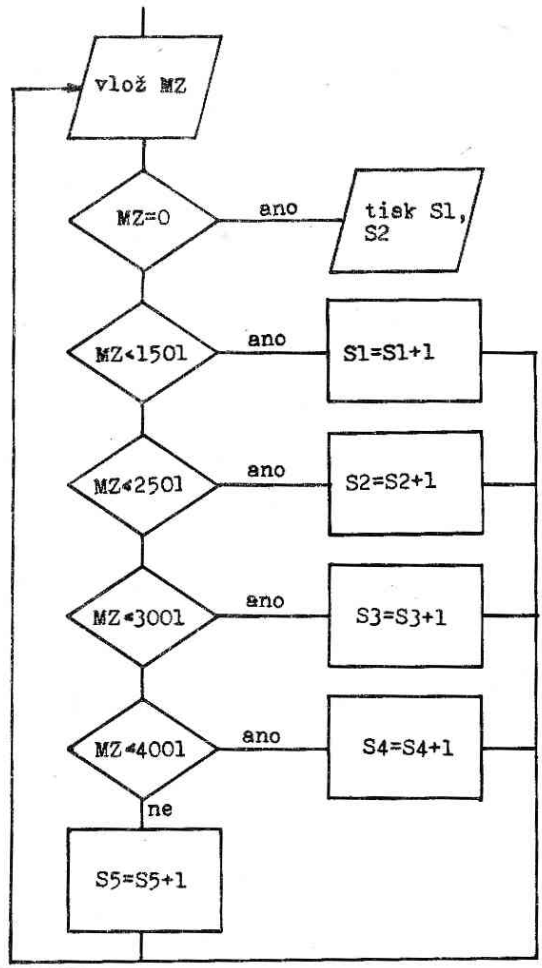
Příklad (načítání hodnot)

```
10 INPUT A
20 IF A = -1 THEN 90
30 S = S + A
40 GOTO 10
90 PRINT "Soucet ="; S
```

Je-li podmínka v příkazu IF pravdivá, provede se příkaz v části THEN, tzn. skok na určené číslo řádku. Je-li podmínka nepravdivá, pokračuje program plněním příkazů v sekvenci čísel programových řádků. Hodnota -1 v příkazu (20) plní funkci tzv. koncového znaku vstupního souboru; chceme-li zpracování ukončit, vložíme z klávesnice -1.

Příklad (četnost mezd v daných intervalech; koncový znak vstupního souboru MZ=0)

```
     - do 1500
od 1501 do 2500
od 2501 do 3000
od 3001 do 4000
nad 4000 -
```




```
10 INPUT MZ
15 IF MZ = 0 THEN 90
20 IF MZ  1501 THEN S1 = S1 + 1: GOTO 10
30 IF MZ  2501 THEN S2 = S2 + 1: GOTO 10
40 IF MZ  3001 THEN S3 = S3 + 1: GOTO 10
50 IF MZ  4001 THEN S4 = S4 + 1: GOTO 10
60 S5 = S5 + 1: GOTO 10
90 PRINT S1; S2; S3; S4; S5
```

4.3.3 Příkaz ON GOTO (přepínač, skok určený výpočtem)

Podmíněný příkaz skoku ON GOTO má tvar:

ON výraz GOTO číslo řádku$_1$, číslo řádku$_2$, ...

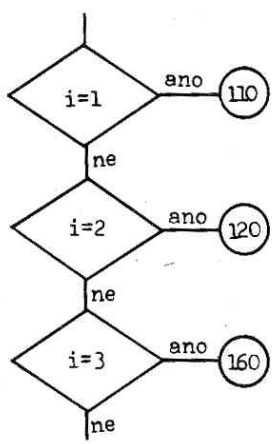
Větvení programu příkazem ON GOTO závisí na hodnotě výrazu; je-li hodnota výrazu i, předá se řízení programu na i-tý programový řádek v pořadí.

Příklad

```
.
.
.
110 ON J GOTO 300, 320, 180, 20, 40
.
.
.
```

Má-li J hodnotu například 4, větví se program na řádek 20, má-li J hodnotu 1, provede se skok na řádek 300 apod.

Čísla řádků uvedená v seznamu musí označovat řádky v programu skutečně existující; případný pokus předat řízení na neexistující programový řádek vede k chybě.



Příklad

```
.
.
.
90 INPUT I
100 ON I GOTO 110, 120, 160
```

nahrazuje

```
IF I=1 THEN 110
IF I=2 THEN 120
IF I=3 THEN 160
```

Příklad (zjištění četnosti známek (ZN) jedniček, dvojek a trojek)

```
10 INPUT ZN
20 ON ZN GOTO,30,40,50,60      koncový znak vstupního sou-
30 S1=S1+1: GOTO 10            boru ZN=4
40 S2=S2+1: GOTO 10
50 S3=S3+1: GOTO 10
60 PRINT S1;S2;S3
```

Řešení je ekvivalentní například s postupem:

```
5 DIM S(3)
10 INPUT ZN: IF ZN=4 THEN 30
20 S(ZN) = S(ZN) + 1: GOTO 10
30 PRINT S(1); S(2); S(3)
```

4.3.4 Příkaz cyklu FOR/NEXT

S cyklickým opakováním příkazu jsme se již setkali v odstavci 5.3.1, v souvislosti s příkazem GOTO. Příkaz cyklu FOR/NEXT zajišťuje řízení průběhu cyklu automaticky.

Příkaz cyklu má tvar

```
FOR i = a TO b  STEP  k
.
.       příkazy cyklu
.
NEXT i,
```

kde:

- i (řídicí proměnná cyklu) je jednoduchá číselná proměnná, řídící průběh cyklu (počet opakování); plní funkci počitadla počtu opakování cyklu a její hodnota se vždy po jednom opakování příkazů cyklu zvyšuje o hodnotu kroku cyklu (případně snižuje, je-li krok záporný)

Počáteční hodnota řídicí proměnné může být kladná, nulová i záporná.

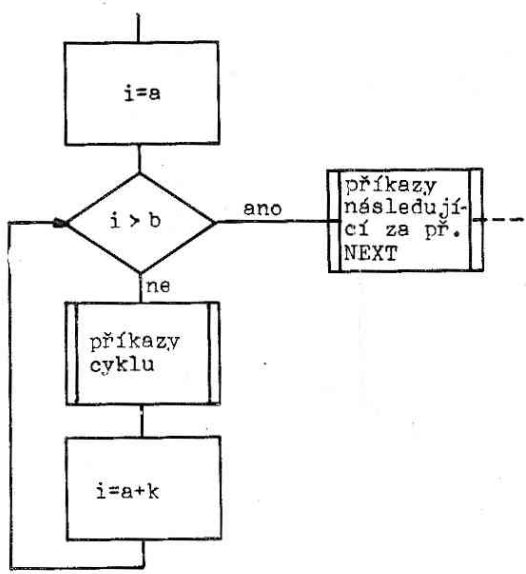
- a (dolní mez cyklu) je číselný výraz udávající počáteční hodnotu proměnné i

- b (horní mez cyklu) je číselný výraz udávající konečnou hodnotu proměnné i
- k (krok) je číselný výraz udávající krok zvyšování proměnné cyklu; není-li část STEP uvedena, platí implicitně krok = 1

Cyklické provádění příkazů se provádí pokud řídicí proměnná nepřekročí horní mez.

Instrukce NEXT ukončuje příkazy cyklu; instrukce následující po části NEXT již není součástí cyklu.

Průběh cyklu lze vyjádřit vývojovým diagramem:

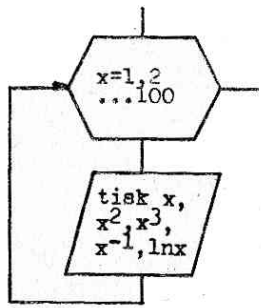


Příklady

(1) Ve všech následujících případech se provede příkaz cyklu 10krát:

```
50 FOR I=1 TO 10 ...                    ... NEXT I
50 FOR I=0 TO 9 ...                     ... NEXT I
50 FOR II=10 TO 1 STEP -1 ...           ... NEXT II
50 FOR I=20 TO 200 STEP 20 ...          ... NEXT I
50 FOR J=1 TO A+1*10 ...                ... NEXT J
50 FOR K=0 TO 1 STEP 0.1 ...            ... NEXT K
```

(2) Tabelace hodnot x^2, x^3, 1/x, ln x pro x od 1 do 100:




```
10 FOR X = 1 TO 100
20 PRINT X; X*X; X ^3; 1/X; LOG(X)
30 NEXT X
```

(3) (a) `FOR I = 1 TO 15 STEP 2` — i nabývá hodnot: 1,3,5,7,9, 11,13,15

(b) `FOR J = 10 TO 1 STEP -1` — j nabývá hodnot: 10,9, 8,...1

(c)
```
10 INPUT N
.
.
FOR K = 1 TO N
```
horní mez cyklu čteme příkazem INPUT

Poznámky

(a) Do cyklu nelze přecházet z jiného místa programu než prostřednictvím příkazu FOR:

```
10 FOR I = 1 TO 10
20
30
40
50 NEXT I
60 GOTO 30          nepřípustný příkaz
.
.
.
```

Naopak je možný přechod z cyklu před jeho ukončením:

```
100 FOR J = 1 TO 15 STEP 2
110 INPUT A(J)
135 IF A(J) = 0 THEN 900
 .
 .
170 NEXT J
 .
 .
900
 .
```

(b) Četné použití "předčasného" ukončení cyklu však může způsobit komplikace; pokud totiž neukončíme cyklus dosažením horní meze cyklu, nevynulují se registry řídící průběh cyklu.

(c) Jeden cyklus může být vložen do jiného cyklu. Mluvíme pak o vnitřním cyklu (vloženém) a cyklu vnějším.

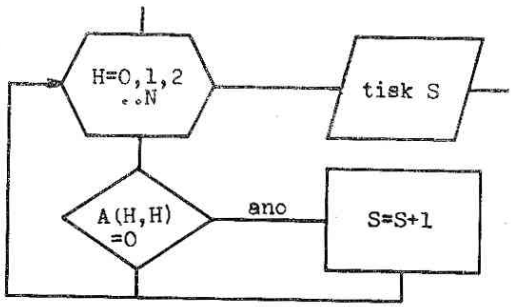
(4) Čtení a uložení prvků matice a_{ij} po řádcích:

```
10 FOR I=1 TO 3
15 FOR J=1 TO 4
17 INPUT A(I,J)
18 NEXT J: NEXT I
```

Poznámka
Příkazy NEXT J: NEXT I lze zapsat též NEXT J,I.
Průběh cyklů (nejdříve se vždy vyčerpají příkazy vnitřního cyklu - proměnná cyklu j):

i=0	čti	i=1	čti	i=2	čti
j=0	A(0,0)	j=0	A(1,0)	j=0	A(2,0)
j=1	A(0,1)	j=1	A(1,1)	j=1	A(2,1)
j=2	A(0,2)	j=3	A(1,2)	j=2	A(2,2)
j=3	A(0,3)	j=3	A(1,3)	j=3	A(2,3)

(5) Zjištění četnosti nulových prvků na hlavní diagonále matice A:




```
35 FOR H=0 TO N
40 IF A(H,H)=0 THEN S=S+1
46 NEXT H
50 PRINT S
.
.
```

Poznámka
Předpokládáme, že prvky matice včetně parametru N jsou již uloženy v paměti.

5 Práce s magnetopáskovou pamětí

Počítač IQ 151 používá jako vnější paměť magnetickou pásku ve formě magnetofonové kazety. Pro pořízení záznamů na kazetu lze použít jakýkoliv standardní magnetofon.

5.1 Čtení programu z kazety

Povel pro čtení programu z magnetické pásky je

MLOAD.

Postup čtení programu z kazety

(1) Zapíšeme povel MLOAD.
(2) Vložíme kazetu s programem.
(3) Stiskneme klávesu pro přehrávání.
(4) Přehrajeme obsah kazety až po návěští (pilotní tón), trvající 10 sekund; páska je nastavena na začátku prázdného místa před programem.
(5) Stiskneme klávesu CR.
(6) Po ukončení pilotního tónu se postupně přehrávají jednotlivé řádky programu z pásky do vnitřní paměti procesoru; tento proces lze sledovat na obrazovce a případně kontrolovat, zda se skutečně jedná o nahrávání požadovaného programu.
(7) Ukončení čtení programu signalizuje systém zprávou READY.

5.2 Zápis (nahrávání) programu na kazetu

Musíme si být vždy vědomi toho, že po vypnutí procesoru počítače je zcela nenávratně ztracen celý obsah vnitřní paměti.

Programy, na nichž nám záleží pro další použití, se musí proto archivovat na magnetické pásce. Tuto operaci je ostatně

vhodné provést vždy, máme-li v paměti uložen rozsáhlejší program.

Na magnetické pásce nelze zřizovat a vést rejstřík uložených souborů (tzv. katalog), tak jak je tomu například při práci s disketami. Proto si musíme sami vést pečlivou evidenci uložených programů.

U kazet, na nichž archivujeme odladěné programy (netýká se pracovních pásek, s nimiž běžně pracujeme), odstraníme pojistky, čímž zabráníme neúmyslnému přepsání programu; jde o obdobnou funkci, jakou je ochrana analogového záznamu na pásce. Ochranu přepisu lze snadno zrušit přelepením bezpečnostního otvoru, například isolepou.

Poznámka
Používáme-li kazetový magnetofon TESLA K-10, odpojíme nahrávací kabel; je-li totiž kabel ve zdířkách magnetofonu, je automaticky vypnut mikrofon.

Postup nahrávání programu na kazetu

(1) Namluvíme zvukové záhlaví (proto dbáme, aby byl ve funkci vestavěný mikrofon). Záhlaví by mělo obsahovat

název programu,

případně též pokyny pro práci s programem; takto lze využít zvukového záhlaví i pro stručnou uživatelskou dokumentaci programu. Rozsah záhlaví je věcí programátora, avšak příliš stručné záhlaví, tzn. krátký záznam, se na pásce obtížně hledá.

(2) Bezprostředně po přehrání záhlaví zastavíme posuv pásky.

(3) V případě práce s magnetofonem TESLA K-10 znovu připojíme nahrávací kabel.

(4) Z klávesnice zapíšeme povel:

MSAVE.

zatím ještě bez současného stisknutí klávesy CR.

(5) Stiskneme klávesy pro posuv pásky vpřed + červenou klávesu nahrávání; ozve se pilotní tón (pronikavé pískání), který necháme znít zhruba 10 sekund.

(6) Stiskneme klávesu CR na klávesnici počítače; tím realizujeme povel MSAVE.

(7) Charakteristický zvuk nahrávání střídavě s pilotním tónem svědčí o tom, že se ukládají jednotlivé programové řádky na magnetickou pásku.

Z hygienických důvodů snížíme hlasitost magnetofonu na minimum!

(8) Ukončení nahrávání programu oznámí systém zprávou:

READY.

(9) Vypneme magnetofon.

5.3 Změna polarity nahrávky

Při práci s počítačem IQ 151 v podmínkách našich škol se nám může stát, že budeme nuceni číst s použitím například magnetofonu M 710 záznam, který byl pořízen na magnetofonu K-10, nebo naopak. V těchto případech je třeba nejdříve změnit polaritu povelem monitoru:

PRINT PEEK(27) CR

Zjistíme, že na adrese 27 je obvykle uložena hodnota 86. V tomto případě změníme hodnoty na 214 dalším monitorským povelem:

POKE 27,214

a tak umožníme přečtení programu z kazety do vnitřní paměti počítače.

6 Grafické možnosti počítače IQ 151

6.1 Zobrazení čárek a háčků nad písmeny

Počítač IQ 151 nemá prostředky pro práci s plnou českou abecedou, tj. nemá možnost generovat na obrazovce písmena s háčky, s čárkami, písmena jako je ů apod. Tuto nevýhodu má dosud, žel, naprostá většina počítačů. Často se setkáváme s názorem, že jsme si díky počítačům již zvykli přijímat česká slova bez těchto znaků, a že je to jednak neúcta k mateřštině, jednak smíření se s předpokladem řady častých nedorozumění vyplývajících ze zkomolenin, zvláště v případě vlastních jmen, názvů obcí apod. Týká se to však i doprovodného textu programů; například ze zprávy: ZADEJ HODNOTU nepoznáme, máme-li údaj zadat, tj. vložit, nebo žádat, požadovat od systému.

Tyto nevýhody lze v případě počítače IQ 151 alespoň částečně zmírnit postupem, který ilustrujeme na následujícím příkladu.

Příklad: Zobrazte text: "VYSOKÁ ŠKOLA"

Řešení

(1) Zobrazíme znak uvozovek (").

(2) Celý zbytek řádku přejedeme kurzorem až na druhý sloupec následujícího řádku (kurzor je tedy o jedno místo vpravo pod znakem (")):

"

—

(3) Vložíme text: VYSOKA SKOLA".

(4) Kurzorem se vrátíme až nad znak A, vložíme apostrof jako symbol čárky (klávesa čísla 7 + klávesa SH) a nad S zapíšeme malé v, symbol háčku.

' v
VYSOKA SKOLA

(5) Kurzorem se vrátíme na začátek řádku; pomocí klávesy IC vložíme číslo řádku, např. 10,a text PRINT:

10 PRINT "

(6) Kurzorem přejedeme celý řádek až za poslední uvozovky a stiskneme CR.

(7) Příkazem RUN se přesvědčíme o tom, že IQ zobrazí požadovaný text VYSOKÁ ŠKOLA.

6.2 Přechod do grafického režimu

Při práci s počítačem IQ můžeme využít některých grafických znaků, jejichž symboly nalezneme na klávesách. Nejprve však musíme zvolit grafický režim:

(a) klávesami:

CTRL + písmene O nebo

(b) použitím funkce CHR$ přímo v textu programu:

PRINT CHR$(15) "zzzzz",

kde: konstanta 15 je parametr volby funkce CHR$
zzzz jsou symboly základních znaků na příslušné klávese

Například příkazem:

10 PRINT CHR$(15) "IIII"

se zobrazí čtyři znaky ++++ ("kříže"), neboť grafický znak, který má společnou klávesu s písmenem "I", je znak "+". Podobně znak "srdíčko" zobrazíme příkazem: PRINT CHR$(15) "G".

Inverzní zobrazení zvolíme:

(a) Klávesami:

CTRL + S nebo

(b) Použitím funkce CHR$ přímo v programu:

PRINT CHR$(19) " t e x t ",

kde: konstanta 19 je parametrem volby inverzního zobrazení a t e x t je libovolný řetězec znaků vložený z klávesnice

Poznámky

(1) Během práce v grafickém režimu zvoleném klávesami CTRL a O se může stát, že nelze pohybovat kursorem, a to ani klávesou pro pohyb kursoru, ani mezerníkem. V těchto případech si pomůžeme použitím tzv. prázdného znaku, který má společnou klávesu se znakem M.

(2) Návrat z grafického do běžného režimu:

CTRL + N

(3) Význam parametrů funkce CHR$ pro grafický režim:
15 přechod do grafického režimu,
14 zrušení grafického režimu,
19 volba inverzního zobrazení,
18 zrušení inverzního zobrazení.

Ostatní parametry funkce CHR$ a jejich význam uvádíme v kapitole Standardní funkce.

7 Zobrazení znaků v libovolném místě obrazovky

Počítač IQ pracuje se standardní televizní obrazovkou a s modulem VIDEO 32, který umožňuje zobrazení 32 řádků na obrazovce. V běžném režimu však využívá pouze každý druhý řádek, a pracuje tedy jen se 16 řádky. To je formát u osobních počítačů zcela neobvyklý a nepraktický (běžný je rozsah obrazovky 24 řádků a 80 sloupců). Tuto výraznou nevýhodu lze poněkud zmírnit nastavením kurzoru na libovolné pole televizní obrazovky pomocí příkazu:

PRINT čř, čs,

kde: čř je číslo řádku v intervalu 0-30,
čs je číslo sloupce v intervalu 0-31.

Příklad

```
PRINT & 2,10 "VSE PRAHA"
```

výstup:

```
     12345678901234567890101234567890
1
2              VSE PRAHA
3
4
5
6
7
.
.
```

Poznámky

(1) V této variantě příkazu PRINT vkládáme abecedněčíselné konstanty (řetězce znaků) do uvozovek, číselné konstanty oddělujeme středníkem.

Příklad

```
PRINT & 26,6;90        (na 26. řádku, počínaje sloupcem 6
                        se zobrazí konstanta 90)
```

(2) Příkaz PRINT & můžeme použít i v případě grafického či inverzního zobrazení.

Příklad

```
5 CLS                              (smazání obsahu obrazovky)
10 FOR I = 1 TO 30
20 PRINT CHR$(15)& 2,I "Q"
30 NEXT I
RUN
```

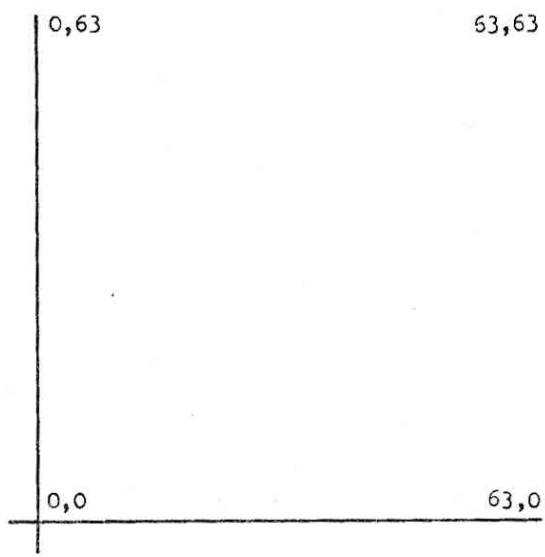
(Programem se zobrazí vodorovná čára v druhém řádku obrazovky.)

8 Semigrafické možnosti počítače IQ - příkaz PLOT

Počítač IQ lze v omezené míře využít i pro semigrafické funkce, pro bodové zobrazení grafů.

Pro bodové zobrazení grafů lze pracovat v souřadnicovém systému 64 × 64 bodů na obrazovce; x-souřadnice mají hodnoty od 0 do 63, y-souřadnice od 0 do 63.

Bod X(0,0) je bod v 1. sloupci posledního řádku obrazovky



Obr. 3 Orientace bodů na obrazovce v semigrafickém režimu

Příkaz:

PLOT X,Y

zobrazí grafický symbol ("čtvereček" sestávající ze 4 × 4 bodů) v místě se souřadnicemi x,y.

Příkaz:

UNPLOT X,Y

odstraní bod v místě se souřadnicemi x,y.

Příklad

```
 5 REM zobrazení sinusoidy
10 CLS
20 REM rozmery A,D,V jsou uvedeny v poctu ctverecku
30 INPUT "amplituda"; A
40 INPUT "delka"; D
50 INPUT "osa ve vysce"; V
60 FOR I=1 TO D+1
70 PLOT I, A*SIN(2*PI/D I)+V
80 NEXT I
```

9 Procedury a podprogramy

Často potřebujeme v programu použít několikrát a v různých místech postupu tentýž algoritmus, byť i s různými údaji. V těchto případech se může s výhodou využít i podprogramů.

Algortimus naprogramujeme v podprogramu s obecnými (formálními) parametry a vždy v případě potřeby použití algoritmu jej v daném místě programu "vyvoláme". Za obecné parametry dosadíme konkrétní údaje - vlastní parametry.

V jazyce BASIC plní funkci podprogram též funkce definované uživatelem.

9.1 Uživatelské funkce

Uživatelské funkce (procedury) používáme především pro definici složitějších aritmetických výrazů, které se v programu opakují a které nejsou v souboru standardních funkcí.

Funkci definujeme příkazem

$$\text{DEF FNi}(p_1, p_2, \ldots p_5) = \text{výraz},$$

kde: DEF FN je klíčové slovo uživatelské funkce
i je název funkce (jednoduché proměnné)
p_j jsou obecné parametry funkce (j=1,2,3,4,5)
výraz tvoří vlastní funkční vztah, tzv. tělo procedury

Uživatelskou funkci vyvoláme příkazem

$$\text{FND}\ (a_1, a_2 \ldots a_n),$$

kde: FN je klíčové slovo
D je název funkce, odpovídající názvu v definici (DEF FND)
a_i jsou vlastní parametry, dosazené za formální parametry použité v příkazu DEF FN

Příklad

definice funkce:

```
10 DEF FNX1(A,B,C) = (-B + SQR(B*B - 4*A*C))/(2*A)
20 DEF FNX2(A,B,C) = (-B - SQR(B B - 4*A*C))/(2*A)
.
.
.
```

vyvolání funkce:

```
100 INPUT P,Q,R
110 PRINT FNX1(P,Q,R)
120 PRINT FNX2(P,Q,R)
.
.
.
200 INPUT I,J,K
210 W = FNX1(I,J,K)
.
.
.
```

za formální parametry A,B,C dosazujeme vlastní parametry P,Q,R

vyvolání funkce:
za formální parametry A,B,C dosazujeme I,J,K

Při realizaci programu příkazem

RUN číslo řádku

musíme dbát na to, aby se uživatelská funkce vždy znovu nadefinovala; příkazem RUN se totiž její definice ruší.

9.2 Podprogram

9.2.1 Příkaz GOSUB

Podprogram začíná na určitém programovém řádku a jeho definice je ukončena příkazem

RETURN

(RETURN - "návrat").

Vyvoláme jej příkazem:

GOSUB

(GO SUB - go to subroutine /"jdi do podprogramu"/).

Příklad

```
10
.
.
.
260 I = 3: GOSUB 410
.
.
.
300 I = 5: GOSUB 410
.
.
.
410 PRINT I
420 FOR J = 1 TO N: PRINT A(I,J)
430 NEXT J
440 RETURN
450 END
```

Podprogram (410 - 440) vytiskne vždy číslo řádku matice (I) a celý i-tý řádek matice.

9.2.2 Příkaz ON GOSUB

Příkaz ON GOSUB slouží pro vyvolání podprogramu, jehož 1. řádek (vstup do podprogramu) se vybírá ze seznamu čísel řádků v závislosti na hodnotě výrazu v příkazu ON. Příkaz ON GOSUB je tedy obdobou příkazu přepínače ON.

Podmíněný příkaz ON GOSUB má tvar:

ON hodnota proměnné GOTO číslo řádku$_1$, číslo řádku$_2$,...

Větvení programu příkazem ON GOSUB závisí na hodnotě výrazu; je-li hodnota výrazu i, předá se řízení programu na programový řádek (vstupní řádek daného podprogramu), který je v seznamu jako i-tý v pořadí.

Příklad

```
.
.
.
100 ON X GOSUB 500, 800, 1200
110
.
.
.
800
.
.
1200 RETURN
```

Má-li X například hodnotu 2, provede se podprogram začínající na řádku 800; po provedení všech příkazů podprogramu se vrátí řízení programu na řádek 110.

Poznámky

(1) Řádky podprogramu mohou být v programu kdekoliv, doporučuje se však umístit podprogramy na konci programu; před sekci podprogramů vložíme příkaz END nebo STOP.

(2) Počet podprogramů není v programu omezen.

(3) Podprogram může být vyvolán pouze příkazem GOSUB nebo ON GOSUB; nelze tedy přejít do těla podprogramu příkazem GOTO a rovněž tak se nesmí přejít do podprogramu v běžné sekvenci plnění řádků programu.

(4) Východ z podprogramu lze uskutečnit pouze přes příkaz RETURN; nelze tedy ukončit plnění příkazů podprogramu příkazem GOTO nebo IF.

```
.
150 GOSUB 500
160 ...
.
.
500 ...
510 GOTO 160
.
.
560 RETURN
.
```

(5) Uvnitř podprogramu může být vyvolán další podprogram; mezi příkazy podprogramu může tedy být i příkaz GOSUB. Rekurzivní podprogramy, t.j. podprogramy, které vyvolávají samy sebe, nejsou dovoleny.

(6) Snaha o vyvolání neexistujícího podprogramu (číslo řádku v příkazu GOSUB v programu neexistuje) vede k chybě; k chybě dojde rovněž tehdy, je-li hodnota proměnné v příkazu ON GOSUB menší než 1, nebo je-li větší než počet čísel řádků v seznamu.

Závěrem poznámek k používání podprogramů však dodejme, že každý podprogram a každé vyvolání podprogramu snižují přehlednost programu a orientaci v jeho stavbě.

10 Generování tónů

V režimu práce s monitorem lze generovat zvukové signály. Program monitoru vyvoláme příkazem:

CALL HEX(F973)

Charakteristiky tónu definujeme příkazem POKE; výšku tónu na adrese 23, délku na adrese 24:

POKE 23, délka tónu
POKE 24, výška tónu

Generování tónu si ukážeme na příkladě:

```
10 FOR I = 9 TO 1 STEP -1
20 POKE 23, I+5
30 POKE 24,90
40 CALL HEX(F973)
50 WAIT (1)
60 NEXT I
```

Zkuste si poslechnout například takto naprogramovanou melodii:

```
10 READ N
20 FOR I=1 TO N
30 READ D, T
40 POKE 24, D*17
50 POKE 23, T
60 CALLHEX (F973)
70 NEXT I
80 DATA   59,4,103,8,77,4,82,8,77,4,61,8,69,4,77,8,69,
          2,61,2,69,8,77,4,77,8,61,4
90 DATA   52,12,46,4,46,8,52,4,61,8,61,4,77,8,69,4,77,
          8,69,2,61,2,69,8,77
100 DATA  4,92,8,92,4,130,12,77
110 DATA  4,46,8,52,4,61,8,61,4,77,8,69,4,77,8,69,4,46,
          8,52,4,61,8,61,4,52,12,46
```

```
120 DATA  4,46,8,52,4,61,8,61,4,77,8,69,4,77,8,69,2,61,
          2,69,8,77,4,92,8,92
130 DATA  4,130,8,77
```

11 Speciální příkazy

11.1 Příkazy STOP a END a povel CONT

Příkazem STOP či END lze zastavit průběh zpracovávání programu; zpracování lze opět obnovit povelem CONT (CONTINUE - pokračuj).

Příkazem END také označujeme konec programu ve výpisech programů pro lepší orientaci v textu programu.

Použití příkazu STOP je výhodné zvláště tam, kde ukončujeme program v jiném řádku než v posledním (například v závislosti na splnění určité podmínky).

Po zastavení programu příkazem STOP sdělí systém zprávu:

BREAK IN číslo řádku

Takto lze, v kombinaci s vloženými příkazy PRINT, sledovat mezivýsledky zpracování.

Povel CONT obnoví realizaci programu počínaje prvním příkazem po příkazu END či STOP. Povel CONT nesmíme použít v programovém řádku.

Příklady

(a)

```
.
20 INPUT A
30 IF A=0 THEN STOP
40
.
.
.
END
```

(b)

```
10 FOR I=1 TO 100
20 PRINT I;I*I;SQR(I)
30 STOP: NEXT I
RUN

výstup:
 1  1  1
BREAK IN 30
CONT
 2  4  1.41421
BREAK IN 30
```

Je zřejmé, že v programu může být několik příkazů STOP, avšak ani END, ani STOP nejsou v programu vždy nutné.

11.2 Příkaz MEM

Příkaz MEM vypíše rozsah volné vnitřní paměti v bytech.

Příklad

```
MEM
```

výstup:

```
32202 bytes free
```

11.3 Příkaz REM - poznámky v programu

Poznámku lze v programu zapsat kdekoliv, vždy po uvedení příkazu REM. Příkaz REM má tvar:

číslo řádku REM text

Příkaz REM lze využít především k dokumentaci v samotném programu.

Příklad

```
10 REM Reseni soustavy rovnic     označení funkce progra-
20 INPUT "a1,a2,b1,b2,c1,c2";     mu
A1,A2,B1,B2,C1,C2
30 REM A parametr x, B parametr y,
C parametr z
.
.
.
```

Jakýkoliv text včetně případných klíčových slov jazyka, který uvedeme za slovem REM, překladač až do konce programového řádku ignoruje.

Uvedeme-li tedy REM na jednom řádku společně s jinými příkazy, budou příkazy následující za REM považovány za součást poznámky.

Příklad

```
60 PRINT "Sestava": REM tisk hlavicky: PRINT "N.P. KOVO-
   DILO"
.
RUN
```

výstup:

```
Sestava
```

11.4 Příkaz SCRATCH

Příkaz SCRATCH vymaže program uložený v paměti.

ČÁST DRUHÁ

V této druhé části textu jsou uvedeny různé úlohy a jejich provedení na počítači. Všechny programy byly odladěny a vyzkoušeny na počítači IQ 151.

Uvedené úlohy nebo jejich části mohou bezprostředně sloužit k programování dalších programů, leckdy rozsáhlejších. To se týká např. programů pro lineární optimalizaci, vyhledávání, třídění, vyhledání extrémních povků, dělitelů atd., ale také prakticky všech podprogramů, které jsou většinou bez úprav použitelné.

Úlohy jsou z různých oblastí, aby se tím demonstrovala šíře možností. Část inspirace pochází z učebnic pro střední školy, zejména z učebnic matematiky pro gymnázia, ale i z dalších. Úlohy jsou koncipovány tak, že se většinou obejdou bez větších dat. Právě počítač IQ 151 vyhovuje svými vlastnostmi spíše pro ty výpočty, které práci s většími daty příliš nepotřebují.

Opis programů bylo nutné provést na odlišném zařízení, protože možnosti, které v tomto směru má počítač IQ 151, jsou omezené a navíc pro přímé použití v knížce nepoužitelné. Totéž se týká i opisů výsledků. Ty odpovídají téměř věrně podobě, se kterou byste se setkali na obrazovce počítače. Jedinou výjimkou je uvádění odpovědí na otázky, pro něž se používá podprogram Souhlas se vstupním příkazem INKEY$. Příkaz INKEY$ totiž nezobrazuje vloženou odpověď na obrazovce, tak jako je tomu u příkazu INPUT. Odpovědi jsou ve výpisech přesto uváděny, aby čtenář věděl, jak program reagoval na jakou odpověď. Naproti tomu příkaz INKEY$ nutně odpověď vyžaduje. Naproti tomu prázdná odpověď (pouhé stisknutí klávesy CR bez předchozího vložení čísla či textu) ukončuje funkci programu.

12 Vývoj programu - sčítačka

Nejprve si ukážeme některé velmi malé programy. Začneme s jednoduchým sčítáním. Budeme ilustrovat postup, při kterém rostoucí požadavky na vlastnosti programu povedou postupně k jeho zdokonalení, ale také ke zvětšení programu samotného.

12.1 Sčítání známého počtu sčítanců

Většina mikropočítačů - a počítač Iq 151 mezi nimi - může být použit také k přímým výpočtům. O tom jste se dozvěděli v předcházející části knížky. Víte, že za příkazem PRINT (před kterým není číslo řádku - tj. v přímém výpočtu) je možné uvést matematický výraz s čísly a že počítač daný výpočet provede a na obrazovce uvede jeho výsledek.

Přímý výpočet je ovšem vhodný pouze pro nevelké matematické výrazy s malým počtem veličin. Pro sčítání je možné za příkazem PRINT uvést několik sčítanců se značkou + mezi nimi. To však není vhodné při velkém počtu sčítanců.

Ukážeme si nejprve jednoduchý program, který sčítání provádí podobně, jako bychom to dělali na sčítačce anebo malé kalkulačce.

Program: Sčítačka 1

```
 10 REM Scitacka 1
 20 REM Znamy pocet scitancu
 30 CLS
 40 PRINT "PROGRAM PRO SCITANI CISEL 1"
 50 PRINT
 60 INPUT "Zadejte pocet cisel";N
 70 S=0
 80 FOR I=1 TO N
 90 INPUT "Vlozte (dalsi) cislo";R
100 S=S+R
110 NEXT I
120 PRINT
130 PRINT "Soucet je";S
```

```
140 PRINT
150 PRINT "Konec."
160 END
```

Příklad

```
PROGRAM PRO SCITANI CISEL ]"

Zadejte pocet cisel: 3
Vlozte (dalsi) cislo: 150
Vlozte (dalsi) cislo: 235
Vlozte (dalsi) cislo: 720

Soucet je 1105

Konec.
```

Pro tři čísla by jistě nebylo potřeba program pro sčítání dělat. Jako ostatně kterýkoliv program vůbec má smysl teprve při jistém rozsahu a určité složitosti práce, kterou nahrazuje. Všimněme si však jiné okolnosti. Aby bylo možné v programu použít cyklu s daným počtem opakování, je potřeba předem znát počet sčítanců. To ale není obvyklé. Častější je případ, kdy máte před sebou dlouhý sloupec čísel, která se mají sčítat. O nich ale předem nevíte, kolik jich je. V tomto smyslu je uvedený program nedokonalý, protože by vás nutil zjistit předem počet sčítanců.

Ve skutečnosti však můžeme postupovat i jinak. Vkládat čísla pro sčítání, nebo v jiných případech pro jiné výpočty, aniž předem známe jejich počet. Pak ovšem potřebujeme při vkládání čísel poznat, kdy jsme u konce. Kdy jsme vložili poslední sčítanec (99999).

Tomuto účelu slouží předem zvolená hodnota. Obvykle se volí taková hodnota, která se mezi sčítanci nevyskytuje, dejme tomu pět devítek.

12.2 Sčítání neznámého počtu sčítanců

Předcházející program upravíme tak, aby bylo možné na něm sčítat předem neznámý počet sčítanců. V programu se pozná, že vkládání je ukončeno právě podle čísla, které se zadává pěti devítkami (99999).

Program: Sčítačka 2

```
 10 REM Scitacka 2
 20 REM Neznamy pocet scitancu
 30 CLS
 40 PRINT "PROGRAM PRO SCITANI CISEL 2"
 50 PRINT
 60 PRINT "Za poslednim cislem vlozte 99999!"
 70 S=0
 80 INPUT "Vlozte (dalsi) cislo";R
 90 IF R=99999 THEN 120
100 S=S+R
110 GOTO 80
120 PRINT
130 PRINT "Soucet je";S
140 PRINT
150 PRINT "Konec."
160 END
```

Příklad

```
PROGRAM PRO SCITANI CISEL 2

Za poslednim cislem vlozte 99999!
Vlozte (dalsi) cislo: 150
Vlozte (dalsi) cislo: 235
Vlozte (dalsi) cislo: 720
Vlozte (dalsi) cislo: 99999

Soucet je 1105

Konec.
```

Ukončení vkládání dat podle zadané hodnoty má jednu slabinu v tom, že se zadaná hodnota může vyskytnout mezi skutečnými sčítanci. Kdyby se to stalo, museli bychom ji vložit jako dvojici hodnot, např. 9999Ø a 9. Pro výpočet součtu takový zásah nevadí, ale pro některé jiné, například výpočet průměru,

by takový postup zkreslil výsledek.

Oba uvedené malé programy mají ovšem jednu slabinu, kterou sčítací stroje s průběžným výpisem sčítanců na pás papíru nemají. Tou je nemožnost kontroly vložených hodnot.

Bylo by nutné postupovat tak, jak se postupuje při sčítání na kalkulačkách, které také nemají průběžný výpis sčítanců: udělat součet dvakrát. Jestliže se výsledky obou součtů shodují, můžeme předpokládat, že jsme počítali správně. Jestliže jsou oba součty různé, musíme si je poznamenat a sčítání opakovat tak dlouho, až dostaneme alespoň dva stejné součty.

12.3 Sčítání s kontrolním výpisem

Mikropočítač IQ 151 (a samozřejmě také další) nám však umožňují nespokojit se s touto nedokonalostí, ale upravit program tak, aby kontrolu vložených sčítanců umožňoval. Právě o tuto možnost je obohacen program, který následuje.

Program: Sčítačka s kontrolním výpisem

```
 10 REM Scitacka 3
 20 REM Scitani s kontrolnim vypisem
 30 DIM A(100)
 40 CLS
 50 PRINT
 60 PRINT "PROGRAM PRO SCITANI AZ 100 CISEL"
 70 PRINT "--------------------------------"
 80 PRINT "s moznosti kontrolniho vypisu."
 90 PRINT
100 PRINT "Cisla mohou byt cela nebo realna"
110 PRINT "se setinami (Kcs a halere)."
120 PRINT "Maximalni hodnota je 100000."
130 PRINT
140 S=0
150 T$="Vite kolik cisel budete scitat?"
160 GOSUB 870
170 IF Q=0 THEN 360
180 INPUT "Vlozte pocet scitanych cisel";N
190 N=ABS(N)
200 N=INT(N)
210 IF N > 100 THEN 820
220 IF N > 1 THEN 250
230 PRINT "Cisla musi byt nejmene 2!"
240 GOTO 180
```

```
250 CLS
260 FOR I=1 TO N
270 PRINT "Vlozte cislo";I;
280 GOSUB 980
290 IF ABS(R) < 100001 THEN 320
300 PRINT "Cislo nesmi byt vetsi nez 100000"
310 GOTO 270
320 S=S+R
330 A(I)=R
340 NEXT I
350 GOTO 500
360 PRINT "Za poslednim cislem vlozte 99999."
370 I=1
380 PRINT "Vlozte cislo";I;
390 GOSUB 980
400 IF R=99999 THEN 490
410 IF ABS(R) < 100001 THEN 440
420 PRINT "Cislo nesmi byt vetsi nez 100000"
430 GOTO 380
440 S=S+R
450 A(I)=R
460 I=I+1
470 IF I < 101 THEN 380
480 GOTO 820
490 N=I-1
500 REM Vypis
510 PRINT "Cisel je celkem";N
520 PRINT "Jejich soucet je";S
530 PRINT
540 T$="Chcete kontrolni vypis?"
550 GOSUB 870
560 IF Q=0 THEN 840
570 REM Kontrolni vypis
580 T$="Muze se pokracovat ve vypise?"
590 GOSUB 1020
600 FOR I=1 TO N
610 IF I < 100 THEN PRINT " ";
620 IF I < 10 THEN PRINT " ";
630 PRINT I;"  ";
640 B=A(I)
650 IF B < 100000 THEN PRINT " ";
660 IF B < 10000 THEN PRINT " ";
670 IF B < 1000 THEN PRINT " ";
680 IF B < 100 THEN PRINT " ";
690 IF B < 10 THEN PRINT " ";
700 PRINT B
710 IF I/10<>INT(I/10) THEN 750
720 GOSUB 870
730 IF Q=0 THEN 760
740 GOSUB 1020
750 NEXT I
760 T$="Jsou vsechna cisla spravna?"
```

```
770 GOSUB 870
780 IF Q=1 THEN 500
790 PRINT
800 PRINT "Musite je vlozit znovu!"
810 GOTO 840
820 PRINT "Cisel je vic nez 100!"
830 PRINT "Musite zvetsit velikost A(100)!"
840 PRINT
850 PRINT "Konec."
860 END
870 REM Souhlas
880 PRINT T$
890 Q$=INKEY$
900 IF Q$="" THEN 890
910 Q=1
920 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
930 Q=0
940 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
950 PRINT "Odpovidejte takto:"
960 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
970 GOTO 870
980 REM Uprava vstupu
990 INPUT R
1000 R=INT(100*R+0.5)/100
1010 RETURN
1020 REM Hlavicka
1030 PRINT "  C.       Kcs,--"
1040 RETURN
```

Příklad

```
PROGRAM PRO SCITANI AZ 100 CISEL
--------------------------------
s moznosti kontrolniho vypisu.

Cisla mohou byt cela nebo realna
se setinami (Kcs a halere).
Maximalni hodnota je 100000.

Vite kolik cisel budete scitat?
N

Za poslednim cislem vlozte 99999.
Vlozte cislo 1 :150
Vlozte cislo 2 :235
Vlozte cislo 3 :720
Vlozte cislo 4 :99999

Cisel je celkem 3
Jejich soucet je 1105
```

```
Chcete kontrolni vypis?
A
  C.      Kcs,--
   1      150
   2      235
   3      720

Jsou vsechna cisla spravna?
:N

Musite je vlozit znovu!

Konec.
```

Program je již o něco delší. Protože jsme chtěli umožnit udělat kontrolní výpis sčítanců, museli jsme předcházející jednoduché programy doplnit hned o několik dalších částí, které právě umožní změnu vlastností, a to:

- dát do programu pole (vektor pro číselné hodnoty), do něhož se jednotlivé vkládané sčítance průběžně ukládají, aby bylo možné se k nim vrátit,
- umožnit kontrolní výpis všech vložených čísel na obrazovku,
- s tím ovšem souvisí další okolnost. Tou je nutnost vypisování sčítanců na obrazovku po částech tak, aby se každý částečný výpis dal porovnávat. Nemůže se stát, aby čísla přeběhla po obrazovce bez možnosti je na obrazovce zastavit. A jestliže se na obrazovce objeví část sčítanců, musí být také možnost ve vypisování pokračovat.

Toto všechno již v programu je a přesto ještě není zcela dokonalý, alespoň do té míry, aby se dal dobře používat. Schází mu ještě možnost vložená data opravovat. Když totiž zjistíme, že v datech je chyba, třeba jen jediná, nezbývá nám u tohoto programu nic jiného, než je všechna vložit znovu.

Tento problém řeší následující program.

12.4 Sčítání s možností oprav vložených dat

Aby se dala vložená data opravovat, musí být program upraven tak, aby to umožňoval. Musí být možné zvolit si číslo, které má být opraveno, a také opravu provést. Pracovně nejjednodušší to je hned při vypisování sčítanců.

Kromě toho se může stát, že chceme data vypisovat opakovaně tolikrát, až budou bezchybná. Konečně - po každé opravě se musí uskutečnit nový součet. Toto všechno je v tomto programu obsaženo.

Program: Sčítačka 4

```
 10 REM Scitacka 4
 20 REM Scitani s kontrolnim vypisem
 30 REM Opravovani dat v pameti
 40 DIM A(100)
 50 CLS
 60 PRINT
 70 PRINT "PROGRAM PRO SCITANI AZ 100 CISEL"
 80 PRINT "--------------------------------"
 90 PRINT "s moznosti kontrolniho vypisu"
100 PRINT "a opravovani."
110 PRINT
120 PRINT "Cisla mohou byt cela nebo realna"
130 PRINT "se setinami (Kcs a halere)."
140 PRINT "Maximalni hodnota je 100000."
150 PRINT
160 T$="Vite kolik cisel budete scitat?"
170 GOSUB 860
180 IF Q=0 THEN 310
190 INPUT "Vlozte pocet scitanych cisel";N
200 N=ABS(N)
210 N=INT(N)
220 IF N > 100 THEN 810
230 IF N > 1 THEN 260
240 PRINT "Cisla musi byt nejmene 2!"
250 GOTO 190
260 CLS
270 FOR I=1 TO N
280 GOSUB 970
290 NEXT I
300 GOTO 400
310 REM Vkladani s posledni hodnotou
320 PRINT "Za poslednim cislem vlozte 99999!"
330 I=1
340 GOSUB 970
350 IF R=99999 THEN 390
360 I=I+1
```

```
370 IF I < 101 THEN 340
380 GOTO 810
390 N=I-1
400 REM Kontrolni vypis
410 T$="Chcete kontrolni vypis?"
420 GOSUB 860
430 IF Q=0 THEN 720
440 T$="Muze se pokracovat ve vypise?"
450 GOSUB 1070
460 FOR J=1 TO N
470 IF J < 100 THEN PRINT " ";
480 IF J < 10 THEN PRINT " ";
490 PRINT J;"  ";
500 B=A(J)
510 IF B < 100000 THEN PRINT " ";
520 IF B < 10000 THEN PRINT " ";
530 IF B < 1000 THEN PRINT " ";
540 IF B < 100 THEN PRINT " ";
550 IF B < 10 THEN PRINT " ";
560 PRINT B
570 IF J/10<>INT(J/10) THEN 640
580 GOSUB 860
590 IF Q=1 THEN 630
600 PRINT "Ktere cislo chcete opravit?"
610 INPUT I
620 GOSUB 970
630 GOSUB 1070
640 NEXT J
650 T$="Jsou vsechna cisla spravna?"
660 GOSUB 860
670 IF Q=1 THEN 720
680 PRINT "Ktere cislo chcete opravit?"
690 INPUT I
700 GOSUB 970
710 GOTO 410
720 REM Soucet
730 S=0
740 FOR I=1 TO N
750 S=S+A(I)
760 NEXT I
770 PRINT "Cisel je celkem";N
780 PRINT "Jejich soucet je";S
790 GOTO 830
800 PRINT
810 PRINT "Cisel je vic nez 100!"
820 PRINT "Musite zvetsit velikost A(100)!"
830 PRINT
840 PRINT "Konec."
850 END
860 REM Souhlas
870 PRINT T$
880 Q$=INKEY$
```

```
890 IF Q$="" THEN 880
900 Q=1
910 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
920 Q=0
930 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
940 PRINT "Odpovidejte takto:"
950 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
960 GOTO 860
970 REM Vstup, kontrola, uprava
980 PRINT "Vlozte hodnotu";I;
990 INPUT R
1000 IF R=99999 THEN RETURN
1010 R=INT(100*R+0.5)/100
1020 IF ABS(R) < 100001 THEN 1050
1030 PRINT "Cislo nesmi byt vetsi nez 100000"
1040 GOTO 980
1050 A(I)=R
1060 RETURN
1070 REM Hlavicka
1080 PRINT "  C.      Kcs,--"
1090 RETURN
```

Příklad

```
PROGRAM PRO SCITANI AZ 100 CISEL
--------------------------------
s moznosti kontrolniho vypisu
a opravovani.

Cisla mohou byt cela nebo realna
se setinami (Kcs a halere).
Maximalni hodnota je 100000.

Vite kolik cisel budete scitat?
:N

Za poslednim cislem vlozte 99999!
Vlozte cislo 1 :150
Vlozte cislo 2 :235
Vlozte cislo 3 :720
Vlozte cislo 2 :99999

Cisel je celkem 3
Jejich soucet je 1105

Chcete kontrolni vypis?
:A
  C.      Kcs,--
   1      150
   2      235
   3      720
```

```
Jsou vsechna cisla spravna?
:N

Ktere cislo chcete opravit?
3
Vlozte hodnotu 3: 740
Chcete kontrolni vypis?
:N
Cisel je celkem 3
Jejich soucet je 1125

Konec.
```

I když se program s možností oprav vložených čísel příliš neliší od předcházejícího, je z hlediska své funkce dokonalejší. Je totiž málo platné zjistit chybu, nemáme-li možnost ji opravit, zejména při velkém počtu sčítaných čísel.

Ve zdokonalování programu a tím také k obecnější práci s ním je možné pokračovat i dále. Bylo by např. vhodné, kdyby se dalo - kromě oprav již vložených čísel - počítat také s jejich rozšířením. To by bylo nutné, kdybychom při kontrole zjistili, že jsme některé číslo vynechali vůbec.

Zobecňování programu může dále pokračovat k volbě rozsahu číslic za desetinnou tečkou atd. Uvedené příklady měly právě ukázat, jak požadavky na funkci programu mění jeho obsah. Čím větší jsou požadavky a čím univerzálnější funkci má program plnit, tím také program sám je větší a komplikovanější.

13 Řešení rovnic

Zde budou popsány programy pro řešení některých typů rovnic, které se vyučují na středních školách. Jedná se o kvadratickou rovnici a o soustavu lineárních rovnic o dvou a třech neznámých.

13.1 Soustava lineárních rovnic o dvou neznámých

Nejčastěji počítané příklady rovnic ve škole jsou rovnice o dvou neznámých. Budeme předpokládat běžný tvar rovnic:

$$A(1,1)*X(1) + A(1,2)*X(2) = B(1)$$
$$A(2,1)*X(1) + A(2,2)*X(2) = B(2)$$

V programu je použit postup pomocí determinantů, který sice není nejvhodnější pro soustavy o velkém počtu rovnic, ale pro tento účel se zdá nejjednodušší.

Program: Lineární rovnice o 2 neznámých

```
 10 REM LINROV 2
 20 REM Linearni rovnice o 2 neznamych
 30 REM Reseni pomoci determinantu
 40 DIM A(2,2),B(2)
 50 CLS
 60 PRINT "Reseni soustavy 2 lin. rovnic."
 70 PRINT "------------------------------"
 80 PRINT
 90 PRINT "Vkladejte koeficienty po radcich"
100 N=2
110 FOR I=1 TO N
120 FOR J=1 TO N
130 PRINT "Vlozte koeficient";I;"-";J;
140 INPUT A(I,J)
150 NEXT J
160 NEXT I
170 PRINT
180 PRINT "Vkladejte absolutni cleny"
190 FOR I=1 TO N
200 PRINT "Vlozte clen";I;
210 INPUT B(I)
```

```
220 NEXT I
230 PRINT
240 D=A(1,1)*A(2,2)-A(1,2)*A(2,1)
250 IF D <> 0 THEN 280
260 PRINT "Soustava nema reseni."
270 GOTO 320
280 X=B(1)*A(2,2)-B(2)*A(1,2)
290 Y=A(1,1)*B(2)-B(1)*A(2,1)
300 PRINT "X =";X/D
310 PRINT "Y =";Y/D
320 PRINT
330 PRINT "Konec."
340 END
```

Příklad 1

```
Reseni soustavy 2 lin. rovnic.
------------------------------

Vkladejte koeficienty po radcich
Vlozte koeficient 1 - 1 : 2
Vlozte koeficient 1 - 2 : -1
Vlozte koeficient 2 - 1 : 1
Vlozte koeficient 2 - 2 : 3

Vkladejte absolutni cleny
Vlozte clen 1 : 5
Vlozte clen 2 : 6

X = 3
Y = 1

Konec.
```

Příklad 2

```
Reseni soustavy 2 lin. rovnic.
------------------------------

Vkladejte koeficienty po radcich
Vlozte koeficient 1 - 1 : 3
Vlozte koeficient 1 - 2 : 4
Vlozte koeficient 2 - 1 : 6
Vlozte koeficient 2 - 2 : 8
```

```
Vkladejte absolutni cleny
Vlozte clen 1 : 5
Vlozte clen 2 : 10

Soustava nema reseni.

Konec.
```

Jestliže je determinant soustavy nulový, neexistuje pro ni řešení. Pokud tento případ nastane, pokračuje se přímo k této informaci a další výpočty se neprovádí. Takový výsledek uvádí další ukázka.

V programu se nerozlišuje případ, kdy daná soustava lineárních rovnic nemá řešení vůbec, jestliže si rovnice vzájemně odporují, od případu, kdy má nekonečně mnoho řešení, jsou-li lineární rovnice na sobě závislé.

V obou těchto situacích je determinant soustavy nulový a tímto zjištěním program končí.

13.2 Soustava lineárních rovnic o třech neznámých

Další program je určen pro řešení rovnic o třech neznámých. Opět budeme předpokládat běžný tvar rovnic:

```
A(1,1)*X(1) + A(1,2)*X(2) + A(1,3)*X(3)  =  B(1)
A(2,1)*X(1) + A(2,2)*X(2) + A(2,3)*X(3)  =  B(2)
A(3,1)*X(1) + A(3,2)*X(2) + A(3,3)*X(3)  =  B(3)
```

V programu je i zde použit postup pomocí determinantů. Porovnáním programů může být patrné, jak se výpočet stává složitější i když se počet rovnic i neznámých zvětšil jen o jednu.

Program: Lineární rovnice o 3 neznámých

```
 10 REM LINROV 3
 20 REM Linearni rovnice o 3 neznamych
 30 REM Reseni pomoci determinantu
 40 DIM A(3,3),B(3)
 50 CLS
 60 PRINT "Reseni soustavy 3 lin. rovnic."
 70 PRINT "------------------------------"
 80 PRINT
 90 PRINT "Vkladejte koeficienty po radcich"
100 N=3
110 FOR I=1 TO N
120 FOR J=1 TO N
130 PRINT "Vlozte koeficient";I;"-";J;
140 INPUT A(I,J)
150 NEXT J
160 NEXT I
170 PRINT
180 PRINT "Vkladejte absolutni cleny"
190 FOR I=1 TO N
200 PRINT "Vlozte clen";I;
210 INPUT B(I)
220 NEXT I
230 PRINT
240 D=A(1,1)*A(2,2)*A(3,3)
250 D=D+A(2,1)*A(3,2)*A(1,3)
260 D=D+A(3,1)*A(1,2)*A(2,3)
270 D=D-A(1,3)*A(2,2)*A(3,1)
280 D=D-A(2,3)*A(3,2)*A(1,1)
290 D=D-A(3,3)*A(1,2)*A(2,1)
300 IF D <> 0 THEN 330
310 PRINT "Soustava nema reseni."
320 GOTO 540
330 X=B(1)*A(2,2)*A(3,3)
340 X=X+B(2)*A(3,2)*A(1,3)
350 X=X+B(3)*A(1,2)*A(2,3)
360 X=X-A(1,3)*A(2,2)*B(3)
370 X=X-A(2,3)*A(3,2)*B(1)
380 X=X+A(3,3)*A(1,2)*B(2)
390 Y=A(1,1)*B(2)*A(3,3)
400 Y=Y+A(2,1)*B(3)*A(1,3)
410 Y=Y+A(3,1)*B(1)*A(2,3)
420 Y=Y-A(1,3)*B(2)*A(3,1)
430 Y=Y-A(2,3)*B(3)*A(1,1)
440 Y=Y-A(3,3)*B(1)*A(2,1)
450 Z=A(1,1)*A(2,2)*B(3)
460 Z=Z+A(2,1)*A(3,2)*B(1)
470 Z=Z+A(3,1)*A(1,2)*B(2)
480 Z=Z-B(1)*A(2,2)*A(3,1)
490 Z=Z-B(2)*A(3,2)*A(1,1)
500 Z=Z-B(3)*A(1,2)*A(2,1)
510 PRINT "X =";X/D
```

```
520 PRINT "Y =";Y/D
530 PRINT "Z =";Z/D
540 PRINT
550 PRINT "Konec."
560 END
```

Příklad

```
Reseni soustavy 3 lin. rovnic.
------------------------------

Vkladejte koeficienty po radcich
Vlozte koeficient 1 - 1 : 1
Vlozte koeficient 1 - 2 : 2
Vlozte koeficient 1 - 3 : -3
Vlozte koeficient 2 - 1 : -3
Vlozte koeficient 2 - 2 : 1
Vlozte koeficient 2 - 3 : -2
Vlozte koeficient 3 - 1 : 2
Vlozte koeficient 3 - 2 : 3
Vlozte koeficient 3 - 3 : 2

Vkladejte absolutni cleny
Vlozte clen 1 : -1
Vlozte clen 2 : 2
Vlozte clen 3 : 11

X =-1
Y = 3
Z = 2

Konec.
```

Metod pro řešení soustav lineárních rovnic je víc. Na středních školách se některé z nich probírají. Např. eliminační metoda se v této knížce používá jako součást simplexové metody pro řešení úloh lineární optimalizace.

13.3 Kvadratická rovnice

Další program je určen pro řešení kvadratické rovnice. V programu je napřed vypočten diskriminant. Další postup se řídí podle toho, jaké hodnoty diskriminant nabývá. Vzhledem k tomu, že se studenti na středních školách ještě neseznamu-

jí s komplexními čísly, je při záporném diskriminantu výpočet ukončen a úloha prohlášena za neřešitelnou.

Pro rozhodnutí o dalším postupu podle hodnoty diskriminanty je v programu výhodně užita funkce SGN. Jak je známo, je hodnota funkce SGN rovna -1, je-li původní hodnota záporná, hodnota SGN se rovná nule, je-li i původní hodnota nula, a rovná se 1, když je původní hodnota kladná.

Protože se v příkazu ON GOTO mohou vyskytovat pouze kladné hodnoty, je výsledek funkce SGN zvětšen o 2. Pak jsou výsledné hodnoty 1, 2 nebo 3 podle znaménka diskriminantu. Právě podle toho se rozlišuje možný výsledek a také rozvětvuje následující program.

Funkce SGN s následujícím příkazem ON GOTO je v programu použita ještě na dalších dvou místech. Obě se vyskytují při tisku výsledu, v němž je potřeba rozlišit, zda jsou koeficienty u neznámé kladné, záporné, nebo nuly. Pro nulové se příslušná část rovnice nepíše vůbec. U kladných koeficientů je nutné zajistit výstup značky + zejména uprostřed rovnice, která v běžné formě nevystupuje. Pro zápornou značku se pak používá stejný postup proto, aby se zachovala stejná výstupní forma jako u kladných koeficientů.

Program: Kvadratická rovnice

```
 10 REM Kvadraticka rovnice
 20 CLS
 30 PRINT
 40 PRINT "Reseni kvadraticke rovnice."
 50 PRINT "---------------------------"
 60 PRINT
 70 PRINT "Vlozte koeficient kvadratickeho clenu (a)";
 80 INPUT A
 90 PRINT "Vlozte koeficient linearniho    clenu (b)";
100 INPUT B
110 PRINT "Vlozte absolutni clen (c)";
120 INPUT C
130 PRINT
140 PRINT " Cela rovnice vypada takto:"
150 PRINT
160 PRINT A;"* X*X ";
170 ON SGN(B)+2 GOTO 200,220,180
```

```
180 PRINT "+";
190 GOTO 210
200 PRINT "-";
210 PRINT ABS(B);"* X ";
220 ON SGN(C)+2 GOTO 250,270,230
230 PRINT "+";
240 GOTO 260
250 PRINT "-";
260 PRINT ABS(C);
270 PRINT "  =   0"
280 PRINT
290 D=B*B-4*A*C
300 PRINT "Diskriminant =";D
310 ON SGN(D)+2 GOTO 320,340,370
320 PRINT "Reseni je v mnozine KOMPLEXNICH cisel."
330 GOTO 430
340 PRINT "Rovnice ma jeden DVOJNASOBNY    koren."
350 Y=(-B/(2*A))
360 GOTO 420
370 PRINT "Rovnice ma 2 koreny."
380 D=SQR(D)
390 X=(-B+D)/(2*A)
400 Y=(-B-D)/(2*A)
410 PRINT "X1 =";X
420 PRINT "X2 =";Y
430 PRINT
440 PRINT "Konec."
450 END
```

Příklad 1:

```
Reseni kvadraticke rovnice.
---------------------------

Vlozte koeficient kvadratickeho
clenu (a): 2
Vlozte koeficient linearniho
clenu (b): -3
Vlozte absolutni clen (c): 1

 Cela rovnice vypada takto:

 2 * X*X - 3 * X + 1   =   0

Diskriminant = 1
Rovnice ma 2 koreny.
X1 = 1
X2 = .5

Konec.
```

Příklad 2:

```
Reseni kvadraticke rovnice.
---------------------------

Vlozte koeficient kvadratickeho
clenu (a): 4
Vlozte koeficient linearniho
clenu (b): -12
Vlozte absolutni clen (c): 9

 Cela rovnice vypada takto:

 4 * X*X - 12 * X + 9    =    0

Diskriminant = 0
Rovnice ma 1 DVOJNASOBNY
koren.
X2 = 24

Konec.
```

Příklad 3:

```
Reseni kvadraticke rovnice.
---------------------------

Vlozte koeficient kvadratickeho
clenu (a): 2
Vlozte koeficient linearniho
clenu (b): 2
Vlozte absolutni clen (c): 2

 Cela rovnice vypada takto:

 2 * X*X - 2 * X + 2    =    0

Diskriminant = -12
Reseni je v mnozine KOMPLEXNICH
cisel.

Konec.
```

14 Hledání v datech

Princip binárního hledání, který se dá použít pouze v uspořádaném souboru, spočívá v tom, že se soubor rozdělí na dvě poloviny. Podle velikosti prostředního prvku se zjistí, ve které polovině je hledaná jednotka. Tím se velikost souboru, ve kterém máme hledat, zmenší na polovinu. Tento postup se opakuje. Znovu se zjistí, ve které ze dvou polovin poloviního souboru se hledaná jednotka nachází, a znovu se soubor zmenší na polovinu atd.

Tento princip zaručuje nalezení hledané jednotky anebo zjištění, že jednotka v souboru není, velmi rychle. Rychlost závisí na velikosti souboru. Dá se popsat vztahem

$$n <= 2^k,$$

kde n je rozsah souboru a k počet kroků, které jsou pro vyhledání nezbytné.

Princip budeme ilustrovat na jednoduchém souboru přirozených čísel. Takový soubor je samozřejmě uspořádaný od 1 až do maximálního čísla, které je v programu uvažováno jako 131072. Ve skutečnosti ovšem takový soubor fakticky ani nebude v programu existovat. Prakticky je ovšem možné, aby existoval skutečný soubor, např. seznam zaměstnanců abecedně uspořádaný. V něm by se hledalo podobným způsobem, jaký je zde popisován.

14.1 Binární hledání počítačem

V tomto programu hledá program počítače číslo, které si myslíte. Přitom právě využívá princip binárního hledání.

Program: Binární hledání

```
 10 REM Binarni hledani
 20 REM Pocitac hleda binarne
 30 CLS
 40 PRINT "Hadam cislo, ktere si zvolite."
 50 PRINT "Zvolte si cislo do 131072."
 60 PRINT "Misto ANO vlozte 1, misto NE 0!"
 70 PRINT "Z A C I N A M E"
 80 A=65536
 90 B=A
100 B=INT(B/2)
110 PRINT "Je Vase cislo vetsi nez";A;"?"
120 INPUT C
130 IF C=0 OR C=1 THEN 160
140 PRINT "Odpovidejte pouze 1 nebo 0!"
150 GOTO 110
160 IF B=0 THEN 220
170 IF C=0 THEN 200
180 A=A+B
190 GOTO 100
200 A=A-B
210 GOTO 100
220 A=A+C
230 "Hledane cislo je";A
240 PRINT "Je to tak?"
250 INPUT C
260 IF C=0 OR C=1 THEN 290
270 PRINT "Odpovidejte pouze 1 nebo 0!"
280 GOTO 230
290 IF C=1 THEN 320
300 PRINT "Podvadite! Dal nehraji!"
310 GOTO 400
320 PRINT "Dekuji Vam!"
330 PRINT "Chcete ve hre pokracovat?"
340 INPUT C
350 IF C=0 OR C=1 THEN 380
360 PRINT "Odpovidejte pouze 1 nebo 0!"
370 GOTO 330
380 IF C=1 THEN 30
390 PRINT "Konec hrani!"
400 END
```

Ukázku nebudeme uvádět, je jednoduchá. Spočívá pouze v opakovaném dialogu, v němž odpovídáte na dotazy počítače. Ten systematicky zmenšuje prostor, v němž se vaše číslo nalézá, tak jak to odpovídá principu binárního hledání.

14.2 Najděte rychle číslo

Program Najděte rychle číslo je opakem předcházejícího programu. Vyzkouší vás, jak sami dokážete rychle hledat zadané číslo, ať znáte princip binárního hledání, anebo ne.

Program je sestaven tak, že se na počátku hledané číslo nevytváří náhodným způsobem, ale že jej do počítače vloží váš partner. Nejprve se tedy žádá vložení čísla, které má váš kamarád hledat. Po jeho vložení se obrazovka vymaže a následuje postup hledání. Pokusy hledání jsou sečítány a také se nakonec dozvíte, kolik kroků bylo pro hledání potřeba nejméně.

Program: Najděte číslo

```
 10 REM Najdete cislo
 20 REM Binární hledání proti počítači
 30 CLS
 40 INPUT "Vlozte cislo pro kamarada";X
 50 X=INT(X)
 60 IF X > 0 AND X < 131072 THEN  90
 70 PRINT "0 < vlozene cislo < 131072!"
 80 GOTO 40
 90 PRINT "Je vlozeno cislo";X
100 K=1
110 FOR I=1 TO 17
120 K=K*2
130 IF K >= X THEN 150
140 NEXT I
150 K=I
160 CLS
170 Z=0
180 Z=Z+1
190 INPUT "Vlozte hadane cislo";Y
200 IF X >= Y THEN 230
210 PRINT "Hledane cislo je MENSI."
220 GOTO 180
230 IF X = Y THEN 260
240 PRINT "Hledane cislo je VETSI."
250 GOTO 180
260 PRINT "G R A T U L U J E M E"
```

```
270 PRINT "Potreboval jste";Z;"pokusu."
280 PRINT "Optimalni pocet pokusu je";K
290 PRINT "Chcete ve hre pokracovat?"
300 INPUT Q$
310 IF Q$="1" OR Q$="A" OR Q$="a" THEN 30
320 PRINT "Konec hrani!"
330 END
```

Ani u tohoto programu není ukázka potřebná. Ilustrovala by pouze dlouhý dialog. Všimněte si pouze jiné formy, kterou je veden dialog na konci programu, resp. jak je vyhodnocována odpověď.

14.3 Dvourozměrné hledání

Hledání, které se vyskytovalo v předcházejících dvou programech, si můžeme představit jako hledání bodu na úsečce. Například místa na trati mezi Prahou a Brnem. Uvažovanou úsečku ani cestu neuvažujeme ovšem jako spojitou, ale jakoby byla rozdělena na konečný a ne příliš velký počet úseků, z nichž jeden máme určit.

Podobně v tomto programu budeme hledat bod v rovině. Také v této situaci se hledání usnadníme tím, že rovinu rozdělíme na myšlený konečný počet menších částí. Ty si můžeme představit jako čtverce uvnitř velké čtvercové plochy.

	1	2	3
1	SZ	S	SV
2	Z	x	V
3	JZ	J	JV

Pro snadnější domluvu budeme plochu uvažovat jako mapu, tak, abychom se mohli snadněji popsat směry pohybu, v nichž se má v hledání pokračovat. Na schématu je formou tabulky zapsáno, jakým způsobem se bude doporučovat směr dalšího postupu.

V centru obrázku je místo, na kterém se právě nacházíte. Toto místo udáte souřadnicí (číslem v rozsahu 1 - 131072) ve směru V-Z, tj. východ-západ, a druhou souřadnicí (rovněž číslem v rozsahu 1 - 131072) ve směru S-J, tj. sever-jih.

Hledání bude o něco obtížnější, protože je třeba dávat pozor na postup ve dvou směrech současně.

Program: Plošné hledání

```
 10 REM Binarni hledani
 20 REM Dvourozmerne
 30 CLS
 40 CLS
 50 PRINT "HLEDANI BODU V ROVINE"
 60 PRINT
 70 PRINT "Rovina se vodorovne i svisle"
 80 PRINT "deli na 131072x131072 poli."
 90 PRINT
100 PRINT "Mate najit pole s pokladem."
110 PRINT "Zadavejte jednu souradnici"
120 PRINT "pole ve smeru vychod-zapad (X)"
130 PRINT "a druhou ve smeru sever-jih (Y)."
140 PRINT "Pak se dovite, kterym smerem"
150 PRINT "mate pokracovat. Zaciname!"
160 T$="Pokracujte na"
170 REM Vytvoreni nahodneho cisla
180 X=INT(131072*RND(0))+1
190 Y=INT(131072*RND(0))+1
200 A=0
210 B=0
220 Z=0
230 REM Pocitani pokusu
240 Z=Z+1
250 PRINT
260 IF A=X THEN 280
270 INPUT "Jaky bod hadate ve smeru V-Z";A
280 IF B=Y THEN 300
290 INPUT "Jaky bod hadate ve smeru S-J";B
300 R=SGN(X-A)+2
310 S=(SGN(B-Y)+1)*3
320 REM Rozvetveni podle smeru
330 ON R+S GOTO 340,360,380,400,500,420,440,460,480
340 PRINT T$;" severozapad!"
350 GOTO 230
360 PRINT T$;" sever!"
370 GOTO 230
380 PRINT T$;" severovychod!"
390 GOTO 230
400 PRINT T$;" zapad!"
410 GOTO 230
420 PRINT T$;" vychod!"
```

```
430 GOTO 230
440 PRINT T$;" jihozapad!"
450 GOTO 230
460 PRINT T$;" jih!"
470 GOTO 230
480 PRINT T$;" jihovychod!"
490 GOTO 230
500 PRINT
510 PRINT "BLAHOPREJI VAM!"
520 PRINT "Potreboval jste";Z;"pokusu."
530 END
```

Při práci s tímto programem se na obrazovce opakovaně objevuje návod pro směr dalšího postupu. Podle toho, jak přesně se budete řídit těmito pokyny, dostanete se na pole s pokladem rychleji, nebo pomaleji. Postup by neměl trvat déle (větší počet kroků) než odpovídá vztahu pro binární hledání. Pokud se nebudete držet pravidel binárního hledání, pak vám sice někdy může být náhoda nápomocná a hledání může být rychlejší, ale většinou to bývá právě naopak.

V programu si můžete všimnout, jakým poměrně jednoduchým způsobem se určí směr, v němž se má dále hledat. Výhodně je pro to využita funkce SGN, která pro oba směry rozliší základní směr postupu. Jejich kombinace pro pohyb v rovině je pomocí úpravy téže funkce udělána tak, že součet hodnot proměnných (v programu S a R) nabývá hodnoty 1, 2, 3, ... , 9. To jsou právě všechny možné směry, včetně případu nalezení pokladu - střed větrné růžice. Příslušný text se pak vyvolá po rozvětvení programu pomocí příkazu ON S+R GOTO ....

14.4 Trinární hledání

Tímto názvem označujeme postup hledání, který využívá stejného principu binárního hledání jako programy předcházející. Liší se pouze v tom, že jako odpověď na nabízené číslo připouští tři možnosti:

a) M - hledané číslo je menší,

b) V - hledané číslo je větší,

c) R - hledané číslo se rovná.

Již u předcházejících programů jste si totiž mohli všimnout, že se číslo, které hledáte, na obrazovce objeví. Protože ale musíte odpovědět pouze na jednu otázku, např. je-li hledané číslo menší, odpovíte, že ne. A pak se může stát, že trvá ještě dosti dlouho, než se dojde ke zjištění, o které číslo se jednalo.

Tuto nevýhodu částečně odstraňuje program, který následuje. Samotné hledání se tím příliš nezrychlí. Pouze v jednom případě z celého souboru se může stát, že se hledané číslo objeví hned poprvé na obrazovce. Pouze ve dvou dalších případech se může objevit ve druhém kroku, jen ve čtyřech případech ve třetím kroku atd.

Dá se ukázat, že při této možnosti hledání, která je obohacena vlastně pouze tím, že připouští trojhodnotovou odpověď (proto trinární hledání), se průměrný počet kroků, které budeme potřebovat na nalezení libovolného čísla, zmenší pouze o jeden. Hledáme-li např. v rozsáhlém souboru, který má 131072 prvků (tj. 2^{13}), najde se libovolný prvek klasickým binárním hledáním nejdéle ve 13 krocích, uvedeným trinárním hledáním přibližně ve 12 krocích.

Program: Trinární hledání

```
 10 REM Trinarni hledani
 20 REM Pocitac hleda "trinarne"
 30 CLS
 40 PRINT "TRINARNI HLEDANI CISLA"
 50 PRINT "Budu hledat vami zadane cislo"
 60 PRINT "v intervalu  1 - 131 072."
 70 PRINT "Pri hledani vam nabidnu cislo."
 80 PRINT "Odpovidejte takto:"
 90 PRINT "M nebo < - je-li cislo MENSI"
100 PRINT "V nebo > - je-li cislo VETSI"
110 PRINT "S nebo = - je-li cislo STEJNE"
120 X=0
130 Y=131072
140 Z=0
150 Z=Z+1
160 C=INT((X+Y)/2)
170 IF C=X OR C=Y THEN 310
```

```
180 PRINT "Hadam:";C
190 PRINT "Jake je vase cislo k memu?"
200 INPUT Q$
210 IF LEN(Q$)<>1 THEN 250
220 IF Q$="S" OR Q$="=" THEN 340
230 IF Q$="V" OR Q$=">" THEN 270
240 IF Q$="M" OR Q$="<" THEN 290
250 PRINT "Odpovezte pouze S,V,M,<,> ,=!"
260 GOTO 180
270 X=C
280 GOTO 150
290 Y=C
300 GOTO 150
310 PRINT "Cislo se jiz nemeni."
320 PRINT "Zadal jste cislo mimo hranice!"
330 GOTO 360
340 PRINT
350 PRINT "Potreboval jsem";Z;"pokusu."
360 PRINT
370 PRINT "Konec."
380 END
```

Ukázka

```
TRINARNI HLEDANI CISLA
Budu hledat vami zadane cislo
v intervalu  1 - 131 072.
Pri hledani vam nabidnu cislo.
Odpovidejte takto:
M nebo < - je-li cislo MENSI
V nebo > - je-li cislo VETSI
S nebo = - je-li cislo STEJNE

Hadam: 65536
Jake je vase cislo k memu?
:M
Hadam: 32768
Jake je vase cislo k memu?
:V
Hadam: 49152
Jake je vase cislo k memu?
:V
Hadam: 57344
Jake je vase cislo k memu?
:M
Hadam: 53244
.
.
.
Hadam: 50000
Jake je vase cislo k memu?
```

```
:S
Potreboval jsem 8 pokusu."

Konec.
```

14.5 Hledání v nesetříděném souboru 1

Jiná situace nastává, jestliže se hledá v souboru, který není setříděný. Stejná situace může nastat také tehdy, jestliže se sice hledá v souboru, který uspořádaný je, ale podle jiného hlediska, než podle kterého vyhledáváme. Například, když se v abecedně uspořádaném souboru osob hledá podle data narození.

Ve všech těchto případech nezbývá, než procházet soubor celý. To je ovšem časově náročné, zejména tehdy, jestliže se jedná o rozsáhlý soubor.

Nejjednodušší postup by se dal ilustrovat následujícím programem. Pro ilustraci je zvolen příklad, v němž se v souboru prvočísel do 1000 hledá vložené číslo. Najde-li se, jde o prvočíslo, jestliže ne, jde o číslo složené. Pořadí prvočísel bylo po řádcích přeházeno, aby soubor byl neuspořádaný.

Program: Hledání v nesetříděném souboru 1

```
 10 REM Obecne hledani 1
 20 REM Nesetrideny soubor
 30 DIM P(168)
 40 CLS
 50 N=168
 60 FOR I=1 TO N
 70 READ P(I)
 80 NEXT I
 90 PRINT "Hledani v souboru prvocisel."
100 INPUT "Vlozte zkoumane cislo";X
110 IF INT(X)=X AND X > 0 THEN 140
120 PRINT "Vlozte PRIROZENE cislo!"
130 GOTO 100
140 IF X < 1000 THEN 170
150 PRINT "Cislo musi byt mensi nez  1000."
160 GOTO 100
170 I=0
180 I=I+1
```

```
190 IF X = P(I) THEN 230
200 IF I < N THEN 180
210 PRINT "Cislo";X;"NENI prvocislo."
220 GOTO 240
230 PRINT "Cislo";X;"JE prvocislo."
240 PRINT "Konec vypoctu."
250 DATA 73,79,83,89,97,101,103,107
260 DATA 523,541,547,557,563,569,571
270 DATA 479,487,491,499,503,509,521
280 DATA 191,193,197,199,211,223,227
290 DATA 151,157,163,167,173,179,181
300 DATA 709,719,727,733,739,743,751
310 DATA 659,661,673,677,683,691,701
320 DATA 37,41,43,47,53,59,61,67,71
330 DATA 2,3,5,7,11,13,17,19,23,29,31
340 DATA 229,233,239,241,251,257,263
350 DATA 269,271,277,281,283,293,307
360 DATA 617,619,631,641,643,647,653
370 DATA 577,587,593,599,601,607,613
380 DATA 811,821,823,827,829,839,853
390 DATA 757,761,769,773,787,797,809
400 DATA 353,359,367,373,379,383,389
410 DATA 311,313,317,331,337,347,349
420 DATA 907,911,919,929,937,941,947
430 DATA 857,859,863,877,881,883,887
440 DATA 439,443,449,457,461,463,467
450 DATA 397,401,409,419,421,431,433
460 DATA 953,967,971,977,983,991,997
470 DATA 109,113,127,131,137,139,149
480 END
```

Příklad

```
Hledaní v souboru prvocisel.
Vlozte zkoumane cislo: 999
Cislo 999 NENI prvocislo."
```

V programu se prochází daným souborem postupně od prvního prvku. Zakončení může být dvojí. Buď se vložený prvek rovná některé hodnotě ze souboru (v případě hledání podle jména nastane shoda mezi vloženým jménem a jménem v souboru, a pak o tom uživatel dostane informaci. Jestliže se ale projde bez úspěchu celý soubor, pak se dané číslo v souboru nevyskytuje a také o tom budete informováni.

V této situaci je zajímavé si uvědomit, že takto realizované hledání v neuspořádaném souboru prochází v každém kroku cyklu dvojím porovnáváním. Jednak je to porovnávání hledané-

ho prvku s prvkem daného souboru a jednak je to ještě porovnávání uskutečňované přímo cyklem, související s tím, že je potřeba poznat konec souboru dat.

14.6 Hledání v nesetříděném souboru 2

V teorii programování se pro tento případ uvádí úspornější postup, který by se uskutečnil následujícím programem.

Program: Hledání v nesetříděném souboru 2

```
 10 REM Obecné hledani 2
 20 REM Neusporadaný soubor
 30 DIM P(169)
 40 CLS
 50 N=168
 60 FOR I=1 TO N
 70 READ P(I)
 80 NEXT I
 90 PRINT "Hledaní v souboru prvocisel II."
100 INPUT "Vlozte zkoumane cislo";X
110 IF INT(X)=X AND X > 0 THEN 140
120 PRINT "Vlozte PRIROZENE cislo!"
130 GOTO 100
140 IF X < 1000 THEN 170
150 PRINT "Cislo musi byt mensi nez  1000."
160 GOTO 100
170 P(N+1)=X
180 I=0
190 I=I+1
200 IF X = P(I) THEN 220
210 GOTO 190
220 IF I = N+1 THEN 250
230 PRINT "Cislo";X;"JE prvocislo."
240 GOTO 260
250 PRINT "Cislo";X;"NENI prvocislo."
260 PRINT "Konec vypoctu."
270 DATA 479,487,491,499,503,509,521
280 DATA 191,193,197,199,211,223,227
290 DATA 151,157,163,167,173,179,181
300 DATA 709,719,727,733,739,743,751
310 DATA 659,661,673,677,683,691,701
320 DATA 37,41,43,47,53,59,61,67,71
330 DATA 2,3,5,7,11,13,17,19,23,29,31
340 DATA 229,233,239,241,251,257,263
350 DATA 269,271,277,281,283,293,307
360 DATA 617,619,631,641,643,647,653
370 DATA 577,587,593,599,601,607,613
380 DATA 811,821,823,827,829,839,853
```

```
390 DATA 757,761,769,773,787,797,809
400 DATA 353,359,367,373,379,383,389
410 DATA 311,313,317,331,337,347,349
420 DATA 907,911,919,929,937,941,947
430 DATA 857,859,863,877,881,883,887
440 DATA 439,443,449,457,461,463,467
450 DATA 397,401,409,419,421,431,433
460 DATA 953,967,971,977,983,991,997
470 DATA 109,113,127,131,137,139,149
480 DATA 73,79,83,89,97,101,103,107
490 DATA 523,541,547,557,563,569,571
500 END
```

Příklad

```
Hledaní v souboru prvocisel II.
Vlozte zkoumane cislo: 997
Cislo 997 JE prvocislo."
```

Úspora tohoto algoritmu pro hledání v neuspořádaném souboru spočívá v tom, že se o hledaný prvek rozšíří daný soubor. Hledaný prvek se umístí na poslední místo v daném souboru. Tím se zaručí, že se daný prvek v souboru zcela určitě najde. To ale znamená, že není nutné hlídat ukončení cyklu, protože hledání musí v každém případě skončit úspěšně a tím se program dostane z cyklu ven.

Po úspěšném nalezení prvku je ještě třeba zjistit, na kterém místě se nalezený prvek našel. Jestliže je to na místě N+1, tedy na posledním místě rozšířeného souboru, pak to znamená, že vlastně daný prvek v souboru není. V opačném případě v souboru je. V algoritmu je jenom jediné porovnávání, a to s hodnotou hledaného prvku. Tím se vlastní proces hledání urychluje o polovinu času ve srovnání s postupem předcházejícím.

15 Třídění

15.1 Třídění hledáním maximálního prvku

Třídění patří k operacím, které je potřeba vykonávat poměrně často. V moderních programových systémech existují přímo prostředky pro třídění libovolných souborů, často i několikastupňové. Jestliže ovšem daný systém či programovací jazyk takové možnosti nemá, je nutné třídění programovat.

Uvedeme zde několik postupů a začneme postupem, který je jednoduchý pro vysvětlování. Z předchozích programů již známe postupy vedoucí k nalezení extrémních prvků (minimálního či maximálního). Tento postup je možné pro třídění uplatnit tím způsobem, že v daném souboru nalezneme například minimální prvek a vyměníme jej s prvním prvkem souboru, třídíme-li vzestupně. Pro soubor zmenšený o první prvek budeme hledat další minimum, které zaměníme s druhým prvkem souboru atd.

Program: Třídění hledáním minimálního prvku

```
 10 REM Trideni I
 20 DIM A(15)
 30 CLS
 40 PRINT "Trideni I."
 50 N=15
 60 DATA 9,3,8,11,2,16,5,7,12,10,13,4,16,3,8
 70 FOR I=1 TO N
 80 READ A(I)
 90 NEXT I
100 FOR J=1 TO N-1
110 M=A(J)
120 K=J
130 FOR I=J+1 TO N
140 IF A(I) >= M THEN 170
150 M=A(I)
160 K=I
170 NEXT I
180 L=A(K)
190 A(K)=A(J)
200 A(J)=L
210 NEXT J
220 PRINT "Setrideny soubor."
```

```
230 FOR I=1 TO N
240 PRINT A(I);
250 NEXT I
260 END
```

Příklad

```
Trideni I.
Setrideny soubor.
2 3 3 4 5 7 8 8 9 10 11 12 13 16 16
```

V programu není zvlášť ošetřen případ, kdy by se minimální prvek nacházel právě na místě J-1 a kdy by nebylo nutné přehazování prvků. Takto se může stát, že se vlastně prohodí prvek se sebou samým.

15.2 Třídění "probubláváním"

Mezi efektivnější třídící postupy patří algoritmus, který se označuje jako "probublávání". Název, který používá analogie s procesem, který se při tomto způsobu třídění uskutečňuje, naznačuje, že prvky se postupně dostávají na svá místa, jako když lehčí bublinky kysličníku uhličitého vystupují k hladině limonády.

Realizace tohoto principu je velmi jednoduchá. Porovnávají se vždy dvojice sousedních prvků v souboru. Jestliže splňují podmínku na vzájemnou velikost podle daného uspořádání, zůstanou na svých místech. Jestliže je jejich velikost opačná, než by pro uspořádaný soubor měla být, vzájemně se prohodí. Proces se musí opakovat tak dlouho, až jsou všechny prvky na svých místech. To se pozná podle toho, že se v předchozím kroku žádná dvojice prvků neprohazovala.

Program: Třídění "probubláváním"

```
 10 REM Trideni II
 20 DIM A(15)
 30 CLS
 40 PRINT "Trideni II."
 50 N=15
 60 DATA 9,3,8,11,2,16,5,7,12,10,13,4,16,3,8
 70 FOR I=1 TO N
 80 READ A(I)
 90 NEXT I
100 K=0
110 FOR I=2 TO N
120 IF A(I) >= A(I-1) THEN 170
130 L=A(I)
140 A(I)=A(I-1)
150 A(I-1)=L
160 K=1
170 NEXT I
180 IF K=1 THEN 100
190 PRINT "Setrideny soubor."
200 FOR I=1 TO N
210 PRINT A(I);
220 NEXT I
230 END
```

Příklad

```
Trideni II.
Setrideny soubor.
2 3 3 4 5 7 8 8 9 10 11 12 13 16 16
```

Výsledek je samozřejmě stejný.

Podle toho, jak byl uspořádán výchozí soubor, postupuje rychle i toto třídění. Jestliže je hned od začátku setříděný, nedojde k žádnému přehazování prvků a hned po prvním kroku se zjistí, že soubor je setříděný. Třídící proces trvá nejdéle tehdy, byl-li původní soubor setříděn právě opačným způsobem, než je požadováno.

Protože je program napsán tak, že prochází od nižších indexů k vyšším, "probublá" největší prvek již při prvním průchodu na své místo na konci souboru. Naopak prvky, které se přemisťují dolů, se v každém kroku přemístí vždy jen o jed-

no místo. Tuto nevýhodu nemá postup uvedený v následujícím programu.

15.3 Třídění "obousměrným probubláváním"

Jestliže při průchodu souborem se podle předchozího postupu dostávají rychle velké prvky na své místo, můžeme pomocí cyklu, který by postupoval souborem opačně, urychlit umísťování malých prvků.

Takový postup bychom mohli nazvat "obousměrným probubláváním".

Program: Třídění "obousměrným probubláváním"

```
 10 REM Trideni III
 20 DIM A(15)
 30 CLS
 40 PRINT "Trideni III."
 50 N=15
 60 DATA 9,3,8,11,2,16,5,7,12,10,13,4,16,3,8
 70 FOR I=1 TO N
 80 READ A(I)
 90 NEXT I
100 D=2
110 H=N
120 K=0
130 FOR I=D TO H
140 IF A(I) >= A(I-1) THEN 190
150 L=A(I)
160 A(I)=A(I-1)
170 A(I-1)=L
180 K=1
190 NEXT I
200 IF K=0 THEN 330
210 H=H-1
220 K=0
230 FOR I=H TO D STEP -1
240 IF A(I) >= A(I-1) THEN 290
250 L=A(I)
260 A(I)=A(I-1)
270 A(I-1)=L
280 K=1
290 NEXT I
300 IF K=0 THEN 330
310 D=D+1
320 GOTO 120
```

```
330 PRINT "Setrideny soubor."
340 FOR I=1 TO N
350 PRINT A(I);
360 NEXT I
370 END
```

Příklad se stejným zadáním by vedl ke shodnému výsledku.

Výsledek je stejný. Rychlost třídění, zejména u velkých a neuspořádaných souborů, je však vyšší. Postup by se dal ještě dále zrychlit. Existují ještě další metody pro třídění.

16 Faktoriály a kombinační čísla

Výpočty faktoriálů a kombinačních čísel patří k látce, která se probírá na středních školách v několika ročnících. Jedná se zejména o gymnázia, ale učí se i na dalších školách.

Přitom hodnoty faktoriálů i kombinačních čísel velice rychle rostou. To znesnadňuje výpočty a proto jistě patří mezi takové, které je vhodné provádět na počítači. I na počítačích nemusí být výpočty faktoriálů a kombinačních čísel snadnou záležitostí.

Přestože z charakteru faktoriálů i kombinačních čísel plyne, že se jedná o čísla přirozená (celá), nemusí být často snadno možné dostat výsledek na počítači v celých číslech. Velká čísla se totiž v počítači běžně zobrazují jako čísla reálná a často přináší zmenšení přesnosti. U některých řad je zase velmi obtížné určit přesněji vyšší členy řady přímým výpočtem, ale je nutné výpočet provést postupně od nižších čísel k vyšším.

16.1 Výpočet faktoriálu

Faktoriál přirozeného čísla je součin všech přirozených čísel počínaje jedničkou až do daného čísla. Výjimku tvoří faktoriál nuly, který se rovná jedné.

Program: Faktoriál

```
10 REM Faktorial 1
20 REM Vypocet nasobenim
30 CLS
40 PRINT "Vypocet faktorialu."
50 PRINT
60 INPUT "Vlozte cislo pro faktorial";N
70 N=ABS(N)
80 N=INT(N)
90 IF N < 34 THEN 120
```

```
100 PRINT "Pro N < 33 nemohu N! vypocitat."
110 GOTO 230
120 IF N < 10 THEN 140
130 PRINT "Pro N > 9 bude N! v semilog.tvaru!"
140 F=1
150 IF N < 2 THEN 200
160 REM Vypocet N!
170 FOR I=2 TO N
180 F=F*I
190 NEXT I
200 PRINT "Faktorial";N;"=";F
210 PRINT
220 PRINT "Konec programu."
230 END
```

Příklad

```
Vypocet faktorialu.
Vlozte cislo pro faktorial: 6
Faktorial 6 = 720

Konec programu.
```

16.2 Výpočet velkého faktoriálu

Praktický problém při výpočtu faktoriálů spočívá v tom, že rostou velice rychle. Obvykle brzy je nemůžeme zobrazit jako celé číslo. V semilogaritmickém tvaru pak již nedávají přesný výsledek, ačkoliv již z jejich definice plyne, že to jsou celá čísla. Rovněž na ručních kalkulačkách dostanete hodnotu čísla faktoriál v semilogaritmickém tvaru, navíc málokdy pro číslo převyšující 70.

Program, který si nyní pro výpočet faktoriálu uvedeme, obsahuje obě možnosti řešení. Jednak obvyklý postup, který pro velkou hodnotu faktoriálu připouští zobrazení v semilogaritmickém tvaru, a jednak postup zobrazující výsledný faktoriál ve velké textové proměné. Právě možnost výpočtu velkých čísel v textové proměnné je hlavním smyslem těchto ukázek.

Program: Faktoriál v textové proměnné

```
 10 REM Faktorial 2
 20 REM Vypocet v textove promenne
 30 CLEAR 3000
 40 REM Prostor pro textove promenne
 50 CLS
 60 PRINT "Vypocet velkeho faktorialu."
 70 PRINT
 80 INPUT "Vlozte cislo pro faktorial";N
 90 N=ABS(N)
100 N=INT(N)
110 IF N < 34 THEN 140
120 PRINT "Pro N > 33 nemohu N! vypocitat."
130 GOTO 80
140 IF N < 10 THEN 180
150 PRINT "Pro N > 9 bude N! v semilog.tvaru!"
160 INPUT "Chcete celociselny vysledek?";Q$
170 IF Q$="A" THEN 280
180 IF N > 1 THEN 210
190 PRINT "Faktorial";N;"= 1"
200 GOTO 650
210 REM Vypocet N!
220 F=1
230 FOR I=2 TO N
240 F=F*I
250 NEXT I
260 PRINT "Faktorial";N;"=";F
270 GOTO 680
280 REM Vypocet v textove promenne
290 A$="1"
300 FOR I=1 TO 9
310 A$=A$+"**********"
320 NEXT I
330 FOR I=2 TO N
340 L=0
350 D=0
360 B$=""
370 L=L+1
380 IF L=101 THEN 670
390 IF MID$(A$,L,1)="*" THEN 490
400 A=VAL(A$(L,1))
410 C=I*A+D
420 IF C > 9 THEN 450
430 D=0
440 GOTO 470
450 D=INT(C/10)
460 C=C - 10*D
470 B$=B$+STR$(C)
480 GOTO 370
490 IF D=0 THEN 550
500 C=D - 10*(INT(D/10))
510 B$=B$+STR$(C)
520 D=INT(D/10)
```

```
530 L=L+1
540 GOTO 490
550 Z=LEN(B$)
560 FOR J=Z+1 TO 100
570 B$=B$+"*"
580 NEXT J
590 A$=B$
600 NEXT I
610 PRINT N;"! = ";
620 FOR I=100 TO 1 STEP -1
630 IF MID$(A$,I,1) <> "*" THEN PRINT MID$(A$,I,1);
640 NEXT I
650 PRINT
660 GOTO 680
670 PRINT "Cislo";N;"je prilis velke."
680 PRINT "Konec programu."
690 END
```

Příklad

```
Vypocet velkeho faktorialu.
Vlozte cislo pro faktorial: 30
Pro N > 9 bude N! v semilog.tvaru!
Chcete celociselny vysledek?
:A
 30 ! = 265252859812191058636308480000000
Konec programu.
```

Klíčem k výpočtu velkých celých čísel je jejich zaznamenávání ve velké textové proměnné.

Při procesu výpočtu je třeba číslice zpracovávat postupně od posledních řádů uložených v textové proměnné A$ a transformovat je na číslo, tj. násobit příslušným násobitelem - I - a pak uložit do textové proměnné B$. Při zpětném ukládání je třeba rozlišit případ jedno a víceciferné hodnoty čísla a při víceciferných připočítat vyšší řády k příštímu násobku.

Musí se hlídat počet číslic dosud vypočteného čísla v textové proměnné. Po každém násobení pak přesunout čísla mezi textovými proměnnými a proces opakovat tak dlouho, jak je to zapotřebí. V daném případě se velké číslo ukládá do textové proměnné v obráceném pořadí číslic. To poněkud usnadňuje prá-

ci s textovou proměnnou. Při výstupu výsledku je pak samozřejmě nutné vypisovat číslo z textové proměnné pozpátku.

16.3 Permutace

Faktoriál udává, jaký je počet možných uspořádání prvků. Nedává však odpověď na otázku, jaká jsou jednotlivá uspořádání.

Pro malý počet prvků a pro nalezení všech jejich permutací to ještě nemusí být zvlášť obtížný problém. Situace se komplikuje, jestliže je počet prvků permutace větší - o tom ostatně podává základní informaci samotná hodnota daného faktoriálu. Ještě obtížnější je zjistit danou konkrétní permutaci.

Předpokládejme, že jednotlivé permutace jsou uspořádány lexikograficky. To znamená, že jsou uspořádány podle velikosti čísla, které představují, jsou-li tvořeny číslicemi. Když jsou tvořeny písmeny, pak lexikografické uspořádání odpovídá uspořádání abecednímu.

Program, který následuje, dokáže provést dvě věci. Zadáme-li pořadové číslo permutace, vytvoří danou permutaci samu. Naopak, zadáme-li některou konkrétní permutaci, zjistí, jaké má pořadové číslo.

Program: Permutace

```
 10 REM Permutace
 20 REM Permutace <-> pořadí
 30 CLEAR
 40 DIM F(15),K(15),N(15)
 50 CLS
 60 PRINT "         PERMUTACE"
 70 PRINT
 80 PRINT "Vypocet PORADI dane permutace a"
 90 PRINT "vypocet PERMUTACE daneho poradi."
100 PRINT
110 F(1)=1
120 FOR I=2 TO 15
130 F(I)=I*F(I-1)
```

```
140 NEXT I
150 INPUT "Vlozte pocet prvku permutace";B
160 B=INT(B)
170 IF B > 0 AND B < 16 THEN 200
180 PRINT "Maximalne 15 prvku!"
190 GOTO 150
200 C=B-1
210 S=C
220 D=B+1
230 IF B > 10 THEN 380
240 Z$="0123456789"
250 T$="Je permutace tvorena cislicemi?"
260 GOSUB 1080
270 IF Q=0 THEN 340
280 IF B=10 THEN 350
290 T$="Je mezi cislicemi NULA?"
300 GOSUB 1080
310 IF Q=1 THEN 350
320 Z$="123456789"
330 GOTO 350
340 Z$="ABCDEFGHIJKLMNO"
350 REM Z poradoveho cisla permutace
360 T$="Chcete k porad.cislu permutaci?"
370 GOSUB 1080
380 IF Q=0 THEN 730
390 INPUT "Vlozte PORADOVE CISLO permutace";X
400 X=INT(X)
410 IF X > 0 AND X <= F(B) THEN 440
420 PRINT "Poradove cislo musi byt 1 -";F(B)
430 GOTO 390
440 A=X-1
450 P$=MID$(Z$,1,B)+" "
460 FOR I=1 TO 15
470 N(I)=1
480 K(I)=0
490 NEXT I
500 V$=""
510 FOR J=C TO 1 STEP -1
520 IF A < F(J) THEN 560
530 A=A-F(J)
540 N(J)=N(J)+1
550 GOTO 520
560 NEXT J
570 FOR J=C TO 1 STEP -1
580 M=N(J)
590 DV=LEN(V$)
600 IF B-J > 1 THEN V$=LEFT$(V$,B-J-1)
610 V$=V$+MID$(P$,M,1)
620 IF DV-B > 1 THEN V$=V$+RIGH$(V$,DV-B+J
630 DP=LEN(P$)
640 IF M > 1 THEN P$=LEFT$(P$,M-1)+MID$(P$,M+1,DP-M)+" "
650 IF M=1 THEN P$=MID$(P$,M+1,DP-M)+" "
660 NEXT J
670 IF B > 1 THEN V$=LEFT$(V$,B-1)
680 V$=V$+LEFT$(P$,1)
```

```
690 IF DV-B > 0 THEN V$=V$+RIGHT$(V$,DV-B)
700 PRINT "Poradovemu cislu";X;"z";B;"prvku"
710 PRINT "patri permutace: ";LEFT$(V$,B)
720 GOTO 1050
730 REM Z permutace poradove cislo
740 T$="Chcete k permutaci porad.cislo?"
750 GOSUB 1080
760 IF Q=0 THEN 1050
770 P$=LEFT$(Z$,B)+" "
780 FOR I=1 TO 15
790 N(I)=1
800 NEXT I
810 INPUT "Vlozte permutaci";V$
820 FOR I=1 TO B-1
830 FOR J=I+1 TO B
840 IF MID$(V$,I,1) <> MID$(V$,J,1) THEN 870
850 PRINT "Zadny znak se nesmi opakovat!"
860 GOTO 810
870 NEXT J
880 NEXT I
890 FOR I=1 TO B
900 FOR J=1 TO B
910 IF MID$(V$,I,1) <> MID$(P$,J,1) THEN 970
920 DP=LEN(P$)
930 N(I)=J-1
940 IF J > 1 THEN P$=LEFT$(P$,J-1)+MID$(P$,J+1,DP-1)+" "
950 IF J=1 THEN P$=MID$(P$,J+1,DP-J)+" "
960 GOTO 980
970 NEXT J
980 NEXT I
990 P=1
1000 FOR I=2 TO B
1010 P=P+N(I-1)*F(D-I)
1020 NEXT I
1030 PRINT "Permutace ";LEFT$(V$,B);" z";B;"prvku"
1040 PRINT "ma poradove cislo";P
1050 PRINT
1060 PRINT "Konec."
1070 END
1080 REM Souhlas
1090 PRINT T$
1100 Q$=INKEY$
1110 IF Q$="" THEN 1100
1120 Q=1
1130 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
1140 Q=0
1150 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
1160 PRINT "Odpovidejte takto:"
1170 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
1180 GOTO 1080
```

Ukázka 1

```
        PERMUTACE

Vypocet PORADI dane permutace a
vypocet PERMUTACE daneho poradi.

Vlozte pocet prvku permutace: 10
Je permutace tvorena cislicemi?
A
Chcete k porad.cislu permutaci?
A
Vlozte PORADOVE CISLO permutace: 1000000

Poradovemu cislu 1000000 z 10 prvku
patri permutace: 2783915460

Konec.
```

Ukázka 2

```
        PERMUTACE

Vypocet PORADI dane permutace a
vypocet PERMUTACE daneho poradi.

Vlozte pocet prvku permutace: 10
Je permutace tvorena cislicemi?
A
Chcete k porad.cislu permutaci?
N
Vlozte permutaci: 2143658709

Permutace 2143658709 z 10 prvku
ma poradove cislo 777159

Konec.
```

16.4 Kombinační čísla

Také kombinační čísla patří k látce, která se učí na středních školách. I pro tato čísla platí, že princip jejich výpočtu je sice snadný, ale že i kombinační čísla rychle rostou. Tím nejenom vzrůstá výpočetní náročnost, ale může se

stát, že ani na běžných počítačích nemůžeme snadno vypočítat kombinační číslo jako číslo celé.

Uvedeme si nejprve program, který tuto úlohu řeší přímým výpočtem, tj. násobením a dělením, a poté program, který umožní vypočítat i velkou hodnotu kombinačního čísla v celočíselné podobě.

Program: Kombinační číslo

```
 10 REM N nad M
 20 REM Vypocet kombinacniho cisla
 30 CLS
 40 PRINT "KOMBINACNI CISLO 'N nad M'"
 50 PRINT
 60 PRINT "Vypocet kombinacniho cisla"
 70 PRINT "'N nad M' pomoci faktorialu"
 80 PRINT "a jejich delenim."
 90 PRINT
100 PRINT "Vlozte cislo N";
110 INPUT N
120 N=INT(N)
130 IF N >= 0 THEN 160
140 PRINT "Musite zadat prirozene cislo!"
150 GOTO 100
160 IF N <= 34 THEN 190
170 PRINT "Pro N > 33 nemohu vypocet provest!"
180 GOTO 400
190 PRINT "Vlozte cislo M";
200 INPUT M
210 M=INT(M)
220 IF N >= 0 THEN 250
230 PRINT "Musite zadat prirozene cislo!"
240 GOTO 190
250 IF M <= N THEN 280
260 PRINT "Cislo M nemuze byt vetsi nez N!"
270 GOTO 190
280 REM Vypocet
290 Z=N
300 GOSUB 430
310 V=F
320 Z=M
330 GOSUB 430
340 V=V/F
350 Z=N-M
360 GOSUB 430
370 V=V/F
380 PRINT
390 PRINT "Kombinacni cislo";N;"nad";M;"je";V
400 PRINT
410 PRINT "Konec programu."
```

```
420 END
430 REM Vypocet faktorialu
440 F=1
450 IF Z < 2 THEN RETURN
460 FOR I=2 TO Z
470 F=F*I
480 NEXT I
490 RETURN
```

Ukázka

```
KOMBINACNI CISLO 'N nad M'

Vypocet kombinacniho cisla
'N nad M' pomoci faktorialu
a jejich delenim.

Vlozte cislo N: 10
Vlozte cislo M: 5

Kombinacni cislo 10 nad 5 je 252

Konec programu.
```

Ukázka

```
KOMBINACNI CISLO 'N nad M'

Vypocet kombinacniho cisla
'N nad M' pomoci faktorialu
a jejich delenim.


Vlozte cislo N:
10
Vložte číslo M:

5
Kombinační číslo 10 nad 5 je 252
Konec programu.
```

16.5 Velká kombinační čísla

Další program vznikl rozšířením programu předcházejícího. Umožňuje vypočítat také velké kombinační číslo, které by se jinak vypočítalo pouze v semilogaritmickém tvaru.

Program: Velké kombinační číslo

```
 10 REM N nad M v textove promenne
 20 CLEAR 3000
 30 CLS
 40 PRINT "KOMBINACNI CISLO 'N nad M'"
 50 PRINT
 60 PRINT "Vypocet kombinacniho cisla"
 70 PRINT "'N nad M'  pomoci velke"
 80 PRINT "textove promenne."
 90 PRINT
100 PRINT "Vlozte cislo N";
110 INPUT N
120 N=INT(N)
130 IF N >= 0 THEN 160
140 PRINT "Musite zadat prirozene cislo!"
150 GOTO 100
160 IF N <= 34 THEN 190
170 PRINT "Pro N > 33 nemohu vypocet provest!"
180 GOTO 400
190 PRINT "Vlozte cislo M";
200 INPUT M
210 M=INT(M)
220 IF N >= 0 THEN 250
230 PRINT "Musite zadat cele nezaporne cislo!"
240 GOTO 190
250 IF M <= N THEN 280
260 PRINT "Cislo M nemuze byt vetsi nez N!"
270 GOTO 190
280 REM Vypocet
290 Z=N
300 GOSUB 1180
310 V=F
320 Z=M
330 GOSUB 1180
340 V=V/F
350 Z=N-M
360 GOSUB 1180
370 V=V/F
380 PRINT
390 PRINT "Kombinacni cislo";N;"nad";M;"je";V
400 REM Vypocet v textove promenne
410 PRINT
420 INPUT "Chcete N nad M celociselne (A/N)";Q$
430 IF Q$="N" OR Q$="0" THEN 1160
440 REM Nasobeni (citatel)
```

```
450 A$="1"
460 FOR I=1 TO 9
470 A$=A$+"**********"
490 FOR I=N-M+1 TO N
500 L=0
510 D=0
520 B$=""
530 L=L+1
540 IF L > 100 THEN 1150
550 IF MID$(A$,L,1)="*" THEN 650
560 A=VAL(A$(L,1))
570 C=I*A+D
580 IF C > 9 THEN 610
590 D=0
600 GOTO 630
610 D=INT(C/10)
620 C=C - 10*D
630 B$=B$+STR$(C)
640 GOTO 530
650 IF D=0 THEN 710
660 C=D - 10*(INT(D/10))
670 B$=B$+STR$(C)
680 D=INT(D/10)
690 L=L+1
700 GOTO 650
710 Z=LEN(B$)
720 FOR J=Z+1 TO 100
730 B$=B$+"*"
740 NEXT J
750 A$=B$
760 NEXT I
770 REM Deleni (jmenovatel)
780 IF M < 2 THEN 1060
790 B$=""
800 FOR I=1 TO 100
810 IF MID$(A$,I,1)="*" THEN 830
820 NEXT I
830 J=I
840 FOR I=1 TO 100
850 IF I < J THEN B$=B$+MID$(A$,J-I,1)
860 IF I >= J THEN B$=B$+"*"
870 NEXT I
880 FOR I=2 TO M
890 A$=B$+"*****"
900 B$=""
910 D=0
920 L=1
930 IF MID$(A$,L,1)="*" THEN 1070
940 D=10*D+VAL(MID$(A$,L,1))
950 IF D >= I THEN 1010
960 L=L+1
970 IF L > 100 THEN 1070
980 GOTO 930
990 IF MID$(A$,L,1)="*" THEN 1070
```

```
1000 D=10*D+VAL(MID$(A$,L,1))
1010 C=INT(D/I)
1020 B$=B$+STR$(C)
1030 D=D-C*I
1040 L=L+1
1050 IF L > 100 THEN 1070
1060 GOTO 990
1070 NEXT I
1080 PRINT
1090 PRINT N;"nad";M;"= ";
1100 FOR I=1 TO 100
1110 IF MID$(B$,I,1) <> "*" THEN PRINT MID$(B$,I,1);
1120 NEXT I
1130 PRINT
1140 GOTO 1160
1150 PRINT "Cislo";N;"je prilis velke!"
1160 PRINT "Konec programu."
1170 END
1180 REM Vypocet faktorialu
1190 F=1
1200 IF Z < 2 THEN RETURN
1210 FOR I=2 TO Z
1220 F=F*I
1230 NEXT I
1240 RETURN
```

Ukázka

```
KOMBINACNI CISLO 'N nad M'

Vypocet kombinacniho cisla
'N nad M'  pomoci velke
textove promenne.

Vlozte cislo N: 30
Vlozte cislo M: 7

Kombinacni cislo 10 nad 5 je 2035800

Konec programu.
```

16.6 Největší společný dělitel

Při výpočtu velkého kombinačního čísla je možné postupovat také jinak. Vytvořit si čísla, jejichž násobením vzniká čitatel i jmenovatel pro výpočet kombinačního čísla. Tato

čísla pak mezi sebou dělit, pokud to je možné, a teprve poté přistoupit k jejich násobení v textové proměnné.

K tomu může být užitečný program pro hledání největšího společného dělitele. V programu je použit známý Euklidův algoritmus, který je velice efektivní.

Program: Největší společný dělitel

```
 10 REM Delitel
 20 CLS
 30 PRINT "Nejvetsi spolecny delitel."
 40 INPUT "Vlozte PRVNI cislo";A
 50 INPUT "Vlozte DRUHE cislo";B
 60 IF A <> B THEN 90
 70 PRINT "Obe cisla jsou stejna";A
 80 GOTO 170
 90 REM Vypocet
100 IF A < B THEN 140
110 C=A
120 A=B
130 B=C
140 C=A - B*INT(A/B)
150 IF C <> 0 THEN 120
160 PRINT "Nejvetsi spolecny delitel je";B
170 PRINT "Konec vypoctu."
180 END
```

Příklad

```
Nejvetsi spolecny delitel.
Vlozte PRVNI cislo: 28220
Vlozte DRUHE cislo: 18018

Nejvetsi spolecny delitel je 546

Konec vypoctu
```

Program pro nalezení největšího společného dělitele může pracovat také samostatně. V této podobě je zde uveden. Jeho včlenění do programu pro výpočet velkých kombinačních čísel může být úkolem pro vás.

17 Statistické programy

V této části budou uvedeny některé jednoduché statistické programy.

17.1 Aritmetický průměr

Nejprve bude uveden program pro výpočet prostého aritmetického průměru, který je počítán z vkládaných dat.

Program: Aritmetický průměr

```
 10 REM Aritmeticky prumer
 20 CLS
 30 PRINT "Prosty aritmeticky prumer.
 40 P=0
 50 INPUT "Vite kolik bude prvku (A/N)";Q$
 60 IF Q$="A" THEN 140
 70 INPUT "Vlozte hodnotu konce napr. 99999";M
 80 N=0
 90 INPUT "Vlozte hodnotu prvku";A
100 IF A=M THEN 190
110 P=P+A
120 N=N+1
130 GOTO 90
140 INPUT "Vlozte pocet prvku";N
150 FOR I=1 TO N
160 INPUT "Vlozte hodnotu prvku";A
170 P=P+A
180 NEXT I
190 P=P/N
200 PRINT
210 PRINT "Aritmeticky prumer z";N;"prvku  je";P
220 END
```

Příklad nebudeme uvádět. Jeho výsledek závisí na vkládaných hodnotách.

17.2 Vážený aritmetický průměr

Při výpočtu váženého aritmetického průměru se hodnoty prvků násobí jejich vahami (počtem prvků dané velikosti) a vypočtený součet se pak dělí součtem vah. Vlastní program se formálně liší pouze v tom, že data musí vstupovat po dvojicích. Vždy hodnota prvku a jeho váha.

Program: Vážený aritmetický průměr

```
 10 REM Vazeny aritmeticky prumer
 20 CLS
 30 PRINT "Vazeny aritmeticky prumer.
 40 P=0
 50 V=0
 60 INPUT "Vite kolik bude prvku (A/N)";Q$
 70 IF Q$="A" THEN 150
 80 INPUT "Vlozte hodnotu konce napr. 99999";M
 90 INPUT "Vlozte hodnotu prvku";A
100 IF A=M THEN 220
110 P=P+A
120 INPUT "Vlozte vahu prvku";B
130 V=V+B
140 GOTO 90
150 INPUT "Vlozte pocet prvku";N
160 FOR I=1 TO N
170 INPUT "Vlozte hodnotu prvku";A
180 P=P+A
190 INPUT "Vlozte vahu prvku";B
200 V=V+B
210 NEXT I
220 P=P/V
230 PRINT
240 PRINT "Vazeny aritmeticky prumer je";
250 PRINT P
260 END
```

Také zde není příklad potřebný. Jeho výsledek závisí na vkládaných veličinách.

17.3 Rozptyl a směrodatná odchylka

Dalším programem bude ilustrován jednoduchý program pro výpočet rozptylu a směrodatné odchylky, rovněž ze vkládaných dat.

Program: Rozptyl a směrodatná odchylka

```
 10 REM Rozptyl
 20 CLS
 30 PRINT "Rozptyl a smerodatna odchylka"
 40 P=0
 50 R=0
 60 INPUT "Vite kolik je prvku (A/N)";Q$
 70 IF Q$="A" THEN 160
 80 INPUT "Vlozte hodnotu konce napr. 99999";M
 90 N=0
100 INPUT "Vlozte hodnotu prvku";A
110 IF A=M THEN 220
120 P=P+A
130 R=R+A*A
140 N=N+1
150 GOTO 100
160 INPUT "Vlozte pocet prvku";N
170 FOR I=1 TO N
180 INPUT "Vlozte hodnotu prvku";A
190 P=P+A
200 R=R+A*A
210 NEXT I
220 P=P/N
230 R=R/N
240 S=R-P
250 PRINT
260 PRINT "Pocet prvku je";N
270 PRINT "Aritmeticky prumer je";P
280 PRINT "Rozptyl je";S
290 PRINT "Smerodatna odchylka je";SQR(S)
300 END
```

Příklad nebudeme uvádět. Jeho výsledek závisí na vkládaných hodnotách.

Pro výpočet rozptylu a směrodatné odchylky zde byl použit postup, který nepotřebuje předem znát hodnotu průměru. To je výhodné, protože se celý výpočet dá provést bez ukládání jednotlivých dat.

17.4 Hledání extrémních prvků souboru

Princip postupu nalezení maximálního a minimálního prvku, pokud se hledají samostatně, je jednoduchý. Když se ale ve

stejném souboru hledá současně maximální i minimální prvek, pak se nemusí porovnávat všechny prvky s možným maximem i minimem. Postup může být úspornější.

Znalost obou krajních bodů souboru je často potřebná. Jejich rozdíl (maximum - minimum) se nazývá variační rozpětí. Z něj se dělá jednoduchý závěr o rozptýlení prvků souboru.

Jinou častou situací, kdy je znalost obou krajních hodnot potřebná, je informace před kreslením grafického výstupu. Pomocí krajních bodů se pak volí velikosti souřadnic i měřítko kreslených grafů.

Jestliže je možné stejný problém řešit programově různými postupy, přichází otázka, který postup je nejlepší? Nejčastěji se kvalita výpočetních postupů měří určením nezbytného počtu operací, které daný program nebo jeho část potřebuje. Může se uvažovat počet násobení, sčítání, porovnávání anebo jejich vhodný souhrn. V našem případě by měřítkem kvality byl nejlépe počet porovnání. Ukazuje se, že počet porovnání je nejmenší, budeme-li postupovat podle dále uvedeného programu.

Program je zde uveden proto, že postup v něm použitý je úspornější než běžné dvojí porovnávání, a také proto, že není příliš známý.

Program: Extrémní prvky

```
 10 REM Extremni prvky
 20 DIM A(20)
 30 CLS
 40 PRINT "EXTREMNI a MAXIMALNI prvek."
 50 N=15
 60 DATA 8,5,9,34,5,8,39,6,18,4,38,6,11,24,9
 70 FOR I=1 TO N
 80 READ A(I)
 90 NEXT I
100 D=A(N)
110 H=A(N)
120 FOR I=1 TO N-1 STEP 2
130 J=I+1
140 IF A(I) ≥ A(J) THEN 180
150 IF A(I) ≤ D THEN D=A(I)
```

```
160 IF A(J) > H THEN H=A(J)
170 GOTO 200
180 IF A(J) < D THEN D=A(J)
190 IF A(I) > H THEN H=A(I)
200 NEXT I
210 PRINT
220 PRINT "Maximalni prvek je";H
230 PRINT "Minimalni prvek je";D
240 END
```

Příklad pro data, která jsou součástí programu

```
EXTREMNI PRVKY SOUBORU

Maximalni prvek je 39
Minimalni prvek je 4
```

Program vychází z následujícího principu. Platí-li mezi dvěma prvky vztah X > Y, pak je zřejmé, že při hledání maximálního prvku stačí zabývat se pouze prvkem X a opomenout prvek Y, a naopak pro minimum je důležitý pouze prvek Y a X může být přeskočen. Dovedete-li tyto úvahy dál, můžete zjistit, že při použitém způsobu hledání může být vztah mezi dvěma prvky také X >= Y. Na počátku se oběma proměnným pro maximum (H) i minimum (D) přiřadí hodnota posledního prvku souboru. To proto, že se souborem postupuje po dvojicích prvků a je-li počet prvků lichý, byl by právě poslední prvek opomenut.

17.5 Statistické charakteristiky

Další program shrnuje možnost výpočtu základních statistických charakteristik. Umožňuje výpočet aritmetického průměru rozptylu i směrodatné odchylky.

Přitom pracuje tak, že vkládaná data ukládá do pomocného pracovního vektoru,aby dal možnost pro jejich kontrolu a případné opravy. V tomto směru vychází ze zkušeností uvedených v programech pro sčítání velké řady čísel.

Program: Statistické charakteristiky

```
 10 REM STATISTIKA
 20 REM Statisticke charakteristiky
 30 REM pocet, soucet, prumer
 40 REM rozptyl a smerodatna odchylka
 50 REM s opravovanim dat v pameti
 60 DIM A(100)
 70 CLS
 80 PRINT "PPROGRAM PRO STAT.CHARAKTERISTIKY"
 90 PRINT "do 100 hodnot s kontrolním"
100 PRINT "vypisem a opravovanim."
110 PRINT "Max.hodnota velicin je 100000."
120 PRINT
130 T$="Vite kolik je cisel?"
140 GOSUB 910
150 IF Q=0 THEN 280
160 INPUT "Vlozte pocet cisel";N
170 N=ABS(N)
180 N=INT(N)
190 IF N > 100 THEN 860
200 IF N > 1 THEN 230
210 PRINT "Cisla musi byt nejmene 2!"
220 GOTO 160
230 CLS
240 FOR I=1 TO N
250 GOSUB 1020
260 NEXT I
270 GOTO 370
280 REM Vkladani s poslední hodnotou
290 PRINT "Za poslednim cislem vlozte 99999!"
300 I=1
310 GOSUB 1020
320 IF R=99999 THEN 360
330 I=I+1
340 IF I < 101 THEN 310
350 GOTO 860
360 N=I-1
370 REM Kontrolni vypis
380 T$="Chcete kontrolni vypis?"
390 GOSUB 910
400 IF Q=0 THEN 690
410 T$="Muze se pokracovat ve vypise?"
420 GOSUB 1110
430 FOR J=1 TO N
440 IF J < 100 THEN PRINT " ";
450 IF J < 10 THEN PRINT " ";
460 PRINT J;" ";
470 B=A(J)
480 IF B < 100000 THEN PRINT " ";
490 IF B < 10000 THEN PRINT " ";
500 IF B < 1000 THEN PRINT " ";
510 IF B < 100 THEN PRINT " ";
520 IF B < 10 THEN PRINT " ";
530 PRINT B
```

```
540 IF J/10  ..  INT(J/10) THEN 610
550 GOSUB 910
560 IF Q=1 THEN 600
570 PRINT "Ktere por.cislo chcete opravit?"
580 INPUT I
590 GOSUB 1020
600 GOSUB 1110
610 NEXT J
620 T$="Jsou vsechna cisla spravna?"
630 GOSUB 910
640 IF Q=1 THEN 700
650 PRINT "Ktere por.cislo chcete opravit?"
660 INPUT I
670 GOSUB 1020
680 GOTO 380
690 REM Vypocty
700 S=0
710 R=0
720 FOR I=1 TO N
730 A=A(I)
740 S=S+A(I)
750 R=R+A*A
760 NEXT I
770 P=S/N
780 R=R/N-P*P
790 PRINT "Cisel je celkem";N
800 PRINT "Jejich soucet =";S
810 PRINT "Arit. prumer  =";P
820 PRINT "Rozptyl       =";R
830 PRINT "Smer.odchylka =";SQR(R)
840 GOTO 880
850 PRINT
860 PRINT "Cisel je vic nez 100!"
870 PRINT "Zvetsete A(100) v dimenzaci!"
880 PRINT
890 PRINT "Konec."
900 END
910 REM Souhlas
920 PRINT T$
930 Q$=INKEY$
940 IF Q$="" THEN 930
950 Q=1
960 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
970 Q=0
980 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
990 PRINT "Odpovidejte takto:"
1000 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
1010 GOTO 910
1020 REM Vstup hodnot
1030 PRINT "Jake je";I;". cislo";
1040 INPUT R
1050 IF R=99999 THEN RETURN
1060 IF ABS(R) <= 100000 THEN 1090
1070 PRINT "Cislo nesmi byt vetsi nez 100000!"
1080 GOTO 1030
```

```
1090 A(I)=R
1100 RETURN
1110 REM Hlavicka
1120 PRINT "  C. hodnota"
1130 RETURN
```

17.6 Lineární regrese

Posledním statistickým programem, který zde uvedeme, bude program pro lineární regresi. V programu se předpokládá vstup dvojic hodnot nezávisle a závisle proměnných veličin, z nichž se vypočítají koeficienty lineární funkce.

I v tomto případě se v programu uvažují dvě možnosti při vkládání hodnot, lišící se znalostí počtu dvojic hodnot před počátkem vkládaní, tak jako tomu bylo i u ostatních programů pro statistiku. Bez jakékoliv újmy je možné program upravit tak, že z těchto dvou možností bude jedna vypuštěna.

Program: Lineární regrese

```
 10 REM Linearni regrese
 20 REM Dve formy vstupu
 30 DIM A(99),B(99)
 40 CLS
 50 PRINT "Linearni regrese.
 60 PRINT
 70 INPUT "Vite kolik bude prvku (A/N)",Q$
 80 IF Q$="N" THEN 170
 90 INPUT "Vlozte pocet dvojic prvku",N
100 FOR I=1 TO N
110 PRINT I;"nezavisla";
120 INPUT A(I)
130 PRINT I;"zavisla";
140 INPUT B(I)
150 NEXT I
160 GOTO 270
170 INPUT "Vlozte hodnotu konce napr. 99999";N
180 I=0
190 I=I+1
200 PRINT I;"nezavisla";
210 INPUT A(I)
220 IF A(I)=N THEN 260
230 PRINT I;"zavisla";
240 INPUT B(I)
250 GOTO 190
260 N=I-1
```

```
270 U=0
280 V=0
290 X=0
300 Z=0
310 FOR I=1 TO N
320 U=U+A(I)
330 V=V+B(I)
340 X=X+A(I)*A(I)
350 Z=Z+A(I)*B(I)
360 NEXT I
370 B=(N*Z-U*V)/(N*X-U*U)
380 A=(V-B*U)/N
390 PRINT "Rovnice primky je:"
400 PRINT "Y  =";
410 IF A <> 0 THEN PRINT A;
420 ON SGN(B)+2 GOTO 430,480,450
430 PRINT "-";
440 GOTO 460
450 IF B <> 0 THEN PRINT "+";
460 IF ABS(B) <> 1 THEN PRINT ABS(B);"*";
470 PRINT " X"
480 PRINT
490 PRINT "Konec programu."
500 END
```

Příklad

```
Linearni regrese
Vite kolik bude prvku (A/N):
N
Vlozte hodnotu konce napr. 99999:
99999
1 nezavisla: 1
1 zavisla: 3
2 nezavisla: 2
2 zavisla: 5
3 nezavisla: 3
3 zavisla: 7
4 nezavisla: 99999

Rovnice primky je:
Y  =  1  +  2 * X
```

18 Programy pro lineární optimalizaci

V této části textu budou uvedeny některé jednoduché programy zejména z operačního výzkumu, ale také některé další. Byly upraveny a zjednodušeny tak, aby byly provozovatelné na počítači IQ 150.

18.1 Krátká simplexová metoda

Dále uvedený program obsahuje tzv. jednofázovou simplexovou metodu. Mohou se pomocí něj řešit obecné úlohy lineárního programování, jejichž všechna omezení jsou ve tvaru menší nebo se rovná.

Vstup se předpokládá v podobě výchozí simplexové tabulky, včetně jednotkové matice, koeficientů pravých stran a účelové funkce v anulovaném tvaru.

Program: Simplex I

```
  10 REM Simplex I
  20 DIM A(11,26),X(25)
  30 CLS
  40 PRINT "Simplexova metoda - kratka verze"
  50 PRINT "================================="
  60 PRINT "Slouzi pouze pro vypocet MAXIMA"
  70 PRINT "Max.pocet omezeni typu  .= je 10"
  80 PRINT "Max.pocet prom.vcetne baze je 25"
  90 PRINT "Vklada se cela tabulka:"
 100 PRINT "- omezeni jako rovnice"
 110 PRINT "- ke kazdemu prava strana"
 120 PRINT "- UF v anulovanem tvaru."
 130 PRINT "Nuly je take nutno vkladat!"
 140 PRINT
1000 REM Vstupy
1010 INPUT "Vlozte pocet omezeni";M
1020 M1=M+1
1030 INPUT "Vlozte pocet promennych";N
1040 N1=N+1
1050 FOR I=1 TO M1
1060 FOR J=1 TO N1
1070 A(I,J)=0
1080 IF I=M1 AND J=N1 THEN 3000
```

```
1090 IF J < N1 THEN 1120
1100 PRINT "Prava strana";I;
1110 GOTO 1160
1120 IF I < M1 THEN 1150
1130 PRINT "Ucelova funkce";J;
1140 GOTO 1160
1150 PRINT "Prvek";I;J;
1160 INPUT A(I,J)
1170 NEXT J
1180 NEXT I
3000 REM Test optima
3010 GOSUB 9000
3020 D=-99999
3030 FOR J=1 TO N
3040 IF A(M1,J) >= 0 THEN 3090
3050 IF ABS(A(M1,J)) < D THEN 3090
3060 S=J
3070 L=1
3080 D=ABS(A(M1,J))
3090 NEXT J
3100 IF L=1 THEN 4000
3110 CLS
3120 PRINT "Optimum"
3130 GOTO 8000
4000 REM Vystupujici
4010 P=99999
4020 FOR I=1 TO M
4030 IF A(I,S)  .= 0 THEN 4080
4040 F=A(I,N+1)/A(I,S)
4050 IF F >= P THEN 4080
4060 R=I
4070 P=F
4080 NEXT I
4090 IF R <> 0 THEN 5000
4100 PRINT "Neomezeno"
4110 GOTO 8220
5000 REM T. - klicovy radek
5010 IF S=1 THEN 5050
5020 FOR J=1 TO S-1
5030 A(R,J)=A(R,J)/A(R,S)
5040 NEXT J
5050 FOR J=S+1 TO N1
5060 A(R,J)=A(R,J)/A(R,S)
5070 NEXT J
6000 REM T. - tabulka
6010 IF R=1 THEN 6080
6020 IF S=1 THEN 6150
6030 FOR I=1 TO R-1
6040 FOR J=1 TO S-1
6050 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6060 NEXT J
6070 NEXT I
6080 IF S=1 THEN 6200
6090 FOR I=R+1 TO M1
6100 FOR J=1 TO S-1
```

```
6110 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6120 NEXT J
6130 NEXT I
6140 IF R=1 THEN 6200
6150 FOR I=1 TO R-1
6160 FOR J=S+1 TO N1
6170 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6180 NEXT J
6190 NEXT I
6200 FOR I=R+1 TO M1
6210 FOR J=S+1 TO N1
6220 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6230 NEXT J
6240 NEXT I
7000 REM T. - klicovy sloupec
7010 IF R=1 THEN 7050
7020 FOR I=1 TO R-1
7030 A(I,S)=0
7040 NEXT I
7050 FOR I=R+1 TO M1
7060 A(I,S)=0
7070 NEXT I
7080 A(R,S)=1
7090 L=0
7100 S=0
7110 R=0
7120 GOTO 3000
8000 REM Tisk
8010 FOR J=1 TO N
8020 FOR I=1 TO M1
8030 IF A(I,J)=1 THEN 8060
8040 IF A(I,J)=0 THEN 8080
8050 GOTO 8140
8060 K=I
8070 GOTO 8090
8080 T=T+1
8090 NEXT I
8100 IF T > M THEN 8130
8110 X(J)=A(K,N1)
8120 PRINT "X";J;"=";A(K,N1)
8130 T=0
8140 NEXT J
8150 PRINT "Ostatni promenne = 0"
8160 PRINT "Z =";A(M1,N1)
8170 INPUT "Chcete pokracovat (A/N)?",Q$
8180 IF Q$="N" OR Q$="0" THEN 8230
8190 PRINT "Ucelova funkce:"
8200 FOR J=1 TO N
8210 IF A(M1,J) <> 0 THEN PRINT "C";J;"=";A(M1,J)
8220 NEXT J
8220 PRINT "Ostatni ceny = 0"
8230 PRINT "Konec."
8240 END
9000 REM Tisk tabulky
9010 INPUT "Chcete vystup tabulky (A/N)?",Q$
```

```
9020 IF Q$="N" OR Q$="0" THEN RETURN
9030 Y=Y+1
9040 PRINT "Tabulka - krok:";Y
9050 FOR I=1 TO M1
9060 FOR J=1 TO N1
9070 IF LEN(STR$(ABS(A(I,J)))) <= 3 THEN PRINT " ";
9080 IF LEN(STR$(ABS(A(I,J)))) < 2 THEN PRINT " ";
9090 PRINT A(I,J);
9100 NEXT J
9110 PRINT
9120 NEXT I
9130 PRINT
9140 RETURN
```

Program je co nejjednodušší. Může však tvořit základ dalších podobných programů. Zdokonalování by se na prvém místě týkalo možných změn a oprav vložených dat. Na druhém místě by program obohatila možnost návratu k původním datům po provedeném výpočtu. To by ovšem předpokládalo dvojnásobný rozsah ukládaných dat, resp. poloviční rozsah úlohy pro danou velikost paměti.

18.2 Dlouhá simplexová metoda

Další program obsahuje tzv. dvoufázovou simplexovou metodu. Mohou se pomocí něj řešit obecné úlohy lineárního programování s omezením všech druhů.

Vstup se předpokládá v podobě výchozí simplexové tabulky, včetně jednotkové matice, koeficientů pravých stran, účelové funkce v anulovaném tvaru i pomocné účelové funkce. Program předpokládá umístění pomocných proměnných na posledních místech určených pro proměnné.

Program: Simplex II

```
10 REM Simplex II
20 DIM A(11,26)
30 CLS
40 PRINT "Simplexova metoda - dlouha verze"
50 PRINT "================================"
60 PRINT "MAXIMUM nebo MINIMUM"
70 PRINT "Maximalni pocet omezeni je 10"
80 PRINT "Max.pocet prom.vc. baze je 25"
```

```
  90 PRINT "Vklada se cela tabulka takto:"
 100 PRINT "- omezeni jako rovnice,"
 110 PRINT "- pomocne promenne na konci,
 120 PRINT "- ke kazdemu prava strana,"
 130 PRINT "- UF v anulovanem tvaru."
 140 PRINT "Nuly je take nutno vkladat!"
 150 PRINT
1000 REM Vstupy
1010 INPUT "Vlozte MAX nebo MIN!";U$
1020 IF U$="MAX" OR U$="MIN" THEN 1050
1030 PRINT "JEN NEKTEROU UVEDENOU MOZNOST!"
1040 GOTO 1010
1050 INPUT "Vlozte pocet omezeni";M
1060 IF M < 11 THEN 1090
1070 PRINT "Maximalne 10!"
1080 GOTO 1050
1090 M1=M+1
1100 M2=M+2
1110 INPUT "Vlozte poc.prom.vcetne baze";N
1120 IF N < 26 THEN 1150
1130 PRINT "Maximalne 25!"
1140 GOTO 1110
1150 N1=N+1
1160 INPUT "Vlozte pocet POMOCNYCH prom.";NP
1170 IF NP < N THEN 1200
1180 PRINT "Maximalne";N-1;"!"
1190 GOTO 1160
1200 IF NP=0 THEN M2=M1
1210 FOR I=1 TO M2
1220 FOR J=1 TO N1
1230 IF J < N1 THEN 1260
1240 PRINT "Prava strana";I;
1250 GOTO 1330
1260 IF I <> M1 THEN 1290
1270 PRINT "Ucelova funkce";J;
1280 GOTO 1330
1290 IF I < M2 THEN 1320
1300 PRINT "Pom.ucel.funkce";J;
1310 GOTO 1330
1320 PRINT "Prvek";I;J;
1330 INPUT A(I,J)
1340 NEXT J
1350 NEXT I
2000 REM Test optima pom. UF
2010 GOSUB 9000
2020 D=-99999
2030 FOR J=1 TO N
2040 IF A(M2,J) >= 0 THEN 2090
2050 IF ABS(A(M2,J)) < D THEN 2090
2060 S=J
2070 K=1
2080 D=ABS(A(M2,J))
```

```
2090 NEXT J
2100 IF K=1 THEN 4000
2110 IF ABS(A(M2,N1)) < 0.01 THEN 2140
2120 PRINT "Nepripustne"
2130 GOTO 8420
2140 M2=M1
2150 GOTO 3000
3000 REM Test optima
3010 GOSUB 9000
3020 IF U$="MIN" THEN 3500
3030 REM Test pro maximum
3040 D=-99999
3050 FOR J=1 TO N
3060 IF A(M1,J) >= 0 THEN 3110
3070 IF ABS(A(M1,J)) < D THEN 3110
3080 S=J
3090 L=1
3100 D=ABS(A(M1,J))
3110 NEXT J
3120 IF L=1 THEN 4000
3130 CLS
3140 PRINT "Optimum"
3150 GOTO 8000
3500 REM Test pro minimum
3510 D=-99999
3520 FOR J=1 TO N
3530 IF A(M1,J) >= 0 THEN 3580
3540 IF ABS(A(M1,J)) <D THEN 3580
3550 S=J
3560 L=1
3570 D=ABS(A(M1,J))
3580 NEXT J
3590 GOTO 3120
4000 REM Vystupujici
4010 P=99999
4020 FOR I=1 TO M
4030 IF A(I,S) <= 0 THEN 4080
4040 F=A(I,N1)/A(I,S)
4050 IF F >= P THEN 4080
4060 R=I
4070 P=F
4080 NEXT I
4090 IF R <> 0 THEN 5000
4100 PRINT "Neomezeno"
4110 GOTO 8420
5000 REM T. - klicovy radek
5010 IF S=1 THEN 5050
5020 FOR J=1 TO S-1
5030 A(R,J)=A(R,J)/A(R,S)
5040 NEXT J
5050 FOR J=S+1 TO N1
5060 A(R,J)=A(R,J)/A(R,S)
5070 NEXT J
6000 REM T. - tabulka
```

```
6010 IF R=1 THEN 6080
6020 IF S=1 THEN 6150
6030 FOR I=1 TO R-1
6040 FOR J=1 TO S-1
6050 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6060 NEXT J
6070 NEXT I
6080 IF S=1 THEN 6200
6090 FOR I=R+1 TO M1
6100 FOR J=1 TO S-1
6110 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6120 NEXT J
6130 NEXT I
6140 IF R=1 THEN 6200
6150 FOR I=1 TO R-1
6160 FOR J=S+1 TO N1
6170 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6180 NEXT J
6190 NEXT I
6200 FOR I=R+1 TO M1
6210 FOR J=S+1 TO N1
6220 A(I,J)=A(I,J)-A(I,S)*A(R,J)
6230 NEXT J
6240 NEXT I
7000 REM T. - klicovy sloupec
7010 IF R=1 THEN 7050
7020 FOR I=1 TO R-1
7030 A(I,S)=0
7040 NEXT I
7050 FOR I=R+1 TO M1
7060 A(I,S)=0
7070 NEXT I
7080 A(R,S)=1
7090 L=0
7100 S=0
7110 R=0
7120 GOTO 3000
8000 REM Tisk
8010 T$="Chcete tisk vysledku?"
8020 GOSUB 9500
8030 IF Q=0 THEN 8420
8040 REM Priprava
8050 T$="Chcete pokracovat?"
8060 E=0
8070 REM Nenulove promenne
8080 PRINT "Hodnoty promennych:"
8090 FOR J=1 TO N-NP
8100 IF A(M1,J) <> 0 THEN 8220
8110 T=0
8120 FOR I=1 TO M
8130 IF A(I,J)=1 THEN K=I
8140 IF A(I,J)=0 THEN T=T+1
8150 NEXT I
8160 IF T <> M-1 THEN 8220
```

```
8170 PRINT "X";J;"=";A(K,N1)
8180 E=E+1
8190 IF E/6 <> INT(E/6) THEN 8220
8200 GOSUB 9500
8210 IF Q=0 THEN 8240
8220 NEXT J
8230 PRINT "Ostatni promenne = 0"
8240 PRINT "Z =";ABS(A(M1,N1))
8250 REM Ceny z UF
8260 T$="Chcete ceny z ucelove funkce?"
8270 GOSUB 9500
8280 IF Q=0 THEN 8420
8290 T$="Chcete pokracovat?"
8300 E=0
8310 PRINT "Ceny ucelove funkce:"
8320 FOR J=1 TO N-NP
8330 A=A(M1,J)
8340 IF A=0 THEN 8400
8450 PRINT "C";J;"=";A
8360 E=E+1
8370 IF E/6 <> INT(E/6) THEN 8400
8380 GOSUB 9500
8390 IF Q=0 THEN 8420
8400 NEXT J
8410 PRINT "Ostatni ceny = 0"
8420 PRINT
8430 PRINT "Konec."
8440 END
9000 REM Tisk tabulky
9010 T$="Chcete vystup tabulky (A/N)?"
9020 GOSUB 9500
9030 IF Q=0 THEN RETURN
9040 PRINT "Jen pro malou tabulku!"
9050 Y=Y+1
9060 PRINT "Tabulka - krok:";Y
9070 FOR I=1 TO M1
9080 FOR J=1 TO N1
9090 A=A(I,J)
9100 E=LEN(STR$(ABS(A)))
9110 IF E > 2 THEN PRINT " ";
9120 IF E < 3 THEN PRINT " ";
9130 PRINT A;
9140 NEXT J
9150 PRINT
9160 NEXT I
9170 PRINT
9180 RETURN
9500 REM Souhlas
9510 PRINT T$
9520 Q$=INKEY$
9530 IF Q$="" THEN 9520
9540 Q=1
9550 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
9560 Q=0
```

```
9570 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
9580 PRINT "Odpovidejte takto:"
9590 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
9600 GOTO 9500
9610 REM Konec programu Simplex II
```

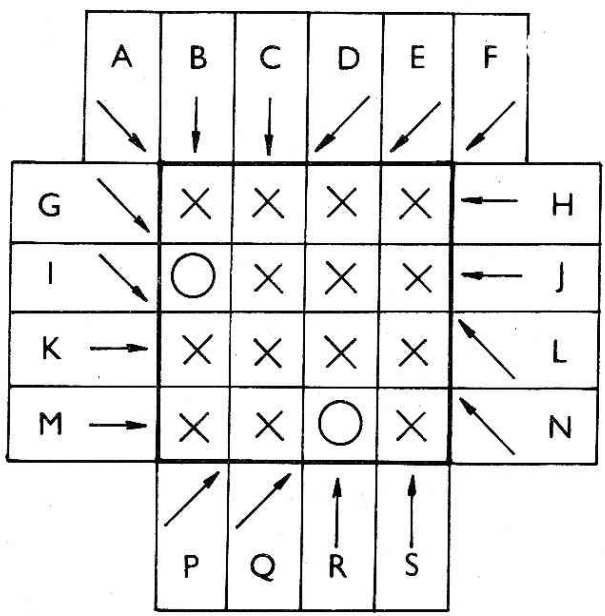
Program může být použit i pro reálné výpočty, ale omezené velikosti. Nemá zabudovány kontroly vkládaných veličin, ani možnost jejich oprav. Stejně tak neobsahuje možnost návratu k úpravám vstupních hodnot, jestliže se ve výsledku ukáže potřeba takových změn. O to zde ostatně nejde. Reálné úlohy by bylo stejně nutné řešit na jiné technice.

19 Další úlohy a hry

V této kapitole jsou uvedeny některé zajímavé úlohy a hry, které mohou zpestřit předkládané ukázky a ukázat některé programátorské obraty, které mohou být užitečné v mnoha jiných situacích.

19.1 Gardnerova hra

Gardnerova hra byla publikována v časopise KVIZ, nikoliv však jako program, ale jako hra s mincemi.



Obr. 4 Gardnerova hra

Na obrázku 4 je uveden plán hry. Je to pole 4x4 políčka. Na všech šestnácti políčkách leží mince. Jsou položeny náhodně nahoru rubem, nebo lícem, které na obrázku zastupují kolečka a křížky. Cílem hry je obrátit všechny mince stejnou stranou nahoru. Přitom není rozhodující, zda to budou křížky nebo kroužky.

Přitom je možné obracet vždy celou řadu mincí. Řadou se rozumí všechny vodorovné a všechny svislé řady. Řadami se rozumí také všechny úhlopříčné řady v obou směrech, pokud ovšem obsahují nejméně dvě mince. Samostatné rohové mince se obracet nemohou. Pro označení řad, které se budou obracet pomocí programu na obrazovce počítače, jsou kolem hracího pole uvedeny úsečky, naznačující směr, ve kterém budou mince obraceny. U úseček jsou napsána písmena, pomocí nichž se do počítače vkládá označení obrácené řady mincí.

Program je sestaven tak, že se na obrazovce objeví obrázek shodný s uvedeným. Pouhým zadáním odpovídajícího písmene se "mince obrátí", na obrazovce se ukáže nové řešení.

Hru můžete ukončit vložením nuly nebo písmene O. To může být tehdy, jestliže dospějete k cíli, ale i tehdy, když si s dalším postupem nevíte rady. Hra nemá vždycky řešení. Po jejím ukončení (nulou nebo O) se dozvíte, zda řešení měla, nebo ne.

Program: Gardnerova hra

```
 10 REM Gardner
 20 REM Gardnerova hra z Kvizu
 30 CLS
 40 DIM M(4,4)
 50 DIM T$(30)
 60 PRINT "     GARDNEROVA HRA"
 70 PRINT "V tabulce 4x4 jsou krouzky a"
 80 PRINT "krizky. Zastupuji mince, ktere"
 90 PRINT "mate obratit na stejnou stranu."
100 PRINT "Obrací se cele radky a sloupce."
110 PRINT "Obrací se take sikme rady,"
120 PRINT "maji-li dve nebo vice minci."
130 PRINT "Radu k obraceni zadate pismenem"
140 PRINT "u tabulky, kde je take smer."
150 PRINT
160 T$="Chcete si zahrat (A/N)"
```

```
170 GOSUB 1300
180 IF Q=0 THEN 1270
190 K=0: M=0: R=0
200 FOR I=1 TO 4
210 FOR J=1 TO 4
220 M(I,J)=1
230 IF RND(0) > 0.5 THEN 270
240 L=I*J
250 M(I,J)=-1
260 IF L=2 OR L=3 OR L=8 OR L=12 THEN R=R+1
270 K=K+M(I,J)
280 NEXT J
290 NEXT I
300 IF ABS(K)=16 THEN 180
310 CLS
320 PRINT TAB(10);"A B C D E F"
330 PRINT TAB(10);"\ ! ! / / /"
340 PRINT TAB(8);"G \";
350 FOR K=1 TO 4
360 IF M(1,K)=1 THEN PRINT " x";
370 IF M(1,K) <> 1 THEN PRINT " o";
380 NEXT K
390 PRINT " - H"
400 PRINT TAB(8);"I \";
410 FOR K=1 TO 4
420 IF M(2,K)=1 THEN PRINT " x";
430 IF M(2,K) <> 1 THEN PRINT " o";
440 NEXT K
450 PRINT " - J"
460 PRINT TAB(8);"K -";
470 FOR K=1 TO 4
480 IF M(3,K)=1 THEN PRINT " x";
490 IF M(3,K) <> 1 THEN PRINT " o";
500 NEXT K
510 PRINT "\  L"
520 PRINT TAB(8);"M -";
530 FOR K=1 TO 4
540 IF M(4,K)=1 THEN PRINT " x";
550 IF M(4,K) <> 1 THEN PRINT " o";
560 NEXT K
570 PRINT "\  N"
580 PRINT TAB(12);"/ / ! !"
590 PRINT TAB(12);"P Q R S"
600 PRINT
610 PRINT "Chcete-li koncit - vlozte 0!"
620 PRINT
630 PRINT "Vlozte pismeno oznacujici radu,"
640 INPUT "kterou chcete obratit.";Q$
650 IF Q$="0" THEN 1060
660 N=ASC(Q$)
670 N=N-64
680 IF N > 0 AND N < 20 THEN 720
690 PRINT CHR$(7)
700 PRINT "Chybny vstup!"
710 GOTO 630
```

```
 720 REM Cykly pro ruzne rady
 730 X=1: Y=4
 740 IF N > 10 THEN 860
 750 ON N GOTO 760,770,780,790,800,810,820,830,840,850
 760 V=1: W=0: GOTO 950
 770 V=0: W=1: GOTO 950
 780 V=0: W=2: GOTO 950
 790 Y=2: V=-1: W=3: GOTO 950
 800 Y=3: V=-1: W=4: GOTO 950
 810 V=-1: W=5: GOTO 950
 820 X=2: V=1: W=-1: GOTO 950
 830 W=1: GOTO 1010
 840 X=3: V=1: W=-2: GOTO 950
 850 W=2: GOTO 1010
 860 ON N-10 GOTO 870,880,890,900,1060,910,920,930,940
 870 W=3: GOTO 1010
 880 Y=2: V=1: W=2: GOTO 950
 890 W=4: GOTO 1010
 900 Y=3: V=1: W=1: GOTO 950
 910 X=2: V=-1: W=6: GOTO 950
 920 X=3: V=-1: W=7: GOTO 950
 930 V=0: W=3: GOTO 950
 940 V=0: W=4
 950 REM Zmena
 960 FOR I=X TO Y
 970 J=I*V+W
 980 M(I,J)=-1*M(I,J)
 990 NEXT I
1000 GOTO 310
1010 FOR I=X TO Y
1020 J=W
1030 M(J,I)=-1*M(J,I)
1040 NEXT I
1050 GOTO 310
1060 REM Zaver
1070 FOR I=1 TO 4
1080 FOR J=1 TO 4
1090 M=M+M(I,J)
1100 NEXT J
1110 NEXT I
1120 IF ABS(M) <> 16 THEN 1170
1130 PRINT
1140 PRINT "Gratuluji k uspechu!"
1150 PRINT
1160 GOTO 1240
1170 IF R/2=INT(R/2) THEN 1230
1180 PRINT
1190 PRINT "Mate pravdu!"
1200 PRINT "Ve hre nema smysl pokracovat."
1210 PRINT "Nema reseni!"
1220 GOTO 1240
1230 PRINT "Ale tato hra MELA reseni!"
1240 T$="Chcete pokracovat jinou hrou?"
1250 GOSUB 1300
```

```
1260 IF Q=1 THEN 190
1270 PRINT
1280 PRINT "Na shledanou!"
1290 END
1300 REM Souhlas
1310 PRINT T$
1320 Q$=INKEY$
1330 IF Q$="" THEN 1320
1340 Q=1
1350 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
1360 Q=0
1370 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
1380 PRINT "Odpovidejte takto:"
1390 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
1400 GOTO 1300
```

Příklad bez úvodních informací

```
    A B C D E F
    \ ! ! / / /
G \ x x x x - H
I \ o o x x - J
K - x o x o \ L
M - x o o x \ N
    / / ! !
    P Q R S
```

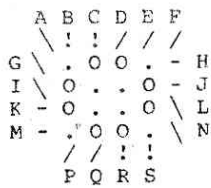
```
Chcete-li koncit - vlozte 0!

Kterou radu chcete obratit?
:D
    A B C D E F
    \ ! ! / / /
G \ x o x x - H
I \ x o x x - J
K - x o x o \ L
M - x o o x \ N
    / / ! !
    P Q R S

Chcete-li koncit - vlozte 0!

Kterou radu chcete obratit?
:C

    A B C D E F
    \ ! ! / / /
G \ x x x x - H
I \ x x x x - J
K - x x x o \ L
M - x x o x \ N
    / / ! !
    P Q R S
```

```
Chcete-li koncit - vlozte 0!

Kterou radu chcete obratit?
:Q

        A B C D E F
         \ ! ! / / /
      G \ x x x x - H
      I \ x x x x - J
      K - x x x x \ L
      M - x x x x \ N
          / / ! !
         P Q R S

Chcete-li koncit - vlozte 0!

Kterou radu chcete obratit?
:0

Chcete pokracovat jinou hrou?
:N
```

Krátká ukázka - vlastně pouhé tři kroky - vedla rychle k nalezení správného výsledku. Ne vždy to jde tak rychle, ne vždy je snadné určení řady, která se má otáčet. Nic ovšem nebrání tomu, abyste zadali kteroukoliv řadu, a když se vám výsledek nebude zdát správný, vrátili se k původní tabulce opakovanou volbou téže řady k obrácení.

Studiem programu byste také jistě zjistili, jak se pozná, zda zadání má, nebo nemá řešení. Na tomto místě vám to prozradíme. Vycházíme přitom ze zkušenosti s hraním této hry. Ukazuje se totiž, že pro správné obracení řad příliš nepomáhá znalost toho, jestli úloha řešení má, nebo ne. Naopak se ukazuje, že hráči jsou při hře vytrvalejší, jestliže ví, že pro dané zadání řešení existuje, než když to neví.

Jiná ukázka



Spočítejte křížky (ane' ' kroužky) na osmi políčkách, která jsou na uvedeném schématu označena velkým písmenem O. Bude-li počet křížků a tím také 'roužků lichý, pak úloha nemá řešení, a žádným obracením řad se k žádanému výsledku nedostanete. Když bude počet křížků i kroužků sudý, úloha řešení má. Nezáleží přitom vůbec na počtu kroužků a křížků na ostatních políčkách, tj. na těch, která jsou na schématu označena tečkami.

19.2 Desetiúhelník

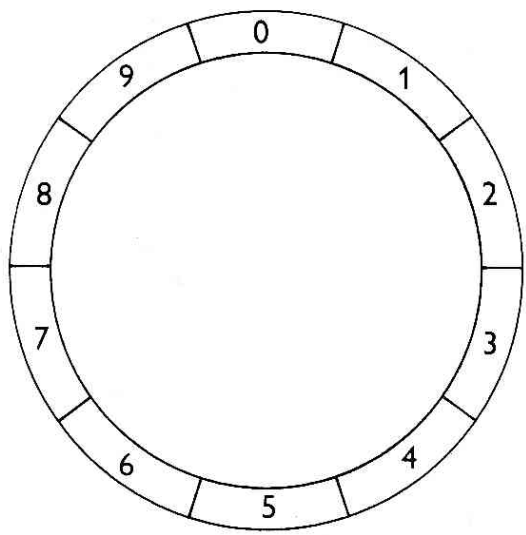
V učebnici matematiky pro gymnázia jsou občas uváděny i hry, které mají studenti analyzovat a u kterých mají najít nejlepší postup. V učebnici pro první ročník je tato zajímavá hra. Zadání použijeme přímo z této učebnice.

Hrají dva hráči. Na papír si načrtnou desetiúhelník, jeden z jeho vrcholů označí O a postaví na něj figurku (minci, knoflík, kousek papíru). Potom figurku střídavě - jednou jeden, podruhé druhý - přemísťují o 3, 4 nebo 5 vrcholů, vždy ve směru chodu hodinových ručiček. Vyhrává hráč, který opět postaví figurku na vrchol O.

Pokuste se určit nejmenší počet tahů, který umožní začínajícímu hráči vyhrát.

Tuto hru převedeme na počítač. Na obrázku 5 je nakresleno schéma hry, které se bude podobně také objevovat na obrazovce. Polohu figurky bude přitom zastupovat pomlčka, která se bude objevovat na místě některého vrcholu namísto čísla daného vrcholu.

Program je sestaven tak, že zastupuje jednoho hráče. Přitom je možné začínat, nebo nechat začínat počítač.



Obr. 5 Desetiúhelník

Program: Desetiúhelník

```
10 REM Desetiuhelnik
20 REM Matematika pro gymnazia
30 REM Strana 99
40 CLS
50 PRINT "DESETIUHELNIK
60 PRINT "Pres sebou mate desetiuhelnik:"
70 PRINT
80 T=11
```

```
 90 GOSUB 880
100 PRINT "Zacina se od 0. Jde se ve smeru"
110 PRINT "hodin az se znovu dojde k 0."
120 PRINT "Postupujte o 3,4 nebo 5 vrcholu."
130 PRINT "Ve hre se stridate s pocitacem."
140 PRINT "Kdo opet dosahne NULU VYHRAVA!"
150 T=0
160 L=0
170 T$="Chcete zacinat? (A/N)"
180 GOSUB 770
190 IF Q=1 THEN 280
200 REM Prvni tah pocitace
210 L=1
220 J=4
230 T=4
240 CLS
250 PRINT "Postoupil jsem o";J
260 PRINT
270 GOSUB 880
280 Tah hrace
290 INPUT "Vlozte pocet vrcholu";K
300 IF K > 2 AND K < 6 THEN 330
310 PRINT "Postoupit lze jen o 3,4 nebo 5!"
320 GOTO 290
330 REM Vyhodnoceni tahu hrace
340 T=T+K
350 IF T=10 THEN 670
360 IF T > 9 THEN T=T-10
370 REM Tah pocitace
380 J=8-K
390 IF L=1 THEN 540
400 REM Novy vypocet
410 ON T GOTO 440,420,460,480,500,500,500,520,520
420 J=INT(RND(0)*3)+3
430 GOTO 540
440 J=3
450 GOTO 530
460 J=5
470 GOTO 540
480 J=INT(RND(0)*2)+4
490 GOTO 540
500 J=10-T
510 GOTO 540
520 J=12-T
530 L=1
540 REM Vyhodnocení tahu pocitace
550 T=T+J
560 IF T > 9 THEN T=T-10
570 CLS
580 PRINT "Postoupil jsem o";J
590 PRINT
600 GOSUB 880
610 IF T=0 THEN 640
620 PRINT
630 GOTO 280
```

```
640 PRINT
650 PRINT "PROHRAL JSTE!"
660 GOTO 690
670 PRINT
680 PRINT "VYHRAL JSTE!"
690 PRINT
700 T$="Chcete hrat dal? (A/N)"
710 GOSUB 770
720 IF Q=0 THEN 740
730 GOTO 80
740 PRINT
750 PRINT "Konec hry!"
760 END
770 REM Souhlas
780 PRINT T$
790 Q$=INKEY$
800 IF Q$="" THEN 790
810 Q=1
820 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
830 Q=0
840 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
850 PRINT "Odpovidejte takto:"
860 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
870 GOTO 770
880 REM Kresleni desetiuhelniku
890 IF T=0 THEN PRINT TAB(15)"-"
900 IF T <> 0 THEN PRINT TAB(15)"0"
910 IF T=9 THEN PRINT TAB(11)"-";
920 IF T <> 9 THEN PRINT TAB(11)"9";
930 IF T=1 THEN PRINT TAB(19)"-"
940 IF T <> 1 THEN PRINT TAB(19)"1"
950 IF T=8 THEN PRINT TAB(9)"-";
960 IF T <> 8 THEN PRINT TAB(9)"8";
970 IF T=2 THEN PRINT TAB(21)"-"
980 IF T <> 2 THEN PRINT TAB(21)"2"
990 IF T=7 THEN PRINT TAB(9)"-";
1000 IF T <> 7 THEN PRINT TAB(9)"7";
1010 IF T=3 THEN PRINT TAB(21)"-"
1020 IF T <> 3 THEN PRINT TAB(21)"3"
1030 IF T=6 THEN PRINT TAB(11)"-";
1040 IF T <> 6 THEN PRINT TAB(11)"6";
1050 IF T=4 THEN PRINT TAB(19)"-"
1060 IF T <> 4 THEN PRINT TAB(19)"4"
1070 IF T=5 THEN PRINT TAB(15)"-"
1080 IF T <> 5 THEN PRINT TAB(15)"5"
1090 RETURN
```

Ukázka

```
DESETIUHELNIK
Pred sebou mate desetiuhelnik:
                0
            9       1
        8               2
        7               3
            6       4
                5
Zacina se od 0. Jde se ve smeru
hodin az se znovu dojde k 0.
Postupujte o 3,4 nebo 5 vrcholu.
Ve hre se stridate s pocitacem.
Kdo opet dosahne NULU - VYHRAVA!
Chcete zacinat?
0

Postoupil jsem o 4
                0
            9       1
        8               2
        7               3
            6       -
                5
O kolik vrcholu postupujete?
:5

Postoupil jsem o 3
                0
            9       1
        8               -
        7               3
            6       4
                5
O kolik vrcholu postupujete?
:3


Postoupil jsem o 5
                -
            9       1
        8               2
        7               3
            6       4
                5
PROHRAL JSTE!

Chcete hrat dal?
0

Konec hry.
```

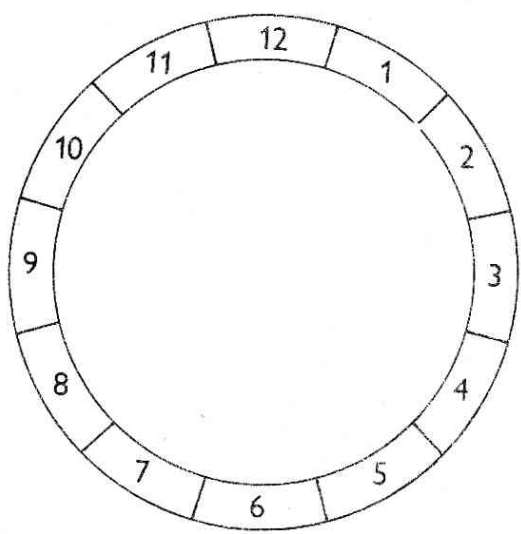
19.3 Dvanáctiúhelník - hodiny

V zadání k desetiúhelníku je v učebnici matematiky pro první ročník gymnázií uveden také dodatek, pomocí něhož je možné základní podobu hry, která je popsána u předchozí hry Desetiúhelník, modifikovat.

Zobecnění, které je zde použito, se týká zvětšení rozměru n-úhelníku. Místo deseti vrcholů budeme tentokrát uvažovat 12, tak jak je to znázorněno obrázkem 6.

Protože se v tomto případě dvanáctiúhelník podobá hodinám, budeme pro rozlišení užívat někdy i tento název. Také provedeme - kromě zvětšení velikost - další změnu. Budeme uvažovat číslici 1 - 12, tentokrát bez nuly.

Princip programu je jinak shodný. Můžete porovnat, v čem se mění jeho podmínky a jak se změna zadání úlohy jeví v programu samotném i ve hře s ním.



Obr. 6 Dvanáctiúhelník - hodiny

Program: Dvanáctiúhelník - hodiny

```
 10 REM Dvanactiuhelnik
 20 REM Matematika pro gymnazia
 30 REM 1. rocnik
 40 REM Strana 99
 50 CLS
 60 PRINT "DVANACTIUHELNIK - HODINY
 70 PRINT "Pres sebou mate dvanactiuhelnik:"
 80 T=13
 90 GOSUB 930
100 PRINT "Zacina se od 12. Jdete ve smeru"
110 PRINT "hodin az se znovu dojde k 12."
120 PRINT "Postupujte o 3,4 nebo 5 vrcholu."
130 PRINT "Ve hre se stridate s pocitacem."
140 PRINT "Kdo dosahne DVANACTKU VYHRAVA!"
150 T=12
160 L=0
170 T$="Chcete zacinat? (A/N)"
180 GOSUB 820
190 IF Q=1 THEN 280
200 REM Prvni tah pocitace
210 L=1
220 J=4
230 T=4
240 CLS
250 PRINT "Postoupil jsem o";J
260 PRINT
270 GOSUB 930
280 REM Tah hrace
290 INPUT "Vlozte pocet vrcholu";K
300 IF K > 2 AND K < 6 THEN 330
310 PRINT "Postoupit lze jen o 3,4 nebo 5!"
320 GOTO 290
330 REM Vyhodnoceni tahu hrace
340 T=T+K
350 IF T=12 THEN 710
360 IF T > 12 THEN T=T-12
370 REM Tah pocitace
380 J=8-K
390 IF L=1 THEN 580
400 REM Novy vypocet
410 ON T GOTO 440,500,460,420,480,520,560,560,560,540,540
420 J=INT(RND(0)*3)+3
430 GOTO 580
440 J=3
450 GOTO 570
460 J=3
470 GOTO 580
480 J=5
490 GOTO 580
500 J=INT(RND(0)*2)+3
510 GOTO 580
520 J=INT(RND(0)*2)+4
```

```
530 GOTO 580
540 J=16-T
550 GOTO 580
560 J=12-T
570 L=1
580 REM Vyhodnocení tahu pocitace
590 T=T+J
600 IF T > 12 THEN T=T-12
610 CLS
620 PRINT "Postoupil jsem o";J
630 PRINT
640 GOSUB 930
650 IF T=12 THEN 680
660 PRINT
670 GOTO 280
680 PRINT
690 PRINT "PROHRAL JSTE!"
700 GOTO 730
710 PRINT
720 PRINT "VYHRAL JSTE!"
730 PRINT
740 T$="Chcete hrat dal? (A/N)"
750 GOSUB 820
760 IF Q=0 THEN 780
770 GOTO 80
780 PRINT
790 PRINT
800 PRINT "Konec hry!"
810 END
820 REM Souhlas
830 PRINT T$
840 Q$=INKEY$
850 IF Q$="" THEN 840
860 Q=1
870 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
880 Q=0
890 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
900 PRINT "Odpovidejte takto:"
910 PRINT "ANO=A nebo 1,  NE=N nebo 0!"
920 GOTO 820
930 REM Kresleni dvanactiuhelniku
940 IF T=12 THEN PRINT TAB(15)"-"
950 IF T <> 12 THEN PRINT TAB(14)"12"
960 IF T=11 THEN PRINT TAB(11)"-";
970 IF T <> 11 THEN PRINT TAB(10)"11";
980 IF T=1 THEN PRINT TAB(19)"-"
990 IF T <> 1 THEN PRINT TAB(19)"1"
1000 IF T=10 THEN PRINT TAB(9)"-";
1010 IF T <> 10 THEN PRINT TAB(8)"10"
1020 IF T=2 THEN PRINT TAB(21)"-"
1030 IF T <> 2 THEN PRINT TAB(21)"2"
1040 IF T=9 THEN PRINT TAB(8)"-";
1050 IF T <> 9 THEN PRINT TAB(8)"9";
1060 IF T=3 THEN PRINT TAB(22)"-"
1070 IF T <> 3 THEN PRINT TAB(22)"3"
```

```
1080 IF T=8 THEN PRINT TAB(9)"-";
1090 IF T <> 8 THEN PRINT TAB(9)"8";
1100 IF T=4 THEN PRINT TAB(21)"-"
1110 IF T <> 4 THEN PRINT TAB(21)"4"
1120 IF T=7 THEN PRINT TAB(11)"-";
1130 IF T <> 7 THEN PRINT TAB(11)"7";
1140 IF T=5 THEN PRINT TAB(19)"-"
1150 IF T <> 5 THEN PRINT TAB(19)"5"
1160 IF T=6 THEN PRINT TAB(15)"-"
1170 IF T <> 6 THEN PRINT TAB(15)"6"
1180 RETURN
```

Ukázka:

```
DVANACTIUHELNIK - HODINY
Pred sebou mate dvanactiuhelnik:
              12
          11        1
       10             2
       9               3
        8             4
          7         5
              6
Zacina se od 12. Jde se ve smeru
hodin az se znovu dojde ke 12.
Postupujte o 3,4 nebo 5 vrcholu.
Ve hre se stridate s pocitacem.
Kdo dosahne DVANACTKU VYHRAVA!
Chcete zacinat?
0

Postoupil jsem o 4
              12
          11        1
       10             2
       9               3
        8             -
          7         5
              6

O kolik vrcholu postupujete?
:3

Postoupil jsem o 5
              -
          11        1
       10             2
       9               3
        8             4
          7         5
              6
```

```
PROHRAL JSTE!

Chcete hrat dal?
0

Konec hry.
```

Ukázková partie je úmyslně zvlášť krátká. Odhalíte-li správnou strategii, může být utkání delší a hlavně, můžete dokázat svoji převahu nad počítačem a zvítězit nad ním.

U této i u předchozí hry to je možné, musíte ale také vzít v úvahu první tah. Při správné strategii někdy musí vést k vítězství, jindy žádný první tah nepomůže. Optimální strategie je na straně druhého hráče. Vy ovšem máte možnost se rozhodnout také o tom, kdo bude ve hře začínat. Včetně tohoto prvního tahu musí být konečné vítězství vaše.

19.4 Muž nebo žena?

Abychom si od her odpočinuli, předkládáme vám tentokrát program, jehož smyslem je analýza vloženého jména tak, aby se poznalo, zda toto jméno patří chlapci, nebo dívce, muži, nebo ženě. Hned na začátku ale musíme upozornit, že výsledek analýzy nemusí být vždy správný. Lidská řeč a samozřejmě také vlastní jména mohou být natolik rozmanitá, že výsledek analýzy bude opačný proti skutečnosti. Přesto budete možná překvapeni, jak poměrně dokonale může takový program fungovat.

Na tomto místě vám už víc neprozradíme. Ten, kdo se bude o věc zajímat více, najde některé další informace za programem a ukázkou. Mnoho se také může dovědět studiem samotného programu a také četbou doporučené literatury.

Program: Muž nebo žena?

```
10 REM Muz nebo Zena
20 REM Program pozna muze nebo zenu podle jmena
30 CLEAR 500
40 CLS
50 PRINT "MUZ nebo ZENA?"
```

```
60 PRINT
70 PRINT "Vklada se prijmeni a krestni jmeno."
80 PRINT "Poznam jde-li o muze nebo zenu."
90 PRINT
100 INPUT "Vlozte PRIJMENI";P$
110 P=LEN(P$)
120 FOR I=1 TO P
130 IF ASC(MID$(P$,I,1)) < 65 THEN 160
140 IF ASC(MID$(P$,I,1)) > 90 THEN 160
150 NEXT I: GOTO 170
160 PRINT "Jen VELKA pismena!";CHR$(7): GOTO 100
170 IF P > 2 THEN 200
180 PRINT "Jmeno musi mit vic nez 2 pismena!";CHR$(7)
190 GOTO 100
200 INPUT "Vlozte KRESTNI jmeno";J$
210 J=LEN(J$)
220 FOR I=1 TO J
230 IF ASC(MID$(J$,I,1)) < 65 THEN 260
240 IF ASC(MID$(J$,I,1)) > 90 THEN 260
250 NEXT I: GOTO 270
260 PRINT "Jen VELKA pismena!";CHR$(7): GOTO 200
270 IF J > 2 THEN 300
280 PRINT "Jmeno musi mit vic nez 2 pismena!";CHR$(7)
290 GOTO 200
300 IF MID$(J$,J,1) <> "A" THEN 470
310 REM Konec na A
320 IF J > 6 THEN 1040
330 ON J-2 GOTO,340,360,400,430
340 IF J$="OTA" OR J$="ULA" THEN 1020
350 IF J$ <> "OTA" AND J$ <> "ULA" THEN 1040
360 IF J$="MILA" OR J$="SASA" THEN 980
370 K=10
380 W$="ALVAGEZAILJAJUDAKUBAPEPASABASAVAVASAVITA"
390 GOTO 570
400 K=6
410 W$="ATILAGEJZAHONZAMNATATONDALOJZA"
420 GOTO 570
430 IF J$="STANDA" THEN 980
440 K=5
450 W$="FRANTAMILOTANIKITANIKOLASTASTA"
460 GOTO 570
470 REM Konec na E
480 IF MID$(J$,J,1) <> "E" THEN 620
490 IF J > 5 THEN 1040
500 ON J-2 GOTO 510,520,550
510 IF J$="UVE" THEN 980
520 IF J$="RENE" THEN 980
530 IF J$="ARNE" THEN 1020
540 IF J$ <> "ARNE" THEN 1040
550 K=3
560 W$="JOSUEJOZUESHANE"
570 REM Poznavani muzskeho jmena
580 FOR I=1 TO K
590 IF J$=MID$(W$,1*J-J+1,J) THEN 1020
```

```
600 NEXT I
610 GOTO 1040
620 REM Jiny konec nez A nebo E
630 L=ASC(MID$(J$,J,1))-64
640 IF L > 18 THEN 860
650 IF L > 9 THEN 730
660 ON L GOTO 1040,1020,1020,670,1040,1020,690,1020,710
670 IF J$="ASTRID" OR J$="INGRID" THEN 1040
675 IF J$="MAUD" OR J$="MILDRED" THEN 1040
680 GOTO 1020
690 IF J$="INGEBORG" OR J$="SOLVEIG" THEN 1040
700 GOTO 1020
710 IF J$="HEIDI" OR J$="KAZI" THEN 1040
715 IF J$=MARI" OR J$="NOEMI" THEN 1040
720 GOTO 1020
730 ON L-9 GOTO 1020,1020,740,770,790,1020,1020,1020,840
740 IF J$="ABIGAIL" OR J$="MIRIEL" THEN 1040
750 IF J$="MURIEL" OR J$="NIKOL" OR J$="RACHEL" THEN 1040
760 GOTO 1020
770 IF J$="MIRIAM" OR J$="MIRJAM" THEN 1040
780 GOTO 1020
790 IF J$="CARMEN" OR J$="KARMEN" OR J$="GUDRUN" THEN 1040
800 IF J$="HEIDRUN" OR J$="KATRIN" THEN 1040
810 IF J$="KERSTIN" OR J$="KIRSTIN" THEN 1040
820 IF J$="MARILYN" OR J$="MARION" THEN 1040
830 GOTO 1020
840 IF J$="DAGMAR" OR J$="ESTER" OR J$="PILAR" THEN 1040
850 GOTO 1020
860 ON L-18 GOTO 870,900,1020,1020,1020,1020,920,960
870 IF J$="DAMARIS" OR J$="DORIS" OR J$="INES" THEN 1040
880 IF J$="IRIS" OR J$="MERCEDES" THEN 1040
890 GOTO 1020
900 IF J$="BERIT" OR J$="BIRGIT" OR J$="RUT" THEN 1040
910 GOTO 1020
920 IF J$="JENNY" OR J$="KITTY" OR J$="MARY" THEN 1040
930 IF J$="PEGGY" OR J$="POLY" OR J$="SALY" THEN 1040
940 IF J$="SALLY" OR J$="SANDY" THEN 1040
950 GOTO 1020
960 IF J$="INEZ" THEN 1040
970 GOTO 1020
980 REM Podle krestniho jmena nelze rozhodnout
990 PRINT
1000 PRINT "Pravdepodobny vysledek:"
1010 IF MID$(P$,P-2,3)="OVA" THEN 1040
1020 PRINT J$;" ";P$;" je muz!"
1030 GOTO 1050
1040 PRINT J$;" ";P$;" je zena!"
1050 PRINT
1060 PRINT "Konec programu."
1070 END
```

Ukázka

```
MUZ nebo ZENA?

Vklada se prijmeni a krestni jmeno.
Poznam jde-li o muze nebo zenu.

Vlozte PRIJMENI: JANKU
Vlozte KRESTNI jmeno: Jitka
Vkladejte jen velka pismena!
Vlozte KRESTNI jmeno: JITKA

JITKA JANKU je zena!
```

K tomu, abychom poznali, zda vložené jméno patří chlapci, nebo dívce, se používá pouze příjmení a křestní jméno. V češtině je sice obvyklé, že ženská příjmení končí na -ová, avšak tato informace nestačí. Nejenom, že mnoho ženských příjmení končí jinak (např. Krejčí, Janků, Petrů nebo Janů), ale jsou dokonce také mužská příjmení, která naopak na koncovku -ova končí (např. SOVA). Není to sice koncovka s dlouhým á, ale s diakritickými znaménky mohou pracovat pouze málokteré počítače.

Takže analýza jména staví na křestních jménech. Jako základ pro analýzu, která je obsažena v programu, posloužila velice pěkná a důkladná knížka J. Knappové - viz literatura. Ukazuje se, že seznam např. doporučených křestních jmen pro účely zápisu v matrice je nepříliš dlouhý.

Nejlépe se analyzují podle koncovky. Dívčí křestní jména převážně končí na -a nebo -e. Na tato písmena zase naopak končí jen velmi málo mužských jmen. Těch několik je možné porovnat a máme "téměř jistotu". Naopak jiné poslední písmeno se u dívčích křestních jmen vyskytuje málo a opět je možné je jednotlivě porovnat. Tak se skutečně v programu postupuje.

Analýza samozřejmě selhává u cizích jmen, i když i u nich by analýza pomocí křestního jména byla nejnadějnější. Selhává navíc u několika jmen dalších, zejména, nejsou-li zapsána ve správné spisovné podobě, např. Míla, René, Saša, Stan-

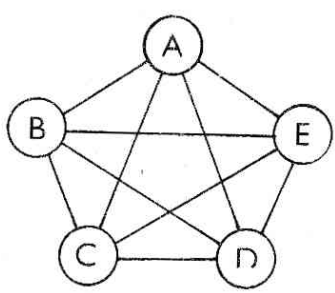
da. V těch několika málo situacích se přece jenom zjišťuje, jaká je koncovka u příjmení. Pro jistotu se však v takových situacích na obrazovce objevuje méně určité tvrzení.

Je zajímavé, že hovorová podoba mnoha chlapeckých jmen končí také písmenem - a. Např. Pepa, Franta, Honza a další. Některá z nich byla také do programu včleněna, aby proces poznávání byl dokonalejší. Mnohotvárnost hovorových jmen, jejich zdrobnělých či osobitých podob je však bez hranic. Proto je nutné se pro přesnou analýzu uchýlit do bezpečné oblasti spisovných tvarů.

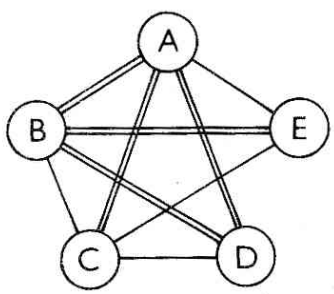
19.5 Trojúhelníky v pětiúhelníku

Hra, která je nazvána Trojúhelníky v pětiúhelníku, má jednoduchá pravidla. Dva protihráče (z nichž jeden je počítač) doplňují střídavě úsečky, spojující pět vrcholů pětiúhelníka.

Takových úseček je celkem 10. Můžete se o tom přesvědčit na obrázku 7. Vtip hry je v tom, že žádný hráč nemá zakreslit takovou trojici úseček, která by vytvořila trojúhelník. Kdo první sestrojí trojúhelník, ten prohrává.



Obr. 7 Trojúhelníky v pětiúhelníku



Obr. 8 Příklad hry trojúhelníky v pětiúhelníku

Na obrázku 8 je uvedeno jedno z možných zakončení hry. Úsečky, které vybrali jednotliví hráči, jsou odlišeny tím, že pro jednoho hráče jsou zakreslovány jako jednoduché, kdežto pro druhého jako dvojité čáry.

Možných trojúhelníků pro pět vrcholů pětiúhelníku je také deset. Vrcholy pětiúhelníku jsou označeny velkými písmeny A, B, C, D a E. Hrany se zapisují dvojicí těchto písmen, např. AD. Počítač kontroluje správnost zadání i to, zda daná hrana byla již vybrána, nebo ne.

Musíte se bránit tomu, abyste to byli právě vy, kdo bude muset trojúhelník sestrojit. Stav hry se v paměti počítače zaznamenává, ale celkový stav se nezobrazuje. Budete-li chtít hrát pečlivě a vyhrát, raději si vybrané úsečky kreslete v průběhu hry sami.

Program: Trojúhelníky v pětiúhelníku

```
 10 REM 3 v 5
 20 REM Trojuhelniky v petiuhelniku
 30 DIM C(10),R(10,3),S(10),T(10,3)
 40 CLS
 50 PRINT "HRA TROJUHELNIKY V PETIUHELNIKU"
 60 PRINT "Petiuhelnik ma pet vrcholu: ABCDE."
 70 PRINT "Kazde 3 vrcholy tvori trojuhelnik."
 80 PRINT "Takovych trojuhelniku je 10."
 90 PRINT "V kazdem tahu se zadava strana."
100 PRINT "V tazich se stridate s pocitacem."
110 PRINT "Kdo sestavi 3-uhelnik PROHRAVA"!
120 PRINT "Vkladejte jen velka pismena!"
130 PRINT
140 REM Priprava dat
150 S$="ABACADAEBCBDBECDCEDE"
160 T$="ABCABDABEACDACEADEBCDBCEBDECDE"
170 DATA 1,2,3,1,4,5,2,4,6,3,5,6,1,7,8
180 DATA 2,7,9,3,8,9,4,7,10,5,8,10,6,9,10
190 FOR I=1 TO 10
200 FOR J=1 TO 3
210 READ R(I,J)
220 NEXT J
230 NEXT I
240 FOR I=1 TO 10
250 C(I)=1
260 S(I)=0
270 FOR J=1 TO 3
280 T(I,J)=0
290 NEXT J
300 NEXT I
```

```
310 U$="Chcete zacinat? (A/N)"
320 GOSUB 1070
330 IF Q=1 THEN 470
340 K=INT(10*RND(0))+1
350 REM Zacatek dvojtahu
360 L=1
370 S(K)=L
380 GOSUB 1170
390 PRINT "Volim stranu: ";MID$(S$,2*K-1,2)
400 GOSUB 1270
410 REM Je jeste volna strana?
420 FOR I=1 TO 10
430 IF S(I)=0 THEN 470
440 NEXT I
450 GOTO 940
460 REM Volba hrace
470 INPUT "Vlozte stranu 3-uhelnika";V$
480 IF LEN(V$) <> 2 THEN 560
490 IF LEFT$(V$,1) < RIGHT$(V$,1) THEN 530
500 IF LEFT$(V$,1) = RIGHT$(V$,1) THEN 560
510 Q$=LEFT$(V$,1)
520 V$=RIGHT$(V$,1)+Q$
530 FOR I=2 TO 20 STEP 2
540 IF V$=MID$(S$,I-1,2) THEN 580
550 NEXT I
560 PRINT "Chybne zadani!";CHR$(7)
570 GOTO 470
580 K=I/2
590 IF S(K)=0 THEN 620
600 PRINT "Strana ";V$;" uz byla vybrana!";CHR$(7)
610 GOTO 470
620 L=-1
630 S(K)=L
640 GOSUB 1170
650 GOSUB 1270
660 REM Tah pocitace
670 Y=100
680 Z=0
690 FOR I=1 TO 10
700 C(I)=0
710 IF S(I) <> 0 THEN 890
720 FOR J=1 TO 3
730 M=R(I,J)
740 IF T(M,1)=0 THEN 820
750 IF T(M,3) <> 0 THEN 850
760 X=T(M,1)+T(M,2)+3
770 ON X GOTO 780,800,820,780,840
780 C(I)=C(I)+4
790 GOTO 850
800 C(I)=C(I)+2
810 GOTO 850
820 C(I)=C(I)+1
830 GOTO 850
840 C(I)=C(I)+20
850 NEXT J
```

```
 860 IF C(I) >= Y THEN 890
 870 Y=C(I)
 880 Z=I
 890 NEXT I
 900 IF Y=100 THEN 940
 910 K=Z
 920 GOTO 360
 930 REM Tisk vysledku
 940 PRINT "Nerozhodne!"
 950 GOTO 1010
 960 IF T(I,1) <> 1 THEN 990
 970 PRINT "VYHRAL JSTE!"
 980 GOTO 1000
 990 PRINT "PROHRAL JSTE!"
1000 PRINT "Trojuhelnik: ";MID$(T$,3*I-2,3)
1010 PRINT
1020 U$="Chcete novou hru? (A/N)"
1030 GOSUB 1070
1040 IF Q=1 THEN 240
1050 PRINT "Konec hry."
1060 END
1070 REM Souhlas
1080 PRINT U$
1090 Q$=INKEY$: IF Q$="" THEN 1090
1100 Q=0
1110 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
1120 Q=1
1130 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
1140 PRINT "Musite odpovedet podle pravidel:"
1150 PRINT "ANO = 1 nebo A,  NE = 0 nebo N."
1160 GOTO 1070
1170 REM Strany trojuhelnika
1180 FOR I=1 TO 3
1190 M=R(K,I)
1200 FOR J=1 TO 3
1210 IF T(M,J) <> 0 THEN 1240
1220 T(M,J)=I
1230 GOTO 1250
1240 NEXT J
1250 NEXT I
1260 RETURN
1270 REM Vyhodnoceni
1280 FOR I=1 TO 10
1290 IF T(I,3)=0 THEN 1310
1300 IF T(I,1)=T(I,2) AND T(I,2)=T(I,3) THEN 960
1310 NEXT I
1320 RETURN
```

Příklad

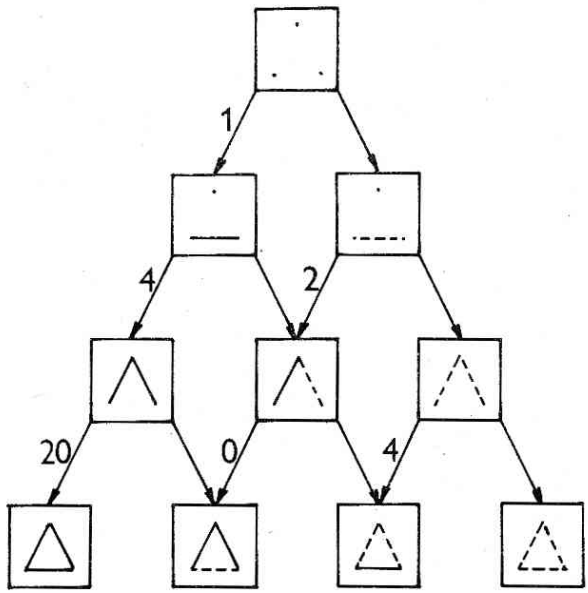
```
HRA TROJUHELNIKY NA PETIUHELNIKU
Petiuhelnik ma 5 vrcholu: ABCDE.
Kazde 3 vrcholy tvori 3-uhelnik.
Takovych trojuhelniku je 10.
V kazdem tahu se zadava strana.
V tazich se stridate s pocitacem.
Kdo sestavi 3-uhelnik PROHRAVA!
Vkladejte jen velka pismena!

Chcete zacinat? (A/N):
0
Volim stranu: BD
Vlozte stranu 3-uhelnika:
AE

Volim stranu: AC
Vlozte stranu 3-uhelnika:
CE

Volim stranu: BE
Vlozte stranu 3-uhelnika:
BC

Volim stranu: AB
Vlozte stranu 3-uhelnika:
CD

Volim stranu: AD
VYHRAL JSTE!
Trojuhelnik: ABD

Chcete novou hru? (A/N):
0
Konec hry.
```

V programu se používají data o všech stranách trojúhelníků (jsou v textové proměnné S$), data o všech trojúhelnících (jsou v textové proměnné T$) a data o zastoupení úseček v těchto trojúhelnících. Ta jsou v matici R.

△	Strana									
	AB	AC	AD	AE	BC	BD	BE	CD	CE	DE
ABC										
ABD										
ABE										
ACD										
ACE										
ADE										
BCD										
BCE										
BDE										
CDE										

Obr. 9 Zastoupení úseček v trojúhelnících

Na obrázku 9 je v tabulce zaznamenáno, jak se jednotlivé úsečky vyskytují v trojúhelnících.



Obr. 10 Hodnocení tahů počítačem

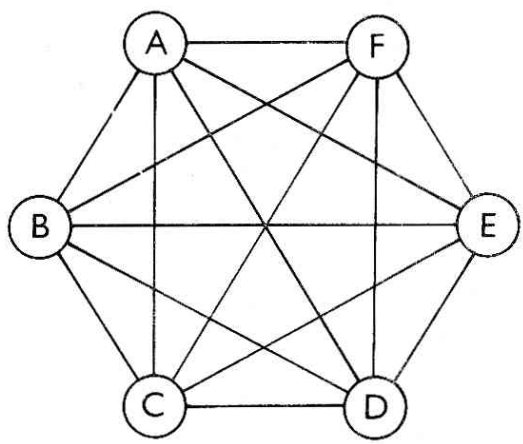
Zajímavý je způsob, jímž program počítače vyhodnocuje svůj vlastní tah. K tomu jsou na obrázku 10 zachyceny všechny možné situace, které se vyskytují z hlediska každého trojúhelníka. Jako nejvýhodnější je považována situace takového trojúhelníka, v němž jsou již dvě strany zadány, každá z nich jedním ze soupeřů. Je zřejmé, že ani hráč, ani počítač nemůže v této situaci nic ztratit.

Ostatní situace jsou hodnoceny hůř. Nejhůř - 20 penalizačními body - je hodnocena situace trojúhelníka, v němž jsou již dvě strany využity a obě počítačem. Takový tah počítač provede jedině tehdy, nemá-li žádnou jinou možnost. Podrobnosti o penalizační funkci a jejím používání se můžete pokusit vyčíst z programu.

19.6 Trojúhelníky v šestiúhelníku

Předcházející hra Trojúhelníky v pětiúhelníku je pouze jednodušší formou původní hry, která se nazývá Trojúhelníky v šestiúhelníku. Princip samotné hry je podobný, hra je však pro hráče obtížnější zejména proto, že při jejím vyhodnocování je potřeba větší pečlivosti.

Zde se opět v tazích střídají hráč s počítačem. Doplňují střídavě úsečky spojující šest vrcholů šestiúhelníku.



Obr. 11 Trojúhelníky v šestiúhelníku

Takových úseček je tentokrát celkem 15. Můžete se o tom přesvědčit na obrázku 11. Zde opět prohrává ten, kdo první sestrojí trojúhelník.

Možných trojúhelníků pro šest vrcholů šestiúhelníku je tentokrát dvacet. V tom je právě hra na šestiúhelníku obtížnější. Vrcholy pětiúhelníku jsou zde označeny velkými písmeny A, B, C, D, E a F. Všechno ostatní je obdobné programu předcházejícímu.

Hodně úspěchů při hře.

Program: Trojúhelníky v šestiúhelníku

```
 10 REM Trojuhelnik 6
 20 REM Trojuhelniky v sestiuhelniku
 30 DIM C(15),R(15,4),S(15),T(20,3)
 40 CLS
 50 PRINT "HRA TROJUHELNIKY V SESTIUHELNIKU"
 60 PRINT "Sestiuhelnik ma 6 vrcholu: ABCDEF"
 70 PRINT "Kazde 3 vrcholy tvori trojuhelnik."
 80 PRINT "Takovych trojuhelniku je 20."
 90 PRINT "V kazdem tahu se zadava strana."
100 PRINT "V tazich se stridate s pocitacem."
110 PRINT "Kdo sestavi 3-uhelnik PROHRAVA"!
120 PRINT "Vkladejte jen velka pismena!"
130 PRINT
140 REM Priprava dat
150 S=15: T=20: U=4
160 S$="ABACADAEAFBCBDBEBFCDCECFDEDFEF"
170 T$="ABCABDABEABFACDACEACFADEADFAEF"
180 T$=T$+"BCDBCEBCFBDEBDFBEFCDECDFCEFDEF"
190 DATA 1,2,3,4,1,5,6,7,2,5,8,9,3,6,8,10,4,7,9
200 DATA 10,1,11,12,13,2,11,14,15,3,12,14,16,4
210 DATA 13,15,16,5,11,17,18,6,12,17,19,7,13,18
220 DATA 19,8,14,17,20,9,15,18,20,10,16,19,20
230 FOR I=1 TO S
240 FOR J=1 TO U
250 READ R(I,J)
260 NEXT J
270 NEXT I
280 FOR I=1 TO S
290 C(I)=1
300 S(I)=0
310 NEXT I
320 FOR I=1 TO T
330 FOR J=1 TO 3
340 T(I,J)=0
350 NEXT J
360 NEXT I
370 U$="Chcete zacinat? (A/N)"
380 GOSUB 1100
390 IF Q=1 THEN 520
400 K=INT(S*RND(0))+1
410 REM Zacatek dvojtahu
420 L=1
430 S(K)=L
440 GOSUB 1200
450 PRINT "Volim stranu: ";MID$(S$,2*K-1,2)
460 GOSUB 1300
470 REM Je jeste volna strana?
480 FOR I=1 TO S
490 IF S(I)=0 THEN 520
500 NEXT I
```

```
510 GOTO 970
520 INPUT "Vlozte stranu 3-uhelnika";V$
530 IF LEN(V$)  ..  2 OR LEFT$(V$,1)=RIGHT$(V$,1) THEN 590
540 IF LEFT$(V$,1)  .= RIGHT$(V$,1) THEN 560
550 V$=RIGHT$(V$,1)+LEFT$(V$,1)
560 FOR I=2 TO 2*S STEP 2
570 IF V$=MID$(S$,I-1,2) THEN 610
580 NEXT I
590 PRINT "Chybne zadani!";CHR$(7)
600 GOTO 530
610 K=I/2
620 IF S(K)=0 THEN 650
630 PRINT "Strana ";V$;" uz byla vybrana!";CHR$(7)
640 GOTO 520
650 L=-1
660 S(K)=L
670 GOSUB 1200
680 GOSUB 1300
690 REM Tah pocitace
700 Y=100
710 Z=0
720 FOR I=1 TO S
730 C(I)=0
740 IF S(I) <> 0 THEN 920
750 FOR J=1 TO U
760 M=R(I,J)
770 IF T(M,1)=0 THEN 850
780 IF T(M,3) <> 0 THEN 880
790 X=T(M,1)+T(M,2)+3
800 ON X GOTO 810,830,850,810,870
810 C(I)=C(I)+4
820 GOTO 880
830 C(I)=C(I)+2
840 GOTO 880
850 C(I)=C(I)+1
860 GOTO 880
870 C(I)=C(I)+20
880 NEXT J
890 IF C(I) >= Y THEN 920
900 Y=C(I)
910 Z=I
920 NEXT I
930 IF Y=100 THEN 970
940 K=Z
950 GOTO 420
960 REM Tisk vysledku
970 PRINT "Nerozhodne!"
980 GOTO 1040
990 IF T(I,1) <> 1 THEN 1020
```

```
1000 PRINT "VYHRAL JSTE!"
1010 GOTO 1030
1020 PRINT "PROHRAL JSTE!"
1030 PRINT "Trojuhelnik: ";MID$(T$,3*I-2,3)
1040 PRINT
1050 U$="Chcete novou hru? (A/N)"
1060 GOSUB 1100
1070 IF Q=1 THEN 280
1080 PRINT "Konec hry."
1090 END
1100 REM Souhlas
1110 PRINT U$
1120 Q$=INKEY$: IF Q$="" THEN 1120
1130 Q=0
1140 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
1150 Q=1
1160 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
1170 PRINT "Musite odpovedet podle pravidel:"
1180 PRINT "ANO = 1 nebo A,  NE = 0 nebo N."
1190 GOTO 1100
1200 REM Strany trojuhelnika
1210 FOR I=1 TO U
1220 M=R(K,I)
1230 FOR J=1 TO 3
1240 IF T(M,J) <> 0 THEN 1270
1250 T(M,J)=L
1260 GOTO 1280
1270 NEXT J
1280 NEXT I
1290 RETURN
1300 REM Vyhodnoceni
1310 FOR I=1 TO T
1320 IF T(I,3)=0 THEN 1340
1330 IF T(I,1)=T(I,2) AND T(I,2)=T(I,3) THEN 990
1340 NEXT I
1350 RETURN
```

Příklad

```
HRA TROJUHELNIKY V SESTIUHELNIKU
Petiuhelnik ma 6 vrcholu: ABCDEF
Kazde 3 vrcholy tvori 3-uhelnik.
Takovych trojuhelniku je 20.
V kazdem tahu se zadava strana.
V tazich se stridate s pocitacem.
Kdo sestavi 3-uhelnik PROHRAVA!
Vkladejte jen velka pismena!
```

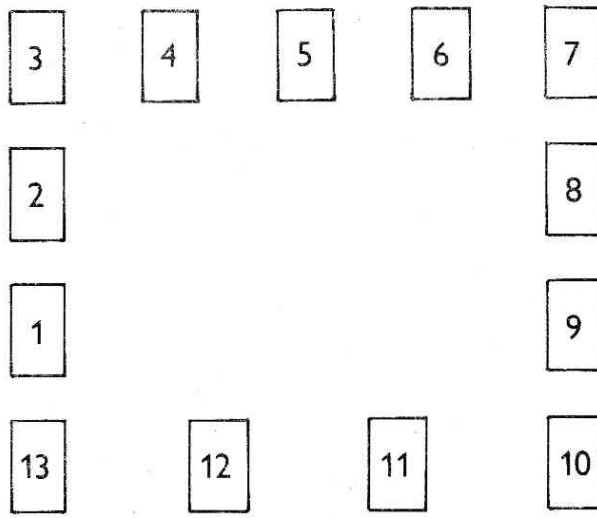
```
Chcete zacinat? (A/N):
0
Volim stranu: BD
Vlozte stranu 3-uhelnika:
AE

Volim stranu: AC
Vlozte stranu 3-uhelnika:
CE

Volim stranu: BE
Vlozte stranu 3-uhelnika:
BC
Volim stranu: AB
Vlozte stranu 3-uhelnika:
CD

Volim stranu: AD
VYHRAL JSTE!
Trojuhelnik: ABD

Chcete novou hru? (A/N):
0
Konec hry.
```

Princip programu je obdobný s předchozím. Jen se všechny informace rozšiřují a tím se hra stává méně "průhledná". Doporučujeme začít se snadnější hrou Trojúhelníky v pětiúhelníku. V obou případech je vhodné zakreslovat si celou měnící se situaci průběžně na papír a rozlišovat přitom ty úsečky, které jste vybral sám, od úseček, které byly vybrány počítačem.

Pak můžete zjistit, že ani nemusí záležet na tom, kdo začíná. Na rozdíl od her s přesně formulovanou strategií zde tomu tak není a šance jsou otevřenější.

19.7 Šťastná třináctka

Šťastná třináctka patří do kategorie her, podobných hře NIM. I zde se odebírají předměty (zde to je třináct figurek, kaménků nebo karet) a vyhrává ten, kdo vezme poslední předmět. Jediný rozdíl je v tom, že předměty nejsou na hromádce nebo ve sloupci, ale jsou uspořádány v kruhu.



Obr. 12 Šťastná třináctka

Uspořádání je na obrázku 12. Pravidla pro odebírání předmětů jsou tato: Je možné odebrat jeden nebo nejvíc dva předměty. Jestliže se odebírá jeden předmět, není jeho odebrání nijak podmiňováno. Odebírají-li se však dva předměty, musí to být takové, které spolu sousedí. Sousedí např. 1 a 2, nebo 8 a 9, ale také 1 a 13.

V tazích se opět střídáte s počítačem. Před zahájením můžete rozhodnout, chcete-li začínat, anebo nechat první tah na počítači. Toto rozhodnutí může být pro vítězství ve hře podstatné. Také zdaleka nemusí vždy platit, že je první tah výhodný. Můžete ovšem i v této hře vyzkoušet známou taktiku: Nechat počítač začínat a použít se z jeho tahů, tak jak je to ostatně možné i v dalších hrách, kde je vám počítač partnerem.

Program: Šťastná třináctka

```
 10 REM 13
 20 REM Odebirani karet z kruhu
 30 REM Hra pro hrace a pocitac
 40 DIM D(2,5),K(13)
 50 CLS
 60 PRINT
 70 PRINT "   ODEBIRANI KARTICEK Z KRUHU"
 80 GOSUB 1220
 90 GOSUB 710
100 PRINT "Na obrazovce je 13 cisel v kruhu."
110 PRINT "Cisla zastupuji karticky."
120 PRINT "Bere se jedna nebo dve sousedni."
130 PRINT "Kdo bezme posledni kartu vyhrava."
140 PRINT "V tazich s pocitacem se stridate."
150 PRINT
160 N=0
170 T$="Muze se zacit?"
180 GOSUB 600
190 IF Q=0 THEN 570
200 CLS
210 GOSUB 710
220 GOSUB 1270
230 Z=A
240 IF B <> 13 ABD B > A THEN Z=B
240 FOR J=1 TO 5
260 GOSUB 1370
270 D(1,J)=K(Z)
280 NEXT J
290 GOSUB 1370
300 X=K(Z)
310 IF B > 0 THEN 340
320 GOSUB 1370
330 Y=K(Z)
340 FOR I=1 TO 5
350 GOSUB 1370
360 D(2,J)=K(Z)
370 NEXT J
380 CLS
390 PRINT "Odebiram kartu";X
400 K(X)=0
410 N=N+1
420 IF B > 0 THEN 460
430 PRINT "Odebiram kartu";Y
440 K(Y)=0
450 N=N+1
460 GOSUB 710
470 GOSUB 1270
480 GOSUB 1410
490 IF N < 13 THEN 460
500 GOSUB 710
```

```
510 PRINT "PROHRAL JSTE!"
520 T$="Chcete si jeste zahrat?"
530 GOSUB 600
540 IF Q=0 THEN 570
550 GOSUB 1220
560 GOTO 200
570 PRINT
580 PRINT "Konec."
590 END
600 REM Souhlas
610 PRINT T$
620 Q$=INKEY$
630 IF Q$="" THEN 620
640 Q=0
650 IF Q$="N" OR Q$="0" OR Q$="O" THEN RETURN
660 Q=1
670 IF Q$="A" OR Q$="1" OR Q$="Y" THEN RETURN
680 PRINT "Musite odpovedet podle pravidel:"
690 PRINT "ANO = 1 nebo A,  NE = 0 nebo N."
700 GOTO 600
710 REM Vylozeni karticek
720 PRINT
730 PRINT "     ";
740 FOR I=3 TO 7
750 PRINT " ";
760 IF K(I)=0 THEN PRINT " - ";
770 IF K(I) > 0 THEN PRINT I;
780 NEXT I
790 PRINT
800 FOR I=8 TO 9
810 PRINT "      ";
820 IF K(10-I)=0 THEN PRINT " - ";
830 IF K(10-I) > 0 THEN PRINT 10-I;
840 PRINT TAB(23);
850 IF K(I)=0 THEN PRINT " -"
860 IF K(I) > 0 THEN PRINT I
870 NEXT I
880 PRINT "     ";
890 FOR I=13 TO 10 STEP -1
900 PRINT " ";
910 IF K(I)=0 THEN PRINT " -- ";
920 IF K(I) > 0 THEN PRINT I;
930 NEXT I
940 PRINT
950 PRINT
960 RETURN
970 REM Odebirani karet
980 IF C$="0" THEN RETURN
990 X=INT(SCS(C$))
```

```
1000 IF X > 0 AND X < 14 THEN 1030
1010 PRINT "Muzete vybrat pouze 1 - 13!"
1020 GOTO 1170
1030 IF K(X) > 0 THEN 1060
1040 PRINT "Karta jiz byla odebrana!"
1050 GOTO 1170
1060 IF A > 0 THEN 1090
1070 IF A=0 THEN A=X
1080 GOTO 1140
1090 IF ABS(A-X)=1 THEN 1130
1100 IF ABS(A-X)=12 THEN 1130
1110 PRINT "Nevybral jste SOUSEDNI karty!"
1120 GOTO 1170
1130 B=X
1140 K(X)=0
1150 N=N+1
1160 RETURN
1170 PRINT CHR$(7)
1180 PRINT "Vlozte odebirane cislo ZNOVU!"
1190 INPUT C$
1200 IF A > 0 THEN 970
1210 GOTO 990
1220 REM Naplneni
1230 FOR I=1 TO 13
1240 K(I)=I
1250 NEXT I
1260 RETURN
1270 REM Odebirani karet
1280 A=0
1290 B=0
1300 PRINT "Vlozte PRVNI odebirane cislo";
1310 INPUT C$
1320 GOSUB 990
1330 PRINT "Vlozte DRUHE cislo nebo NULU";
1340 INPUT C$
1350 GOSUB 970
1360 RETURN
1370 REM Nacitani indexu
1380 Z=Z+1
1390 IF Z > 13 THEN Z=1
1400 RETURN
1410 REM Tah pocitace
1420 CLS
1430 FOR I=1 TO 2
1440 FOR J=1 TO 5
1450 IF D(I,J) <> A THEN 1520
1460 D(I,J)=0
1470 K=3-I
1480 X=D(K,J)
1490 K(X)=0
1500 PRINT "Odebiram kartu";X
```

```
1510 N=N+1
1520 IF B=0 THEN 1600
1530 IF D(I,J)   ..  B THEN 1600
1540 D(I,J)=0
1550 K=3-I
1560 Y=D(K,J)
1570 K(Y)=0
1580 PRINT "Odebiram kartu";Y
1590 N=N+1
1600 NEXT J
1610 NEXT I
1620 RETURN
```

Ukázka hry

```
Na obrazovce je 13 cisel v kruhu.
Cisla zastupuji karticky.
Bere se jedna nebo dve sousedni.
Kdo vezme posledni kartu vahrava.
V tazich se stridate s pocitacem.

Chcete zacinat?
A
          3   4   5   6   7
          2                8
          1                9
          13   12   11   10

Vlozte PRVNI odebirane cislo: 7
Vlozte DRUHE odebirane cislo: 0

Odebiram kartu 1
Odebiram kartu 13

          3   4   5   6   -
          2                8
          -                9
          --   12   11   10

Vlozte PRVNI odebirane cislo: 4
Vlozte DRUHE odebirane cislo: 5

Odebiram kartu 11
Odebiram kartu 10

          3   -   -   6   -
          2                8
          -                9
          --   12   --   --

Vlozte PRVNI odebirane cislo: 2
Vlozte DRUHE odebirane cislo: 3
```

```
Odebiram kartu 8
Odebiram kartu 9

          -   -   -   6   -
          -                 -
          -                 -
          --   12   --   --

Vlozte PRVNI odebirane cislo: 6
Vlozte DRUHE odebirane cislo: 0

Odebiram kartu 12

          -   -   -   -   -
          -                 -
          -                 -
          --   --   --   --

PROHRAL JSTE!

Chcete hrat dalci hru?
:N

Konec hry.
```

I pro tuto hru vám můžeme dát dobrou radu. Pokud neumíte odhalit strategii, která vede k vítězství, pokuste se poučit z tahů počítače.

V této hře záleží na tom, který hráč začíná. Jestliže to nedokážete poznat, můžete se poučit tím, že budete hrát víc her, a v nich někdy začínat sami a někdy nechat začínat počítač.

Strategie není příliš obtížná. Dá se samozřejmě také vyčíst z programu. Pak můžete přistoupit ke tvorbě podobných programů, lišících se počtem karet v kruhu, počtem možných odebíraných i pravidly pro jejich odebírání.

Přesto, že se jedná o zábavný program, můžete se tak naučit o programování a analýze problémů leckdy víc, než při programování jednoduchých reálných úloh.

PŘÍLOHY

Seznam hlášení o chybách PŘÍLOHA 1

00	K příkazu NEXT chybí příkaz FOR
01	Syntaktická chyba
02	Použít příkazu RETURN, aniž by byl použit příkaz GOSUB
03	Nedovolený příkaz v ne číslovaném řádku. Příkaz READ nenašel data
04	Použitý parametr je větší než 32767
05	Číselné přeplnění
06	Zaplnění paměti vyhražené pro interpret jazyka BASIC
07	Odkaz na neexistující řádek
08	Překročení dovolené nebo nadeklarované velikosti indexu
09	Opakovaná deklarace téhož pole
10	Dělitel je roven nule
11	Neočíslovaný příkaz INPUT nebo DEF FN
12	Nedovolená operace s řetězcem
13	Přeplnění oblasti CLEAR nebo USR
14	Řetězec. Alfanumerická proměnná má větší rozsah než 255 znaků
16	V dané situaci nelze pokračovat příkazem CONT
17	Pro použitou operaci chybí příkaz DEF FN
18	Parametr je větší než 65535
19	Překročen parametr v příkazu PLOT nebo PRINT
20	Pokus o zrušení neexistujícího pole
21	Identifikátor proměnné nezačíná písmenem
22	Překročení počtu parametrů definované operace. Počet skutečných parametrů definované operace není roven počtu formálních parametrů

Seznam řídicích znaků PŘÍLOHA 2

Řídicí znaky (řz) lze použít v příkazu:

PRINT CHR$(řz)

7	zvukový signál
8	posun kurzoru vlevo
9	posun kurzoru po osmi znacích
12	posun kurzoru na řádek 0, sloupec 0
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

Seznam řídicích 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, ve 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 [	zrušení povelu AUTO, automatické řádkování

Přehled povelů, příkazů, funkcí a logických operátorů

PŘÍLOHA 3

Klíčové slovo	funkce
(1)	Povely
AUTO	automatické číslování programových řádků
CONT	pokračování programu nebo výpisu, po předchozím použití příkazu STOP nebo END
LIST	výpis programu
MEM	výpis volného místa v paměti (v bytech)
MLOAD	čtení programu z kazety do vnitřní paměti
MSAVE	uložení programu z vnitřní paměti na kazetu
(2)	Příkazy
READ/ DATA	čtení dat (konstant) ze seznamu příkazu DATA a jejich ukládání do proměnných v seznamu příkazu READ
RESTORE	znovu čtení konstant ze seznamu v části DATA
END	ukončí zpracování programu
DIM	deklarace proměnných; vymezení místa ve vnitřní paměti
FREE	zruší dimenzování pole
DEF FN	definuje uživatelskou funkci
FN	vyvolá funkci definovanou příkazem DEF FN
FOR TO STEP	příkaz cyklu
NEXT	ukončení příkazu cyklu
GOTO	příkaz skoku
GOSUB	vyvolání podprogramu
RETURN	ukončení příkazů podprogramu
IF THEN	podmíněný příkaz (podmínka)
INKEY$	vložení hodnoty znaku stisknuté klávesy do řetězce

Klíčové slovo	funkce
INPUT	příkaz vstupu
LET	přiřazovací příkaz
ON GOTO	skok určený výpočtem (přepínač)
ON GOSUB	vyvolání podprogramu přepínačem
PRINT	příkaz výstupu (zobrazení)
PLOT	zobrazí grafický znak
UNPLOT	zruší zobrazení grafického znaku
REM	poznámka
SCRATCH	vymaže vložený program
(3)	Funkce
PI	Ludolfovo číslo
WAIT	přerušení zpracování programu
LEN	počet znaků v řetězci
LEFT$	počet znaků řetezce zleva
RIGHT$	počet znaků řetězce zprava
MID$	vytvoří podřetězec
VAL	konverze abecedněčíselné hodnoty na hodnotu číselnou
STR$	inverzní funkce k VAL
ABS	absolutní hodnota
ASC	dekadické číslo odpovídají v kódu ASCII vloženému znaku nebo prvnímu znaku řetězce
ATN	arctangens
COS	cosinus
EXP	e^x
HEX	dekadické vyjádření hexadecimálního čísla
CHR$	znak odpovídají v ASCII dané hodnotě
INT	celočíselná dolní hodnota výrazu
LOG	přirozený logaritmus
RND	generátor pseudonáhodných čísel
SIN	sinus

Klíčové slovo	funkce
(4)	Operátory
NOT	negace
AND	logický součin
OR	logický součet
(5)	Příkazy pro práci v monitoru
CALL	vyvolání podprogramu ve strojovém kódu
PEEK	obsah paměťového místa
POKE	zápis do paměťového místa
CLEAR	nulování číselných proměnných, přiřazení znaků "mezera" abecedněčíselným proměnným, zrušení všech deklarací

Tabulka kódu ASCII PŘÍLOHA 4

0	NUL	32	␣	64	@	96	
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	↑	126	~
31	US	63	?	95	←	127	DEL

LITERATURA

1. Coan, J. S.: Basic BASIC. Rochelle Park, New Jersey, Hayden Book Company 1978 (druhé vydání).
2. Coan, J.S.: Advanced BASIC. Rochelle Park, New Jersey, Hayden Book Company 1977.
3. Dávid, A.: Programy pre malé kalkulátory. Bratislava, Alfa 1985.
4. Jedlička, Z. - Feil, M.: BASIC pro začátečníky, příručka k počítači IQ 151. Praha, Komenium 1985.
5. Knappová, M.: Jak se bude jmenovat? Praha, Academia 1978
6. Kollert, E.: Programování počítače IQ 151 v jazyku BASIC. Praha, Komenium 1984.
7. Kollert, E.: Výpočetní technika. Praha, SNTL 1985.
8. KVIZ - časopis
9. Machačka, I. - Pavlů, J.: Programování v jazvku Basic. Praha, SNTL 1985 .
10. Laga, J. - Ježowicz, E.: Základy algoritmizace. Praha, VŠE 1984.
11. Morávek, J.: Složitost výpočtů a optimální algoritmy. Praha, Academia, 1984
12. Osobný mikropočítač PMD-85, Příručka pre používatela, I - návod na použitie, Tesla.
13. Schwartz, B. L., red.: Mathematical Solitaires and Games. New York, Baywood 1978
14. Smida, J., a kol.: Matematika pro I. ročník gymnázií. Praha, SPN, 1984
15. Vejmola, S.: Operační výzkum (skriptum). Praha, SPN 1984.
16. Vejmola, S.: Konec záhady hlavolamů. Praha, SPN 1986.
17. Vejmola, S.: Hry s počítačem. Praha, SPN (v tisku).
18. Výrut, K.: Programování počítače WANG 2200 MVP, Příručka pro školení. Praha, ČSVTS, VŠE 1983.
19. Výrut, K. - Němec, M.: Základy programování. Praha, SPN 1985, 3., upravené vydání.
20. Výrut, K. - Václavíková, D.: Programování minipočítačů

WANG v jazyce Basic. Praha, SPN 1986.
21. Výrut K. - Vejmola, S.: Programy pro počítač IQ 151. Praha, SPN 1986.

Ing. Stanislav Vejmola, CSc.

Ing. Karel Výrut, CSc.

Programy pro počítač IQ 151

Obálku navrhl Václav Konečný
Vydalo Státní pedagogické nakladatelství, n. p., Praha,
roku 1987 jako svou publikaci č. 7-42-11/1
Edice Pomocné knihy pro žáky
Odpovědná redaktorka Ing. Zdeňka Valentová a Zora Vedralová
Výtvarná redaktorka Jitka Baliharová
Technická redaktorka Jitka Greplová
Z nové sazby písmem Public
vytiskly Tiskařské závody, n. p., provoz 52, Praha 1
Formát papíru 70 × 100 cm
Počet stran 208
AA 9,16 (8,61 AA textu, 0,55 AA grafiky) — VA 10,03
Náklad 40 000 výtisků
Tematická skupina a podskupina 01/12
1. vydání
Cena vázaného výtisku Kčs 14,00
505/21,826

14 - 174 - 87 Kčs 14,00

SPN 7 - 42 - 11/1

14 - 174 - 87
01/12 Kčs 14,00

Stanislav Vejmola, Karel Výrut - Programy pro počítač IQ 151