

A K T U A L I T Y 1 9

B A S I C Q B A S I C

Obsah:	strana:
Slovo vydavatele	2
Nabídka C2717-PMD: SWD 72 - Testy IQ	3
QBasic: práce s proměnnými	5
přiřazení hodnoty proměnné	10
číselné funkce	11
řetězcové funkce	14
struktura a zápis programu	15
příkazy	16
chybová hlášení	18
Nabídka C2717-PMD: SWD 73 - Hry 10	21
SWD 74 - Matematika 7/8 ..	22
SWD 75 - Čeština 5	23
Informace	24

Vážená čtenářko,
vážený čtenáři.

Další číslo Aktualit nevybočuje z jejich ustáleného obsahu. Především přináší další 4 tituly v nabídce programů pro C2717/PMD a tak počet souborů programů pro tyto počítače dosáhl již čísla 75, v nichž je asi 650 jednotlivých programů. Tento úctyhodný počet vznikl především zásluhou externích spolupracovníků ze škol prakticky v celé republice. Dovolte mi, abych jim i touto cestou znovu poděkoval nejen za sebe, ale i ty z vás, kteří jejich programy používáte nebo jste se jimi dali inspirovat při výuce.

Řada programů pochází i od neznámých autorů, kteří je kdysi dávno dali k dispozici počítačové veřejnosti, čímž se z nich staly programy volně šířitelné, různě upravované a doplňované. Patří sem i některé ze souboru HRY-10, kde jsou mimo dvou variant hry DÁMA, dostihů a seskoku parašutistů i 3 hry podnikatelského typu charakteristické pro současnou ekonomickou atmosféru.

Vyšším třídám ZŠ poslouží 23 matematických testovacích programů Z.Rubého z oboru mocnin a práce s matematickými výrazy. Jako český jazyk hrou by se dalo mluvit o osmisměrkách E.Kabrhelové, které doplňují testovací programy, na jejichž závěr je za odměnu nakreslen obrázek, nebo si lze kreslit jednoduchým editorem.

Soubor TESTY IQ je především seznamovacího typu - jak mohou vypadat otázky těchto testů a jak se řeší, ale zas tak jednoduché to není. I při nápovědě je nutno přemýšlet, neboť ta ukazuje jen cestu k řešení problému a nenabízí jednoznačnou odpověď. Protože se v testech používají jednoduché matematické operace na úrovni základní školy nebo slovní a písmenné problémy, lze je doporučit pro doplnění výuky matematiky i českého jazyka.

Nabídka pro PC zatím není rozšířena, pracuji na převodu některých programů z C2717. Pokud si myslíte, že by některé měly být dříve k dispozici než za rok nebo dva, napište mi svůj názor. Pokud jste nějaké programy už vytvořili, pošlete nám je k posouzení a možnému zařazení do nabídky. Váš podíl na prodeji pak bude předmětem smlouvy o využití autorských práv.

Třetí pokračování kursu QBasicu je tentokrát podrobněji věnováno proměnným, ukazuje, jak se liší od starších variant jazyka Basic. Téměř standardní jsou přehledně uvedené číselné a řetězové funkce, zatímco počet možných chybových hlášení se podstatně rozšířil vzhledem k širokým možnostem jazyka QBasic. Tato hlášení využijete při svých pokusech s programováním v prostředí QBasicu. Podle lekcí tohoto kursu se rýsuje i obsah příručky Úvod do jazyka QBasic, která vyjde na podzim v rozsahu asi 120 stran a mohla by se stát jednou z příruček pro předmět INFORMATIKA. Její cena by mohla být kolem 40 Kč a bude záviset na počtu objednaných kusů zákazníkem. Přijímáme nezávazné objednávky a náměty, co do příručky ještě zařadit, předpokládáme i vydání diskety s příklady.

Těším se na spolupráci i s Vámi

Pavel Hlaváček

Programové vybavení CONSUL 2717/PMD85-2,3

Kč 198,-

SWD72/SWK72

TESTY IQ

SWD72/SWK72

Autor: Pavel Hlaváček

Soubor programů pro seznámení se s různými druhy testů, které se mohou vyskytovat v tzv. IQ-testech. Je známo, že inteligence je těžko postižitelný pojem, všezahrnující směr rozumových schopností, zatímco k životu je třeba o mnoho víc, než jen schopnost uspět v určitém druhu testů. Testy stanoví schopnost lidí, řešit určitý druh problémů, a tak jim naznačit, jak si asi budou počínat i v budoucnosti. Často se proto používají i k hodnocení školáků.

Následující testy tvoří soubory podobných úloh, jejichž cílem není testovat inteligenci, nejsou časově limitovány jako standardní testy, naopak poskytují nápovědu, jak je řešit. Nápovědy jsou většinou jen naznačením, jak problém řešit, i při jejich použití je nutno přemýšlet, neumožňují bezmyšlenkovitě projít testy. Proto jsou programy spíše myšlenkovým tělocvikem.

Zakladním pravidlem pro zvyšování IQ je zkusit samostatně vyřešit nějaké úlohy z IQ testů (vědět, co se může v těchto testech vyskytnout a jak se to asi řeší). Proto je v závěru všech testů, obsahujících kolem 30 otázek, uvedena bilance řešení: počet bodů z celkové možných, počet využitých nápověd, počet chyb a úspěšnost s poznámkou, jak je možné se zlepšit.

Každá cesta k úspěchu začíná poznáním pravdy především o sobě samém. Jednou z cest je vyzkoušet si testy inteligence. Důležité je vědět, jak se tyto testy tvoří a především, jak se řeší.

MYSLENI 1

Úlohy s číselnými řadami typu, jaké je další číslo uvedené řady s otazníkem na některém místě. Mezi každým číslem řady a číslem po něm následujícím je určitý vztah, který je nutno nejprve odhalit a potom uplatnit. Jsou uvedeny řady s chybějícím číslem na konci, začátku nebo uprostřed s uvedením postupu řešení. Jsou uvedeny příklady řad se stejnými intervaly mezi členy, rostoucími nebo klesajícími intervaly, sečítáním sousedních členů řady, násobením nebo dělením členů řady konstantou, mocnina předchozího členu řady, mocninné řady, uplatnění dvoukrokových formulí se dvěma matematickými operacemi, dvojité řady. Většina úloh užívá jen jednoduché matematické operace na úrovni základní školy.

MYSLENI 2

Početní operace ve dvojicích čísel v mřížce, v kruhu nebo v tabulce. Podobně početní operace v trojicích vedle sebe, nad sebou s chybějícím číslem na konci nebo uprostřed. Magický čtverec a úlohy typu PODOBY (podobné operace mezi čísly v různých obrazcích).

MYSLENI 3

Písmenné (alfabetické) řady, jejichž řešení je podobné číselným řadám, jenom je nutno písmena v anglické abecedě (bez ch a písmen s háčky, čárkami a kroužky) nahradit čísly. Během těchto testů je trvale zobrazeno přiřazení písmen k číslům jako nápověda, nápovědy u jednotlivých příkladů jsou číselného typu. Proto je vhodné zkoušet testy tohoto typu až po seznámení se s číselnými řadami, neboť zakódování čísel písmeny ztěžuje pochopení úloh tohoto typu. Jsou uvedeny i příklady dvojitých písmenných řad, kouzelných mřížek s číselnými operacemi nebo jen doplnění chybějícího písmene ve slově. Dalším typem jsou tzv. dominové testy.

MYSLENI 4

Písmenné a slovní testy, v nichž nenahrazují písmena žádná čísla, i když mohou nabývat také různých hodnot. Slova z písmen sestavená se pak mezi sebou liší buď počtem písmen, vahou samohlásek či souhlásek, pořadím písmen ve slovech aj. Další úlohy jsou typu doplňování jednoho nebo dvou slov, slov na sebe navazujících, slov se stejnými příponami nebo předponami, slovními můstky, souvztažnostmi, přesmyčkami.

MYSLENI 5

Obrázkové úlohy jsou prý kulturně spravedlivější, neboť k jejich řešení není třeba tolik všeobecných znalostí či početních a jazykových dovedností ze školy. V těchto testech však často neexistuje žádné pravidlo nebo vzor. Mnoho úloh tohoto typu je založeno na orientaci obrázků, hledání pravidla dělení nebo opakování, kombinací, natočení a zrcadlení, postupující matrice, počtů v řádcích i sloupcích, ručičky na hodinách i slunečních, podobností a různé variace na hrací kostky, domino, logické úlohy. Nemohou být uvedeny všechny možnosti příkladů testů, jsou uvedeny některé typické obrázkové úlohy samozřejmě s nápovědou řešení a vysvětlením, co se od zkoušeného požaduje a jak lze postupovat.

MYSLENI

Dříve publikovaný program (SWD57 - testy 3) podobného typu, ve kterém je mj. uvedena tabulka jednotlivých kategorií IQ s jejich procentuálním výskytem v lidské populaci.

QBASIC - práce s proměnnými

Modulová struktura jazyka QBasic umožňuje používat proměnné se stejnými názvy v jednotlivých modulech. Každý modul má ve své paměti vymezeno místo pro své proměnné, které tak nejsou přístupné jiným modulům. Proměnné hlavního programu jsou běžně platné pouze pro hlavní program a proměnné procedur platí jen v procedurách. Lokální jsou tedy všechny proměnné, které se nacházejí v hlavním modulu programu nebo některé procedury. Existuje však možnost sdílení proměnných mezi moduly a procedurami (klíčové slovo nebo příkaz SHARED).

Některé konstanty pro práci programu mohou být trvalého charakteru a pro zjednodušení se označují podobně jako proměnné pomocí jména. Odlišení se provádí na začátku programu tzv. deklarací symbolické konstanty pomocí příkazu CONST:

```
CONST constantname = expression [,constantname = expression]...  
jméno_konst1 = výraz1      [,jméno_konst2 = výraz2
```

jméno_konst - identifikátor symbolické konstanty, který může obsahovat znak deklarace typu (%,&!,#,\$), ten však není součástí jména konstanty. Není-li uveden, bude konstantě přidělen typ dle výsledné hodnoty výrazu.

výraz - výraz tvořený konstantami a libovolnými aritmetickými a logickými operátory (mimo mocníka ^).

Nejčastěji se takto vymezuje Ludolfovo číslo: pi

... - pomocí jediného příkazu lze definovat více konstant

```
CONST pi = 3.141593: print pi                                'vypíše: 3.141593  
V programu pak již nelze používat proměnné se stejným jménem a příponou typu, nebo proměnné jednopísmenné (např. p$), shodné s prvním písmenem konstanty. Interpret QB by hlásil: DUPLICATE DEFINITION.
```

Konstanty deklarované v procedurách SUB a FUNCTION jsou lokální vzhledem k dané proceduře, jiné moduly k nim nemají přístup. Konstanty deklarované na úrovni modulu jsou definovány v rámci celého modulu, a proto platí i v jím vyvolaných procedurách. To znamená, že CONST pi definovaná v hlavním modulu programu na jeho začátku platí i ve všech procedurách volaných z tohoto modulu.

Definujeme-li takto řetězcovou konstantu například jako často používané slovo v programu, bude se dále používat již bez přípony \$, s čímž je nutno počítat:

```
CONST kcd = "Kč včetně DPH", kcb = "Kč bez DPH"  
print "9 "+kcd                                'vypíše:9 Kč včetně DPH  
print "7 "+kcb                                7 Kč bez DPH
```

Základním prostředkem reprezentace dat jsou proměnné, představované určitou hodnotou, která se může během činnosti programu libovolně měnit. Proměnnou tvoří určitý typ dat (číslo, řetězec, záznam) a její jméno (název, identifikátor). Jméno proměnné musí začínat písmenem a nesmí je tvořit tzv. vyhrazená nebo klíčová slova totožná s názvy příkazů nebo funkcí QBasicu, délka je omezena na 40 znaků bez mezer.

Jednoduchá proměnná je představována jednou hodnotou:
a = 54: b\$ = "slovo": adresa.psc = 63900: adresa.misto = "Brno"

Pole neboli indexovaná proměnná je představována množinou hodnot označených stejným identifikátorem:

p(0) = 10: p(1) = 11: p(2) = 12: p(3) = 13 ...

Číselné (numerické) proměnné lze přiřadit jen číselnou hodnotu, řetězcové proměnné lze přiřadit jen řetězec znaků. Řetězcové proměnné jsou odlišeny obvykle posledním znakem \$, pokud nejsou definovány bez této typové přípony pomocí:

CONST, DEFSTR, DIM ... AS STRING nebo DIM ... AS STRING * n.

QBasic využívá také mechanismus pro popis typu proměnné podle prvního písmene identifikátoru. Chceme-li vyjádřit, že všechny proměnné začínající písmenem I jsou typu integer (i) napíšeme DEFINT I, mají-li být celočíselnými proměnnými všechny proměnné začínající písmeny I,J,K,L,M,N, zapíšeme to deklarací DEFINT I-N.

Deklarace DEFTyp slouží pro deklaraci standardních typů všech proměnných v programu, jejichž identifikátor začíná některým z písmen uvedených v příkazu DEFTyp. Proto se tento příkaz uvádí v záhlaví programu, aby byl přehled o všech proměnných na určitém místě. Příkaz DEFTyp platí opět jen v rámci programového modulu v němž se nachází, nebo v proceduře, ve které je použit:

DEFINT	letterrange [,letterrange]...	pro krátká celá čísla
DEFLNG	letterrange [,letterrange]...	pro dlouhá celá čísla
DEFSNG	letterrange [,letterrange]...	pro reálná čísla jednoduchá
DEFDBL	letterrange [,letterrange]...	pro reálná čísla dvojnásobná
DEFSTR	letterrange [,letterrange]...	pro řetězce proměnné délky
	rozsa_h_písmen,rozsa_h_písmen	např.A-F, I-N, X-Z

DEFDBL a: a=sqr(3): print a	'vypíše: 1.732050807568877
DEFSNG b: b=sqr(3): print b	'vypíše: 1.732051
DEFINT c: c=sqr(3): print c	'vypíše: 2

Z těchto příkladů plyne, že již deklarací proměnné můžeme ovlivnit přesnost výpočtů, neboť procesor QBasicu zaokrouhluje podle typu proměnné na určitý počet desetinných míst.

Nejobecnějším příkazem pro deklaraci proměnných je DIM, který je určen pro deklaraci polí nebo specifikaci typů dat pro proměnné nevytvářející pole:

DIM [SHARED] variable[(subscripts)] [AS type] [,variable...]
[sdílená] proměnná[(indexy)] [jako typ]

sdílená - určuje že proměnná bude sdílena všemi procedurami SUB nebo FUNCTION v programu. Příkaz SHARED je nepovinný.

proměnná - jméno proměnné nebo identifikátor pole proměnných

indexy - číselný rozsah pole proměnných, vyjádřený následovně:
[lower TO] upper [, [lower TO] upper]...
[dolní TO] horní

- jeden index určuje, že se jedná o jednorozměrné pole, čárkou oddělené další indexy přidávají další rozměry:

DIM c(19,19) -dvourozměrné pole s 20 * 20 prvky,

DIM d\$(5,5,5)-třírozměrné řetězcové pole s 6*6*6 prvky

dolní - dolní hranice rozsahu proměnné, není-li uvedena, je =0

TO - vyhrazené slovo pro oddělení hranic indexů,

horní - horní hranice rozsahu.

DIM a(5 TO 8) -deklaruje 4 prvky: a(5),a(6),a(7),a(8)

DIM b\$(5) -deklaruje 6 prvků: b\$(0),b\$(1),...,b\$(5)

DIM e(-99 TO -5, 1 TO 5) -dvourozměrné pole 95 * 5

AS typ - definuje typ dat pole nebo typ proměnné jako (AS):

INTEGER, LONG, SINGLE, DOUBLE, STRING nebo uživatelem definovaný typ pomocí příkazu TYPE...END TYPE.

DIM c AS STRING - deklaruje proměnnou c jako řetězec.

c = "ahoj":print c 'vypíše: ahoj

DIM delka AS INTEGER - deklaruje proměnnou delka jako číslo v rozmezí (-32768,32767)

Bez použití příkazu DIM lze používat pole pouze s 11 prvky (indexy 0-10). Pro přehlednost programu je však vhodné deklarovat pomocí DIM i pole s méně prvky. Přípustný rozdíl hodnot 'dolní' a 'horní' je 32767, možné rozmezí indexů je -32768 a +32767.

Pokud nám nevyhovuje implicitní minimální hodnota 'dolní'= 0 je možno zvolit 1 pro všechna pole v programu pomocí příkazu:
OPTION BASE 1

Proměnné v QBasicu mají tzv. lokální charakter, platí pouze v tom modulu, v němž jsou definovány a používány. To je velkou výhodou, neboť není nutno sledovat, která proměnná již byla dříve použita či nikoli, a jednotlivé moduly mohou programovat nezávisle na sobě různí programátoři. Bylo by však omezením, kdyby nebylo možno předávat si proměnné mezi moduly. Pro tento účel má QBasic některé další příkazy.

Pokud potřebujeme v proceduře SUB nebo FUNCTION pracovat s proměnnými deklarovanými v hlavním modulu programu, můžeme si tyto proměnné převzít pomocí příkazu SHARED (sdílet).

SHARED proměnná [AS typ][,proměnná [AS typ]]...
proměnná - jméno proměnné, definované na úrovni hlavního modulu, kterou bude procedura nebo funkce používat; pokud jde o pole, musí být za jménem uvedeny prázdné závorky:
SHARED b\$() - sdílené pole řetězcových proměnných.

Někdy je naopak nutno v hlavním modulu určit, že ostatní moduly mají používat společné (COMMON) proměnné. Takové proměnné se nazývají globální. Mají-li platit nejen v modulech, ale i v procedurách a funkcích, musí za COMMON následovat SHARED v hlavním modulu programu:

COMMON [SHARED] proměnná[()] [AS typ] [,proměnná[()] [AS typ]]...
- definuje globální proměnné sdílené mezi všemi moduly, pokud jde o pole, musí být za jménem proměnné opět (),
SHARED - indikuje, že globální proměnné platí i ve všech procedurách SUB a FUNCTION.

S použitím globálních proměnných však mizí výhoda modulového programování s nezávislými lokálními proměnnými a naopak vzrůstá riziko, že některá procedura změní nevhodně hodnoty proměnné. Proto je vhodnější jejich používání co nejvíce omezit a raději používat příkaz STATIC, zachovávající hodnoty proměnných v procedurách. Pokud má být v proceduře zachována hodnota proměnné mezi jejími voláními, je nutno to proceduře předepsat příkazem STATIC:
STATIC proměnná[()] [AS typ] [,proměnná[()] [AS typ]]...

Je to nutné proto, že proměnné mohou být buď statické nebo dynamické, čímž je myšlena délka jejich trvání. Dynamické proměnné existují jen během provádění procedury, při každém vyvolání se znovu vytvoří a inicializují s hodnotou 0 nebo "" podle typu proměnné. Použitím příkazu STATIC na začátku procedury si vyjmenované proměnné zachovají své hodnoty i pro další volání procedury.

Uvedené příkazy znamenají, že s jejich pomocí přímo můžeme definovat proměnné na odpovídající úrovni programu, za definicí musí pak ještě následovat vymezení indexů v případě polí pomocí příkazů DIM nebo raději REDIM (mění rozsah indexů proměnných).

Uživatelsky definovaný typ proměnných patří mezi tzv. struktury nebo záznamy či skupinová data, tvořené různými proměnnými. Na rozdíl od polí mohou být jednotlivé proměnné různých typů, pouze řetězce musí být pevné délky o n znacích (typu STRING*n). Definice tohoto typu se provádí v bloku příkazů TYPE ... END TYPE:

TYPE usertype
 elementname AS type
 [elementname AS type]
 ...
END TYPE

TYPE uživatelský_typ
 jméno_prvku AS typ
 [jméno_prvku AS typ]
 ...
END TYPE -konec definice

uživat._typ- jméno uživatelsky definovaného datového typu, tj.
strukturované proměnné,
jméno_prvku- jednotlivý prvek strukturovaného datového typu,
typ - některý z povolených typů dat:
INTEGER, LONG, SINGLE, DOUBLE, STRING*n,
- není povolen řetězec libovolné délky STRING.

Typickou strukturovanou proměnnou bývá adresa zákazníka:

```
TYPE ADRTYP
  jmeno AS STRING * 10      'jméno jako řetězec o 10 znacích
  prijmeni AS STRING * 15   'příjmení jako řetězec o 15 znacích
  ulice AS STRING * 15      'ulice a popisné číslo o 15 znacích
  psc AS LONG               'psc jako dlouhé celé číslo >32767
  misto AS STRING * 20      'dodávací pošta o max. 20 znacích
END TYPE
```

Tento typ strukturované proměnné vytváří vlastně větu pevné délky $(10 + 15 + 15 + 4 + 20) = 64$ bytů, s pevnou strukturou:

jmeno-----prijmeni-----ulice-----psc-misto-----

kde pomlčky nahrazují mezery. Pokud jsou přiřazené proměnné delší než vymezený počet znaků, jsou přebytečné znaky ořezány a ztraceny. Proto je nutno při deklaraci TYPE předem analyzovat možné rozsahy proměnných

Tento blok příkazů definuje nový typ dat ADRTYP (seznamuje s ním interpret QBasicu), ale nedeklaruje proměnné a nevytváří pro ně místo v paměti. Vlastní deklaraci je nutno vytvořit pomocí následujícího příkazu

```
DIM adresa AS ADRTYP
```

Teprve tento příkaz vymezí v paměti pro adresu výše uvedených 64 bytů. Proměnná 'adresa' není číselná ani řetězcová, ale je právě proměnná typu ADRTYP, jedná se o strukturovanou (složenou) proměnnou s více položkami. Chceme-li přiřadit hodnoty jednotlivým položkám, musíme uvádět jejich plný název s tečkou:

'proměnná.položka', například:

```
adresa.jmeno = "Pavel": print adresa.jmeno      'vypíše: Pavel
adresa.prijmeni = "Hlaváček"
adresa.ulice = "Hluboká 5"
adresa.psc = 63900
adresa.misto = "Brno"
```

Proměnné tohoto typu jsou nazývány také jako proměnné typu záznam (věta, record) a můžeme jim přiřadit i jiné proměnné stejného uživatelského typu, definované pomocí DIM, např.:

```
DIM adr1 AS ADRTYP, adr2 AS ADRTYP
```

Po vzájemném přiřazení můžeme pracovat i s položkami nových proměnných shodného typu:

adr1 = adresa: print adr1.ulice 'vypíše: Hluboká 5

Proměnné typu záznam mohou vytvářet i pole pomocí deklarace:
DIM adrpole (1 TO 19) AS ADRTYP

Tato deklarace pak v paměti vymezí 19*64 bytů pro 'adrpole'.

Po přiřazení můžeme opět pracovat s položkami pole jak výše:
adrpole(5) = adresa: print adrpole(5).psc 'vypíše: 63900

Je-li rozsah proměnné vyjádřen čísly, hovoříme o statickém dimenzování, pokud je vyjádřen pomocí proměnných, jejichž hodnota se může měnit, mění se i rozsah pole a pak se jedná o dynamické dimenzování:

DIM p(9,9) - dvourozměrné pole p (10 * 10 prvků) je statické,
DIM d(pocet) - jednorozměrné pole d je dynamické, neboť proměnná
'pocet' se v programu může měnit a proto procesor
QBasicu mu přiřadí paměťový prostor až při provedení instrukce DIM.

Pomocí DIM můžeme deklarovat jak statická, tak i dynamická pole, rozlišení se provádí metapříkazy: REM \$STATIC nebo REM \$DYNAMIC, uvedenými v programu před příkazy DIM. Statické pole je přiděleno po spuštění programu a nemění se během jeho běhu, dynamické je přiděleno až během programu podle hodnot proměnných.

Příkazem pro deklaraci pouze dynamických polí je REDIM, definovaný podobně jako DIM. Příručka QB doporučuje používat příkaz DIM pro statická pole a REDIM pro dynamická a nepoužívat metapříkazy.

Přiřazení hodnoty proměnné

Pro přiřazení hodnoty nějaké proměnné je určen příkaz LET, ale je nepovinný a proto se běžně nepoužívá.

Proměnné může být jako hodnota přiřazena:

- konstanta: k = 5: m\$ = "Konec souboru": r\$ = "5"
- hodnota jiné proměnné: x = k: q\$ = m\$
SWAP q\$,r\$ (vzájemné prohození)
- hodnota uvedeného výrazu: y = sin(3)/4: z\$ = "a: " + m\$ + " !"
- složka strukturované proměnné: jm\$ = adresa.jmeno
- strukturované proměnné hodnota stejného typu strukt.proměnné:
adr1 = adresa
- příkazem READ hodnoty ze seznamu v řádcích DATA.

DATA je nevykonaný příkaz, který v programu poskytuje seznam

konstant, které mají být načteny příkazem READ. Řádků s příkazy DATA může být v programu libovolný počet:

DATA constant[,constant]...
konstanta

konstanta - libovolná číselná nebo řetězcová konstanta, řetězec musí být v uvozovkách jen tehdy, obsahuje-li čárky, dvojtečky nebo mezery na začátku nebo konci textu. Nulové hodnoty nebo prázdné řetězce jsou představovány chybějící konstantou mezi čárkami (oddělovači).

Příkazy DATA se použijí ve stejném pořadí, ve kterém jsou uvedeny v programu. Mají-li se stejná data číst vícekrát, je nutné nastavit ukazatel jejich čtení pomocí příkazu RESTORE:

RESTORE [line]
[řádek]

řádek - návěští nebo číslo řádku s příkazem DATA, je-li vynecháno bude se číst z prvního řádku programu s příkazem DATA.

Příkaz DATA může být uveden pouze na úrovni modulu (nikoli v proceduře). Příkaz pro jejich čtení READ může být kdekoliv v programu nebo proceduře, stejně jako příkaz RESTORE:

READ variablelist
seznam_proměnných

seznam_proměnných - jedna nebo několik číselných nebo řetězcových proměnných oddělených čárkou, do nichž mají být načtena data z řádků DATA.

Například:

DATA 5,Hluboká

READ číslo, ulice\$

PRINT ulice\$;" č.";cislo

'vypíše: Hluboká č.5

Typ proměnných musí souhlasit s typem čtené konstanty. Zatím co číslo se do řetězcové proměnné načte jako řetězec, řetězec se v číselné proměnné necítí dobře a ohlásí se chybovým hlášením: Type mismatch. Pokud budou data pro čtení vyčerpána nebo budou chybět, ohlásí se z: Out of DATA

QBASIC - číselné funkce

Všechny funkce jazyka QBasic mají stejnou úlohu - předat jednu funkční hodnotu. Každá funkce má obvykle nejméně jeden parametr, uvedený v závorce za jménem funkce. Podle toho, zda předávaná funkční hodnota byla definována jako číselný výraz nebo řetězec znaků, hovoříme o číselných nebo řetězcových funkcích. Přitom číselné funkce mohou pracovat i s řetězcem (předávají nějakou číselnou informaci o řetězcích). Řetězcové funkce mohou mít v zá-

vorce jako parametr také číslo (např. určuje počet funkčních zna-
ků nebo převádí číslo na řetězec).

Pro výpočty standardních matematických funkcí je používáno v
QBasicu 18 funkcí s jedním parametrem, jímž může být i složitý vý-
raz, využívající řady matematických operátorů (+, -, *, /, ^, \ ap.).
Uvedeme si pouze jejich přehled s několika příklady:

- ABS(výraz)** - vrací absolutní hodnotu numerického výrazu:
print abs(45.5 - 100!) 'vypíše: 54.5
- ATN(výraz)** - arcustangens výrazu, předá úhel v úhlové míře, jehož
tangenta je rovna hodnotě numerického výrazu (musí
odpovídat poměru stran pravouhlého trojúhelníka):
print atn(tan(3/4)) 'vypíše: 3/4
- CDBL(výraz)** - převede výraz na číslo s dvojitou přesností:
print 1/3 'vypíše: .3333333
print cdbl(1/3) .3333333333333333
Pokud by se tato funkce použila na zpřesnění čísla s
jednoduchou přesností, došlo by k chybě přiřazení na
posledních 9 místech. Vyzkoušejte si:
x = 1/3: print cdbl(x) 'vypíše: .3333333432674408
- CINT(výraz)** - zaokrouhlí výraz (rozsah: -32768, 32767) na celé číslo
print cint(2.49); cint(2.51) 'vypíše: 2 3
Zaokrouhluje zvláštním způsobem poloviny:
int(7.5) = 8, zatímco int(8.5) = 8, int(9.5) = 10
- CLNG(výraz)** - zaokrouhlí výraz (rozsah: -2147483648, 2147483647) na
dlouhé celé číslo:
print clng(98765.4321) 'vypíše: 98765
- CSNG(výraz)** - převede výraz na číslo s jednoduchou přesností:
t# = 21.123456789
print (300 + t#) 'vypíše: 321.123456789
print csng(300 + t#) 321.1234
- COS(úhel)** - vrací hodnotu cosinu úhlu, zadaného v obloukové míře
(radiánech): rad = stupně * pi/180, kde pi=3.141593
pi = 3.141593: print cos(pi/3) 'vypíše: .4999999
- EXP(výraz)** - vrací hodnotu exponenciální funkce se základem při-
rozených logaritmů e=2.71828: exp(x) = e^x
print exp(0); exp(1) 'vypíše: 1 2.718282
- FIX(výraz)** - vrací celočíselnou část výrazu oddělením míst za
desetinnou tečkou:
print fix(-1.23) 'vypíše: -1
- FRE(výraz)** - vypíše velikost paměti dostupné programu nebo paměti
vymezené pro řetězce je-li výraz řetězec:
print fre(x\$) 'vypíše např.: 30768
- je-li výraz -1 vrací FRE velikost nejdelšího čísel-
ného pole v bytech, které by bylo možné dimenzovat:
print fre(-1) 'vypíše např.: 158192
- je-li výraz -2 předá FRE velikost nevyužitého pro-
storu v zásobníkové paměti:

```

print fre(-2) 'vypíše např.: 562
- používá se pro informaci jenom v přímém režimu.
FREEFILE - vypíše nejnížší volné číslo, které je k dispozici
           pro otevření souboru.
INT(výraz) - vrací největší celé číslo menší nebo rovno hodnotě
             numerického výrazu:
             print int(7.25); int(-7.25) 'vypíše: 7 -8
LOG(výraz) - vrací přirozený logaritmus numerického výrazu:
             print log(1); log(exp(1)) 'vypíše: 0 1
RND[(n#)] - vrací náhodné číslo jednoduché přesnosti mezi 0 a 1,
            hodnota n určuje, jak bude RND generovat toto číslo:
            n=0 zopakuje generování posledního čísla,
            n<0 generuje se stejné číslo pro libovolné n#,
            n>0 generuje se následující náhodné číslo, např.:
            print rnd(1) 'vypíše: .2260668
            print rnd(-1) .4431317
            print rnd(0) .4431317
SGN(výraz) - znaménko hodnoty numerického výrazu:
             print sgn(9); sgn(-0.1); sgn(0) 'vypíše: 1 -1 0
SIN(úhel) - vrací hodnotu sinu úhlu zadaného v obloukové míře:
             print sin(pi/6) 'vypíše: .5000001
SQR(výraz) - vrací druhou odmocninu z numerického výrazu >=0:
             print sqr(25); sqr(2) 'vypíše: 5 1.414214
TAN(úhel) - vrací hodnotu tangens úhlu zadaného v radiánech:
             print tan(pi/4) 'vypíše: 1

Ostatní číselné funkce pracují s řetězcí. Jsou to funkce:
ASC(x$) - vrací číslo ASCII kódu (0-255) prvního znaku řetěz-
          ce x$ nebo řetězcového výrazu v závorce:
          print asc("DEN") 'vypíše: 68
INSTR([n,]x$,y$) - pozice prvního výskytu řetězce y$ v řetězcí x$
                  prohlédávání začíná od n-tého znaku řetězce x$, ale
                  pozice se uvádí od počátku řetězce:
                  print instr(5,"Rimini","i") 'vypíše: 6
                  print instr("helemese","le") 3
LEN(x$) - vrací počet znaků řetězce x$ nebo řetězcového výrazu
          print len("string") 'vypíše: 6
          k$="posloupnost":print len(k$) 11
VAL(x$) - předá číselnou hodnotu řetězce číslic nebo číslic ji-
          miž řetězec začíná:
          print val("5678") 'vypíše: 5678
          print val("12/4") 12
          print val("2abcd") 2

```

Zvláštní číselnou funkcí je TIMER (bez parametrů), která udává počet sekund, které uplynuly od půlnoci, pokud má počítač instalovány hodiny reálného času. Čas je udáván na dvě desetinná místa, lze tedy odečítat i setiny sekund:

```
print timer
```

QBASIC - řetězcové funkce

Řetězcovou konstantou (zkráceně řetězcem) rozumíme posloupnost alfanumerických znaků uzavřenou do uvozovek. Řetězcovým výrazem je pak kombinace řetězcových konstant, proměnných a funkcí.

Řetězcová proměnná je proměnná typu STRING (řetězcového typu), které smí být jako hodnota přiřazen pouze řetězec. Řetězcové funkce předávají do řetězcových proměnných upravené nebo vytvořené řetězce:

- CHR\$(n) - vrátí znak, jehož ASCII kód je roven n:
print chr\$(69) 'vypíše: E
- HEX\$(n) - převede dekadické číslo n na řetězec vyjadřující jeho hexadecimální (šestnáctkovou) hodnotu:
print hex\$(65535) 'vypíše: FFFF
- LCASE\$(x\$) - změni všechna velká písmena v řetězci x\$ na malá:
print lcase\$("CoCaCoLa") 'vypíše: cocacola
- LEFT\$(x\$,n) - vybere z řetězce x\$ zleva prvních n znaků:
print left\$("levandule",3) 'vypíše: lev
- LTRIM\$(x\$) - předá řetězec x\$ bez mezer vlevo:
x\$=" 24":print "1"+ltrim\$(x\$) 'vypíše: 124
- MID\$(x\$,n[,m]) - předá m znaků řetězce x\$ od n-tého znaku, nebo zbylý počet znaků od n-tého do konce řetězce není-li hodnota m uvedena:
print mid\$("Kč...15",3,3) 'vypíše: ...
print mid\$("Kč...15",3) 'vypíše: ...15
- OCT\$(n) - předá řetězec s osmičkovou hodnotou čísla n.
print oct\$(8) 'vypíše: 10
- RIGHT\$(x\$,n) - vrátí n znaků řetězce x\$, které jsou zcela vpravo:
print right\$("Postoloprty",3) 'vypíše: rty
- RTRIM\$(x\$) - vrátí řetězec x\$ zbavený pravostranných mezer:
print rtrim\$("4 ")+ "Kč" 'vypíše: 4Kč
- SPACE\$(n) - předá řetězec n mezer (vytvoří prázdný řetězec délky až 32767 mezer).
- STR\$(x) - převede číslo, číselnou konstantu x nebo výraz na řetězec, místo znaménka + vyhradí zleva mezeru.
- STRING\$(m,n) - vytvoří řetězec m znaků s ASCII kódem n, nebo znaků uvedených v uvozovkách místo n:
print string\$(4,48) 'vypíše: 0000
print string\$(6,"*") 'vypíše: *****
- UCASE\$(x\$) - změni všechna malá písmena v řetězci x\$ na velká:
print ucase\$("UpperCASE") 'vypíše: UPPERCASE

Mezi řetězcové funkce řadíme i údaje o čase a datumu, platné pouze při instalaci a nabitých bateriích systémových hodin. Obě funkce mohou být použity též jako příkazy k nastavení proměnných:

- TIME\$** - vrací údaj o aktuálním času systémových hodin:
print time\$ 'vypíše např.: 09:58:04
- stejnojmenný příkaz slouží k nastavení správného času systémových hodin, lze zadávat buď jen hodiny nebo nebo jinou kombinací času HH:MM:SS, neuvedená položka je nastavena na nulu.
- DATE\$** - vrací řetězec obsahující aktuální datum ve tvaru určeném příkazem COUNTRY v DOS, např.: MM-DD-RRRR
MM - měsíce v rozsahu 01-12
DD - dny v rozsahu 01-31
RRRR - roky v rozsahu 1980-2099.
- stejnojmenný příkaz následovaný řetězcem data umožňuje nastavení aktuálního data. Řetězec může být: date\$="06/03/94" nebo: date\$="06-03-1994"

Struktura a zápis programu v QBasicu.

Program v QBasicu je tvořen modulem hlavního programu (MAIN MODUL) a libovolným počtem procedur typu SUB nebo FUNCTION. Funkce a procedury nejsou nezbytně nutné, budeme se jimi zabývat později.

Modul programu je vytvářen postupně editorem QBasicu tak, že se na jednotlivé řádky na obrazovce zapisují postupně příkazy jazyka v souladu s algoritmem programu. Struktura modulů je potom:

Hlavní modul:	Modul procedury:	Modul funkce:
-----	-----	-----
příkaz 1	SUB proc	FUNCTION max(a,b)
příkaz 2	příkaz p1	příkaz f1
CALL proc	PRINT max(1,2)	příkaz f2
.....
příkaz n	příkaz pn	příkaz fn
END	END SUB	END FUNCTION

Hlavní modul se liší tím, že obsahuje jako poslední příkaz END (nemusí být na posledním řádku, jak uvidíme později). V těle tohoto modulu mohou být příkazy volání procedury (CALL) nebo příkazy používající uživatelské funkce [např. PRINT max(a,b)] z modulu funkce.

Modul procedury je pro odlišení zahajován vyhrazeným slovem SUB a názvem procedury a ukončen příkazem END SUB. Může také používat uživatelské funkce, jak je naznačeno: PRINT max(1,2).

Modul funkce tvoří v záhlaví vyhrazené slovo FUNCTION a název funkce s parametry v závorce a je ukončen standardním příkazem END FUNCTION.

Základním stavebním prvkem jazyka QBasic je příkazový řádek,

obsahující jeden příkaz jazyka, nebo více příkazů, oddělených od sebe dvojtečkou.

Programový řádek může sice obsahovat až 256 znaků, ale není to přehledné - na monitoru lze zobrazit jen 80 znaků na řádek. Je často doporučováno zadávat na řádek jediný příkaz, aby byl program přehledný, ale protože interpret QBasicu upozorňuje na případné chyby označením chybného příkazu vyšším jasnem, lze na řádek sdružovat příkazy tvořící určitý logický celek.

QBasic nevyžaduje číslování řádků, pokud jsou čísla řádků uvedena slouží pouze jako návěští nebo identifikátor řádku. Program jinak číslování řádků nevyužívá.

Úplná struktura (syntaxe) programového řádku je následující:

[identifikátor][příkaz][:příkaz]...[komentář]

Hranaté závorky u všech článků řádku naznačují, že je platný i prázdný řádek - oddělují se jím například části programu v modulech, aby byly přehlednější (ve výpisech programu jsou jím odděleny moduly a procedury).

- | | |
|---------------|---|
| identifikátor | - označuje řádek pro případ, že je nutno na tento řádek nepodmíněně skočit (GOTO identifikátor) |
| | - může jím být číslo řádku nebo návěští řádku, musí být první na řádku, ale v libovolném sloupci: |
| číslo řádku | - libovolné číslo v rozmezí 1 - 65529, řádek 0 bývá použit jako návěští při chybě: ON ERROR GOTO 0 |
| návěští | - může být až 40 znakové, začínat musí písmenem, nerozlišují se písmena velká a malá, končit musí dvojtečkou i když za ním nenásleduje příkaz; nesmí jím být žádné vyhrazené slovo QBasicu. |
| opaki: CLS | - příklad návěští s následujícím příkazem CLS. |
| komentář | - za příkazy na řádku lze napsat komentář programu, buď dvojtečka, příkaz REM a text komentáře, nebo apostrof a komentář (může být i celým řádkem). |

Příkazy QBasicu

Popis akcí, které má program provádět, je formulován pomocí příkazů. Příkazy lze formálně rozlišovat na výkonné nebo nevýkonné, nebo na jednoduché a strukturované.

Nevýkonné příkazy jsou nutnými přípravnými operacemi, které slouží například k přiřazení paměti proměnným (DIM), deklaraci typu proměnných (DEFTYP, ...AS typ, DATA), vyčlenění proměnných pro předávání parametrů procedurám.

Výkonné příkazy vytváří algoritmus programu, popisují akce organizace pracovní části obrazovky (SCREEN, CLS, VIEW), načtení dat (INPUT, READ) nebo jejich výstup (PRINT, WRITE) včetně nastavení kurzoru (LOCATE, POS, CSRLIN), provedení matematické funkce nebo řetězcové funkce, přiřazení výsledku operace určité proměnné a pod.

Jednoduchým příkazem je například přiřazení, jež provádí nepovinný příkaz LET (LET a = 7), nebo příkaz nepodmíněného skoku (GOTO), volání podprogramu (GOSUB) nebo procedury (CALL).

Strukturované příkazy vytvářejí postupné, podmíněné nebo opakované provedení výpočetních nebo jiných akcí. Skládají se z dílčích příkazů a obsahují také výrazy udávající podmínky pro větvení či opakování. Patří sem složené příkazy, podmíněné příkazy a příkazy cyklu.

Zatímco funkce mají v závorce parametr, příkazy jsou spojeny s operandem, kterým může být:

- konstanta: print 5
- proměnná: a = 2: print a
- funkce: print sqr(a)
- výraz: print a * a / 4

Výraz popisuje posloupnost operací. Nejjednodušším výrazem je konstanta nebo proměnná. Složitější výrazy mohou být tvořeny kombinováním konstant, proměnných a funkcí, které jsou svázány operátory (matematickými nebo logickými). Provedení všech operací předepsaných výrazem se nazývá vyhodnocení výrazu. Výsledkem tohoto vyhodnocení je jediná hodnota - hodnota výrazu. Jednotlivé předepsané operace se provádějí v pořadí, které je určeno prioritou použitých operátorů (závorky, umocňování, násobení a dělení před sečítáním a odčítáním...).

Pokud se ve výrazu objeví více typů číselných proměnných nebo konstant, QBasic provádí převod čísel:

- při přiřazování se hodnota proměnné převede na typ odpovídající typu proměnné se zaokrouhlením, nikoli oříznutím (viz výše),
- pokus o přiřazení číselné proměnné řetězcové nebo naopak se vypíše chybové hlášení: TYPE MISMATCH,
- při výpočtu číselného výrazu s operandy různých datových typů se před výpočtem převedou všechny operandy na stejnou přesnost, kterou má nejpřesnější operand výrazu.

Při tomto převodu může dojít k chybám, způsobeným nesprávným zadáním typů proměnných, například:

```
x = 1/3: print cdbl(x)        'vypíše: .3333333432674408
         print cdbl(1/3)        .3333333333333333
b# = 3 * x: print x,b#        .3333333        1.000000029802322
```

Chybová hlášení QBasicu podle abecedy

Při odladování vytvářeného programu může dojít k některým chybám, které si uvedeme v abecedním pořádku i s jejich číselným kódem (ten je vrácen funkcí ERR a může být použit v programu k ošetření této chyby, pokud ji předpokládáme).

- Argument-count mismatch ... 37
nesouhlasí počet argumentů při volání procedury.
- Array not defined ... 38
program pracuje s polem proměnných, které není definováno.
- Bad file mode ... 54
pokus o práci s neotevřeným datovým souborem pro daný režim (zápis do souboru otevřeného pro čtení a pod.).
- Bad file name or number ... 52
pokus o práci se souborem neotevřeným pomocí OPEN nebo jehož číslo je mimo zadanou oblast čísel souborů.
- Bad file name ... 64
zadáno nepřipustné jméno souboru pro BSAVE,SAVE,KILL,OPEN.
- Bad record length ... 59
délka věty v příkazech GET nebo PUT není v souladu s údajem v příslušném příkazu OPEN.
- Bad record number ... 63
v příkazu PUT nebo GET je číslo záznamu ≤ 0 .
- Cannot continue ... 17
pokus o pokračování v programu přerušeném chybou, neexistujícím nebo opravovaném po zastavení.
- Communication-buffer overflow ... 69
přetečení vyrovnávací paměti pro příjem dat ze zařízení.
- Device fault ... 25
porucha v zařízení, s nímž program komunikoval.
- Device I/O error ... 57
chyba služby DOSu ve vstup/výstupní operaci se zařízením.
- Device timeout ... 24
ve stanoveném čase nedostal program očekávanou informaci od zařízení (např. se ve stanovené době neohlásilo).
- Device unavailable ... 68
přístup k zařízení vypnutému, neexistujícímu či vadnému.
- Disk not ready ... 71
disketa nevložena, neuzavřena šachta, nalezen vadný sektor.
- Disk full ... 61
plný disk/disketa, celý program nelze zapsat.
- Division by zero ... 11
pokus o dělení nulou nebo mocnění nuly záporným číslem.
- Duplicate definition ... 10
pokus o novou definici identifikátoru již dříve použitého.

- Duplikate label ... 33
pokus o novou definici návěští již dříve použitého.
- Feature unavailable ... 73
pokus o použití vyhrazeného slova, které QBasic nezná.
- FIELD overflow ... 50
pokus o přiřazení více bytů příkazem FIELD než je udaná délka záznamu v souboru s přímým přístupem.
- FIELD statement active ... 56
příkazy GET nebo PUT se pokouší operovat s proměnnou, které již bylo v paměti přiřazeno místo příkazem FIELD.
- File already exists ... 58
jméno souboru uvedené v příkazu NAME již existuje.
- File already open ... 55
příkaz OPEN nebo KILL byl použit pro již otevřený soubor.
- File not found ... 53
soubor nenalezen, neexistuje nebo byla chybně zadána cesta.
- FOR without NEXT ... 53
neukončena smyčka, za příkazem FOR nebyl nalezen NEXT.
- Function not defined ... 18
pokus o volání nedefinované procedury typu FUNCTION.
- Illegal function call ... 5
funkci byl předán parametr ležící mimo možný rozsah hodnot.
- Illegal in direct mode ... 12
v přímém režimu (IMMEDIATE) byl zadán příkaz, použitelný jen uvnitř programového modulu.
- Input past end of file ... 62
příkaz INPUT čte z prázdného souboru nebo ze souboru, jehož obsah už byl celý přečten.
- Internal error ... 51
bližší nespecifikovaná chyba interpreteru QBasicu.
- Label not defined ... 8
odvolání na návěští řádku, které nebylo definováno.
- NEXT without FOR ... 1
nalezen příkaz NEXT bez předchozího příkazu FOR; může vzniknout při ukončení cyklu dvěma NEXT (např. v řádku podmínky).
- No RESUME ... 19
při zpracování obslužné rutiny pro zpracování chyb bylo dosaženo konce programu a je nutno doplnit příkaz RESUME.
- Out of DATA ... 4
program obsahuje příkaz READ, ale chybí řádky DATA, nebo se ukazatel čtení posunul za poslední konstantu v řádku DATA.
- Out of memory ... 7
program vyžaduje pro činnost více paměti, než je k dispozici:
- je možno zmenšit počet BUFFERS v souboru CONFIG.SYS,
- je možno z paměti odstranit rezidentní programy a ovladače nebo je převést do volných oblastí HIMEM, je-li to možné.
- omezit rozsáhlá pole ve vlastním programu.

Out of paper ... 27
chybějící papír v tiskárně nebo není tiskárna zapnuta.

Out of string space ... 14
řetězcové proměnné překročily povolený rozsah 64kB.

Overflow ... 6
výsledek výpočtu je příliš velký a nemůže být znázorněn jako číselný typ, do něhož měl být přiřazen.

Path/File access error ... 75
služba DOSu nemohla vytvořit soubor udaného jména nebo nemohla k němu vytvořit cestu v příkazech OPEN, MKDIR, CHDIR, RMDIR

Path not found ... 76
služba DOSu nemohla nalézt cestu k souboru.

Permission denied ... 70
pokus o zápis do souboru, disku, diskety chráněných proti zápisu.

Rename across disks ... 74
při pokusu o přejmenování souboru byl uveden jiný disk/disketa než u původního jména.

RESUME without error ... 20
program narazil na příkaz RESUME, ač dosud nebyla vyvolána žádná obslužná rutina pro zpracování chyb.

RETURN without GOSUB ... 3
program narazil na příkaz RETURN aniž byl volán podprogram pomocí příkazu GOSUB, obvykle po chybném nepodmíněném skoku.

String formula too complex ... 16
řetězcový výraz je příliš složitý nebo příkaz INPUT vyžaduje vstup více než 15 řetězcových proměnných

Subprogram not defined ... 35
pokus o volání procedury typu SUB jež nebyla definována.

Subscript out of range ... 9
index pole leží mimo zadaný rozsah, nebo pokus o manipulaci s prvkem dynamického pole, které nebylo dimenzováno.

Syntax error ... 2
syntaktická chyba (zadaný nesprávný příkaz nebo funkce), nebo nesprávně formátovaný příkaz DATA.

Too many files ... 67
pokus o otevření více než 255 souborů pomocí OPEN nebo SAVE.

Type mismatch ... 13
přiřazovaný výsledek výrazu je jiného typu než proměnná.

Variable required ... 40
požaduje chybějící parametry příkazů INPUT, LET, READ, SHARED.

WEND without WHILE ... 30
program narazil na příkaz WEND aniž byl předtím příkaz WHILE

WHILE without WEND ... 29
program přečetl příkaz WHILE ale nenašel k němu příkaz WEND.

Programové vybavení CONSUL 2717/PMD85-2,3

Kč 98,-

SWD73/SWK73

HRÝ - 10

SWD73/SWK73

BURZA.BAS

Hra spočívá v nakupování a prodeji akcií 4 podniků. Základní kapitál je 1000 Kč. Po každém kole se mění ceny akcií, mění se i úrok konta v bance. Po dosažení konta > 10000 se obchoduje již s 5 podniky a od 100000 se šesti. Občas burza také zkrachuje a je nutno začít znovu.

DAMA.BAS

Varianta hry dáma, v níž má program kameny X a začíná, hráč má kameny O. Tahy se zadávají dvojicí souřadnic (sloupec, řádek) nejprve ODKUD a pak KAM. Je-li přeskokem brán kámen X zmizí oba kameny a je nutno znovu zadat souřadnice ODKUD (jsou zobrazeny pod šachovnicí), aby se kámen O zobrazil na nové pozici. Dámy jsou zobrazeny znakem kamene a závorkou: X(nebo O(. Zadáním pole s kamenem protihráče se vytvoří zmatený obraz a je nejlepší začínat znovu. Přeskakovat kámen dámou lze pouze ze sousedního pole.

DAMA2.BAS

Klasická dáma s pěšáky i dámami v grafickém provedení pro 2 hráče. Souřadnice nejsou zadávány, ale určovány pomocí čtvercového kurzoru, kterým se pohybuje pomocí kláves: A-vlevo, D-vpravo, W-nahoru a Y-dolů. Potvrzení tahu se provádí klávesou mezerník a předání soupeři pomocí klávesy EOL. Pomocí mezerníku je nutné vyznačovat i každou mezípolohu figur, např. při dvojskoku.

DOLOVANI.BAS

Jednoduchá manažerská hra. Úkolem je zajišťovat úspěšnou činnost důlní společnosti nejméně po dobu 12 měsíců, především:

- prodejem vytěžené rudy za nejvyšší cenu,
- prodejem některého dolu, je-li cena rudy nízká,
- nákupem dalších dolů, jsou-li peníze a horníci,
- nákupem potravin ke spokojenosti horníků.

Občas klesne cena rudy, horníci nespokojení nebo onemocní...

DOSTIHY.BAS

Jednoduchá sázková hra simulující dostihy 8 koní.

PARASUTI.BAS

Pomocí klíčů F1/F2 se ovládá padák parašutisty při měnícím se větru a místu seskoku. Cílem je dopadnout na značku NULA a ne do rybníka nebo lesa.

VEKSLAK.BAS

Při cestování po 25 městech ČR lze nakupovat a prodávat 6 druhů zboží podle aktuální cenové nabídky, ale auto uveze jenom něco, občas zabaví náklad i s autem policie, je nutno prodat meziklad a dělat vše pro zvyšování konta v bance a snížení dluhu..

Programové vybavení CONSUL 2717/PMD85-2,3

Kč 298,-

SWD74/SWK74

MATEMATIKA -7/8

SWD74/SWK74

Autor: Zdeněk Rubý, Plesná

Soubor programů pro podporu výuky matematiky a kontrolu pochopení probírané látky žáky sedmých a osmých tříd. Jde o první operace s výrazy v sedmé třídě, počítání s mocninami a mnohočleny včetně užití vzorců v 8. třídě. Všechny programy obsahují náhodný výběr 10 příkladů ze 30-60 možných, směsí až ze 120 příkladů. Při rozkladech je nutno dodržet zásadu, že v první závorce je součet členů a rozdíl až ve druhé. Platí desetinná čárka i tečka, mocnitel je přemykač (shift) s číslicí, + a - s přemykačem i bez něj.

Použití vzorců:

- VZORCE1 Součin součtu a rozdílu typu: $(9z+8x)(9z-8x) = 81z^2 - 64x^2$
 VZORCE2 Rozklad rozdílu druhých mocnin: $(1-x^2) = (1+x)(1-x)$
 VZORCE3 Druhá mocnina součtu: $(m+7n)^2 = m^2 + 14mn + 49n^2$
 VZORCE4 Rozklad druhé mocniny součtu: $a^2 + 4ab + 4b^2 = (a+2b)(a+2b)$
 VZORCE5 Druhá mocnina rozdílu: $(6x-0.2)^2 = 36x^2 - 2.4x + 0.04$
 VZORCE6 Rozklad druhé mocniny rozdílu: $9-6y+y^2 = (3-y)(3-y)$
 VZORCE7 Užití vzorců - směr (výběr 10 ze 120 příkladů)
 VZORCE8 Rozklad podle vzorců - směr
 VZORCE9 Vytknutí a rozklad dle vzorce: $r-r^3 = r(1-r^2) = r(1+r)(1-r)$
 VZORCE10 Rozklad dle vzorce s mnohočleny: $(a^2+2b)^2 - b^2 = (a^2+2b+b)(a^2+2b-b)$

Práce s mocninami:

- MOCNINA2 Sčítání a odečítání mocnin: $3m^3 + 2m - 2m^3 + m^2 - 2m = m^3 + m^2$
 MOCNINA3 Násobení mocnin: $-1.5ab^2 \cdot 2ab = -3a^2b^3$
 MOCNINA4 Dělení mocnin: $6a^3b^4 : 4a^2 = 1.5ab^4$
 MOCNINA5 Umocňování mocnin: $(-0.3a^2b)^2 = 0.09a^4b^2$
 MOCNINA6 Násobení mnohočlenu jednočlenem: $(4n-5m)(3m^2) = 12m^2n - 15m^3$
 MOCNINA7 Násobení mnohočlenu mnohočlenem: $5mn(2m-3n) = 10m^2n - 15mn^2$

Vytýkání:

- VYTYK1 Vytýkání před závorkou: $a(ab+c) - b(ab+c) = (ab+c)(a-b)$
 VYTYK2 Postupné vytýkání: $3a+3b+ac+bc = 3(a+b)+c(a+b) = (a+b)(3+c)$

Práce s výrazy:

- VYRSCI Sčítání výrazů: $(3a+4)+(-2a-7) = 3a+4-2a-7 = a-3$
 VYRODC Odčítání výrazů: $(-9-3x)-(8+9x) = -9-3x-8-9x = -17-12x$
 VYRNAS Násobení výrazů: $(3-2b)(-b) = -3b + 2b^2$
 VYRDEL Dělení výrazů: $(20y - 8y^2) : 4y = 5 - 2y$
 VYRVYT Vytknutí z výrazu: $3a + 9 = 3(a + 3)$

NAVOD Stručný popis použití kláves při zadávání výrazů.

Programové vybavení CONSUL 2717/PMD85-2,3

Kč 148,-

SWD75/SWK75

ČEŠTINA - 5

SWD75/SWK75

Autor: Erika Kabrhelová, Náchod

Soubor programů pro podporu výuky českého jazyka a procvičení probírané látky žáky na nižším stupni základních škol.

OSMI1.BAS

První z programů typu "český jazyk hrou". Úkolem žáka je najít 10 slov na téma výživa v osmisměrce (vodorovně i obráceně, svisle dolů i nahoru, šikmo zleva doprava či naopak). Všechna slova jsou uvedena správně česky malými písmeny. Žák je má napsat pokud je najde pouze velkými písmeny bez diakritiky. Nalezená slova jsou z osmisměrky vymazána pro jednodušší hledání zbylých slov z daného tématu. Úspěšné řešení je slovně ohodnoceno.

OSMI2.BAS, OSMI3.BAS, OSMI4.BAS, OSMI5.BAS

Podobné osmisměrky s tématy: domácí zvířata, naše rodina, kalendář a naše tělo.

KROUCAR.BAS

Program k procvičování psaní slov s u-ú-ů obsahující celkem 30 slov nebo slovních spojení. Je vhodný pro žáky od 2.ročníku. Žák má za úkol vybrat správný výraz z nabídky 3 možností A -B -C stiskem správné klávesy a EOL. Potvrzení správné volby je označeno hvězdičkou, po chybě je zprávou "zde je správně" označen řádek se správným tvarem slov.

SHODA.BAS

Jednoduchý program na procvičování shody podmětu s přísudkem obsahující 30 vět. Vhodné pro žáky 4. a 5. ročníku.

HACEK.BAS

Program se 30 slovy na procvičování háčků a čárek od 2.třídy

SLOVESA2.BAS

Procvičování znalostí o slovesech na 20 větách, v nichž má žák určit mluvnickou osobu, číslo, způsob a někdy i čas. Vhodné pro žáky 4. a 5. ročníku se slovním ohodnocením a obrázkem.

PRJMPR1.BAS

Procvičování psaní y-i ve 30 slovech přídavných jmen přivlastňovacích, vhodné pro žáky 5.ročníku.

STUPNO2.BAS

Stupňování přídavných jmen ve 30 slovech pro 5. a 6.ročník.

Informace:

Dovybavení počítačů C2717: MAGNA Brno, s.s r.o., Šumavská 15,

60200 Brno, tel.(05)4121 2619:

- disketový subsystém JPD 2x5.25"7 300 Kč
- interface Centronics pro připojení tiskárenn 850 Kč
- modul pro připojení televizoru 900 Kč
- připojení zapisovačů ARITMA XY1 450 Kč

Ceny jsou uvedeny bez DPH, která činí 23%.

Společnost dodává i jednotlivé počítače PC, jejich propojení do sítě, tiskárny, dodává i kompletní počítačové sítě. Cenové relace jsou v nabídce, kterou zašle na vyžádání.

Opravy PC, tiskáren, C2717, disketových mechanik a PMD-85

zajišťuje skupina techniků INCOTEXu jak u uživatelů tak v Brně na základě dohody. Adresa: Hybešova 42, 65664 Brno, tel.(05)4332 1268 línak 125, fax. (05)4321 1234.

Nabídka programů pro PC od INCOTEXu, Hybešova 42, 65664 Brno

- Elementární geometrie (lekce 1-5)398 Kč + 5% DPH
- sleva 50% při vrácení disket SWD20/22 C2717199 Kč + 5% DPH
- demoverse lekcí 2-4 na vratné disketě 5.25/3.5" 40 Kč + 5% DPH
(bude odečtena z ceny při vrácení diskety)
- RECORD, ZAPOR, DELENIpo 390 Kč + 5% DPH
při objednávce: dvou různých programů 690 Kč + 5% DPH
všech tří programů 890 Kč + 5% DPH
demoverse všech programů na vratné disketě 40 Kč + 5% DPH
- Matematika ZŠ 490 Kč + 5% DPH
demoverse na vratné disketě 40 Kč + 5% DPH

Publikace vztahující se k jazyku QBasic:

- QBasic-kompendium, S. Baloui, 1993, UNIS Smetanova 3, Brno 540 Kč
- Quick Basic 4.5, E.+J. Špalovi, 1993, Grada, Dlouhá 39, Praha 295 Kč
- Quick Basic, P. Kroha, 1991, SWS Software, p.b. 219, Zlín 298 Kč

Ceník 3/94: Počítače: C2717 PMD85-2 PMD85-3 Diskety: 8"....5,25"

Disk./kazety Ceny jsou uvedeny bez DPH 5 % Cena: Kusů:

SWD40/SWK40	-ZEMEPIS-D (P.Cibulka)	198 Kč
SWD41/SWK41	-ČEŠTINA-2 (J.Rosecký a kol.)	198 Kč
SWD42/SWK42	-CHEMIE-2 (P.Hlaváček)	198 Kč
SWD43/SWK43	-CHEMIE-3 (několik autorů)	198 Kč
SWD44/SWK44	-GEOM+MAT (Z.Lusk)	148 Kč
SWD45/SWK45	-HRY-3 (freeware)	98 Kč
SWK46	-HRY-4 (freeware, stroj.kód)	98 Kč
SWD47/SWK47	-HRY-5 (I.Blažek)	148 Kč
SWD48/SWK48	-EKOLOGIE-2 (T.Hlavička)	198 Kč
SWD49/SWK49	-NĚMČINA-4 (NEM5-NEM10, CITATY2)	198 Kč
SWD50/SWK50	-MAT+GEOM2 (O.Kánský)	198 Kč
SWD51/SWK51	-HRY-6 (freeware)	98 Kč
SWD52/SWK52	-FYZ-OPT (několik autorů)	198 Kč
SWD53/SWK53	-NĚMČINA-5 (NEM11 - NEM15)	198 Kč
SWD54/SWK54	-NĚMČINA-6 (NEM16 - NEM20)	198 Kč
SWD55/SWK55	-ASTRONOMIE (několik autorů)	198 Kč
SWD56/SWK56	-TESTY-2 (několik autorů)	198 Kč
SWD57/SWK57	-TESTY-3 (P.Hlaváček)	198 Kč
SWD58/SWK58	-HRY-7 (I.Blažek)	148 Kč
SWD59/SWK59	-MATEMATIKA 1-3 (E.Kabrhelová).....	148 Kč
SWD60/SWK60	-ČEŠTINA-3 (E.Kabrhelová).....	148 Kč
SWD61/SWK61	-Soubor Foltýn	298 Kč
SWD62/SWK62	-HRY-8 (I.Blažek)	148 Kč
SWD63/SWK63	-STAROVĚK (P.Hlaváček)	198 Kč
SWD64/SWK64	-MOCNINY (Komenium)	198 Kč
SWD65/SWK65	-LINEARNÍ funkce (Komenium)	198 Kč
SWD66/SWK66	-KVADRATICKÉ fkce + podobnost (Komenium)	198 Kč
SWD67/SWK67	-HRY-9 (Freeware)	98 Kč
SWD68/SWK68	-Soubor RUBÝ (Zdeněk Rubý).....	298 Kč
SWD69/SWK69	-ČEŠTINA 4 (E.Kabrhelová).....	148 Kč
SWD70/SWK70	-TESTY-4 a PÍSNÍČKY (několik autorů)....	148 Kč
SWD71/SWK71	-MATEMATIKA-5 (Alois Poštulka).....	198 Kč
SWD72/SWK72	-TESTY 10 (P.Hlaváček).....	198 Kč
SWD73/SWK73	-HRY 10 (Freeware)	98 Kč
SWD74/SWK74	-MATEMATIKA 7/8 (Z.Rubý).....	298 Kč
SWD75/SWK75	-ČEŠTINA 5 (E.Kabrhelová)	148 Kč

Předchozí ceník SW1-39, příruček a Aktualit platí i nadále.

Programy pro počítače typu PC:

SWPC: Elementární geometrie:.... disketa	3.5	5.25	390,-
Demoverse:.....	3.5	5.25
Record Zapor Delení (po 390,-)...	3.5	5.25
Demoverse:.....	3.5	5.25
Matematika ZČ:.....	3.5	5.25	490,-
Demoverse:	3.5	5.25

Objednavatel:..... PSČ:.....

Vyřizuje: Telefon:

Anketa: Prosíme Vás o několik minut Vašeho času a zodpovězení několika následujících otázek pro naši informaci.

Které programy z naší nabídky považujete za nejzdařilejší a proč?

Které programy mají naopak nedostatky a jaké?

Jaké programy postrádáte a co by měly obsahovat (pro C2717 i PC)?

Ve kterých předmětech a třídách pracujete s počítači?

Využíváte počítače i mimo výuku a jak (např. pro rodiče žáků)?

Co brání většímu využití počítačů C2717 ve Vaší škole?

Kdy předpokládáte výměnu počítačů C2717/PMD85 za PC?

Máte-li již PC, jaké mají diskety? 360KB - 1.2MB - 1.44MB

A jakou grafickou kartu? Hercules - EGA - VGA - SVGA -....

Váš názor na kurs QBASIC v Aktualitách BASIC-QBASIC:

Máte předběžně zájem o příručku s tímto kursem? ks... cena ...

Děkuji za odpovědi a těším se na další spolupráci.

Za INCOTEX, s.s r.o., Hybešova 42, 65664 Brno
Telefon: (05) 4332 1267-8, fax: (05) 4321 1234

Pavel Hlaváček