Code-it! QBasic knjiga

Quick Baisc je inačica BASIC programskog jezika. To je programski jezik baziran na DOS-u.

QBasic za apsolutne početnike 1

Uvod

Dakle, kao što vam sâm naslov govori, ovo je vodič za totalne početnike u programiranju. Dobro, ne baš TOTALNE... Trebali biste već znati neke PRAVE osnove, tipa *"Što je programiranje?"* ili *"A compiler?"* ili nešto slično (npr. da se program izvršava red po red odozgo prema dolje). Takve osnove prepuštam nekome drugom. Ovo je za one koji ne znaju skoro ništa, a ono nešto što znaju je samo u teoriji. I odmah vam dajem prvo **upozorenje:** s QBasicom nećete moći ništa. To je samo za učenje, od njega ništa upotrebljivog. Dobra vijest je da je poslije poprilično lako prijeći na Visual Basic, što je itekako upotrebljivo. Eto, to je to od uvoda, znači, krećemo...

Naredba CLS

Svaki programski jezik, pa tako i QBasic, ima komad kôda koji ide na sami početak programa jer... jer to tako jednostavno je. Kasnije ćete saznati što to točno znači i zašto to tako je, za sada morate znati samo da na početku SVAKOG programa ide:

```
CLS
```

Reći ću vam da je CLS kratica od Clear Screen (Očisti Ekran), tek toliko da je lakše zapamtite. Ajmo dalje...

Naredba PRINT

Prva "prava" naredba koju ćete učiti je naredba PRINT. Ona služi za ispis texta na ekran. I nema **nikakve veze** ni sa kakvim printerom niti printanjem (da ne bi bilo nisam znao/la)! Ono što piše poslije naredbe PRINT bit će ispisano na ekranu kad se program pokrene. Pa napišite:

```
CLS
PRINT "Hello world!"
```

i pokrenite program pritiskom na tipku F5. Ekran će pocrnjeti i na njemu će biti napisano:

Citat:

> Hello world!

Qbasic izvršava programe kao u DOS-u, pa na raspolaganju imate 80 znakova u svakom od 25 redova na ekranu. Kao što možete vidjeti, da biste nešto ispisali, morate to staviti u navodnike. Ako želite koristiti novu naredbu PRINT, to ćete napraviti u novom redu. I kad pokrenete program, to što ste napisali će se pojaviti u novom redu. Ako vam je sve to još malo mutno, igrajte se. To mi je savjet za svaki put kad vam nešto bude "malo mutno" (osim u subotu navečer), pa ga zapamtite.

Citat:

> **Napomena:** Sve možete pisati malim slovima. Kad prijeđete u novi red, Qbasic će automatski naredbe napisati velikim slovima, a znakove matematičkih operacija razdvojiti (npr. **3+2** će postati **3 + 2**. Naravno, ono pod navodnicima ostaje isto.

Sad napišite:

```
CLS
PRINT 3
```

Pogledate li rezultat (F5), vidjet ćete da radi i bez navodnika. No, ako umjesto **3** napišete **Hello**, definitivno ne radi (probajte). A ako napišete **3 + 2**... sad vam je već jasnije. Matematičke operacije bivaju izračunate (u, al profesionalno zvučim!) i njihov rezultat ispisan (rezultat od 3 je 3, da ne bi bilo zabune). A slova moraju ići u navodnike. Inače ih Qbasic pokušava shvatiti kao brojeve, a to onda nikako ne valja.

Citat:

> **Napomena:** Neke stvarčice o naredbi PRINT koje je jako dobro znati, a zapravo i nisu toliko važne naći ćete na kraju ovog članka.

Osnove varijabli

Varijable su (zapravo) prilično komplicirane stvari koje imaju veze s RAM-om i onom nečem u procesoru itd. (neću napisati ništa više da ne ispadnem glup). Ono što vama treba je da znate da su to... kao posude u kojima čuvate podatke. Svaka od njih ima svoje ime (naljepnica na posudi) i svoju vrijednost (ono što je **u** posudi). Npr. Ja želim spremiti podatak (koji će mi poslije trebati) da Mirko ima 23 godine. Znači, uzmem posudu, na nju nalijepim naljepnicu "Mirkove godine" i unutra ubacim broj 23 (ne pitajte kako, samo zamislite!). U Qbasicu bi to bilo da kreiram varijablu naziva **Mirko.godine** i dodijelim joj vrijednost **23**, odnosno:

```
Mirko.godine = 23
```

```
Mirko.godine = 23
```

Onda kreiram još jednu varijablu u koju napišem da Slavko ima 26 godina:

```
Slavko.godine = 26
```

Sada s tim varijablama mogu raditi kao da su one - njihove vrijednosti, odnosno, kad napišem **Mirko.godine**, Qbasic će to razumjeti kao **23**. Shodno tome (evo me opet profesionalnog), ako napišemo

```
PRINT Mirko.godine + Slavko.godine
```

računalo će, kad se program pokrene, ispisati **49**.
A možemo kreirati još jednu varijablu koja će biti zbroj Mirkovih i Slavkovih godina:

```
MiS.Godine = Mirko.godine + Slavko.godine
```

Sada je u varijabli **MiS.Godine** (ovo je slučajno ispalo slično kao miss of the year, šta da radim) vrijednost **49**, i s njom možemo raditi što nas volja. O imenima varijabli više na kraju ovog dijela, inače, ona uopće nisu važna – ove tri varijable su se isto tako mogle zvati **a**, **b** i **c** ili **Jozo**, **Pero** i **Djuro** ili bilo kako drugačije (dobro, ne bilo kako... napisat ću na kraju).

A što ako želimo sačuvati podatak (staviti u posudu) da se Mirkova žena zove Fata? (dobro, malo miješam viceve, ali to nema veze). Kod naredbe PRINT sam rekao da brojke (koje služe za matematičke operacije) ne idu pod navodnike, a slova idu. To vrijedi i ovdje. Jedini je problem što odmah poslije varijable u koju idu slova uvijek treba napisati znak $ ([Shift] + [4]). Dakle:

```
Mirkova.zena$ = "Fata"
```
O varijablama će se pisati još, ali to ćemo poslije.

Naredba INPUT

Služi za to da bi korisnik (onaj koji se služi vašim programom) mogao nešto upisati. Nakon upisivanja riječi INPUT, obično se piše poruka pod navodnicima (tipa "Upiši jedan troznamenkasti broj."). Ako se napišu navodnici, poslije njih ide zarez (,) pa zatim varijabla u koju će biti upisana vrijednost koju korisnik napiše (opet, ako se očekuje unos slova, poslije varijable treba ići znak $).

```
CLS
INPUT "Koliko Mirko ima godina?", Mirko.godine
INPUT "Koliko Slavko ima godina?", Slavko.godine
MiS.Godine = Mirko.godine + Slavko.godine
PRINT "Mirko i Slavko zajedno imaju ovoliko godina:"
PRINT MiS.Godine
PRINT
INPUT "Kako se zove Mirkova žena?", Mirkova.zena$
PRINT "Ime Mirkove žene:"
PRINT Mirkova.zena$
```

Citat:

> **Napomene:**
> - Onaj jedan **PRINT** koji je sâm u jednom redu služi za to da bi dobili jedan red razmaka (rekoh da svaki PRINT piše svoje u sljedeći red, pa je i ovaj napisao svoje ništa u sljedeći red).
> - Pošto s imenom Mirkove žene ne mogu ništa, samo sam ga ispisao (zadnji **PRINT**)

Još samo da kažem da s jednom naredbom INPUT možete "natjerati" korisnika da upiše više stvari (ali bi bilo dobro da mu to i kažete - za to služe navodnici pokraj INPUT). To se radi tako da umjesto jedne varijable na kraju, napišete više varijabli odvojenih zarezom. Kad korisnik to bude morao upisati, i on će varijable odvajati zarezom. Tako će naredba

```
INPUT "Upiši tri broja: ", a, b, c
```

zahtijevati od korisnika da upiše npr.

Citat:

> 23, 1, 58

i stisne tipku [Enter]. (Brojevi su proizvoljni, naravno, a razmaci u upisivanju nisu nužni. To je moglo biti i **23,1,58**.)
Evo vam i primjer...

```
CLS
INPUT "Upiši tri broja koja želiš zbrojiti: ", a, b, c
zbroj = a + b + c
PRINT "Zbroj ta tri broja je:"
PRINT zbroj
```
Sve kužite? Na ekranu bi (na kraju programa) trebalo biti nešto tipa:

Sve kužite? Na ekranu bi (na kraju programa) trebalo biti nešto tipa:

```
Upiši tri broja koja želiš zbrojiti: 23, 1, 58
Zbroj ta tri broja je:
82
```

I to je to.

Citat:

> **Napomena:** različiti tipovi varijabli (slova/brojke) se smiju miješati, npr. ako umjesto **a, b, c** napišete **a, b$, c**, korisnik će moći upisati **23, bla, 3511**.

Dodaci

- Naredba PRINT

Kad sam trebao napisati zbroj Mirkovih i Slavkovih godina, u kôdu sam napisao:

```
PRINT "Mirko i Slavko zajedno imaju ovoliko godina:"
PRINT MiS.Godine
```

No, to se moglo napraviti i ljepše, ovako:

```
PRINT "Mirko i Slavko zajedno imaju"; MiS.Godine; "godina."
```

Pa bi ispalo:

Citat:

> Mirko i Slavko zajedno imaju 49 godina.

Tu se nema što objašnjavati... sami skužite. Broj mijenjanja između texta i varijable je neograničen, samo treba na prijelazu staviti točku-zarez (;). Zapravo, ne treba. Qbasic će to napraviti umjesto vas! (fuj, ko reklama za usisavač, multipraktik i tenk u jednom!). Prema tome, ako napišete

```
PRINT "Mirko i Slavko zajedno imaju" MiS.Godine "godina."
```

nakon prelaska u novi red, to će postati

```
PRINT "Mirko i Slavko zajedno imaju"; MiS.Godine; "godina."
```

Kao što ćete vidjeti, Qbasic automatski dodaje razmake između varijabli i texta. Moguće je staviti i više varijabli zaredom, pa i više textova zaredom, makar to i nema previše smisla (mislim na textove, više varijabli ima smisla). Ako stavite zarez (a ne točku-zarez), bit će kao da ste stisnuli tipku TAB (malo žešće razdvojeno – omogućava pisanje u stupcima). Isprobavajte!

Umjesto da pišete PRINT svako malo, možete napisati i upitnik [?]. Kad prijeđete u novi red, upitnik će automatski postati PRINT.

- Varijable

Ime varijable može imati **do 256 znakova**, ne smije počinjati brojem, velika i mala slova nisu važna, smije sadržavati znakove: **abcdefghijklmnopqrstuvwxyz.0123456789** (ova točka se isto ubraja)
i mislim da je to sve.
Dajte imena varijablama pametno, tako da znate što je što (kad budete radili program s parsto varijabli, znat ćete o čemu pričam). Sad vam je vjerojatno bezveze svako malo pisati **Mirko.godine** i **Slavko.godine** – zašto ne biste pisali **a** i **b** kad je lakše? – ali, poslije će vam to dobro doći.

- Ostalo...

Decimalna točka je točka, ne zarez.
Umjesto **0.25**, možete napisati samo **.25**. Štoviše, sámo će se promijeniti u **.25**.

U vašem kodu će vam često trebati vaši komentari (tipa: "ovo je tu tak da bi mi poslije bilo laše napravit ono tam" il sl.). A pogotovo ako budete radili na nekom projektu zajedno s nekim. Komentari se u QBasicu pišu tako da ispred njih (tj. lijevo od njih) jednostavno stavite apostrof (') {kao [?], ali bez tipke [Shift]}. Sve desno od apostrofa QBasic ne gleda. Primjere ćete moći vidjeti u rješenjima zadataka uz ovaj vodič.

I na kraju...

Evo, ovo su najosnovnije osnove osnova Qbasica koje morate znati. Sad to probavljajte neko vrijeme, jer, koliko god lagano ovo izgledalo (i bilo), saznali ste jako puno informacija odjednom i bilo bi dobro da sad malo stanete s ovim vodičem i prihvatite se zadataka tak da vam to "sjedne". A zadaci su u privitku.

Ako netko ima ikakvih pitanja... pitajte na forumu.

QBasic za apsolutne početnike 2

QBasic za apsolutne početnike 2

IF-THEN-ELSE naredbe

Ako znate imalo engleskog, shvatili ste o čemu se radi. IF-THEN-ELSE znači AKO-ONDA-INAČE, dakle imamo posla s uvjetima. Evo jednostavnog primjera za početak. Treba nam program koji će nam reći je li učitan broj veći od nule.

```
CLS
INPUT "Upiši broj: ", x
IF x > 0 THEN PRINT "Je."
```

Stvarno je jednostavno. Naredbu IF (zadnji red) čitamo onako kako je napisano:

```
AKO je x veći od nule, ONDA ispiši "Je".
```

I sad vama čudno... pričalo se o nekom ELSEu... Pa evo vam! Prvom primjeru dodajte da napiše je li veći **ili manji** od nule.

```
CLS
INPUT "Upiši broj: ", x
IF x > 0 THEN PRINT "Je." ELSE PRINT "Nije."
```

Ovo čitate kao i prvi primjer, samo mislim da bolje zvuči kad umjesto INAČE kažete A AKO NIJE:

```
AKO je x veći od nule, ONDA ispiši "Je", A AKO NIJE (INAČE), ispiši "Nije".
```

A sad zamislite da smo to sve htjeli napisati lijepo: "Broj (taj i taj) je veći od nule" (ili nije). Gledajte:

```
CLS
INPUT "Upiši broj: ", x
IF x > 0 THEN PRINT "Broj"; x; "je veći od nule." ELSE PRINT "Broj"; x; "nije veći od
nule."
```

Ajme gužve u jednom redu!!! Zato je bolje pisati u više redova, a naredbe uvlačiti (TAB):

```
CLS
INPUT "Upiši broj: ", x
IF x > 0 THEN
    PRINT "Broj"; x; "je veći od nule."
ELSE
    PRINT "Broj"; x; "nije veći od nule."
END IF
```

Napomena: Pošto je u web stranici poprilično nemoguće napisati nešto uvučeno (tipkom TAB), a uvlačenja su (bar meni) podosta važna u programiranju, koristio sam bijelu boju texta da dobijete privid uvučenog. Žao mi je ako budete morali kopirati neki kod, onda ćete svaki bijeli "TAB" morati brisati i ...

Eto, sad je ljepše, jedino morate zapamtiti da, kad pišete u više redova, na kraju mora pisati END IF. Kad pišete u jednom redu, NE SMIJE!!!

Još jedan jednostavan primjer: password. Evo programčića koji kaže je li lozinka ispravna (u ovom primjeru, lozinka je KjuBejzik).

```
CLS
INPUT "Password? ", a$
IF a$ = "KjuBejzik" THEN
    PRINT "Password confirmed"
ELSE
    PRINT "Error: wrong password"
END IF
```

Vratimo se sad drugom primjeru. Tamo imamo grešku. Ako upišemo broj 0, program će napisati da je nula manja od nule!!! Sad nam treba naredba ELSEIF (ne ELSE IF!!!):

```
CLS
INPUT "Upiši broj: ", x
IF x > 0 THEN
    PRINT "Broj"; x; "je veći od nule."
ELSEIF x < 0 THEN
```

```
ELSEIF x < 0 THEN
    PRINT "Broj"; x; "nije veći od nule."
ELSE
    PRINT "Upisali ste nulu."
END IF
```

Čitamo:

```
AKO je x veći od nule, ONDA
    ispiši "Je"
INAČE, AKO je x manji od nule (odnosno, "AKO nije veći od nule, nego je manji")
    ispiši "Nije".
INAČE
    ispiši "Upisali ste nulu."
KRAJ
```

Napomena: broj ELSEIF naredbi je neograničen.

Evo još jednog (stvarno debilnog) primjera koji postoji samo da bi se ljudi zapitali ima li smisla nastaviti s učenjem programiranja. Uglavnom, program pita pada li kiša i kaže je li potrebno ponjeti kišobran.

```
CLS
INPUT "Pada li kiša? ", a$
IF a$ = "Da" THEN
    PRINT "Ponesi kišobran."
ELSEIF a$ = "Ne" THEN
    PRINT "Nemoj ponijeti kišobran."
ELSE
    PRINT "Daj, piši razgovjetnije!!!"
END IF
```

Ovaj zadnji PRINT je tu jer, ako korisnik napiše "ne" (s malim n), QBasic to neće pročitati kao "Ne", i to neće valjat.

CASE naredba

Pogledajte ovaj primjer. Recimo da vam jednom u nekom programu za rad s datumima zatreba dio programa koji će ispisivati hrvatski naziv mjeseca (a poznat je redni broj):

```
CLS
INPUT "Broj mjeseca: ", x

IF x = 1 THEN
    PRINT "Siječanj"
ELSEIF x = 2 THEN
    PRINT "Veljača"
ELSEIF x = 3 THEN
    PRINT "Ožujak"
ELSEIF x = 4 THEN
    PRINT "Travanj"
ELSEIF x = 5 THEN
    PRINT "Svibanj"
ELSEIF x = 6 THEN
    PRINT "Lipanj"
ELSEIF x = 7 THEN
    PRINT "Srpanj"
ELSEIF x = 8 THEN
    PRINT "Kolovoz"
ELSEIF x = 9 THEN
    PRINT "Rujan"
ELSEIF x = 10 THEN
    PRINT "Listopad"
ELSEIF x = 11 THEN
    PRINT "Studeni"
ELSEIF x = 12 THEN
    PRINT "Prosinac"
ELSE
    PRINT "Taj mjesec ne postoji."
END IF
```

```
    PRINT "Taj mjesec ne postoji."
END IF
```

Puno pisanja, jel'da? Za to služi CASE naredba. Najbolje ćete skužiti na primjeru, pa evo vam isto ovo pomoću naredbe CASE:

```
CLS
INPUT "Upiši broj mjeseca: ", x

SELECT CASE x
CASE 1
    PRINT "Siječanj"
CASE 2
    PRINT "Veljača"
CASE 3
    PRINT "Ožujak"
CASE 4
    PRINT "Travanj"
CASE 5
    PRINT "Svibanj"
CASE 6
    PRINT "Lipanj"
CASE 7
    PRINT "Srpanj"
CASE 8
    PRINT "Kolovoz"
CASE 9
    PRINT "Rujan"
CASE 10
    PRINT "Listopad"
CASE 11
    PRINT "Studeni"
CASE 12
    PRINT "Prosinac"
CASE ELSE
    PRINT "Taj mjesec ne postoji."
END SELECT
```

Manje texta, preglednije, lakše za copy/paste... Znači, na početku naredbe ide:

```
SELECT CASE varijabla
```

Poslije toga za svaku vrijednost:

```
CASE vrijednost
(vrijednost može biti i niz slova, onda se TO stavlja u navodnike i treba biti $ poslije
varijable)
```

Umjesto ELSE:

```
CASE ELSE
```

I na kraju:

```
END SELECT
```

GOTO naredba

GOTO se sastoji do dvije riječi: GO TO (ko bi reko!), odnosno IDI NA. Služi za preskakanje dijela kôda ili vraćanje nazad, ali nema neku pretjeranu primjenu. Mislim da se svako korištenje naredbe GOTO može napisati na drugačiji način (bilo grananjem, petljama...). Uglavnom, na početku jednog reda se napiše broj, a kad bilo gdje u programu bude napisano GOTO taj broj, QBasic se vraća nazad ili preskače kôd do linije na čijem je početku taj broj. (sad sam ga malo zakomplicirao, al skužit ćete.)
Primjer: program upisuje brojeve dok se ne upiše nula. Onda ispiše zbroj svih upisanih brojeva:

```
CLS
zbroj = 0

[b]20[/b] INPUT "Upiši broj: ", x

IF x = 0 THEN
```

```
IF x = 0 THEN
    GOTO 10
ELSE
    zbroj = zbroj + x
    GOTO 20
END IF

10 PRINT zbroj
```

Znači, kad QBasic dođe na **GOTO 10**, on ode na 10, odnosno, izvrši naredbu **PRINT zbroj**. Onda ide dalje, a dalje nema ništa, pa je program gotov.
Kad dođe na **GOTO 20**, ode na 20, odnosno, vrati se na **INPUT**, izvrši ga i ide dalje, na naredbu **IF**.
Isprobavajte, pa ćete skužiti.

Matematika i logika

Uskoro će vam trebati neke žešće matematičke operacije (i funkcije itd.) osim +, -, *, /, <, > i =. Evo vam pregled svih za koje se sad mogu sjetiti.

+, -, *, /, =
--- to znate
<, >
--- manje od, veće od
<=, >=, <>
--- manje ili jednako, veće ili jednako, nejednako
MOD
--- daje ostatak pri dijeljenju.
13 MOD 5 = 3
(13 / 5 = 2 i ostatak 3)
ABS
--- daje apsolutnu vrijednost broja.
ABS(-5) = 5
(matematički napisano: |-5| = 5)
^ ([ALT] + [94])
--- exponencija.
2 ^ 3 = 8
SQR
--- kratica od square root - kvadratni korijen
SQR(16) = 4

SIN, COS, TAN
--- daje sin, cos i tan nekog kuta U RADIJANIMA.
COS(pi) = -1

Napomene:
- Da biste pretvorili stupnjeve u radijane, pomnožite stupnjeve sa (pi / 180).
SIN(90 * (pi / 180)) = 1
- kad radite s brojem pi, bilo bi dobro da na početku programa napišete
pi = 3.141592654

EXP
--- e na n-tu.
EXP(1) = 2.718281828
LOG
--- limes.
LOG(EXP(1)) = 1

INT
--- zaokružuje na prvi manji cijeli broj
INT(5.76) = 5
INT(-5.2) = -6
FIX
--- miče decimalni dio broja
FIX(1.8) = 1
FIX(-5.2) = -5
CINT
--- zaokružuje na najbliži cijeli broj
CINT(2.9) = 3

```
CINT(2.9) = 3
CINT(-2.2) = -2
CINT(2.5) = 2
```

Bilo bi dobro da ovo imate uz sebe stalno (dobro, ne stalno, samo kad programirate!!!).
Pa si ovo isprintajte, praktičnije je: skinite privitak.

Logički operateri (tako ih svi zovu, nemam pojma šta točno znači "operater", i nije važno) su male riječi koje nam pomažu u nekim stvarima. Prvo pregled, onda primjer. Dobro ga proučite.

AND - i
obje strane moraju biti istinite
OR - ili
bar jedna strana mora biti istinita
XOR
jedna strana mora biti istinita, ali ne i druga.
(iskreno, ovo nikad nisam koristio)

Primjer:

```
x = 2
y = 7
IF x = 2 AND y > 5 THEN PRINT "Da." ELSE PRINT "Ne.";
IF x = 3 AND y > 5 THEN PRINT "Da." ELSE PRINT "Ne.";
IF x <> 3 AND y >= 5 THEN PRINT "Da." ELSE PRINT "Ne."

IF x = 3 OR y > 5 THEN PRINT "Da." ELSE PRINT "Ne.";
IF x = 2 OR y = 5 THEN PRINT "Da." ELSE PRINT "Ne.";
IF x = 1 OR y = 5 THEN PRINT "Da." ELSE PRINT "Ne."

IF x => 2 XOR y = 5 THEN PRINT "Da." ELSE PRINT "Ne.";
IF x > 2 XOR y = 7 THEN PRINT "Da." ELSE PRINT "Ne.";
IF x = 2 XOR y < 8 THEN PRINT "Da." ELSE PRINT "Ne."
```

Ispis:

```
Da. Ne. Da.
Da. Da. Ne.
Da. Da. Ne.
```

Evo vam i **zadaci** u privitku.

QBasic za apsolutne početnike 3

FOR-NEXT petlja

Zamislite da dobijete zadatak da morate ispisati prvih 10 prirodnih brojeva. Postoji nekoliko rješenja (ja ću napisati tri):

1.) Koristeći naredbu PRINT (i copy/paste). Najlošije rješenje.

```
CLS
PRINT 1
PRINT 2
PRINT 3
PRINT 4
PRINT 5
PRINT 6
PRINT 7
PRINT 8
PRINT 9
PRINT 10
```

2.) Koristeći naredbu PRINT i GOTO. To je najpametnije što možete napraviti s trenutnim znanjem (iz prošla 2 vodiča):

```
CLS
i = 1
10 PRINT i
i = i + 1
```

```
i = i + 1
IF i <= 10 THEN GOTO 10 ELSE END
```

Pojašnjenje: i je 1. To se ispiše. Tada i postaje 2. Ako je još uvijek u skupu brojeva do 10, vraća se na ispis i ponovo povećava za 1. Kad i postane 11, ne ispisuje se, nego se program završava.

3.) Koristeći petlju FOR-NEXT:

```
CLS
FOR i = 1 TO 10
        PRINT i
NEXT i
```

Pojašnjenje:
- Naredba FOR označava početak FOR-NEXT petlje; naredba NEXT označava kraj.
- Sve između te dvije naredbe se najčešće zove "niz naredbi koji se ponavlja".
- Prvi put kad se ponavlja (u ovom slučaju naredba PRINT i), i je 1. Sljedeći put je 2, pa 3... do 10. To piše odmah poslije naredbe FOR – i ide od 1 do 10.
- Dakle, program ide odozgo prema dolje. Kad dođe do NEXT, povećava varijablu i vraća se nazad na red poslije FOR (ako varijablja još nije dosegla konačnu vrijednost) ili nastavlja dalje (ako jest).

Još jedan primjer – program zbraja prvih n prirodnih brojeva:

```
CLS
INPUT "Koliko brojeva? ", n
zbroj = 0

FOR i = 1 TO n
        zbroj = zbroj + i
NEXT i

PRINT zbroj
```

Naredba zbroj = zbroj + i će se ponavljati n puta, i će prvi put biti 1, pa 2... i tako sve do n. Kad i dostigne broj n, zbroj će se ispisati. Jednostavno je!!!

Napišite program koji će množiti samo prirodne parne brojeve do broja n. A šta sad? Jedna je opcije da unutar FOR petlje provjeravate je li taj broj paran ili ne (IF i MOD 2 = 0 THEN... ELSE...). Ali, lakše je koristiti naredbu STEP:

```
CLS
INPUT "Do koliko? ", n
umnozak = 1

FOR i = 2 TO n STEP 2
        umnozak = umnozak * i
NEXT i
PRINT umnozak
```

Prvi put će i biti 2 (jer je i = 2 TO n...). Dalje će se uzimati svaki drugi broj (...STEP 2), dakle 4, 6, 8...

Broj poslije STEP naredbe može biti i negativan. Npr. ispiši sve brojeve od -5 do 5 unazad:

```
CLS
FOR i = 5 TO -5 STEP -1
        PRINT i
NEXT i
```

```
[small]Napomena: varijabla koja se mijenja (ona poslije FOR i NEXT) ne mora uvijek biti
i. Može biti i humuhumunukunukuapuaa, ali je i uobičajeno. [/small]
```

DO-LOOP petlja

Koristi se za ponavljanje nekih naredbi ("niz naredbi koji se ponavlja") dok neki uvjet ne bude zadovoljen / dok je neki uvjet istinit. Recimo, primjer s početka ovog članka (ispisati brojeve do 10) se može napisati ovako:

```
CLS
i = 1
DO
        PRINT i
        i = i + 1
```

```
        i = i + 1
LOOP UNTIL i > 10
```

Naredbe unutar petlje (PRINT i & i = i + 1) će se ponavljati dok i ne postane veće od 10 (UNTIL i > 10). Opet, kad program dođe do naredbe LOOP, provjerava uvjet i ovisno o tome se vraća na red poslije DO ili nastavlja dalje.
Ovo isto se moglo napraviti malčice drukčije:

```
CLS
i = 1
DO
        PRINT i
        i = i + 1
LOOP WHILE i <= 10
```

Opet ista priča, jedino što se te dvije naredbe ponavljaju dok god je i <= 10 (WHILE i <= 10). Pomaže ako znate značenje engleskih riječi "until" i "while", problem je što se one na hrvatski prevode skoro jednako ("dok").

Napomena: naredbe UNTIL ili WHILE se mogu staviti poslije naredbe DO umjesto poslije LOOP. Razlika je u tome što se onda uvjet provjerava na početku. Logikom ćete lako doći do rješenja na bilo koji način (obično možete na sva četiri – UNTIL kod LOOP, UNTIL kod DO, WHILE kod LOOP, WHILE kod DO).

No, ovo je puno lakše napraviti pomoću naredbe FOR-NEXT. Čemu onda služi DO-LOOP petlja? Evo vam opet odgovor pomoću primjera: Program upisuje brojeve dok se ne upiše 0. Onda ispiše zbroj.

```
CLS
zbroj = 0

DO
        INPUT "Upiši broj: ", x
        zbroj = zbroj + x
LOOP UNTIL x = 0

PRINT zbroj
```

Sve vam je jasno: petlja se vrti i vrti i vrti dok god x nije = 0. Ne smeta što se ta nula na kraju zbroji – rezultat je isti.

Evo vam još: Program upisuje niz brojeva dok se ne upiše broj veći od 999 i ispisuje najmanji od njih.

```
CLS
min = 1000
DO
        INPUT "Upiši broj: ", x
        IF x < min THEN min = x
LOOP UNTIL x > 999
PRINT min
```

Skužite sami.

WHILE-WEND petlja

Potpuno bespotreban dio QBasica. Prvi primjer (s početka članka):

```
CLS
i = 1
WHILE i <= 10
        PRINT i
        i = i + 1
WEND
```

Čitajući i prevodeći na hrvatski: dok god je i <= 10, ispiši i i povećaj ga za 1. Potpuno isto kao:

```
CLS
i = 1
DO WHILE i <= 10
        PRINT i
        i = i + 1
LOOP
```

Jedino što se s WHILE-WEND petljom uvjet uvijek provjerava na početku i nema korištenja riječi UNTIL. Jadno. Dakle, koristite vi petlju DO-LOOP i sve će biti OK.

"Ugniježđene" petlje (nested loops)

Nitko vam ne brani da napravite jednu petlju unutar druge. Recimo da hoćete napisati tablicu množenja (do 10x10):

```
CLS
FOR i = 1 TO 10
        FOR j = 1 TO 10
                PRINT i * j;
        NEXT j
        PRINT
NEXT i
```

Malo je grbavo, al nema veze...

Svaki put kad se izvršava PRINT i * j, varijable su drugačije:

1.) i = 1, j = 1
2.) i = 1, j = 2
3.) i = 1, j = 3
(...)
10.) i = 1, j = 10
11.) i = 2, j = 1
12.) i = 2, j = 2
(...)
100.) i = 10, j = 10

Poslije PRINT i * j stoji točka-zarez (;) da bi se sljedeći put ispisivalo u istom retku. Nakon NEXT j piše samo PRINT da bi se ispisalo ništa, i prešlo u novi red.

Da biste potpuno savladali petlje, treba vam puno zadataka (privitak)... Ali [b]stvarno puno, ovo ovdje vam nije ni približno dovoljno!!!

Qbasic za apsolutne početnike 4

Tipovi varijabli

Dosad su varijable koje smo koristili bile brojevi ili slova (ostali znakovi). Sad ćemo upoznati ostale oblike brojevnih varjiabli (i malo produbiti znanje o varijablama koje sadrže slova). QBasic koristi pet vrsta varijabli:

integer: cijeli broj između -32,768 i 32,767
long: cijeli broj između -2,147,483,648 i 2,147,483,647
single: decimalni broj "jednostruke preciznosti" (što god to značilo)
double: decimalni broj "dvostruke preciznosti" (isto ko i iznad)
string: znakovi koji se ne shvaćaju kao brojevi (mislim da najviše 256 znakova)

Naredba DIM

U većini programskih jezika, potrebno je "deklarirati" varijablu prije nego što ju upotrijebite. To znači da morate napisati naredbu u kojoj ćete reći programu ime varijable i njezin tip. U QBasicu se to radi naredbom DIM:

```
DIM ime_varijable AS tip_varijable
```

Dakle, da bi rekli QBasicu da je varijabla blah tipa integer, moramo napisati:

```
DIM blah AS INTEGER
```

```
NAPOMENA: Ako se to ne napiše, QBasic pretpostavlja da je varijabla tipa single.
```

Drugi način

Sjećate se da ste, kad vam je trebala tekstualna varijabla, poslije njezinog imena uvijek stavljali znak **$** ? E, to ima svoj razlog koji ćete sada saznati. Svaki tip podataka ima takav znak, tako da naredba DIM zapravo ne služi skoro ničemu – ne morate deklarirati varijable, samo ih počnite koristiti i upotrebljavajte ove znakove:

integer: **%**
long: **&**
single: **!**
double: **#**
string: **$**

double: #
string: $

Puta deset na...

Ako je neki broj prevelik (ili premalen) za tipove single ili double, on će biti napisan u obliku iz podnaslova – broj 1234567890123456789 će se pretvoriti u:
- ako je tip **single**: 1.234568**E+18** (1.234568 x 10 ^ 18)
- ako je tip **double**: 1.234567890123457**D+18** (1.234567890123457 x 10 ^ 18)

U tom obliku broj možete upisati i kad pišete kôd i kad korisnik upisuje nešto (naredba INPUT).

Čemu to služi?

U "pravim" programima (koji se ne pišu u QBasicu ;)), to služi za to da bi se smanjila količina memorije (RAM-a) koju program ždere. Znači, ako vam treba varijabla za broj ljudi nekih, nećete koristiti tip single ili double jer vam ne trebaju decimale (ne može negdje biti 156 i pol ljudi!), pa ćete koristiti integer (ili, u nekim ekstremnim slučajevima, long. Pa koliko koji tip podataka zauzima memorije?

integer: 16 bitova (2 bajta)
long: 32 bita (4 bajta)
single: 32 bita (4 bajta)
double: 64 bitova (8 bajtova)
string: 8 bitova (1 bajt) po znaku, a pošto je dozvoljeno 256 znakova – 2048 bitova (2 kilobita (256 bajtova))`Zato se koristi "tip podataka" STRING * n. Ako znate broj znakova koji neki string smije sadržavati, koristite ovo.DIM a AS STRING * 10`Ovime će se duljina niza a ograničiti na 10 znakova. Ovaj se tip podataka koristi i za stringove duže od 256 znakova.

Pošto QBasic ne služi za "prave" programe, tipovi podataka služe samo za programe u kojima dobivate malo žešće rezultate. Primjer: Znate onu pričicu kad je neki tip nekom Kinezu (ko zna zašto) rekao da hoće onoliko riže koliko bi bilo na šahovskoj ploči kad bi se na prvo polje stavilo 1 zrno, a na svako slijedeće polje duplo više zrna nego na prethodno? E, pa koliko bi na ploči bilo zrnja?

```
zbroj = 0
FOR i = 1 TO 64
        x = 2 ^ (i - 1)        ' polje 1: x = 2 ^ 0 (1); polje 2: x = 2 itd.
        zbroj = zbroj + x
NEXT i
PRINT zbroj
```

Kad bi varijabla **zbroj** bila tipa integer ili long, dobili biste grešku "Overflow". Da je single, dobili bi rezultat, ali malo jadan: 1.844674E+19 (1.844674 x 10^19). Tipom double, dobivate pristojniju vrijednost: 1.844674407370955D+19 (1.844674407370955 x 10^19).

Brojke i slova

Za tipove podataka važne su još dvije naredbe koje služe za pretvaranje iz brojevnog u znakovni tip i obrnuto. Recimo da u nekoj varijabli tipa string imate "234", i tome sad želite pribrojiti 1. To neće ići jednostavnom naredbom zbrajanja. Prvo vrijednost iz prve varijable ("234") morate prepisati u drugu, s tim da je ona druga brojčana (bilo koji brojčani tip). Za to služi naredba VAL (od eng. "value").

```
a$ = "234"
x% = VAL(a$)
```

Za obrnutu radnju koristi se naredba STR (od "string").

```
x% = 25
y$ = STR(x)
```

Eto, to su sve "tehnikalije" za sad koje trebate znati. Za ovu lekciju nema zadataka jer mislim da su bespotrebni.