RETRO KUTAK

Lekcija druga

Dobrodošli na drugi čas "Male škole programiranja na Commodore 64 u BASIC-u". Danas ćemo dati neke savete i upoznati se sa još nekim važnim komandama, mogućnostima kao i napraviti primere u vezi istih.

Za početak mali uvod

Vrste greške koje se javljaju kod programiranja
Prilikom programiranja mogu se javiti dve vrste grešaka sintetičke i logičke. Sintetičke greške su one koje se dešavaju prilikom pogrešnog unosa naziva naredbi koje kompjuter odmah prepozna i signalizira porukom "Syntax error" i brojem programske linije u kojoj je ista nastala. Pod ovom greškom se smatraju i one koje nastaju pogrešnim unosom tipa podataka na primer kada u promenljivu koja zahteva numerički podatak pokušavamo da upišemo tekst. Tada dobijamo poruke tipa "Redo from start" i sl. Mnogo nezgodnije su tzv.logičke greške. To su greške koje nastaju u programiranju i koje nisu uočljive na prvi pogled ali daju netačne programske rezultate. Tada je potrebno detaljno pregledati program, uneti razne ulazne podatke i pratiti kako se iste izvršavaju korak po korak dok na kraju ne uvidimo gde je greška.

Važna napomena: Kod pisanja programskih linija vodite računa da za tu liniju imate unos u dva reda. Ukoliko imate više redova dobićete poruku "Syntax error" (računar je uzeo samo prva dva reda i smatra da treći red počinje kao nova linija, a pošto na početku nije broj pokušaće da izvrši naredbu kao da je u direktnom modu što će rezultirati pomenutom porukom) i linija neće biti upisana u memoriju. U tom slučaju idite na broj linije koje ste unosili, pritisnite return/enter na nju i biče prihvaćena prva dva reda u toj programskoj liniji. Zato ako imate potrebe za unosom većeg broja karaktera, npr.kod naredbe PRINT, imajte u vidu da to razbijete u više programskih linija.

Mali savet: Prilikom pisanja u C64 postoje dva seta karaktera prvi set velikih slova i simbola i drugi set malih i velikih slova. Preko tastature se iz jednog u drugi set prelazi istovremenim pritiskom na commodore taster (ctrl u emulatoru) i shift taster. Preporučujem rad u prvom setu sem u slučaju da vam je bitno korišćenje velikih i malih slova.

Listanje kataloga na disketi
Da bi ste videli sadržaj diskete ako radite u emulatoru dovoljno je da pretisnete F10, zatim izberete 1541 device 8, odaberete .d64 fajl koji želite i vidite šta on sadrži. Takođe možete da odaberete program na virtualnoj disketi koji ima ekstenziju .prg i računar će isti učitati i odmah startovati.
Kod C64 (narano i kod emulatora) ako želimo da vidimo sadržaj diskete kucamo LOAD "$",8. Ova naredba će učitati sadržaj kataloga u memoriju računara koju možete izlistati naredbom LIST. Imajte u vidu da će ovaj postupak izbrisati bilo koji prethodni bejzik program koji ste imali u memoriji računara pa budite oprezni.

Krenimo sa današnjom lekcijom

: - Unos više naredbi u jednoj programskoj liniji
Dve tačke omogućavaju da se unesu više naredbi u jednoj programskoj liniji.
Primer:
10 PRINT "C64 JE ODLICAN ": PRINT "RACUNAR"
Posle startovanja ispisaće se C64 JE ODLICAN a u sledećem redu RACUNAR

π - Matematička konstanta (Pi)
Približna vrednost ove konstante je 3,14159 i predstavlja odnos obima i prečnika kruga ili odnos površina kruga i kvadrata nad njegovim poluprečnikom. Koristi se za obračun obima i površine kruga i td. U emulatoru da bi dobili ovaj simbol treba pretisnuti shift+delete

↑ - Stepenovanje
Ako želimo neki broj da pomnožimo samim sobom željeni broj puta to nazivamo stepenovanjem. Npr. želimo da broj 2 stepenujemo na 3 što je isto kao i 2*2*2 i daje rezultat 8. U emulatoru oznaka za stepen se dobija pritiskom na delete

END - Kraj programa
Ako želimo da zaustavimo izvršavanje programa u nekoj programskoj liniji unosimo u njoj naredbu END.

REM - Unos komentara
Ova naredba pruža mogućnost korisniku da unese komentare za svaku programsku liniju što veoma olakšava onima koji vrše pregled programa. Veliki nedostatak ove naredbe je što ista, zavisno od veličine komentara, iskorišćava dosta računarske memorije, kao i to što uvećava količinu podataka koji se upisuje na kaseti ili disketi. Zbog toga je kod velikih programa koji zahtevaju dosta memorije treba izbegavati.
Format naredbe je:
REM komentar
Primer:
10 PRINT "C64" : REM OVO CE ISPISATI NA EKRANU TEKST C64

EKRAN, ISPIS I KURZOR - Komande za brisanje ekrana. pomeranje ispisa i kurzora
Kod C64 računara konstruktori su izmislili jedinstveni način za manipulaciju ekranom i kurzorom putem programa. Naime kada želimo da kurzor pošaljemo na početak ekrana u direktnom modu pritisnuli bi taster CLR/HOME (u emulatoru to je taster HOME). Ukoliko bi hteli da ekran obrišemo stisnuli bi kombinaciju SHIFT+CLR/HOME. Pomeranje kurzora, a samim ti i mesta ispisa, bi vršili preko strelica.
Kako to preneti u program? Konstruktori računara su to rešili tako što bi se, kada programer otvori znake navoda, umesto da se izvršavaju pomenute akcije ispisivali simboli u negativu, a prilikom zatvaranja navodnica ponovo bi se vratilo na normalno izvršavanje akcija. To sve ovako izgleda:
CTRL/HOME - premeštanje kursora na početak ekrana - "inverzno slovo S"
SHIFT+CTRL/HOME - brisanje ekrana - "inverzni simbol srca"
KURZOR LEVO - "inverzna crtica na sredini"
KURZOR DESNO - "inverzna desna srednja zgrada"
KURZOR GORE - "inverzni krug"
KURZOR DOLE - "inverzno slovo Q"
Imajte u vidu da bi se ovi simboli na C64/emulatoru videli kao jedan znak. Da bi se ove akcije startovale koristi se naredba PRINT. Iste je moguće startovati kako u programskom tako i u direktnom načinu rada. U daljem tekstu kada budemo koristili ove akcije pisaćemo u zagradi tastere koje stiskamo, a koja će za rezultat ispisivati odgovarajući simbol, npr.za brisanje ekrana:
PRINT "(SHIFT+CTRL/HOME)"

LET - Dodeljivanje vrednosti promenljivima
Formati ove naredbe su:
LET A=10 - dodeljuje se numeričkoj promenljivi A vrednost 10
LET N$="C64" dodeljuje se tekstualnoj promenljivi N$ vrednost C64
Moramo priznati da ova naredba i nema neku svrhu kod C64. Razlog je to što računar podrazumeva naredbu LET tako da je dovoljno napisati A=10 i N$="C64" i tako ćemo nadalje i raditi.

CLR - Brisanje sadržaja svih promenljivih
Ova naredba svim numeričkim promenljivama daje vrednost 0 i briše sadržaje svih tekstualnih promenljivi.

INPUT – Unos podataka u numeričkim i tekstualnim promenljivama iz programa
Videli smo kako dodeljujemo vrednosti promenljivima u direktnom modu i programu ali naravno najbitnije je da omogućimo korisniku da unese vrednosti u istima kada program radi da bi ih mogao obraditi i dati željeni rezultat. Tu dolazi do izražaja naredba INPUT. Ova naredba omogućava unos podataka sa tastature. Sama naredba ima mogućnost ispisa teksta na ekranu poput naredbe Print.
Format naredbe je:
INPUT A - unesi numeričku vrednost
INPUT A$ - unesi tekstualnu vrednost
ili
INPUT "UNESI BROJ";A
INPUT "UNESI TEKST";A$
Ukoliko pogrešno unesemo tekstualni podatak u numeričku promenljivu dobićemo poruku "Redo from start" i program će ponovo zatražiti unos. Ako unesemo numerički podatak u tekstualnu promenljivu neće se ništa desiti jer računar smatra da smo uneli tekst.

Dajemo male primere:
```
10 INPUT "KAKO SE ZOVES?";IME$
20 PRINT "ZDRAVO, ";IME$
```
Računar će zatražiti ime korisnika a zatim ispisati na ekranu Zdravo, ime korisnika.

```
10 INPUT "UNESI POLUPRECNIK KRUGA";R
20 PRINT "OBIM KRUGA JE ";2*R*π
30 PRINT "POVRSINA KRUGA JE ";R↑2*π
```
Računar će zatražiti vrednost poluprečnika kruga, a onda će na osnovu njega izračunati obim i površinu kruga.

FOR-TO-STEP-NEXT - Programska petlja
Ove naredbe idu uvek zajedno (osim naredbe STEP koja je opcionalna) i izvršavaju niz komandi koje se nalaze između njih zadati broj puta za određeni korak (opcionalno). Petlje se u programiranju često koriste i imaju mnoge različite primene.
Format naredbi su:
FOR - za neku numeričku promenljivu od zadate početne vrednosti
TO - idi do njene krajnje zadate vrednosti
STEP - za neki zadati korak (bilo pozitivni ili negativni, ceo ili decimalni broj, ukoliko se izostavi podrazumeva se +1)
NEXT - uvećava vrednost promenljive i vraća na prvu naredbu posle FOR-TO-STEP ukoliko vrednost promenljive nije veća od zadate krajnje vrednosti (u tom slučaju se prelazi na sledeću naredbu). Takođe imajte u vidu da se u NEXT naredbi ne mora uneti naziv promenljive jer računar istu podrazumeva, ali vam toplo preporučujem da to bar u početku radite jer iz mog ličnog iskustva mogu se stvoriti neugodne konfuzne situacije naročito kod kompleksnih programa sa mnogo programskih linija)

Sledeći prosti primer će to ilustrovati:
```
10 FOR I = 1 TO 9 STEP 2
20 PRINT I
30 NEXT I
```

U prvoj programskoj liniji 10 programskom brojaču se zadaje 1 za početnu vrednost promenljive I koja će se u svakom ciklusu uvećavati za 2 do krajnje vrednost 9. U programskoj liniji 20 prikazuje se vrednost ove promenljive. Linija 30 uvećava ovu vrednost za 2 i vraća tok programa na prvu naredbu posle naredbe FOR-TO-STEP u ovom slučaju PRINT u programskoj liniji 20 osim ako promenljiva I ima veću vrednost od krajnje zadate vrednosti u ovom slučaju 9 (kada zadnji prolaz bude 9 uvećanjem za 2 daće rezultat 11 što je veće od krajnje zadate vrednosti a samim tim se prelazi na sledeću naredbu a ukoliko nema drugih naredbi i programskih redova program se završava). Na ovaj način smo prikazali sve neparne brojeve od 1 do 9.

IF - THEN - Izvršavanje naredbi ako je uslov zadovoljen
Naredba IF i THEN su bitne naredbe koje izvršavaju programske naredbe u zavisnosti da li je uslov zadovoljen.
IF - Ako je neki uslov zadovoljen
THEN - Onda uradi ovo
Uslovi mogu biti = jednako, < manje od, > veće od, <= manje ili jednako, >= veće ili jednako, < > različito kao i logički operatori NOT, AND, OR

Najbolje ćemo ilustrirati ovu naredbu sledećim programom:

```
10 PRINT "(SHIFT+CTRL/HOME)"
20 CLR
30 PRINT "OVO JE IGRA POGADJANJA GRADOVA"
40 PRINT "POTREBNO JE POGODITI GRAD IZ 10 PUTA"
50 PRINT "MOLIM DA DRUGI IGRAC KOJI POGADJA NE GLEDA U EKRAN":PRINT
60 INPUT "PRVI IGRAC NEKA UPISE IME GRADA ";GRAD$
70 PRINT "(SHIFT+CTRL/HOME)"
80 FOR I=1 TO 10
90 INPUT "POKUSAJ DA POGODIS NAZIV GRADA";POG$
100 IF GRAD$=POG$ THEN PRINT "BRAVO POGODIO SI IZ " ;I; " PUTA":END
110 PRINT "NISI POGODIO": NEXT I
120 PRINT "ZAO MI JE NISI POGODIO IZ 10 PUTA"
```

Da objasnimo šta se ovde dešava. U programskoj liniji 10 brišemo ekran. Linija 20 briše sve vrednosti promenljiva koje ćemo koristiti u programu. Linija 30 - 50 ispisuje uputstvo za igrače. Linija 60 prvi igrač koji smišlja ime grada koji će se pogađati upisuje to ime i računar ga smešta u tekstualnu promenljivu GRAD$. Linija 70 ponovo briše ekran. Linija 80 pokreće petlju koja će se izvršavati 10 puta kao i sve naredbe između FOR i NEXT naredbi.
Linija 90 drugi igrač koji pogađa upisuje ime grada za koji misli da se traži i program ga unosi u tekstualnu promenljivu POG$.
Linija 100 vrši upoređivanje dve tekstualne promenljive GRAD$ koja sadrži ime grada koji se traži sa POG$ koja promenljiva sadrži ime grada za koje drugi igrač misli da se traži. Ukoliko je uslov tačan (=) i dve tekstualne promenljive su identične onda se ispisuje "Bravo pogodio si iz",dodaje se u ispisu numerička vrednost pokušaja sadržana u promenljivoj I, "puta" i program se završava. Ukoliko uslov nije zadovoljen ide se na liniju 110 ispisuje se da grad nije pogođen, naredba next uvećava promenljivu I za 1 i vraća se na liniju 90 naredba koja dolazi posle FOR-TO gde se ponovo traži da drugi igrač pogodi grad. Ukoliko se došlo do 10 puta i grad nije pogođen NEXT više ne vraća tok programa, odlazi se na na sledeću naredbu koja se nalazi u liniji 120 i igri je kraj.

Toliko za danas. Završićemo drugu lekciju sa porukom budite kreativni. Pokušajte da sa ovim naredbama koje smo danas naučili napravite svoj program. Nije bitno ako ne uspete da dobijete traženi rezultat iz prvog ili više puta važno je da neodustanete.