9. RAD S DATOTEKAMA PODATAKA

U programiranju se često radi sa skupovima podataka koji se čuvaju na jedinicama perfernih memorija, a koji se organizuju u posebne cjeline koje nazivamo datotekama. Svaka datoteka predstavlja fizičku cjelinu kojoj se može fizički prići radi čitanja ili upisa podataka.

9.1 Organizacija datoteka podataka u Liberty BASICu

Podaci unutar datoteke se organizuju u obliku slogova. Svaki slog sadrži određen broj podataka, smještenih u polja sloga.

Kada program komunicira s datotekom, zahtijeva da pročita jedan slog iz datoteke, ili da upiše jedan slog u datoteku. Prema tome kako se može pronaći onaj slog koji program zahtijeva, odnosno prema tome kako se prilazi slogu datoteke, razlikuje se i organizacija datoteka.

Different Methods of File Access in Liberty BASIC:

1. **Sequential Files**
   Sequential Files are accessed from beginning to end, sequentially. It is not possible to read or write a piece of data to the center of the file. Files opened for INPUT can only be read. Files opened for OUTPUT or APPEND can only be written.

2. **Binary Files**
   Files opened for binary access may be read or written, beginning at any location within the file.

3. **Random Files**
   Files opened for random access are read or written one record at a time. The length of records in the file is determined in the OPEN statement.

9.2 Programske datoteke podataka

Skup podataka koji je sastavni dio programa čini programsku datoteku podataka. Podaci programske datoteke definišu se pomoću naredbe:

```
DATA lista podataka (brojčanih i/ili znakovnih),
                    odvojenih zarezima
```

U jednom programu može biti proizvoljan broj naredbi podataka. Sve ovakve naredbe zajedno definišu jednu organizovanu datoteku, čiji podaci se redom učitavaju.

Primjer: Dvije naredbe **DATA** obrazuju programsku datoteku od 8 podataka u slijedećem redoslijedu: `5, 10, 18, 4.1, "Sarajevo", 3.8, "Datum", 15`, kako slijedi:

```
DATA 5, 10, 18, 4.1, "Sarajevo"
DATA 3.8, "Datum", 15
```

Podaci iz programske datoteke mogu se dodjeljivati promjenljivim u programu pomoću posebne naredbe za čitanje iz ovakve datoteke:

READ lista promjenljivih, (brojčanih i/ili znakovnih i/ili indeksnih), odvojenih zarezima

Podaci iz programske datoteke i promjenljive u listi naredbe READ, kojima se ovi podaci dodjeljuju, moraju se redom slagati po tipu. To znači da se brojčanoj promjenljivoj može dodijeliti brojčani podatak iz datoteke, a znakovnoj promjenljivoj znakovni podatak.

O tome koji će podatak biti dodijeljen nekoj promjenljivoj u listi naredbe READ, određuje pokazivač podataka programske datoteke, koji sadrži redni broj podatka koji je na redu za čitanje, i koji se poslije svakog čitanja podatka uvećava za jedan. Time ukazuje na slijedeći podatak koji je na redu za čitanje.

Primjer: Naredba za čitanje:

```
READ X, Y, T$
```

čita tri podatka iz programske datoteke, pri čemu su prva dva brojčani podaci, koji se dodjeljuju promjenljivim X i Y, dok je treći podatak znakovni i dodjeljuje se znakovnoj promjenljivoj T$.

Pokazivač programske datoteke može se programski vratiti na prvi podatak datoteke naredbom:

```
RESTORE
```

Programska datoteka podataka formira se kada se kreira i program. Za vrijeme izvršavanja programa, mogu se samo čitati podaci iz ovakve datoteke, a ne može se vršiti upi u istu.

Primjer P901

Program koji izračunava i štampa vrijednosti zadane funkcije y(x) za 10 fiksnih vrijednosti argumenta x:

```
'Program P901

PRINT " X", "Y(X)"
PRINT

FOR I = 1 TO 10
    READ X
    Y = X*X-2*X+3.5
    PRINT X, Y
NEXT I
PRINT
PRINT "Kraj tabele"

DATA 0, 1, 1.5, 2,5, 4,6
DATA 8, 10, 12.5, 15

END
```

9.3 Dataoteke podataka na perifernim memorijama

9.3.1 Sekvencijalne datoteke podataka

Sadržaj ovih datoteka je u ASCII kodu.

Otvaranje datoteke

Postupak koji se mora provesti da bi se datoteka učinila dostupnom za upis ili čitanje podataka u programu, zove se otvaranje datoteke. Otvaranje datoteke vrši se posebnom naredbom:

```
OPEN "adress of file" FOR purpose AS #handle {LEN = n}
```

adress of file

If the device to be opened is a file, the device parameter must be a valid disk filename. This may be expressed as a string variable, or as a literal text expression enclosed in quotes. For more on coding file specifications, see Path and Filename.

purpose

Files may be opened for the purpose of INPUT, OUTPUT, APPEND, RANDOM or BINARY access. The final {LEN=n} parameter applies to files opened for RANDOM access. For more on opening files, please see File Operations.

#handle

The #handle is a unique name given to the device so that it can be accessed by functions in the program. Use a descriptive word for the handle. It must start with a # and may contain any alpha-numeric characters, but no spaces. This special handle is used to identify the open device in later program statements. Some possible handles are as follows:

```
#commHandle
#newfile
#main
#win
#gdi32
#2
```

Changing the handle of a device dynamically at runtime can be accomplished with the MAPHANDLE command.

Primjer: Naredba za otvaranje datoteke za čitanje:

```
open "c:\readme.dat" for input as #2
```

otvara datoteku podataka pod imenom **`readme.txt`** na disku **`C:`**, za čitanje, i dodjeljuje joj se broj datoteke 2.

Za definisanje adrese datoteke može se uvesti opisna promjenljiva, kojoj se prije izvršenja naredbe otvaranja datoteke mora dodijeliti znakovni podatak koji predstavlja adresu datoteke:

```
x$ = "c:\readme.dat"
open x$ for input as #2
```

Primjer: Naredba za otvaranje datoteke za upis:

```
open "c:\write.dat" for output as #5
```

vrši otvaranje datoteke podataka s imenom **write.dat**, za upis, i dodjeljuje joj se broj datoteke 5.

Zatvaranje datoteke

Nakon završetka rada s datotekom, mora se sprovesti postupak njenog zatvaranja kojim će se ranije otvorena datoteka učiniti nedostupnom za upis ili čitanje podataka. Zatvaranje datoteke vrši se naredbom:

```
close #2
```

Upis u datoteku

Podaci se iz centralne memorije računara upisuju u datoteku naredbom:

```
print #2, lista entiteta za štampanje odvojenih znakom tačka-zarez
```

Lista može biti sastavljena od izraza čije se izračunate vrijednosti upisuju u datoteku i/ili od varijabli. Entiteti iz liste se odvajaju znakovima tačka-zarez, kojima se određuje da vrijednosti koje se upisuju u datoteku slijede jedna za drugom, bez razmaka.

Naredba izlaza:

```
print #2, ""
```

izdaje prazan red u datoteci.

Naredba izlaza:

```
print #2, A; B; C
```

upisuje vrijednosti promjenljivih A, B i C u jedan slog datoteke, bez razmaka među vrijednostima.

Naredba izlaza:

```
print #2, "X = "; X; "□□Y = "; Y
```

upisuje vrijednosti promjenljivih X i Y, u jedan slog datoteke, uz odgovarajući popratni tekst.

Čitanje iz datoteke

Podaci koji se čitaju iz datoteke (koja se nalazi na nekoj od jedinica periferne memorije) prenose se u centralnu memoriju računara. Čitanje iz datoteke ostvaruje se naredbom:

```
input #2, lista promjenljivih, odvojenih zarezima,
                               čije vrijednosti se učitavaju
```

Lista sadrži spisak promjenljivih, među sobom odvojenih zarezima, kojima se dodjeljuju učitane vrijednosti.

U jednom ulaznom slogu datoteke može se nalaziti jedan ili više brojčanih podataka, među sobom razdvojenih zarezom. Ako se radi sa znakovnim podacima, tada se u jednom ulaznom slogu mora nalaziti jedan znakovni podatak.

Ako je vršeno čitanje sekvencijalne datoteke, može se naredbom:

```
restor #2
```

omogućiti ponovno čitanje datoteke, počevši od prvog sloga datoteke.

Naredba ulaza:

```
input #2, A, X, W
```

vrši čitanje tri brojčana podatka iz datoteke s brojem 2, i učitane vrijednosti dodjeljuje promjenljivim: A, X i W.

Primjer P901

Program za upisivanje podataka u datoteku:

```
'Program P901
OPEN "C:\Help\Radni\Izlaz.dat" FOR OUTPUT AS #1

FOR N = 1 TO 5
    PRINT #1, N; "□"; N*N; "□"; SQR(N)
NEXT N

CLOSE #1

END
```

Rezultat će biti štampan u datoteku Izlaz.dat u slijedećem obliku:

```
1□1□1.0
2□4□1.41421356
3□9□1.73205081
4□16□2.0
5□25□2.23606798
```

Primjer P902

Program za učitavanje podataka iz jedne datoteke i za upisivanje podataka u drugu datoteku:

```
'Program P902

OPEN "C:\Help\Radni\Ulaz1.dat" FOR INPUT AS #2
```

```
OPEN "C:\Help\Radni\Izlaz1.dat" FOR OUTPUT AS #3

FOR N = 1 TO 5
    INPUT #2, X, Y
    PRINT X; "□"; Y    'Provjera podataka ucitanih iz datoteke
    Z = X * Y
    PRINT #3, X; "□"; Y; "□"; Z
NEXT N

CLOSE #2
CLOSE #3

END
```

Datoteka Ulaz1.dat se može kreirati pomoću Editora, ili pomoću programa Word, kada datoteku treba snimiti kao tekstualni fajl. U primjeru P902 datoteka Ulaz1.dat ima oblik:

```
1, 1
2, 4
3, 9
4, 16
5, 25
```

pri čemu su vrijednosti varijabli X i Y u svakom ulaznom slogu odvojene zarezom.

Datoteka Ulaz1.dat se može formirati i posebnim LB programom, kako je to pokazano u slijedećem primjeru:

Primjer P903

```
'Program P903
OPEN "C:\Help\Radni\Ulaz1.dat" FOR OUTPUT AS #1

FOR N = 1 TO 5
    PRINT #1, N; ","; N*N
NEXT N

CLOSE #1

END
```

Sekvencijalna datoteka se može čitati i bez petlje, tako da se poslije čitanja svakog sloga provjerava da li se došlo do kraja datoteke (primjer P904). Ovaj način je važan u slučaju kada se čita sekvencijana datoteka kojoj se ne zna dužina, odnosno ne zna se koliko slogova sadrži ovakva datoteka. Zapis o kraju datoteke korisnik ne može vidjeti, a isti je zapisan uz zadnji slog datoteke.

Primjer P904

```
'Program P904

OPEN "C:\Help\Radni\Ulaz1.dat" FOR INPUT AS #5

10   INPUT #5, X, Y
     Z = X * Y
     PRINT X; "□"; Y; "□"; Z
     IF EOF (#5) THEN 100
     GOTO 10

100 PRINT "Kraj datoteke! "
     CLOSE #5
END
```

U primjeru P905, matrice A(3,3) i B(3,3) se sabiru, a prije sabiranja se učitavaju iz datoteke. Ulazna datoteka sadrži elemente matrica A i B poredane red po red i odvojene zarezima, kako slijedi:

```
1,1,1,1,1,1
1,1,1,1,1,1
1,1,1,1,1,1
2,2,2,2,2,2
2,2,2,2,2,2
2,2,2,2,2,2
```

Svaki slog ulazne datoteke sadrži po jedan red prvo matrice A, a zatim matrice B. Učitavanje matrica se realizuje pomoću jedne petlje, pa se u naredbi Input mora navesti svih šest elementa jednog reda matrice. Pri tome se putem indeksa I u indeksnoj promjenljivoj (A(I,J), odnosno B(I,J)) određuje koji red matrice se učitava.

Primjer P905

```
'Program P905

OPEN "C:\Help\Radni\Ulazmat.dat" FOR INPUT AS #2
OPEN "C:\Help\Radni\Izlazmat.dat" FOR OUTPUT AS #3

'Unos matrice A iz datoteke Ulazmat
For I = 1 To 3
     Input #2, A(I,1), A(I,2), A(I,3), A(I,4), A(I,5), A(I,6)
Next I

'Unos matrice B iz datoteke Ulazmat
For I = 1 To 3
     Input #2, B(I,1), B(I,2), B(I,3), B(I,4), B(I,5), B(I,6)
Next I
```

```
For I=1 To 3
    For J=1 To 6
        C(I,J) = A(I,J) + B(I,J)
    Next J
Next I

Print #3, "ZBIR MATRICA A i B JE:"
I = 0
20 I = I + 1
   Print #3, " "
   For J = 1 To 6
       Print #3, C(I,J);" ";
   Next J
   If (I < 3) Then 20

Close #2
Close #3

End
```

Napomena: treba imati na umu da otvaranje datoteke s deklaracijom **Output** briše na disku datoteku s istim imenom, ukoliko je takva ranije kreirana.

U primjeru P905a matrice A i B, koje se sabiru, učitavaju se također iz datoteke, ali kako se kod učitavanja koristi dupla petlja, onda se elementi matrica moraju u ulaznoj datoteci poredati jedan iza drugog. U ovom slučaju, svaki slog ulazne datoteke sadrži samo po jedan element matrice.

Primjer P905a

```
'Program P905a

DIM A(3,6), B(3,6), C(3,6)

OPEN "C:\Help\Radni\Ulazmat1.dat" FOR INPUT AS #2
OPEN "C:\Help\Radni\Izlazmat.dat" FOR OUTPUT AS #3

'Unos matrice A iz datoteke Ulazmat
For I = 1 To 3
    For J = 1 To 6
        Input #2, A(I,J)
    Next J
Next I

'Unos matrice B iz datoteke Ulazmat
For I = 1 To 3
    For J = 1 To 6
        Input #2, B(I,J)
    Next J
Next I
```

```
For I=1 To 3
    For J=1 To 6
        C(I,J) = A(I,J) + B(I,J)
    Next J
Next I

Print #3, "ZBIR MATRICA A i B JE:"
I = 0
20 I = I + 1
   Print #3, " "
   For J = 1 To 6
       Print #3, C(I,J);" ";
   Next J
   If (I < 3) Then 20

Close #2
Close #3

End
```

9.3.2 Datoteke sa direktnim pristupom (Random files)

OPEN filename FOR RANDOM AS #handle LEN=n

To access a file in random access mode, it must be opened with the OPEN statement. When it is no longer needed, and before the program ends, it must be closed with the CLOSE statement. See also: Filedialog, File Operations, Path and Filename , PUT, FIELD, GET, GETTRIM

Random Access files consist of RECORDS. The entire file is divided into many records. Each record has the same length. The length is specified when the file is opened with the LEN parameter. The example below opens a file called "members.dat" and sets the record length to 256:

OPEN "members.dat" FOR RANDOM AS #1 LEN=256

Reading and writing to a file opened for random access is done one record at a time. A record may be read or written anywhere in the file. A record is read with either the GET statement, or with the GETTRIM statement. A record is written to the file with the PUT statement. These statements are explained in more detail below.

FIELD Statement
Each record is subdivided into "fields", each with a given length. The FIELD statement must be included after the OPEN statement to set up the size of each of the fields. Each field is designated by a variable name and given a specified length. When the lengths of all fields are added together, their sum must be equal to the length that was set with the LEN parameter in

the OPEN statement. In the above case, the field lengths must total 256. A "$" character indicates that a field holds data that will be accessed as a string. If there is no "$" character, the field will be accessed as a number. The fields for "members.dat" might look like this:

```
OPEN "members.dat" FOR RANDOM AS #1 LEN=256

   FIELD #1,_    ' set up the fields for file opened as #1
   90 AS Name$,_ ' 1st 90 bytes contains Name$, string
   110 AS Address$,_   ' 2nd 110 bytes contains Address$, string
   50 AS Rank$,_       ' 3rd 50 bytes contains Rank$, string
   6 AS IDnumber       ' 4th 6 bytes contains IDnumber, numeric
```

The value after "LEN=" is 256, which is obtained by adding 90 + 110 + 50 + 6 or the total length of all the fields in FIELD#. The FIELD statement must follow the OPEN statement and must be used before trying to read or write data to the file with GET or PUT.

PUT
PUT #handle, number

This statement is used to write the data that is contained in the variables listed in the FIELD statement to the specified record number. Records in RANDOM files are numbered beginning at 1, not 0. If the length of a variable in a given field is shorter than the field length specified, blank spaces are added to pad it. If the length of the variable is larger, it will be truncated to the length specified in the FIELD statement. To add this data as RECORD number 3 to the file referenced above:

```
Name$ = "John Q. Public"
Address$ = "456 Maple Street, Anytown, USA"
Rank$ = "Expert Programmer"
IDnumber = 99

PUT #1, 3
```

GET
GET #handle, number

This statement reads an entire record and fills the variables listed in the FIELD statement.

```
GET #1,3
```

Print Name$ would produce "John Q. Public "
Print Address$ would produce "456 Maple Street, Anytown, USA "
Print Rank$ would produce "Expert Programmer "

print IDnumber would produce "99"

and so on.

GETTRIM
GETTRIM #handle, number

This command retrieves the record in the same manner as GET, but it removes any blank leading or trailing spaces in the record:

GETTRIM #1,3

Print Name$ would produce "John Q. Public"
' no blank spaces are included

Primjer P906

```
'Program P906

OPEN "c:\Help\Radni\members.dat" FOR RANDOM AS #1 LEN=256

    FIELD #1,_     ' set up the fields for file opened as #1
    90 AS Name$,_ ' 1st 90 bytes contains Name$, string
    110 AS Address$,_    ' 2nd 110 bytes contains Address$, string
    50 AS Rank$,_        ' 3rd 50 bytes contains Rank$, string
    6 AS IDnumber        ' 4th 6 bytes contains IDnumber, numeric

Name$ = "John Q. Public"
Address$ = "456 Maple Street, Anytown, USA"
Rank$ = "Expert Programmer"
IDnumber = 99

PUT #1, 3

GET #1,3

Print Name$
Print Address$
Print Rank$
print IDnumber

GETTRIM #1,3

Print Name$
' no blank spaces are included

Close #1
End
```