USER GUIDE ON UNIDATA AND UNIBASIC

SAMPLE PROGRAMS ON HOW TO START WITH UNIBASIC AND UNIDATA TECHNOLOGY IN AIX SYSTEM.

Author: Partha P Mondal (BCMD-WAC).

September 15, 2006

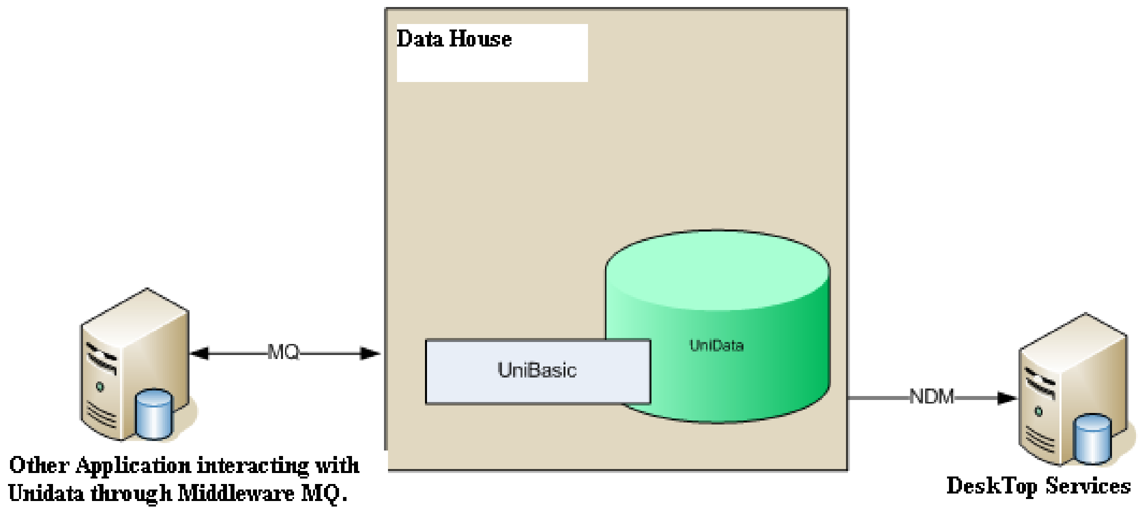

Data House
UniBasic
UniData
MQ
NDM
Other Application interacting with Unidata through Middleware MQ.
DeskTop Services

Table of Contents:

1.0 Introduction to IBM'S UnIData and Unibasic:

1.1 Introduction:

IBM UnIData is an extended relational data server designed for embedding in vertical applications. UnIData simplifies data management and query logic, provIDing more power for online high-transaction applications

It can be accessed by tools such as Visual Interdev, VB, VC++, Delphi or any ODBC, ActiveX or OLEDB tool and any Java-based integrated development environment (IDE).
It includes UniQuery — an easy to use, English-like query language — which provIDes the ability to create meaningful ad hoc data server queries without having to understand underlying data server structures

Includes UniBasic — a unique application development environment — which enables developers to create powerful, transaction-intensive applications.

IBM UnIData is a best-of-breed, extended relational data server for embedding in vertical solutions — IDeal for markets which target small and medium-businesses.

- Highly scalable, low-cost data management system.
- Support for Web services protocols including XML and SOAP.
- Includes UniObjects for .NET — a native, high performance API for developing in the Microsoft .NET framework.
- Robust features for high availability, failover and data replication available.
- Includes innovative features for achieving optimum response times, ease of management and low total-cost-of-ownership (TCO).
- Standards-based SQL interfaces such as JDBC, OLEDB and ODBC that support the full nesting capability of the data server engine.

1.2 Benefits:

- Extended relational architecture requires fewer tables whose size is dynamically managed by the data server. UnIData sites require list data server administration with smaller sites having zero or part-time resources allocated.

- Shortened development cycles due to ease of data server design along with tightly integrated development environment.

- UnIData supports a broad range of information management applications with excellent performance and reliability. As a result, a large number of industry solution provIDers use UnIData to support their applications. This allows the customer to more quickly satisfy their requirements by buying, rather than building, solutions.

2.0 Special feature of UnIData and Unibasic:

2.1 Connection Pooling (CP):

CP enables developers to design and deploy applications that do not use persistent connections to the data server, such as in a Web environment. Connection Pools are authorized separately and do not consume data server licenses.

2.2 External Data server Access (EDA):

EDA provIDes the ability to store one or more UnIData files on an external data server such as IBM DB2 Universal Data server, in a manner that is transparent to the application.

2.3 Network File Architecture (NFA):

NFA offers local access to data on remote UnIData servers.

2.4 Recoverable File System (RFS):

RFS provIDes recoverability in case of system or media failures.

2.5 Transaction Processing (TP):

TP semantics provIDe atomicity, consistency, isolation and durability (ACID) properties for the data server.

2.6 Data Handling (DH):

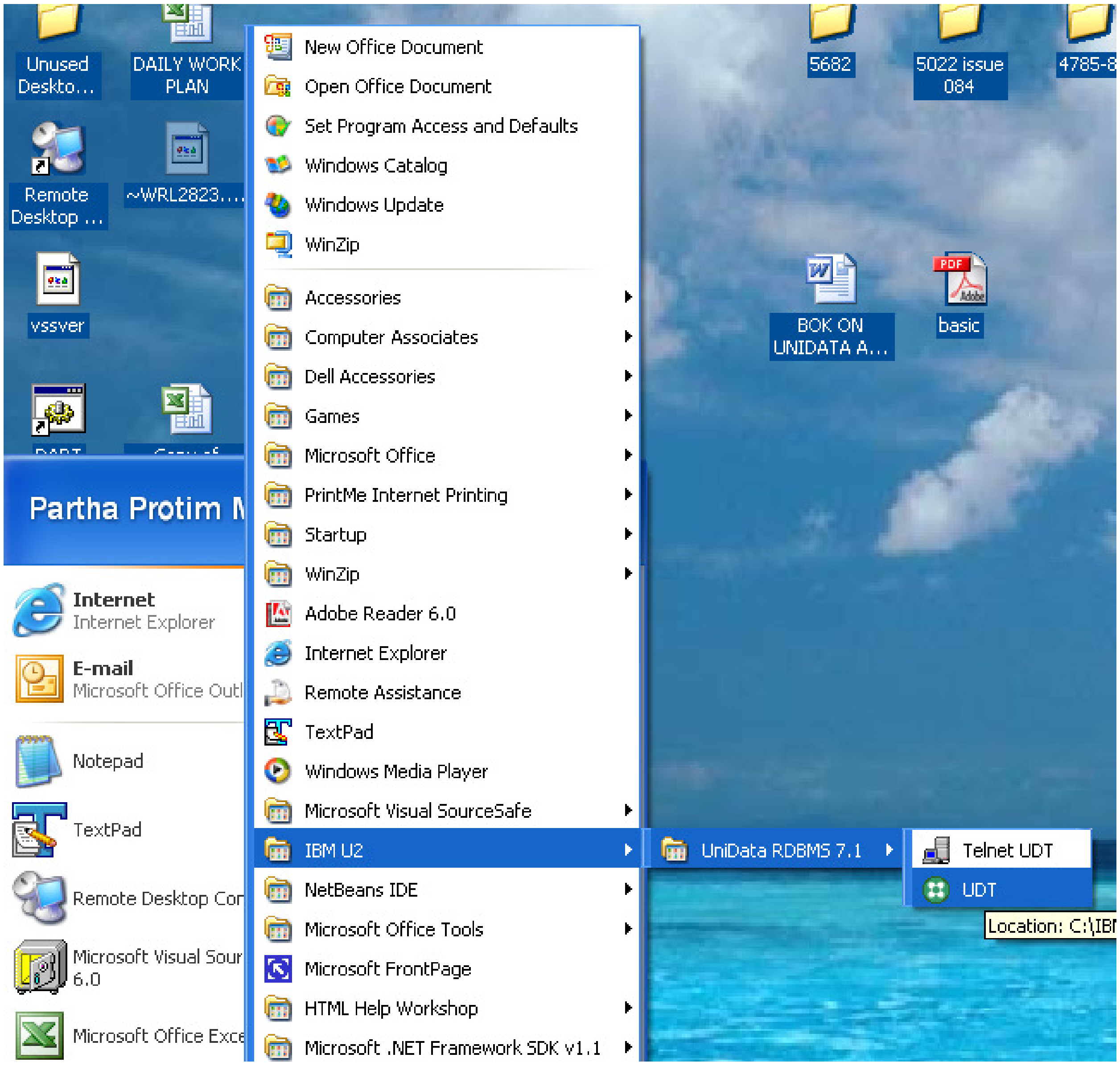
DH automatic data conversion helps programmer to manipulate data in an easiest way.

2.7 Compatibility with AIX:

Compatibility with system helps user to interact with heterogeneous systems.

3.0 Starting with UnIData and Unibasic application:

You want to do a Unidata-UniBasic program? Let’s start coding on Unidata-UniBasic.
Install UDT shell from the site
https://www14.software.ibm.com/webapp/iwm/web/reg/download .
This is a trial version. Extract the zip file and click on the .exe file. You should able to install the UDT shell.
All the path should be default path during installation. Go to the menu shown below-



Now you are in UDT shell. We will be doing programs in this shell. Do the below program-

```
:CREATE.FILE DIR TRNG 101
Create DIR type file TRNG.
Create file D_TRNG, modulo/1,blocksize/1024
Hash type = 0
Added "@ID", the default record for UniData to DICT TRNG.
:
:AE TRNG FIRST.PROG
Top of New "FIRST.PROG" in "TRNG".
*--: I
001= PRINT
002= PRINT @(15,1):"HELLOW FRIEND, SEE MY PROGRAMS !"
003= PRINT @(10,2):"=========================="
004= CHOICES = ''
005= PRINT @(16,5):"ENTER YOUR NAME: "
006= INPUT CHOICES
007= PRINT @(18,8):"YOU SUCESSFULLY FINISHED YOUR FIRST UNIDATA
PROGRAM"
008= PRINT @(19,9):"THANK YOU ":CHOICES
009= PRINT "BYE"
*--: FI
Filed "FIRST.PROG" in file "TRNG".
:BASIC TRNG FIRST.PROG
LsaQueryInfoTrustedDomain() failed, error(8478):
        The requested operation requires a directory service, and none was available.

Compiling Unibasic: TRNG\FIRST.PROG in mode 'u'.
compilation finished

:RUN TRNG FIRST.PROG
             HELLOW FRIEND, SEE MY PROGRAMS !
        ==========================



              ENTER YOUR NAME:
?PARTHA

                YOU SUCESSFULLY FINISHED YOUR FIRST UNIDATA PROGRAM
                 THANK YOU PARTHA
BYE
:
```

So, you are happy now? You can go through the documentation below. I am sure that, you will be a good Unidata-Unibasic programmer, if you glance through the material.

3.1 How to compile and run a Unidata-UniBasic program:

To create and use a BASIC program, follow these steps:

1. Use the CREATE.FILE command to create a type 1 or type 19 file to store your BASIC program source.
2. Use the UniDebugger editor or some other editor to create the source for your BASIC program as a record in the file you created in step 1.
3. Once you have created the record containing your BASIC program source statements, use the BASIC command to compile your program. The BASIC command creates a file to contain the object code output by the compiler. You do not have to know the name of the object file because the program is always referred to by the source filename.
4. If the BASIC compiler detects any errors, use the editor to correct the source code and recompile using the BASIC command.
5. When your program compiles without any errors, execute it using the RUN command. Use the RAID command to debug your program.

3.2 How to run program in Background:

You can use the PHANTOM command to compile BASIC programs in the background. The output from PHANTOM processes is stored in the file named &PH&. For example, the command:
>PHANTOM BASIC FIRST.PROG compiles the programs in background and the out put of the file will be saved in &PH& file (system file).

4.0 <u>Data types and Variables:</u>

This module gives an overview of the fundamental components of the UniBASIC language. It describes types of data, constants, variables, and how data is combined with arithmetic, string, relational, and logical operators to form expressions.

4.1 Types of Data

All data is stored internally as character strings and data typing are done contextually at run time. There are three main types of data: character string, numeric, and unknown (that is, the null Value).

4.1.1 Character String Data

Character string data is represented internally as a sequence of ASCII characters. Character strings can represent either numeric or nonnumeric data. Their LENGT is limited only by the amount of available memory. Numeric and nonnumeric data can be mixed in the same character string (for example, in an address).

4.1.2 Character String Constants

In BASIC source code, character string constants are a sequence of ASCII characters enclosed in single or double quotation marks, or backslashes (\ ).These marks are not part of the character string value. The LENGT of character string constants is limited to the LENGT of a statement. Some examples of character string constants are the following:
'$42,368.99'
'Number of Employees'

4.1.3 Numeric Data

All numeric data is represented internally either as floating-point numbers with the full range of values supported by the system’s floating-point implementation, or as integers. On most systems the range is from 10-307 through 10+307 with 15 decimal digits of precision.

4.1.4 Numeric Constants

Numeric constants can be represented in either fixed-point or floating-point form. Commas and spaces are not allowed in numeric constants.

4.1.5 Fixed-Point Constants

Fixed-point form consists of a sequence of digits, optionally containing a decimal point and optionally preceded by a plus (+) or minus (-) sign. Some examples of valID fixed-point constants are:
12
-132.4
+10428

4.1.6 Floating-Point Constants

Floating-point form, which is similar to scientific notation, consists of a sequence of digits, optionally preceded by a plus (+) or minus (-) sign representing the mantissa. The sequence of digits is followed by the letter E and digits, optionally preceded by a minus sign, representing the power of 10 exponents. The exponent must be in the range of –307 through +307. Some examples of valid floating-point constants are:
1.2E3
-7.3E42
-1732E-4

4.1.7 Unknown Data: The Null Value

The null value has a special run-time data type in UniBASIC. @null is a system variable. I will discuss it later.
A = @NULL
B = ""
C = "JONES"

4.1.8 Constants

Constants are data that do not change in value, data type, or LENGT during program execution. Constants can be character strings or numeric strings (in either integer or floating-point form). A character string of no characters— the empty string—can also be a constant.

4.1.9 Array Variables

An array is a variable that represents more than one data value. There are two types of array: dimensioned and dynamic. Dimensioned arrays can be either standard or fixed.

Dimensioned Arrays

Each value in a dimensioned array is called an *element* of the array. Dimensioned arrays can be one or two-dimensional.
A one-dimensional array is called a *vector*. Its elements are arranged sequentially in memory. An element of a vector is specified by the variable name followed by the index of the element enclosed in parentheses. The index of the first element is 1. The index can be a constant or an expression. Two examples of valid vector element specifies are:
A (1)
COST (35)
A two-dimensional array is called a *matrix*. The elements of the first row are arranged sequentially in memory, followed by the elements of the second row, and so on. An element of a matrix is specified by the variable name followed by two indices enclosed in parentheses. The indices represent the row and column position of the element. The indices of the first element are (1, 1). Indices can be constants or expressions. The indices used to specify the elements of a matrix that has four columns and three rows are illustrated by the following:

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

Dynamic Arrays

Dynamic arrays map the structure of UniBasic file records to character string data. Any character string can be a *dynamic array*. A dynamic array is a character string containing elements that are substrings separated by delimiters. At the highest level these elements are fields separated by field marks (F) (ASCII 254). Each field can contain values separated by value marks (V) (ASCII 253). Each value can contain sub-values separated by sub-value marks (S) (ASCII 252).

Example:
A="Partha":@VM:"Mondal":@FM:1.23:@SM:20:@VM:2.50:@SM:10
or you can say:

```
A = ""
A<1,1> = "Partha"
A<1,2> = "Mondal"
A<2,1,1> = 1.23
A<2,1,2> = 20
A<2,2,1> = 2.50
A<2,2,2> = 10
```

The example has two fields. The first field has two values, and the second field has two values. The first value of the second field has two sub-values, and the second value of the second field also has two sub-values. You must use the following statements to declare that the first field contains the two values Partha and Mondal:

```
A = ""
A<1,1> = "Partha"
A<1,2> = "Mondal"
```

The statement:

```
A = ""
A<1> = "Partha"
```

Declares that the first field contains only Partha with no other values or sub values.

4.1.10 File Variables

A file variable is created by a form of the OPEN statement. Once opened, a file variable is used in I/O statements to access the file. There are two types of file variable: hashed file variable and sequential file variable. File variables can be scalars or elements of a dimensioned array.

4.1.11 Select List Variables

Select list variables are created by a form of the SELECT statements. A select list variable contains a select list and can be used only in the READNEXT statement. Unlike other variables, a select list variable cannot be an element of a dimensioned array.

5.0 Loops and Operators:

I will discuss the loop structures in Unidata-UniBasic in this section. See the loops and their code snippet-
You can use the LOOP statement to start a LOOP…REPEAT program loop. A program loop is a series of statements that executes for a specified number of repetitions or until specified conditions are met.
You can use the CONTINUE statement within LOOP…REPEAT to transfer control to the next iteration of the loop from any point in the loop. Use the EXIT statement within LOOP...REPEAT to terminate the loop from any point within loop.

You can also use multiple WHILE and UNTIL statement in a LOOP…REPEAT loop. You can also nest LOOP…REPEAT loops. If a REPEAT statement is encountered without a previous LOOP statement, an error occurs during compilation.

5.1 FOR LOOP:

Use the FOR statement to create a FOR…NEXT program loop. A program loop is a series of statements that execute repeatedly until the specified number of repetitions has been performed or until specified conditions are met.

```
MY.COUNTER=’’
FOR MY.COUNTER = 0 TO 5
PRINT MY.COUNTER
NEXT MY.COUNTER
```

You can nest the FOR loop in another for loop.

5.2 WHILE-DO LOOP:

You can use this loop for entry control loop. See the code snippet.

```
MY.COUNTER=’’
LOOP
WHILE MY.COUNTER <10 DO
PRINT MY.COUNTER
MY.COUNTER += 1
REPEAT
```

```
SELECT MY.FILE
LOOP
WHILE READNEXT KEY DO
PRINT KEY
REPEAT
```

Here also nesting is possible.

5.3 UNTIL LOOP:

See the snippet for UNTIL LOOP-

```
MY.COUNTER=’’
LOOP
UNTIL MY.COUNTER >10 DO
PRINT MY.COUNTER
MY.COUNTER += 1
REPEAT
```

5.4 Loop control statement:

There are some loop control statements by which you can exit from the loop.

1. CONTINUE
2. EXIT
3. ABORT

You can use these key word anywhere in your program.

5.5 Condition Block:

5.5.1 IF Block:

We have IF block similar to other language. You can also use nested IF block. We can use IF-ELSE block. Follow the below code snippet to understand IF-ELSE block-

```
VAL = 5
IF VAL <10 THEN
PRINT VAL:" LESS THAN 10"
END

REC=''
IF REC = '' THEN
PRINT "REC NOT BLANK"
END ELSE IF REC = 'MY RECORD' THEN
PRINT "MY DATA"
END ELSE
PRINT "RECORD IS BLANK"
END



SEQ = 5
REC = 5
IF SEQ = 5 THEN
IF REC = 5 THEN
PRINT SEQ
PRINT REC
END ELSE
PRINT :REC :" IS NOT EQUAL TO ": SEC
END
END
```

5.5.2 CASE Block:

Use the CASE statement to alter the sequence of instruction execution based on the value of one or more expressions. If *expression* in the first CASE statement is true, the following statements up to the next CASE statement are executed. Execution continues with the statement following the END CASE statement.
If the expression in a CASE statement is false, execution continues by testing the expression in the next CASE statement. If it is true, the statements following the CASE statement up to the next CASE or END CASE statement are executed. Execution continues with the statement following the END CASE statement.
If more than one CASE statement contains a true expression, only the statements following the first such CASE statement are executed. If no CASE statements are true, none of the statements between the BEGIN CASE and END CASE statements are executed.

```
NUM =5
BEGIN CASE
CASE NUM=1
PRINT ”NUM =1”
CASE NUMBER<3
PRINT “NUMBER LESS THAN 3”
CASE NUMBER>3
PRINT “NUMBER GREATER THAN 3”
END CASE
RETURN
```

5.6 Operators:

Operator	**Operation**
()	Function
+	Add
-	Subtraction
*	Multiplication
/	Division
^ **	Power
:	Concatenation
EQ =	Equality
NE # <> ><	Inequality
GT >	Greater than
LT <	Less than

GE >= =>	Greater than equal to.
LE <= =<	Less than Equal to.
AND	Logical AND
OR	Logical OR

6.0 Working with Tables in UniData:

6.1 Structure of table:

I will discuss about table in Unidata in this section. Basically Unidata handles all the table related things, like creation of table and the data quarries. UniBasic is a language which interacts with the tables in Unidata.

Lets see how can I create a table in Unidata and what is the structure of the table-

```
:CREATE.FILE DICT MY.TABLE
Create file D_MY.TABLE, modulo/1,blocksize/1024
Hash type = 0
Added "@ID", the default record for UniData to DICT MY.TABLE.
:
```

My table is created. Now I want to see the structure of my table.

```
LIST DICT MY.TABLE BY TYP BY @ID TYP LOC CONV NAME FORMAT SM ASSOC
10:45:33 Sep 19 2006 1
@ID............ TYP LOC.......... CONV NAME........... FORMAT SM ASSOC.....

@ID             D              0      MY.TABLE        10L    S
1 record listed
```

You saw that one record listed. This is the default primary key which will be always added by the system itself.

If you want to add more key, then you can follow the below steps-

```
:AE DICT MY.TABLE NAME
Top of New "NAME" in "DICT MY.TABLE".
*--: I
001= D
002= 1
003=
004= CUSTOMER NAME
005= 30L
006= S
```

```
007=
*--: FI
Filed "NAME" in file "DICT MY.TABLE".
:
LIST DICT MY.TABLE BY TYP BY @ID TYP LOC CONV NAME FORMAT SM ASSOC
10:50:06 Sep 19 2006 1
@ID............ TYP LOC.......... CONV NAME........... FORMAT SM ASSOC.....

@ID            D            0     MY.TABLE       10L    S
NAME           D            1     CUSTOMER NAME   30L    S
2 records listed
```

So, one more key added with the table. Are you confused with the table structure? Let's get clarified.

001= D-------------------→ this says to database that the record is dictionary type.
002= 1--------------------→ this says to database that the record is in second position.
003=----------------------→ this says to database system that the record conversion is not default.
004= CUSTOMER NAME--→ this says to database that the record name is CUSTOMER NAME
005= 30L---------------→ this says to database that the record length is 30 characters and left aligned.
006= S----------------→ this says to database that the record is single valued.
007=

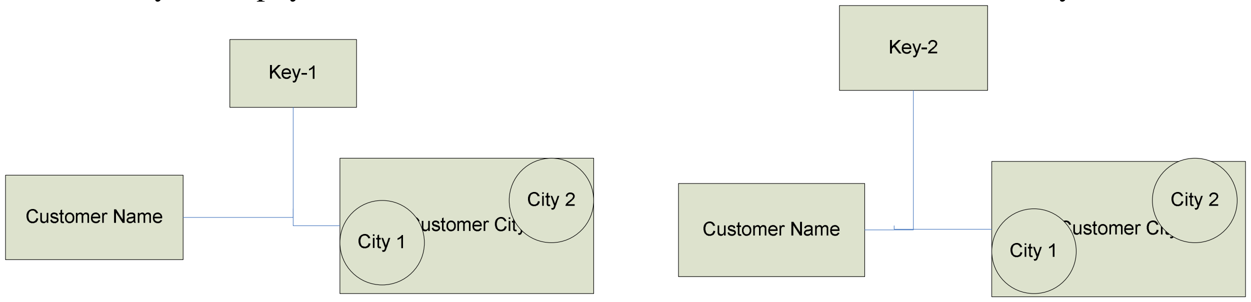
Now I am adding one more record in MY.TABLE-

```
:AE DICT MY.TABLE CITY
Top of New "CITY" in "DICT MY.TABLE".
*--: I
001= D
002= 2
003=
004= CUSTOMER CITY
005= 30L
006= M
007=
*--: FI
Filed "CITY" in file "DICT MY.TABLE".
:
LIST DICT MY.TABLE BY TYP BY @ID TYP LOC CONV NAME FORMAT SM ASSOC
11:02:22 Sep 19 2006 1
@ID............ TYP LOC.......... CONV NAME........... FORMAT SM ASSOC.....

@ID            D            0     MY.TABLE       10L    S
CITY           D            2     CUSTOMER CITY   30L    M
NAME           D            1     CUSTOMER NAME   30L    S
3 records listed
```

Here I added the record CITY. I assumed that a customer city can have more than one value. So my 6th attribute is multi-valued (M). I can write more than one value in CITY.

I will show you the physical data structure, i.e. how would be the data structure in your table-



If you read data from the table, then you will get data in the same structure given above.

6.2 Data-Base quarry:

Like other data-bases, Unidata have some data retrieve quarry, data manipulation quarry and data write quarry. If you create a dictionary file in Unidata, you will get a data file along with the dictionary file. The data definition will be in the dictionary file but the data will be in the data file. So if you want to read a data from the file or write a data on the file, you need to open the data file in a file variable. If you want to change the data dictionary of your file, then you need to open the dictionary file in a file variable. I will cover all types of file related things and data quarry in this section.

6.2.1 OPEN statement:

When you want to read a file or table, you need to open that file, other-wise it can not read the content.

OPEN ‘MY.TABLE’ TO MY.TABLE ELSE STOP “UNABLE TO OPEN THE FILE”

Above statement will open the data file MY.TABLE in the file variable MY.TABLE. If unable to open the file in the file variables it will stop the program and print “UNABLE TO OPEN THE FILE”.

6.2.2 OPENSEQ statement:

When you want to read a file or table sequentially, you need to open that file sequentially, other-wise it can not read the content sequentially.

OPENSEQ "C:/MYTEXT.TXT" TO MYTEXT ELSE STOP "ERROR IN OPENING FILE!!!"

OPENSEQ "C:/MYTEXT.TXT" TO MYTEXT THEN

```
PRINT “FILE OPEND”
ELSE STOP "ERROR IN OPENING FILE!!!"
CLOSESEQ "C:/MYTEXT.TXT"
```

Above first statement will open the file MYTEXT sequentially. The program will be stopped once it can not open the file and print the message as "ERROR IN OPENING FILE!!!" Second statement is best suited to any program.

6.2.3 OPENDEV statement:

Use the OPENDEV statement to open a device for sequential processing.

OPENDEV device TO file variable THEN statements ELSE statements

Device is an expression that evaluates to the record ID of a device definition record in the &DEVICE& (system file) file. If *device* evaluates to the null value, the OPENDEV statement fails and the program terminates with a run-time error message.
If the device does not exist or cannot be opened, the ELSE statements are executed, THEN statements are ignored. The device must have the proper access permissions for you to open it.

6.2.4 OPENPATH statement:

OPENPATH pathname TO file variable ON ERROR statements THEN statements ELSE statements

The OPENPATH statement is similar with OPEN statement, except the pathname of the file is given. This file is opened without any entry in the VOC file. The file must be compatible with UniBasic file.
Path name specifies the relative or absolute pathname of the file to be opened. If the file exists, it is opened and the THEN statements are executed, ELSE statements are ignored. If pathname evaluates to the null value, the OPENPATH statement fails and the program terminates with a run-time error message.

6.2.5 READ statement:

We will use READ statement for reading a file. READ statement will return you a dynamic array of value. See the snippet-

```
REC = ‘’
READ REC FROM MY.TABLE, KEY THEN
PRINT REC
END ELSE
PRINT “DATA NOT FOUND”
```

```
END
```

Here the file MY.TABLE must be opened other-wise the READ command will give error message that, file not opened. Also the key used here must be exist in the data file.
If the file is a table then the effective user of the program must have SELECT privilege to read records in the file.

There are some modes of reading a file. You can lock the file during reading of the file. Let's see those modes-

1. We can use the READL syntax to acquire a shared record lock and then read the record. This allows other programs to read the record with no lock or a shared record lock.

2. We can use the READU statement to acquire an update record lock and then read the record. The update record lock prevents other users from updating the record until the user who owns it releases it.

3. We can use the READV statement to assign the contents of a field in a UniBasic file record to dynamic array.

4. We can use the READVL statement to acquire a shared record lock and then read a field from the record. The READVL statement conforms to all the specifications of the READL and READV statements.

5. We can use the READVU statement to acquire an update record lock and then read a field from the record. The READVU statement conforms to all the specifications of the READU and READV statements.

Examples:

```
READU ITEM FROM MY.FILE,"NAME" ELSE
PRINT "NO RECORD LIKE NAME"
STOP
END

LOOP
WHILE COUNTER < 50 DO
READV ACNT FROM MY.TABLE, KEY, 1 ELSE GOTO SKIP
PRINT ACNT
SKIP:
REPEAT
STOP
```

6.2.6 READSEQ statement:

Use the READSEQ statement to read a line of data from a file opened for sequential processing. Sequential processing lets you process data one line at a time. UniBasic keeps a pointer at the current position in the file. The statement sets this pointer to the first byte of the file.

READSEQ array FROM file-variable ON ERROR statements THEN statements ELSE statements

You have to open a file in sequentially mode i.e. QPENSEQ file-name, other-wise the program will give run time error.

Each READSEQ statement reads data from the current position in the file up to a newline and assigns it to array. The pointer is then set to the position following the newline.

6.2.7 READNEXT statement:

Use the READNEXT statement to assign the next record ID from an active select list to dynamic array. List specifies the select list. If none is specified, select list 0 is used. List can be a number from 0 through 10 indicating a numbered select list, or the name of a select list variable.

READNEXT KEY FROM LIST THEN statements ELSE statements

The BASIC SELECT statements or the GET.LIST, FORM.LIST, SELECT or SSELECT commands create an active select list. These commands build the list of record IDs. The READNEXT statement reads the next record ID on the list specified in the FROM clause and assigns it to the dynamic array.

***Except those READ command, there are some read command like READT for reading tapes and READLIST to read remainder of an active list.

6.2.8 WRITE statement:

You can WRITE statements to write new data to a record in a UniBasic file. The value of expression replaces any data previously stored in the record.

WRITE DAT ON MY.TABLE, KEY ON ERROR PRINT “NOT WRITTEN” THEN PRINT “DADA WRITTEN” ELSE PRINT “KEY NOT FOUND IN THE FILE”

If expression evaluates to the null value, the WRITE statement fails and the program terminates with a run-time error message. File variable specifies an open file. If file variable is not specified, the default file is assumed.

If the file is neither accessible nor open, the program terminates with a run-time error message, unless ELSE statements are specified. The system searches the file for the record specified by record. If the record is not found, WRITE creates a new record.

If the file is a table, the effective user of the program must have INSERT and UPDATE privileges to read records in the file.

How ever, there are some modes of writing the data in the file. I will discuss all those things here.

1. You can use the WRITEU statement to update a record without releasing the update record lock set by a previous READU statement.
   To release the update record lock set by a READU statement and maintained by a WRITEU statement, you must use a RELEASE statement, WRITE statements, MATWRITE statements, or WRITEV statement. If you do not explicitly release the lock, the record remains locked until the program executes the STOP statement. When more than one program or user could modify the same record, use a READU statement to lock the record before doing the WRITE or WRITEU.

2. Use the WRITEV statement to write a new value to a specified field in a record. The WRITEV statement requires that field be specified. Field is the number of the field to which expression is written. It must be greater than 0. If either the record or the field does not exist, WRITEV creates them.

3. Use the WRITEVU statement to update a specified field in a record without releasing the update record lock set by a previous READU statement. The WRITEVU syntax is like that of the WRITEV and WRITEU statements.

Examples:

WRITEV [ID] ON MY.TABLE, [KEY], [POSITION IN DICTIONARY FILE]

6.2.9 WRITESEQ statement:

Use the WRITESEQ statement to write new lines to a file opened for sequential processing. UniBasic keeps a pointer to the current position in the file while it is open for sequential processing. The OPENSEQ statement sets this pointer to the first byte of the file, and it is advanced by the READSEQ statement, READBLK statement, WRITESEQ, and WRITEBLK statement.

WRITESEQ VALUE ON MY.TABLE ON ERROR STATEMENTS THEN STATEMENTS ELSE STATEMENTS.

WRITESEQ writes the value of expression followed by a newline to the file. The data is written at the current position in the file. The pointer is set to the position following the newline. If the pointer is not at the end of the file, WRITESEQ overwrites any existing data byte by byte (including the newline), starting from the current position.

Examples:

WRITESEQ ACCOUNT.NUMBER APPEND TO REPORTFILE ELSE PRINT "BAD REPORTFILE!!!"

6.2.10 WRITELIST statement:

You can use the WRITELIST statement to save a list as a record in the &SAVEDLISTS& file (system file). Dynamic array is an expression that evaluates to a string made up of elements separated by field marks. It is the list to be saved.

WRITELIST dynamic array ON list-name.

*** Except those write statement there are some write statement like, WRUTET is used for writing in a tape and WRITEBLK is used for writing a block of data in a file.

6.2.11 CLOSE statement:

You can use the CLOSE statement after opening and processing a file. Any file locks or record locks will be released. File variable specifies an open file. If file variable is not specified, the default file will be assumed. If the file is neither accessible nor open, or if file variable evaluates to the null value, the CLOSE statement fails and the program terminates with a run-time error message.

CLOSE FILE VARIABLE ON ERROR STATEMENT

6.2.12 CLOSESEQ statement:

You can use the CLOSE statement after opening and processing a file. Any file locks or record locks are released. File variable specifies an open file. If file variable is not specified, the default file is assumed. If the file is neither accessible nor open, or if file variable evaluates to the null value, the CLOSE statement fails and the program terminates with a run-time error message.

CLOSE FILE-VARIABLE ON ERROR STATEMENT

6.2.13 DELETE a file:

Now you are done with all the file handling things. You want to delete the file. See how I can delete the files.

To delete a dictionary file use-
DELETE.FILE MY.TABLE

To delete a record in a dictionary file-
DELETE DICT MY.TABE NAME
The record NAME will be deleted.

To clear a file use-

CLEARFILE FILE-VARIABLE

To clear a active list use-
CLEARSELECT LIST-NAME

7.0 Concept of subroutine:

We can have external subroutine and internal subroutine. For internal subroutine you can use GOSUB statement. For external subroutine you can use CALL statement.

SUBROUTINE SUBROUTINE-NAME [ARGUMENT1, ARGUMENT2…]

We will use RETURN statement for returning to the calling point.

7.1 Internal subroutine:

We will use internal subroutine for readable purpose and proper flow of the programs. We can see the use of GOSUB statement.

Program INT.SUB:

```
VAR = 5
INP = ’’
PRINT “EXAMPLE OF INTERNAL SUBROUTINE”
PRINT “WANT TO TEST INTERNAL SUBROUTINE (Y/N)? “
INPUT INP
IF INP = Y THEN GOTO TEST
PRINT “YOU TESTED SUBROUTINE”
END
TEST:
PRINT “YOU ARE IN INTERNAL SUBROUTINE”
RETURN
```

7.2 External subroutine:

For external subroutine, the subroutine first line should be like-
SUBROUTINE SUBROUTINE-NAME (ARGUMENT1, ARGUMENT2…)

An external subroutine is a separate program or set of statements that can be executed by other programs or subroutines (called calling programs) to perform a task. The external subroutine must be compiled and cataloged before another program can call it.

Use the SUBROUTINE statement to identify an external subroutine. The SUBROUTINE statement must be the first non-comment line in the subroutine. Each external subroutine can contain only one SUBROUTINE statement.

The SUBROUTINE statement can specify a subroutine name for documentation purposes. It need not be the same as the program name or the name by which it is called. The CALL statement must reference the subroutine by its name in the catalog, in the VOC file, or in the object file.

Variables are variable names used in the subroutine to pass values between the calling programs and the subroutine. To pass an array, you must precede the array name with the keyword MAT. When an external subroutine is called, the CALL statement must specify the same number of variables as is specified in the SUBROUTINE statement.
You need to catalog the subroutine with the main program other-wise the main program can not call the subroutine.
CATALOG FOLDER-NAME SUBROUTINE-NAME LOCAL

Program EXT.SUB1

```
INP = ‘’
PRINT “EXAMPLE OF EXTERNAL SUBROUTINE”
PRINT
PRINT “WANT TO TEST EXTERNAL SUBROUTINE(Y/N)? “
INPUT INP
IF INP = Y THEN
CALL EXT.SUB2
PRINT “YOU TESTED EXTERNAL SUBROUTINE”
END
```

Program EXT.SUB2

```
SUBROUTIN EXT.SUB2
MY.NAME = ‘’
PRINT “ENTER YOUR NAME:  “
INPUT MY.NAME
PRINT “HELLO ”:MY.NAME:”, YOU ARE IN EXTERNAL SUBROUTINE”
RETURN
```

8.0 Important APIs:

9.0 TCL commands:

10.0 Concept of Proc:

Proc is similar to JCL in mainframe. When we need to schedule a task at certain time or a workflow in certain direction or need to use some TCL commands, Proc is a good solution. Proc is very strong interface for interacting with system.
Basic structure of a Proc is like-

```
PQN
HMY.FIRST.PROGRAM
P
HMY.SECOND.PROGRAM
P
[ANOTHER.PROC]
RTN
```

Always PQN should be the first statement of a proc.

```
PQN
C NOTICE!!!: All adds and changes to this proc must also be done in
C PL AAA.EARLY.PAYOFFS and PL AAA.ACTIVE.REPORT
C ..................................................................
C * 110896 GBL month end report of all AAA located booked
C* between the begin and end dates
C*
C* 01-23-1997 G B Lindsey: commented out code for "Deed Secured Booked
C* Booked... " per email request from Catherine Looney.
C*
C* 03-31-1997 G B Lindsey: added RC 0120002 (Boston) to RC's
C* tracked.
C* 05-29-1997 G B Lindsey: Added RC 0530123, Auto Finance of America
C*
```

```
C* 06-30-1997 G B Lindsey: Changed division to report RC 0530123 by
C* divisions by coll.code2 A1,I1,and M1,A2.
C*
C* 09-02-1997 G B Lindsey: added 0530123 coll code F1.
C*
C* 09-02-1997 G B Lindsey: added 0010346.
C* 08-17-1998 TINA PENG added 0530123 coll code M5.
C* 05-06-2000 TINA PENG added 0530123 coll code G5
C* 01-14-1999 TINA PENG added "B15" for rc 0530123
C* ----------------------------------------------------------
RTN
HPRINT.LANDSCAPE P044 NET COMPRESSED COPIES_2
P
OBEGIN DATE = +
D3
OEND DATE = +
D4
C *************************************************************
HGETLIST NEW.MONTH
STON
H<
P
HSSELECT LF_LOAN BY BOOK.DATE WITH BANK "20"
H AND WITH RC "0813223" "0805700" "0120002" "0530123" "0010346"
STON
HSAVELIST AAA.MONTHLY.ALL
P
C **************************************************************
HGETLIST AAA.MONTHLY.ALL
P
HSSELECT LF WITH PAY.OFF.DATE AND WITH LOAN.SPAN < "180" BY
PAY.OFF.DATE
STON
HSAVELIST AAA.PAID
P
HGET-LIST AAA.PAID
P
HLIST LF  ACCT RC LOAN.TYPE
H  BOOK.DATE NOTE.DATE PAY.OFF.DATE LOAN.SPAN LOAN.AMT
H HEADING "AAA Origination Report - Accounts Paid from Month    'T'"
H ID-SUPP
H LPTR
P
C **************************************************************
HGET-LIST AAA.MONTHLY.ALL
STON
H<
P
```

```
CHSELECT LF WITH NO PAY.OFF.DATE
CSTON
HSAVELIST AAA.MONTHLY
P
C ***************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0805700"
H AND WITH NO COLL.CODE2 "C2"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0805700 (NO Coll.code C2) -  Titled - Booked
A3
HTo
A4
H 'T'    Page   'PL'"
H  LPTR
P
C ..................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0805700"
H AND WITH NO COLL.CODE2 "C2"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0805700(NO Coll.code C2)  Home Improvement -
Booked
A3
HTo
A4
H 'T'      Page   'PL'"
H  LPTR
P
C ************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
```

```
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0805700"
H AND WITH COLL.CODE2 "C2"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0805700(AAA-Hartford: Coll.Code C2) -  Titled -
Booked
A3
HTo
A4
H 'T'    Page   'PL'"
H  LPTR
P
C .......................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0805700"
H AND WITH COLL.CODE2 "C2"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0805700(AAA-Hartford: Coll.Code C2)  Home
Improvement - Booked
A3
HTo
A4
H 'T'      Page   'PL'"
H  LPTR
P
C .......................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0813223"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0813223 -  Titled - Booked
A3
HTo
A4
```

```
H 'T'    Page   'PL'"
H  LPTR
P
C ........................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0813223"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0813223  Home Improvement - Booked
A3
HTo
A4
H 'T'      Page   'PL'"
H  LPTR
P
C ******************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0120002"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0120002 -  Titled - Booked
A3
HTo
A4
H 'T'    Page   'PL'"
H  LPTR
P
C ....................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0120002"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
```

```
H HEADING "AAA Originations Report  RC: 0120002  Home Improvement - Booked
A3
HTo
A4
H 'T'      Page   'PL'"
H  LPTR
P
C ******************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "A1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Auto Finance of America: Coll Code A1)-
Titled - Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C .....................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "A1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Auto Finance of America: Coll Code A1)
Home Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C****************************************************
HGETLIST AAA.MONTHLY
```

```
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "I1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Info Source)-  Titled - Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ..................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "I1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Info Source) Home Improvement -
Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C ****************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "M5"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
```

```
H HEADING "AAA Originations Report  RC: 0530123 (personal finacial solution)-  Titled -
Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ....................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "M5"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (personal finacial solution) Home
Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C**************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "G5"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (G5)-  Titled - Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ....................................................................
HGETLIST AAA.MONTHLY
```

```
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "G5"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (G5) Home Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C***************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "M1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Marlboro Memorial Cemetery)-  Titled -
Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ...................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "M1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
```

```
H HEADING "AAA Originations Report  RC: 0530123 (Marlboro Memorial Cemetery) Home
Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C ******************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "A2"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Mass Buying Power)-  Titled - Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ......................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "A2"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Mass Buying Power) Home Improvement
- Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C ******************************************************************
HGETLIST AAA.MONTHLY
```

```
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "F1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (First National Bank of New England
(F1))-  Titled - Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ..................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "F1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (First National Bank of New England
(F1)) Home Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C *****************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "A4"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
```

```
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Affordable Auto Finance: Coll Code A4)-
Titled - Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ......................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "A4"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Affordable Auto Finance: Coll Code A4)
Home Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C *******************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "L1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Metro Cabinet: Coll Code L1)-  Titled -
Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
```

```
C ..................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "L1"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Metro Cabinet: Coll Code L1) Home
Improvement - Booked
A3
HTo
A4
H'L"T'     Page   'PL'"
H  LPTR
P
C ******************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "C4"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Computer Institute: Coll Code C4)-
Titled - Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ..................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "C4"
H NAME @ID NOTE.RATE VISIT
```

```
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Computer Institute: Coll Code C4) Home
Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C ******************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "D7"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Dental Institute: Coll Code D7)-  Titled -
Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ..................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "D7"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Dental Institute: Coll Code D7) Home
Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
```

```
P
C ************************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "B15"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE.LOW
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Sanbron Corp: Coll Code B15)-  Titled -
Booked
A3
HTo
A4
H'L"T'    Page   'PL'"
H  LPTR
P
C ..................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0530123"
H AND WITH COLL.CODE2 = "B15"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0530123 (Sanbron Corp: Coll Code B15) Home
Improvement - Booked
A3
HTo
A4
H'L"T'      Page   'PL'"
H  LPTR
P
C****************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH COLL.TYPE = "TITLE"
H AND WITH LOAN.TYPE # "014" "015"
H AND WITH RC "0010346"
```

```
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL TT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0010346 -  Titled - Booked
A3
HTo
A4
H 'T'    Page   'PL'"
H  LPTR
P
C ....................................................................
HGETLIST AAA.MONTHLY
STON
H<
P
HLIST LF_LOAN WITH LOAN.TYPE = "014" "015"
H AND WITH RC "0010346"
H NAME @ID NOTE.RATE VISIT
H BOOK.DATE TOTAL AAA.LOAN.PROCEEDS TOTAL DT.AAA.FEE
H ID-SUPP
H HEADING "AAA Originations Report  RC: 0010346  Home Improvement - Booked
A3
HTo
A4
H 'T'      Page   'PL'"
H  LPTR
P
C*************************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HSORT LF_LOAN BY LOAN.TYPE BREAK-ON LOAN.TYPE TOTAL COUNT
H ID-SUPP DET-SUPP
H HEADING "AAA Originations Report - Loan Type Summary         'T'"
H LPTR
P
C ************************************************************************
HGETLIST AAA.MONTHLY
STON
H<
P
HSORT LF BY LOAN.TYPE WITH  COLL.TYPE # "TITLE" AND WITH COLL.TYPE #
"DEED"
H AND WITH COLL.TYPE # "UNSECURED" AND WITH LOAN.TYPE # "014"
H AND WITH LOAN.TYPE # "015"
H LOAN.TYPE COLL.TYPE COLL.DESC
H HEADING "AAA Originations Report - Unmatched Collateral Types    'T'      Page 'PLL' "
```

```
H LPTR
P
C*****
HLOG.REPORT.JOB AAA_MONTHLY_REPORT PRINT>P044
P
[PL AAA.ACTIVE.REPORT]
```

11.0 A typical project on Infy Bank:

I will discuss one typical UnIData-UniBasic application in this section. This will help a novice programmer to understand UnIData-UniBasic technology a lot. For example, you can say that, *how to do an account maintenance system in UnIData-UniBasic language*?

I will discuss the account maintenance system in a bank from scratch. Assume your requirement is like-
"You want an account maintenance system in Infy Bank where you can add a customer, delete a customer, check the status of balance, balance transfer in another account. More over you want to put some validation over there. Like, Customer name should be with in 20 character, Balance should not be more than 11 digit, City name should not be more than 10 characters, Phone number will be with in 7 digit, Customer can have more than one phone number and residences."

Follow the below steps one by one-

- Create a dictionary file CUSTOMAR.TABLE where number of elements will be Customer name, Account Balance, City, Phone number. This is like a table in other data base engine.
- Every time UnIData will create a data file along with the dictionary file for you to write data over there.
- Create another dictionary file. This dictionary file will generate the customer ID automatically. This file has one element, i.e. "TOREAD".
- So your data base design is over. Now you will create the UniBasic programs that will interact with the UnIData tables.
- Create a module ADDCUSTOMER that should contain add and delete functionality of customer.
- Create another module DISPLAY which will display details of a customer.
- Create a module TRANSACTION which will contain the functionality of balance transfer.
- Finally create another module CUST which will contain a menu driven screen from where you can go any one of the above module.

Let us now see the actual commands and programs-

- Log on to UnIData-Unibasic Shell (Universe, UDT, V2)

- CREATE.FILE CUSTOMAR.TABLE
- Give modulo as 101.
- LIST DICT CUSTOMER.TABLE -----You can see the dictionary file which you created.
- :AE DICT CUSTOMER.TABLE NAME
- Top of New "NAME" in "DICT CUSTOMER.TABLE".
- *--: I
- 001= D
- 002= 1
- 003=
- 004= CUSTOMER NAME
- 005= 20L
- 006= S
- 007=
- *--: FI
- Filed "NAME" in file "DICT CUSTOMER.TABLE".
- :AE DICT CUSTOMER.TABLE BALANCE
- Top of New "BALANCE" in "DICT CUSTOMER.TABLE".
- *--: I
- 001= D
- 002= 2
- 003=
- 004= CUSTOMER BALANCE
- 005= 11L
- 006= S
- 007=
- *--: FI
- Filed "BALANCE" in file "DICT CUSTOMER.TABLE".
- :AE DICT CUSTOMER.TABLE ADDRESS1
- Top of New "ADDRESS1" in "DICT CUSTOMER.TABLE".
- *--: I
- 001= D
- 002= 3
- 003=
- 004= ADRESS LINE 1
- 005= 50L
- 006= M
- 007=
- *--: FI

- Filed "ADDRESS1" in file "DICT CUSTOMER.TABLE".
- :AE DICT CUSTOMER.TABLE ADRESS2
- Top of New "ADRESS2" in "DICT CUSTOMER.TABLE".
- *--: I
- 001= D
- 002= 4
- 003=
- 004= ADRESS LINE 2
- 005= 50L
- 006= M
- 007=
- *--: FI
- Filed "ADRESS2" in file "DICT CUSTOMER.TABLE".
- :AE DICT CUSTOMER.TABLE ADRESS3
- Top of New "ADRESS3" in "DICT CUSTOMER.TABLE".
- *--: I
- 001= D
- 002= 5
- 003=
- 004= ADRESS LINE 3
- 005= 50L
- 006= M
- 007=
- *--: FI
- Filed "ADRESS3" in file "DICT CUSTOMER.TABLE".
- LIST DICT CUSTOMER.TABLE BY TYP BY @ID TYP LOC CONV NAME FORMAT SM ASSOC 12:47:39 Sep 16 2006 1
- @ID............ TYP LOC.......... CONV NAME........... FORMAT SM ASSOC.....
-
- @ID   D   0   CUSTOMER.TABLE 10L   S
- ADDRESS1   D   3   ADRESS LINE 1   50L   M
- ADRESS2   D   4   ADRESS LINE 2   50L   M
- ADRESS3   D   5   ADRESS LINE 3   50L   M
- BALANCE   D   2   CUSTOMER BALANC 11L   S
-                                 E
- NAME   D   1   CUSTOMER NAME   20L   S
- 6 records listed
- :CREATE.FILE CUSTOMER.ID-----This is for automatic generation of customer ID.
- modulos for file CUSTOMER.ID=101

- Create file D_CUSTOMER.ID, modulo/1,blocksize/1024
- Hash type = 0
- Create file CUSTOMER.ID, modulo/101,blocksize/1024
- Hash type = 0
- Added "@ID", the default record for UnIData to DICT CUSTOMER.ID.
- :AE DICT CUSTOMER.ID TOREAD
- Top of New "TOREAD" in "DICT CUSTOMER.ID".
- *--: I
- 001= D
- 002= 1
- 003=
- 004= LAST CUSTOMER ID
- 005= 20L
- 006= S
- 007=
- *--: FI
- Filed "TOREAD" in file "DICT CUSTOMER.ID".
- LIST DICT CUSTOMER.ID BY TYP BY @ID TYP LOC CONV NAME FORMAT SM ASSOC 12:57:03 Sep 16 2006 1
- @ID............ TYP LOC.......... CONV NAME........... FORMAT SM ASSOC.....
-
- @ID   D   0   CUSTOMER.ID   10L   S
- TOREAD   D   1   LAST CUSTOMER I 20L   S
-                                  D
- 2 records listed
- So you successfully created your data bases.
- Now you will create the UniBasic programs to interact with those data bases.
- For that you will create a directory in UniBasic shell. Follow the steps-
- :CREATE.FILE DIR BOK
- Create DIR type file BOK.
- Create file D_BOK, modulo/1,blocksize/1024
- Hash type = 0
- Added "@ID", the default record for UniData to DICT BOK.
- :AE BOK ADDCUSTOMER
- Top of New "ADDCUSTOMER" in "BOK".
- *--: I
- 001= ***Type the below ADDCUSTOMER module. Keep in mind when I pasted the program in word documents the single lines are broken into two lines. You have to take care of those things. Otherwise the program will give you compilation error.***

- Similarly do for all the modules.
- Module ***ADDCUSTOMER:***

```
    SUBROUTINE ADDCUSTOMER

    OPEN "CUSTOMER.TABLE" TO CUSTOMER.TABLE ELSE PRINT "UNABLE TO
OPEN THE FILE"
    OPEN "CUSTOMER.ID" TO ID ELSE PRINT "UNABLE TO OPEN THE FILE ID"
    INITCUSTID=0
    READV INITCUSTID FROM ID,"TOREAD",1 ELSE PRINT "ERROR"

    PRINT "1. ADD CUSTOMER"
    PRINT "2. DELETE CUSTOMER"
    PRINT "ENTER YOUR CHOICE :"
    INPUT CHOICE2

    IF CHOICE2=1 THEN
      TEMPVAR=""
      PRINT "NAME:"
      INPUT NAME1
      LENGT=LEN(NAME1)

            LOOP WHILE LENGT>=21

                    PRINT "CUSTOMER NAME SHOULD NOT BE MORE THAN 20
CHARACTERS:";INPUT NAME1

                    LOOP WHILE (ALPHA(NAME1)=0)

                    PRINT "ENTER CUSTOMER NAME:";INPUT NAME1

                    REPEAT

                    LENGT=LEN(NAME1)

    REPEAT

      PRINT "BALANCE:"
      INPUT BALANCE1
      LENGT=LEN(BALANCE1)

                    LOOP WHILE LENGT>=11

                            PRINT "BALANCE SHOULD NOT BE MORE THAN 20
CHARACTERS:";INPUT BALANCE1

                    LOOP WHILE (NUM(BALANCE1)=0)
```

```
                              PRINT "BALANCE SHOULD BE IN NUMBERS
ONLY:";INPUT BALANCE1
                      REPEAT

                              LENGT=LEN(BALANCE1)

              REPEAT
      PRINT "CITY1:"
      INPUT CITY1
      LENGT=LEN(CITY1)

              LOOP WHILE LENGT>=11
                      PRINT " CITY NAME SHOULD NOT BE MORE THAN 10
CHARACTERS:";INPUT CITY1
              LOOP WHILE (ALPHA(CITY1)=0)
                      PRINT "ENTER CITY NAME:";INPUT CITY1
              REPEAT
                      LENGT=LEN(CITY1)

    REPEAT

      PRINT "PHONENO1:"
      INPUT PHONENO1
      LENGT=LEN(PHONENO1)

              LOOP WHILE LENGT>=8
                      PRINT "PHONE NO SHOULD NOT BE MORE THAN 7
DIGITS:";INPUT PHONENO1
              LOOP WHILE (NUM(PHONENO1)=0)
                      PRINT "ENTER PHONE NUMBER:";INPUT PHONENO1
              REPEAT
                      LENGT=LEN(PHONENO1)

    REPEAT

      PRINT "CITY2:"
      INPUT CITY2
      LENGT=LEN(CITY2)

                      LOOP WHILE LENGT>=11
                              PRINT " CITY NAME SHOULD NOT BE MORE THAN 10
CHARACTERS:";INPUT CITY2
                      LOOP WHILE (ALPHA(CITY2)=0)
                              PRINT "ENTER CITY NAME:";INPUT CITY2
                      REPEAT
                      LENGT=LEN(CITY2)
    REPEAT

     PRINT "PHONENO2:"
     INPUT PHONENO2
     LENGT=LEN(PHONENO2)

                    LOOP WHILE LENGT>=8
                         PRINT "PHONE NO SHOULD NOT BE MORE THAN 7
DIGITS:";INPUT PHONENO2
                    LOOP WHILE (NUM(PHONENO2)=0)
                         PRINT "ENTER PHONE NUMBER:";INPUT PHONENO2
                    REPEAT
                         LENGT=LEN(PHONENO2)

   REPEAT

     PRINT "CITY3:"
     INPUT CITY3
     LENGT=LEN(CITY3)

                    LOOP WHILE LENGT>=11
                         PRINT " CITY NAME SHOULD NOT BE MORE THAN 10
CHARACTERS:";INPUT CITY3
                    LOOP WHILE (ALPHA(CITY3)=0)
                         PRINT "ENTER CITY NAME:";INPUT CITY3
                    REPEAT
                    LENGT=LEN(CITY3)
    REPEAT

     PRINT "PHONENO3:"
     INPUT PHONENO3
     LENGT=LEN(PHONENO3)

                         LOOP WHILE LENGT>=8
                              PRINT "PHONE NO SHOULD NOT BE MORE THAN 7
DIGITS:";INPUT PHONENO3
                         LOOP WHILE (NUM(PHONENO3)=0)
                              PRINT "ENTER PHONE NUMBER:";INPUT
PHONENO3
                         REPEAT
                              LENGT=LEN(PHONENO3)

   REPEAT

     TEMPVAR<1>=NAME1
     TEMPVAR<2>=BALANCE1
```

```
      TEMPVAR<3,1>=CITY1
      TEMPVAR<4,1>=CITY2
      TEMPVAR<5,1>=CITY3
      TEMPVAR<3,2>=PHONENO1
      TEMPVAR<4,2>=PHONENO2
      TEMPVAR<5,2>=PHONENO3
      WRITE TEMPVAR TO CUSTOMER.TABLE,INITCUSTID+1
      INITCUSTID=INITCUSTID+1
      PRINT "GENERATED CUSTID:":INITCUSTID
      WRITE INITCUSTID TO ID,"TOREAD"
    END

      ELSE IF CHOICE2=2 THEN
      PRINT "ENTER THE CUSTOMER ID TO BE DELETED :"
      INPUT ID
      DELETE CUSTOMER.TABLE, ID ON ERROR PRINT "CANNOT DELETE"
      PRINT "CUSTOMER WITH ID ":ID:" IS DELETED"
    END


  END
```

- Module ***DISPLAY:***

```
SUBROUTINE DISPLAY
     PRINT "ENTER THE ACCOUNT ID :"
    INPUT ID2
    flag=1
    OPEN "CUSTOMER.TABLE" TO CUSTOMER.TABLE ELSE PRINT "UNABLE TO
OPEN THE FILE"
    READ TEMP FROM CUSTOMER.TABLE,ID2 ELSE flag=0

    IF flag=0 THEN
         PRINT
            PRINT
            PRINT "ACCOUNT ALREADY DELETED"
            END

    IF flag=1 THEN
            PRINT
            PRINT
            PRINT "NAME :" :TEMP<1>
            PRINT "ACCOUNT ID :":ID2
            PRINT "BALANCE :": TEMP<2>
            PRINT
            PRINT "CITY          PHONE NO"
            PRINT "----          --------"
            PRINT TEMP<3,1>:"        ":TEMP<3,2>
            PRINT TEMP<4,1>:"        ":TEMP<4,2>
            PRINT TEMP<5,1>:"        ":TEMP<5,2>
            PRINT "GIVE THE NAME OF CITY FOR WHICH YOU WANT THE PHONE
NUMBER:";

                    INPUT C1

                    IF C1=TEMP<3,1> THEN
                    PRINT TEMP<1> : "'S CONTACT NUMBER FOR " : C1 : " IS " :
TEMP<3,2>
                    END

                    ELSE IF C1=TEMP<4,1> THEN
                    PRINT TEMP<1> : "'S CONTACT NUMBER FOR " : C1 : " IS " :
TEMP<4,2>
                    END

                    ELSE
                    PRINT TEMP<1> : "'S CONTACT NUMBER FOR " : C1 : " IS " :
TEMP<5,2>
    END

    END

    END
```

- Module ***TRANSACT:***

```
SUBROUTINE TRANSACT

     OPEN "CUSTOMER.TABLE" TO CUSTOMER.TABLE ELSE PRINT "UNABLE TO
OPEN THE FILE"
    OPEN "CUSTOMER.ID" TO ID ELSE PRINT "UNABLE TO OPEN THE FILE
RDCUSTID"

     PRINT "4 : DEPOSIT"
     PRINT "5 : WITHDRAW"
     PRINT "ENTER YOUR CHOICE :"
     INPUT CHOICE1

    IF CHOICE1=4 THEN
            TEMP = ""
            PRINT
            PRINT "ENTER THE ACCOUNT ID :"
            INPUT ID1
            BAL = 0
            PRINT "ENTER THE AMOUNT TO BE DEPOSITED :"
            INPUT AMT1
            READ TEMP FROM CUSTOMER.TABLE,ID1 THEN BAL = TEMP<2> ELSE
PRINT "UNABLE TO FIND THE DATA"
            BAL = BAL + AMT1
            TEMP<2> = BAL
            WRITE TEMP TO CUSTOMER.TABLE, ID1
            IF BAL > AMT1 THEN PRINT "THE BALANCE IS :":BAL
            END

    IF CHOICE1=5 THEN
            PRINT "ENTER THE ACCOUNT ID :"
            INPUT ID2
            PRINT "ENTER THE AMOUNT TO BE WITHDRAWN :"
            INPUT AMT2
            BAL = 0
            READ TEMP1 FROM CUSTOMER.TABLE,ID2 THEN BAL = TEMP1<2>
ELSE PRINT "UNABLE TO GET THE DATA"
            BAL = BAL - AMT2
            TEMP1<2> = BAL
            WRITE TEMP1 TO CUSTOMER.TABLE,ID2
            IF BAL > 0 THEN PRINT "THE FINAL BALANCE IS :":BAL
            END
 END
```

- Module ***CUSTOMER.MAINT:***

```
START:
PRINT "1 : CUSTOMER MAINTENANCE"
PRINT
PRINT
PRINT "2 : TRANSACTION HANDLING"
PRINT
PRINT
PRINT "3 : QUERY"
PRINT
PRINT

PRINT "ENTER YOUR CHOICE"
INPUT CHOICE
OPEN "CUSTOMER.TABLE" TO CUSTOMER.TABLE ELSE PRINT "UNABLE TO
OPEN THE FILE"
IF CHOICE=1 THEN
     CALL ADDCUSTOMER
     GO START:
     END
ELSE IF CHOICE=2 THEN
```

```
       CALL TRANSACTION
       GO START:
       END
ELSE IF CHOICE=3 THEN
        CALL DISPLAY
      GO START:
      END
END
```

- Now you need to compile those programs. Follow the below steps-
- BASIC BOK ADDCUSTOMER
- CATALOG BOK ADDCUSTOMER LOCAL
- BASIC BOK TRANSACT
- CATALOG BOK TRANSACT LOCAL
- BASIC BOK DISPLAY
- CATALOG BOK DIPLAY LOCAL
- BASIC BOK CUSTOMER.MAINT
- RUN BOK CUSTOMER.MAINT
- If you follow properly all the steps which I mentioned above, there should not be any error and you will be able to see the below screen-

```
:RUN BOK CUSTOMER.MAINT
1 : CUSTOMER MAINTENANCE

2 : TRANSACTION HANDLING

3 : QUERY

ENTER YOUR CHOICE
?
?1
1. ADD CUSTOMER
2. DELETE CUSTOMER
ENTER YOUR CHOICE :
?
?1
NAME:
?TOMMY
BALANCE:
?500000
CITY1:
?KOLKATA
PHONENO1:
?00003
```

```
CITY2:
?NEW YORK
PHONENO2:
?704
CITY3:
?LONDON
PHONENO3:
?888
GENERATED CUSTID:2
1 : CUSTOMER MAINTENANCE

2 : TRANSACTION HANDLING

3 : QUERY

ENTER YOUR CHOICE
?
?2
4 : DEPOSIT
5 : WITHDRAW
ENTER YOUR CHOICE :
?
?4

ENTER THE ACCOUNT ID :
?2
ENTER THE AMOUNT TO BE DEPOSITED
?500
THE BALANCE IS :500500
1 : CUSTOMER MAINTENANCE

2 : TRANSACTION HANDLING

3 : QUERY

ENTER YOUR CHOICE
?
?5
ENTER THE ACCOUNT ID :
?2
ENTER THE AMOUNT TO BE WITHDRAWN :
?250
THE FINAL BALANCE IS :500250
1 : CUSTOMER MAINTENANCE

2 : TRANSACTION HANDLING
```

```
3 : QUERY

ENTER YOUR CHOICE
?
?3
ENTER THE ACCOUNT ID :
?2
NAME :TOMMY
ACCOUNT ID :2
BALANCE :500250

CITY         PHONE NO
----         --------
KOLKATA        00003
NEW YORK        704
LONDON        888
GIVE THE NAME OF CITY FOR WHICH YOU WANT THE PHONE NUMBER:
?
?LONDON
TOMMY'S CONTACT NUMBER FOR LONDON IS 888
1 : CUSTOMER MAINTENANCE

2 : TRANSACTION HANDLING

3 : QUERY

ENTER YOUR CHOICE
?
?4
:
```

So, now you can be very happy that you are done with an Infy Bank account maintenance project. Now for details knowledge, you have to go through the complete document up to bottom. I am expecting that, you are quite interested now to go through the material, right? This document is strictly accessible for Infosys and Wachovia employees. If you don't understand any where in this document, you can contact me at partha.mondal@in.ibm.com OR Partha.contact@gmail.com OR reachparthapm@gmail.com .

12.0 <u>Summary:</u>

13.0 <u>References:</u>

14.0 <u>Appendix A:</u>

14.1 Sample_Programs: