Infobasic Programming

TEMENOS

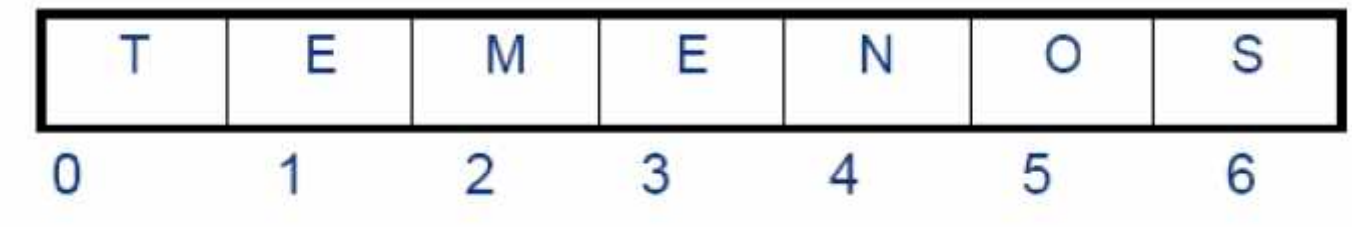
The Banking Software Company

Agenda

- Introduction to Infobasic
- Arrays and types of arrays
- Introduction to subroutines and programs
- Important functions/commands in Infobasic
- Steps to create a subroutine in T24
- Compiling and cataloguing routines and programs
- T24 routines – file operations
- T24 routines – sequential file access
- T24 – Creation of functions and routines with arguments

- Programming language used for T24
- Simple – English like statements
- No declaration of variables required
- No data type specification is required
- All variables by default have infinite variable length
  - **Multi value**
  - **Sub Value**

- Continuous allocation of bytes
- All bytes in an array have the same name as that of the array
- Each byte is uniquely identified with the help of a subscript.



Arrays In Infobasic

- Dynamic Arrays
  - **Dynamic in nature**
  - **Variable length**
  - **Need not be declared**
  - **Can hold any type of data**
  - **Automatically increase or decrease in size depending on the data**
  - **All variables in Infobasic are dynamic arrays**
- Dimensioned arrays
  - **Have a fixed number of rows and columns**
  - **Can hold any type of data**
  - **Needs to be declared**
  - **Used when dimensions and extents are known and are not likely to change**

CUSTOMER.NAME = ‘’

RATE = 0

DATE = “121202”

Can store any type and any amount of data.

Only initialisation is required.

- Dynamic Arrays (Cont.)
  - **Uses delimiters to store data of various fields**

ASCII	Decimal	Description
254		Field Marker
253		Value Marker
252		Sub-Value Marker

Sample Record From The TEMENOS.TRG

1 Name	TemenosTrg
2.1 Address	India
2.2 Address	UK
2.3 Address	Geneva
3.1 Course Category	Technical
4.1.1 Course Name	jBASE
4.1.2 Course Name	T24
3.2 Course Category	Functional
4.2.1 Course Name	Lending
4.2.2 Course Name	Financials
5 Free Text	
6 Inputter	TRAINER.1

How will this record get stored in a dynamic array?

Storage In A Dynamic Array

TemenosTrg**FM**India**VM**UK**VM**Geneva**FM**Technical**VM**Functional**FM**

jBASE**SM**T24**VM**Lending**SM**Financials**FMFM**Trainer.1

Structure Of An Infobasic Program

PROGRAM – Executed from the database prompt

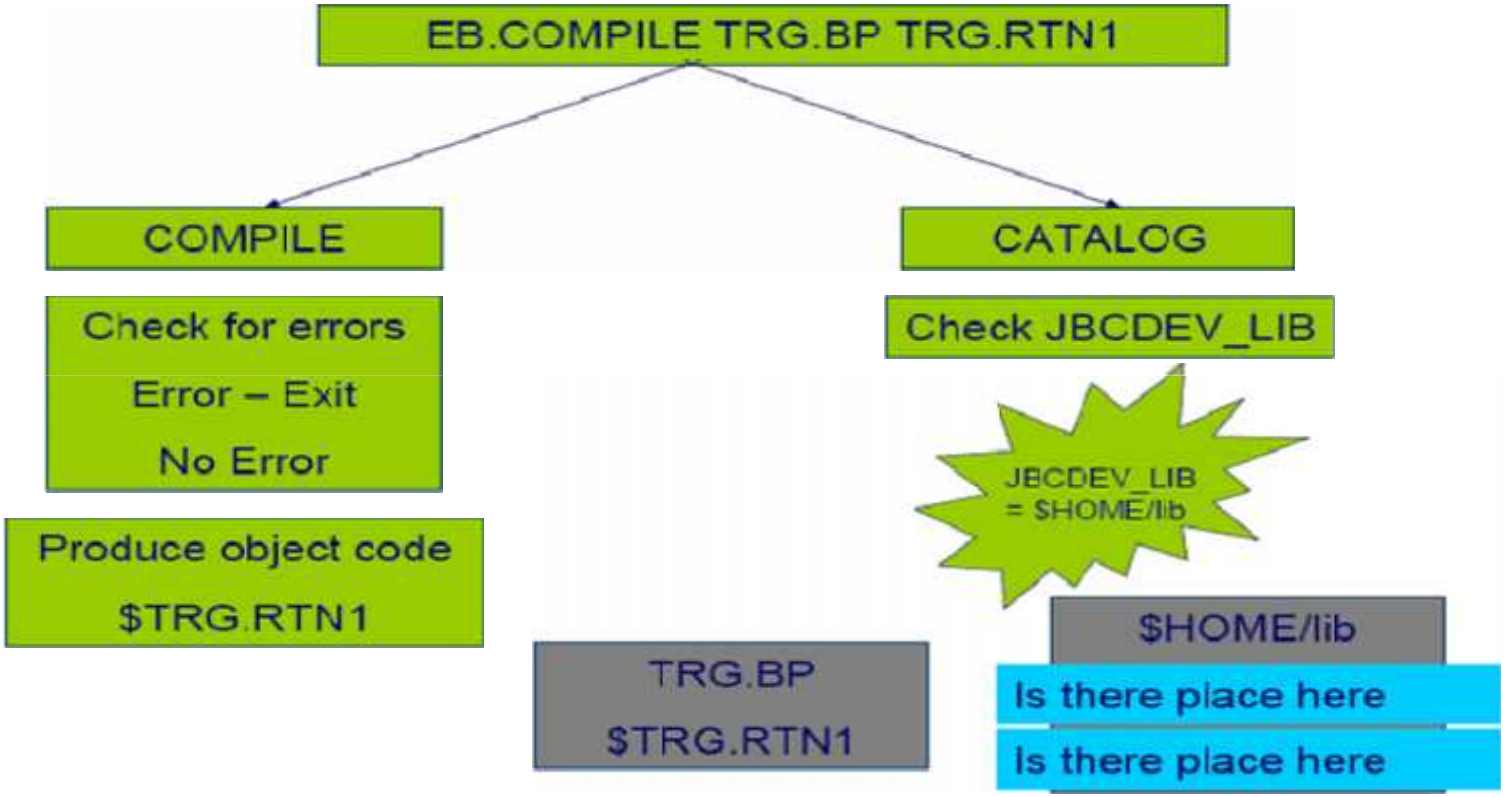
SUBROUTINE – Execute from within Globus

```
*Comments
PROGRAM <Programname>
Statement 1
Statement 2
Statement 3

END
```

```
*Comments
SUBROUTINE <Subroutinename>
Statement 1
Statement 2
Statement 3
RETURN
END
```

Compiling And Cataloguing Routines



Executing Routines

Login into T24

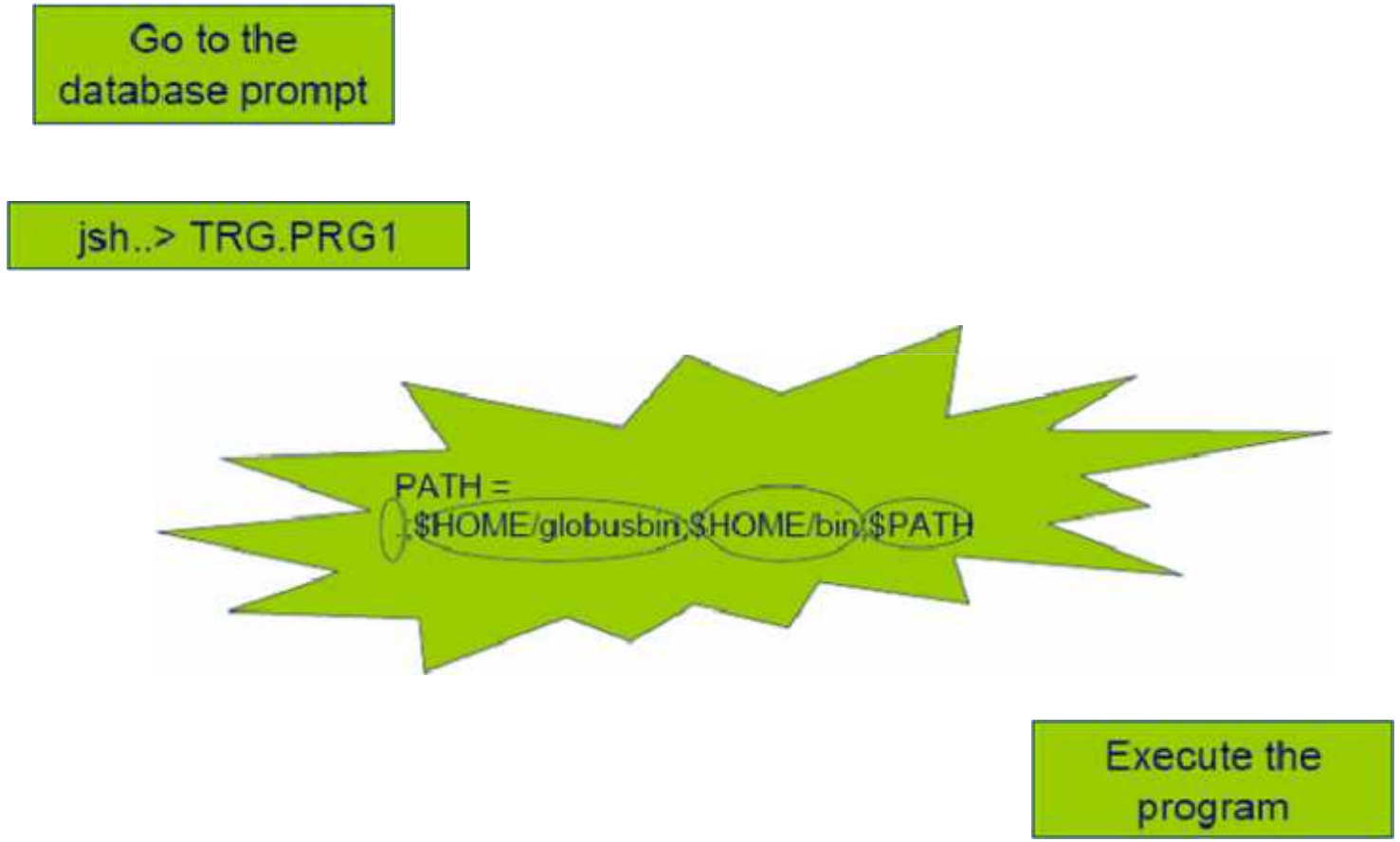
Make an entry in the PGM.FILE

At the command line

TRG.RTN1

Execute the routine

Executing Programs



Writing A Simple Infobasic Program

Program to display ‘Hello World’

```
JED TRG.BP HELLO
PROGRAM HELLO
CRT “Hello World”
END
```

Compile And Execute The Program

- Compile and catalog

  EB.COMPILE TRG.BP HELLO

- Execute the program

  jsh..>HELLO

  Hello World

SUBROUTINES WITH PARAMETERS

- Subroutines can take In Parameters and Return the values as well.
- Example: F.READ

  This Subroutine has five parameters out of which

  File name, record id and File Pointer are incoming parameters.

  Recordvar, Errorvar are Return values

Example:

Step 1:

Create a routine that accepts two values and returns the multiplication of two values in variable.

```
SUBROUTINE TEST.CALLED.RTN(A,B,C)

C = A * B

RETURN

END
```

Step 2: Create another routine that supplies the two values and prints the result.

```
PROGRAM TEST.CALLING.RTN
A = 10 ; B =20
CALL TEST.CALLED.RTN(A,B,RESULT)
PRINT “RESULT : ”:RESULT
END
```

Step 3: Compile and catalog the Program and Subroutine.

Step 4:

Run the Program

Result : RESULT : 200

PATH

- Contains a list of all directories that has executable programs.
- Typically, PATH is already set to locate system and other application executables (like C, java, etc)
- Add paths for jBASE and user executables
- User executables are your cataloged programs for the Globus application

Unix

PATH=$PATH:/apps/bin

export PATH

Windows

SET PATH=%PATH%;D:\apps\bin

JBCOBJECTLIST

- Shows Search path of user subroutines
- This is used during program execution

```
Unix
JBCOBJECTLIST=$HOME/lib:/home/TESTBASE/lib
export JBCOBJECTLIST
Windows
SET JBCOBJECTLIST=%HOME%\lib;C:\dev\TESTBASE\lib
```

JBCDEV_BIN AND JBCDEV_LIB

- Define where the locally developed executables and libraries are stored (when cataloged)
- Tell jBASE where to put your cataloged programs and subroutines
- These paths should be included in your PATH and JBCOBJECTLIST for your programs to be used.

TEMENOS

```
export JBCDEV_BIN=$HOME/bin
```

```
export JBCDEV_LIB=$HOME/lib
```

The various Control structures are

- IF-THEN-ELSE Structure
- CASE Structure
- FOR-NEXT Structure
- LOOP-REPEAT Structure

- IF THEN ELSE

```
IF expression THEN
        statements
   END ELSE
            IF expression THEN
                statements
            END ELSE
                statements
            END
   END
```

IF THEN ELSE - EXAMPLE

```
A = 2
IF  A > 10 THEN
        PRINT “GIVEN NUMBER IS GT 10”
   END ELSE
          IF A  5 THEN
              PRINT “INPUT IS GT THAN 5 AND LE 10”
          END ELSE
              PRINT “INPUT GIVEN IS LT 5”
          END
   END
```

RESULT : INPUT GIVEN IS LT 5

- BEGIN CASE … END CASE

```
BEGIN CASE
CASE <variable> = <value>
        <statements>
CASE <variable> = <value>
        <statements>
CASE <variable> = <value>
        <statements>
CASE 1
        <statements>
END CASE
```

BEGIN CASE … END CASE

Expressions/statements evaluated in sequential order.

- when a true expression is encountered, the corresponding statements

  are executed and control exits from the CASE structure.

- when no true expression is encountered, then the statements under CASE 1 are executed and control exits from the CASE structure.

- when no true expression is encountered, and in the absence of CASE 1 expression, then no statements are executed and control exits from the CASE structure.

CASE Structure - Example

```
PROGRAM TEST4
    NAME="MICHAEL"
    BEGIN CASE
      CASE NAME[1,2]='DA'
        PRINT NAME
      CASE NAME[1,2]='RI'
        PRINT NAME
      CASE NAME[1,2]='BA'
        PRINT NAME
      CASE 1
        PRINT 'NO MATCH'
    END CASE
END
```

Output : No Match

- FOR

FOR - NEXT Structure

- Facilitates repeated execution of a set of statements.
- Control exits from structure after pre-determined number of executions.

FOR <variable> = <initialvalue> TO <maximumvalue>
<statements>
NEXT <variablename>

FOR - NEXT Structure- Example

```
FOR I = 1 TO 10
   PRINT 'Counter Variable:': I
NEXT
```

I is the counter variable.

The PRINT statement executes for a pre-determined 10 times (i.e) till the value of I is 10

Output:
COUNTER VARIABLE:1
COUNTER VARIABLE:2
COUNTER VARIABLE:3
COUNTER VARIABLE:4
COUNTER VARIABLE:5
COUNTER VARIABLE:6
COUNTER VARIABLE:7
COUNTER VARIABLE:8
COUNTER VARIABLE:9
COUNTER VARIABLE:10

FOR - NEXT Structure- Example

```
FOR I = 1 TO 10 STEP 2
   PRINT 'Counter Variable:': I
NEXT
```

I is the counter variable.

The PRINT statement executes for a pre-determined 5 times, I being incremented by two each time.

Output :

```
COUNTER VARIABLE:1
COUNTER VARIABLE:3
COUNTER VARIABLE:5
COUNTER VARIABLE:7
COUNTER VARIABLE:9
```

LOOP - REPEAT Structure

Facilitates repeated execution of a set of statements. Control exits from structure when WHILE expression evaluates to false or UNTIL expression evaluates to true.

- **SYNTAX**

```
LOOP
      set of statements
WHILE/UNTIL expression
      set of statements
REPEAT
```

LOOP - REPEAT Structure - Example

```
PROGRAM TEST2
   X=0
   LOOP
   UNTIL X>4 DO
   PRINT "X= “ :X
   X=X+1
   REPEAT
END
```

Output :

```
X=      0
X=      1
X=      2
X=      3
X=      4
```

Other Flow Control Statements

TEMENOS

EXIT	-	exit from loop structure prematurely.
END	-	mandatory last statement in a program. Signifies end of program.
STOP	-	terminates execution of program and the control transfers back to the process that invoked the program.
CALL	-	executes an external subroutine.
EXECUTE	-	executes unix / jBase commands.

GOSUB	**-**	transfer the control of the program to an internal subroutine available within the program.
RETURN	**-**	used in conjunction with GOSUB. Transfers the control from the subroutine back to the next statement after GOSUB in the main program.
GOTO	-	transfers control of the program to a statement within the program. The control does not transfer back even when a return statement is encountered.

Relational Operators

compare numeric, character string, or logical data.

result of the comparison, either true (1) or false (0), can be used to make a decision regarding program flow.

relational operators are

EQ	or	=	Equal to
NE	or	# >< <>	Not Equal to
LT	or	<	Less than
GT	or	>	Greater than
LE	or	<= =<	Less than or equal to
GE	or	>= =>	Greater than or equal to

Logical Operators

They are operators that combine two or more relational expressions.Logical operators in info Basic are

AND

OR

NOT

Example

IF EMP.DESG EQ ‘MGR’ AND EMP.SAL GT 10000 THEN In the logical expression above, the logical operator ‘AND’ combines the two relational expressions ‘EQ’ and ‘GT’.

Compiler Directives

$INSERT - Inserts and compiles info Basic source code from another program into the program being compiled

REM - Identifies a line as a comment line. Same as the *, !, and $* signs

SUBROUTINE - Identifies a series of statements as a subroutine.

Built In Infobasic Functions

- LEN(e) Length of the text in expression
- COUNT(e,d) Number of occurrences of d in e
- DCOUNT(e,d) Number of occurrences of d in e, +1
- UPCASE(e) Converts e to uppercase
- DOWNCASE(e) Converts e to lowercase
- CHANGE(e,d,c) Change occurrences of d to c in e
- OCONV(e,d) Convert e into the format specified in d

Functions in Infobasic under jBase

INPUT

To Accept a Value for a Variable From User

SYNTAX :

INPUT VARIABLE, LENGTH

EXAMPLE

1. INPUT NAME,10
2. INPUT @(10,5):N,10

PROMPT

Used to change the prompt character displayed at the use of INPUT Statement.

SYNTAX :

PROMPT expr

Example :

PROMPT “#”

DATA

Used to accept data if the response to a INPUT statement is to be given without entering.

SYNTAX : DATA expr1,expr2

Example : A = 10 ; B = 22

```
DATA A,B
INPUT NO.OF.TRAINEES
INPUT AVG.AGE
```

CRT

Used to place Text & Data on the terminal

SYNTAX :

CRT @(COLUMN,ROW): TEXT / VARIABLE

Example : CRT A,B,C

CRT @(10,5): “AMERICAN EXPRESS BANK”

PRINT

This statement is used to print the value of a variable or text. Can be used in conjunction with INPUT statement to prompt the user for input.

SYNTAX :

PRINT @(COLUMN,ROW): TEXT / VARIABLE

Example :

```
PRINT @(10,12):A,B
PRINT @(10,12): “ENTER THE NAME”:
INPUT NAME,12
```

LOCATE

Used to search a string in a dynamic array and get its position. Returns the field Position only.

SYNTAX :

LOCATE <string> IN <array> {<field pos,multivalue pos>} SETTING <var>
THEN/ELSE <statements>

Example :

Y.ARRAY = 'KALLIS':@FM:'JONTY':@VM:'NICKY'

LOCATE 'KALLIS' IN Y.ARRAY<2,1> SETTING POS ELSE NULL

RESULT: 2

TRIMF Removes leading spaces from a string.
SYNTAX TRIMF(string)

TRIMB Removes trailing spaces from a string.
SYNTAX TRIMB(string)

TRIM Removes both leading &trailing spaces
SYNTAX TRIM(string)

LEN

Returns the number of Characters in a string

SYNTAX : LEN(string)

Example :

CRT LEN(“TEMENOS”)

Output : 7

Extracts a sub-string from a Character string using delimiters.

SYNTAX : FIELD(string,delimiter,occur,fields)

Example : 1

X = “A/B/C/D”

Y = FIELD(X,”/”,2,2)

RESULT : Y has a value of B/C

Example : 2

X = “1-2-3-4-5-6”

Y = FIELD(X,”-”,3,2)

RESULT : Y has a Value 3-4

INDEX

Returns the starting column position of a specific occurrence of a specified sub string within a character string.

SYNTAX:

INDEX(string,sub string,occur)

Example :

A = “AMERICAN EXPRESS BANK”
B = INDEX(A,”AN”,2)
B returns the value 19

Used to determine whether the expression is Alphabet or Not. If the expression contains any special characters or numeric value it evaluates false else true.

SYNTAX:	OUTPUT
CRT ALPHA('ABCDEF')	1
CRT ALPHA('10ABCDEF')	0
CRT ALPHA('ABC DEF')	0

This function returns the ASCII character for a given number.

SYNTAX: CHAR(expr)

Example:

X = CHAR(65)

X returns the value “A”

This function returns the ASCII value for a given character.

SYNTAX : SEQ(expr)

Example : SEQ(“A”) returns 65.

Counts number of occurrences of a specified string in a string value.

SYNTAX:

COUNT(string.expr, substr.expr)

Example :

REC=34:”-”:VM:55 :”-”: 88

R=COUNT(REC,@VM)

Result : R has a Value of 2.

Example

A=“TEMENOS”
CRT COUNT(A,’S’)
Output : 1

User Defined functions in InfoBasic under jBase

DEFINING FUNCTIONS

TEMENOS

- Functions can also be defined in jBasic.
- Functions can take in any number of parameters but returns only one value.
- RETURN is used to pass the result to the calling Program.

Example:

STEP 1:

Define a Function that accepts two values and return the result in a variable.

```
FUNCTION TEST.FUN(A,B)

C = A * B

RETURN(C)

END
```

STEP 2:

Compile and Catalog the Function.

Example:

STEP 3: Write a Program that calls the Function(TEST.FUN) and Prints the Result.

```
PROGRAM FUN.CALLING.PRG

A = 5 ; B =10

DEFFUN TEST.FUN(A,B)

RESULT = TEST.FUN(A,B)

PRINT “RESULT : “:RESULT

END
```

STEP 4: Compile and Catalog the Program

STEP 5:Run the Program

Result : RESULT : 50

jDB – Debugger tool of jBase

JDEBUGGER COMMANDS

TEMENOS

- To invoke Jdebugger, include DEBUG in the program/subroutine
- Alternatively at runtime with –jd option before programname
- S debugs each statement
- /VAR.NAME displays the contents stored in the variable
- C to continue execution of program without debugging
- Q to Quit debugger

Core Subroutines in T24

- Used to open a File for reading or writing purposes.
- It has two parameters passed.

SYNTAX :

CALL OPF(File name,File pointer)

Filename : Name of the File (Example : 'FBNK.CUSTOMER')

File pointer : Path of the file

Example :

FN.CUSTOMER = 'FBNK.CUSTOMER'

F.CUSTOMER = ''

CALL OPF(FN.CUSTOMER,F.CUSTOMER)

NOTE:

File pointer is initialized with null value and at the time of execution it will be assigned to the path of the file.

F.READ

- Used to read a record from a file which is already opened using OPF.
- F.READ has five parameters.

SYNTAX:

CALL F.READ(Filename,record.id,dynamic.array,File.var,Error.var)

Filename : File Name

Record.id : ID of the record to be read

Dynamic.array : Dynamic array that will hold the read record

File.var : File Path

Error.var : Error Variable

Example

```
FN.CUSTOMER = 'FBNK.CUSTOMER'
F.CUSTOMER = ''
R.CUSTOMER = ''
CALL OPF(FN.CUSTOMER,F.CUSTOMER)
CALL F.READ (FN.CUSTOMER,Y.CUSTOMER.ID,R.CUSTOMER,
                    F.CUSTOMER,Y.CUS.ERR)
```

F.WRITE

- Used to write details on to a record in a file.
- It has 3 parameters passed.
- Before writing the values in a record, open the file and read the record.

SYNTAX :

CALL F.WRITE (Filename, Record.id, Dynamic array)

Filename : file name

Record.id : Record to be written

Dynamic array : Array that holds the values to be written on a record

<Initialize variables FN.CUSTOMER,F.CUSTOMER,....>

<Open the file using OPF>

<Read the record using F.READ>

<Assign the value to the dynamic array which we are going to write>

R.CUSTOMER<EB.CUS.SHORT.NAME> = 'ABC CORPORATION'

CALL F.WRITE (FN.CUSTOMER,Y.CUSTOMER.ID,R.CUSTOMER)

EXAMPLE 1

Write a subroutine that will display the details (Id, Mnemonic and Nationality) of a customer whose id is 100037

```
 SUBROUTINE CUST.READ
$INSERT I_COMMON
$INSERT I_EQUATE
$INSERT I_F.CUSTOMER
    GOSUB INIT
    GOSUB OPENFILES
    GOSUB PROCESS
    RETURN
INIT:
    FN.CUS = "FBNK.CUSTOMER"
    F.CUS = "  "
    Y.CUS.ID = '100037'
    Y.NATIONALITY = "  "
    Y.MNEMONIC = "  "
    R.CUSTOMER = " "
    Y.ERR = "  "
    RETURN
```

```
OPENFILES:
    CALL OPF(FN.CUS,F.CUS)
    RETURN

 PROCESS:
    CALL F.READ(FN.CUS, Y.CUS.ID , R.CUSTOMER , F.CUS, Y.ERR)
    Y.NATIONALITY = R.CUSTOMER<EB.CUS.NATIONALITY>
    Y.MNEMONIC = R.CUSTOMER<EB.CUS.MNEMONIC>
    PRINT “CUSTOMER ID IS :”:Y.CUS.ID
    PRINT "NATIONALITY IS :":Y.NATIONALITY
    PRINT "MNENOMIC IS :":Y.MNEMONIC
    RETURN
 END
```

EB.READLIST

- To read a set of records from a file we use this core routine.
- It has 5 parameters passed.

SYNTAX

CALL EB.READLIST(1,2,3,4,5)

1 : Select Query.

2 : List variable that contains only the ID of the selected records.

3 : Id of the SAVEDLISTS file (Optional)

4 : No of Records selected (Total Count)

5 : Return code

Example <Initialise File name FN.CUSTOMER>

SEL.CMD = “SELECT “:FN.CUSTOMER

CALL EB.READLIST(SEL.CMD,SEL.LIST,’’,NO.OF.RECORDS,RET.CODE)

REMOVE

REMOVE is the Function that is used to extract a value from a dynamic array.

SYNTAX:

REMOVE <var> FROM <array> SETTING <set var>

Var : variable which holds the extracted string

Array : Dynamic array from which the string is to be extracted.

Set var : Delimiter by which string is extracted from array.

(2 – FM, 3 – VM, 4 – SM, 0 – End of array)

EXAMPLE 2

Write a subroutine that will changes the Account officer from 1 to 2 and display the details (Customer, Mnemonic, Old Acct officer and New Acct officer) for all customers.

```
 SUBROUTINE CUST.READ.WRITE

$INSERT I_COMMON
$INSERT I_EQUATE
$INSERT I_F.CUSTOMER
    GOSUB INIT
    GOSUB OPENFILES
    GOSUB PROCESS
    RETURN
INIT:
    FN.CUS = "FBNK.CUSTOMER"
    F.CUS = "  "
    Y.CUS.ID = ‘’
    Y.NATIONALITY = "  "
    Y.MNEMONIC = "  "
    R.CUSTOMER = " "
    Y.ERR = "  "
    RETURN
```

```
OPENFILES:
    CALL OPF(FN.CUS,F.CUS)
    RETURN

 PROCESS:
    SEL.CMD = 'SELECT  ':FN.CUS
    CALL EB.READLIST(SEL.CMD,SEL.LIST,'',NO.OF.REC,RET.CODE)
    LOOP
    REMOVE Y.CUS.ID FROM SEL.LIST SETTING POS
    WHILE  Y.CUS.ID:POS
    CALL F.READ(FN.CUS, Y.CUS.ID , R.CUSTOMER , F.CUS , Y.ERR)
    Y.OLD.ACCT.OFFICER = R.CUSTOMER<EB.CUS.ACCOUNT.OFFICER>
    IF Y.OLD.ACCT.OFFICER EQ '1' THEN
          Y.NEW.ACCT.OFFICER = '2'
          R.CUSTOMER<EB.CUS.ACCOUNT.OFFICER> = Y.NEW.ACCT.OFFICER
    END
```

```
    Y.DET = Y.CUS.ID:' * ':Y.MNEMONIC:' * ':Y.OLD.ACCT.OFFICER:' * ': Y.NEW.ACCT.OFFICER
    Y.NEW.ACCT.OFFICER = ''
    PRINT Y.DET
    REPEAT
    RETURN
END
```

JOURNAL.UPDATE

- JOURNAL.UPDATE is a core routine that updates the F.JOURNAL when a transaction happens.
- When the system is in Online mode and if we are using F.WRITE, then it will write the data on to the cache and not to the disk. So a Call to Journal.Update is required after a Write Statement.
- When the system is in Batch mode, system takes care of writing on to the disk. Hence call to Journal.Update is not required.

STORE.END.ERROR

- Used for displaying error messages alongside field while validation processing
- Used to display error messages stored in ETEXT
- AF stores the field value
- AV stores the multi-value
- AS stores the sub-value
- Assign the field,multi-value and sub-value as per the requirement to display error message alongside the field no.

Example 2

Write a subroutine that will display the details
(Id, Mnemonic and Nationality)of a customer whose id is 100069

Algorithm

- Step 1. Open the Customer File
- Step 2. Read the Customer file and extract the record with id 100069
- Step 3. From the extracted record obtain the mnemonic and nationality
- Step 4. Display the customer id,mnemonic and nationality.

Open A File

- Use the command OPEN

OPEN FBNK.CUSTOMER…….

But…….

Open A File (Cont.)

- OPF – Open File

CALL OPF(FN.CUS,F.CUS)

FN.CUS = ‘F.CUSTOMER’ (File Name)

F.CUS = ‘’ (File Path)

Read A File

- Use the Globus subroutine

CALL F.READ(1,2,3,4,5)

1 - File name

2 - ID of the record to be read

3 - Dynamic array that will hold the read record

4 - File Path

5 – Error Variable

CALL F.READ(FN.CUS,”100069”,R.CUSTOMER,F.CUS,CUS.ERR1)

F.READ always checks if the record is in cache. If yes, fetches the record from the cache, else retrieves the record from the databse.

Record Returned By F.READ

Contents of R.CUSTOMER

DAOHENGBK؟DAO HENG BANK INC؟DAO HENG BANK INC؟؟119 ASIAN MANSION 209
DELA ROSA ST؟ LEGASPI VILLAGE MAKATI CITY MAN PH ؟؟؟؟ 1111؟
90؟؟8100؟999؟PH؟4؟ PH؟ 20000101 ؟؟؟؟؟ 20000101
؟؟؟1؟؟؟؟؟؟؟؟؟؟؟؟؟؟؟1؟18_RICKBANAT1ÿ28_ANDREABARNES1؟00061204 2؟18_RI
CKBANAT1؟US0010001؟|1؟؟

R.CUSTOMER<1>

R.CUSTOMER<15>

What happens after an upgrade?

I_F.CUSTOMER File

```
0001: * Version 6 15/05/01  GLOBUS Release No. G12.0.00 29/06/01
0002: * File Layout for CUSTOMER Created 15 MAY 01 at 05:02pm by bhatiab
0003: *      PREFIX[EB.CUS.]      SUFFIX[]
0004:        EQU EB.CUS.MNEMONIC TO 1,          EB.CUS.SHORT.NAME TO 2,
0005:              EB.CUS.NAME.1 TO 3,              EB.CUS.NAME.2 TO 4,
0006:              EB.CUS.STREET TO 5,        EB.CUS.TOWN.COUNTRY TO 6,
0007:       EB.CUS.RELATION.CODE TO 7,        EB.CUS.REL.CUSTOMER TO 8,
0008:    EB.CUS.REVERS.REL.CODE TO 9,             EB.CUS.SECTOR TO 10,
0009:   EB.CUS.ACCOUNT.OFFICER TO 11,      EB.CUS.OTHER.OFFICER TO 12,
0010:          EB.CUS.INDUSTRY TO 13,             EB.CUS.TARGET TO 14,
0011:       EB.CUS.NATIONALITY TO 15,   EB.CUS.CUSTOMER.STATUS TO 16,
0012:         EB.CUS.RESIDENCE TO 17,      EB.CUS.CONTACT.DATE TO 18,
0013:        EB.CUS.INTRODUCER TO 19,              EB.CUS.TEXT TO 20,
0014:          EB.CUS.LEGAL.ID TO 21,  EB.CUS.REVIEW.FREQUENCY TO 22,
0015: EB.CUS.BIRTH.INCORP.DATE TO 23,   EB.CUS.GLOBAL.CUSTOMER TO 24,
0016: EB.CUS.CUSTOMER.LIABILITY TO 25,         EB.CUS.LANGUAGE TO 26,
0017:  EB.CUS.POSTING.RESTRICT TO 27,      EB.CUS.DISPO.OFFICER TO 28,
0018:          EB.CUS.POST.CODE TO 29,            EB.CUS.COUNTRY TO 30,
0019:                EB.CUS.BOOK TO 31,         EB.CUS.CONFID.TXT TO 32,
0020:        EB.CUS.RESERVED07 TO 33,         EB.CUS.RESERVED06 TO 34,
0021:        EB.CUS.RESERVED05 TO 35,         EB.CUS.RESERVED04 TO 36,
0022:        EB.CUS.RESERVED03 TO 37,         EB.CUS.RESERVED02 TO 38,
0023:        EB.CUS.RESERVED01 TO 39,           EB.CUS.LOCAL.REF TO 40,
0024:          EB.CUS.OVERRIDE TO 41,      EB.CUS.RECORD.STATUS TO 42,
0025:           EB.CUS.CURR.NO TO 43,           EB.CUS.INPUTTER TO 44,
0026:         EB.CUS.DATE.TIME TO 45,         EB.CUS.AUTHORISER TO 46,
0027:           EB.CUS.CO.CODE TO 47,           EB.CUS.DEPT.CODE TO 48,
0028:       EB.CUS.AUDITOR.CODE TO 49.   EB.CUS.AUDIT.DATE.TIME TO 50
```

Display Parts Of A Record

Y.MNEMONIC = R.CUSTOMER<EB.CUS.MNEMONIC>

Y.NATIONALITY = R.CUSTOMER<EB.CUS.NATIONALITY>

Display Parts Of A Record(Cont.)

CRT “Customer Id: “:Y.CUS.ID

CRT “Customer Mnemonic: “:Y.MNEMONIC

CRT “Customer Nationality: “:Y.NATIONALITY

Solution 2

```
*Subroutine to display the details of customer 100069
SUBROUTINE CUS.DISPLAY.DETAILS
$INSERT I_COMMON
$INSERT I_EQUATE
$INSERT I_F.CUSTOMER
GOSUB INIT
GOSUB OPENFILES
GOSUB PROCESS
RETURN
INIT:
      FN.CUS = 'F.CUSTOMER'
      F.CUS = ''
      Y.CUS.ID = 100069
      Y.MNEMONIC = ''
      Y.NATIONALITY = ''
      R.CUSTOMER = ''
      CUS.ERR1 = ''
RETURN
```

```
OPENFILES:
    CALL OPF(FN.CUS,F.CUS)
RETURN
PROCESS:
    CALL F.READ(FN.CUS,Y.CUS.ID,R.CUSTOMER,F.CUS,CUS.ERR1)
    Y.MNEMONIC = R.CUSTOMER<EB.CUS.MNEMONIC>
    Y.NATIONALITY = R.CUSTOMER<EB.CUS.NATIONALITY>
    CRT “Customer Id: “:Y.CUS.ID
    CRT “Customer Mnemonic: “:Y.MNEMONIC
    CRT ‘Customer Nationality: “:Y.NATIONALITY
RETURN
END
```

Thank You

TEMENOS

The Banking Software Company