**Chapter: N/A**

**Assignment: SDEV 120 FINAL PROJECT**

**Name: Ricky Howe**

**EXECUTIVE DESCRIPTION:**

**“Automated Teller” Vers. 1.0**

This program is called “Automated Teller”. The program is ready for Beta testing.

This program is designed to be a fully functional Automated Teller Machine, or ATM for short. The Beta version includes only 4 customers as a proof of concept.

**INPUT**:   
 This project includes a total of 6 functions that we need to address, 5 being teller functions. The machine needs to be able to login to the user’s account using a username and password, accept a deposit, process a withdrawal, perform a balance inquiry, transfer a balance from one account to another, and log out.

Input variables from the customer will be limited to prompting the user for their username and password, and custom amounts not listed on the menus. The user will make all other selections from prebuilt menus utilizing the keypad on the machine.

Logging into the machine will be performed by prompting the user to enter their login credentials, and then checking those credentials against the array built into the machine.

The menus will be created utilizing case structure. Logout will always be 9 on all submenus. Selection of 9 will end main program.

Users will have three chances total to get their username and password correct. Failure to do so will end main program and require starting fresh.

If a user wishes to login, deposit money, perform a balance inquiry, or transfer funds and needs to use more than one function, a final menu has been implemented asking the user if they would like another transaction. If yes is selected, the final menu will call the main menu again. If no is selected, the program will terminate.

**PROCESS/ASSIGNMENT STATEMENTS**:

This section is broken into three parts. As stated previously this program is merely a proof of concept so these are not real customers, nor is this real customer data. The following snips are the module where all variables are initialized, and the module creating three databases. One contains usernames, one contains passwords, and the final contains both accounts for the corresponding customers. These arrays were offset for ease of reading.

Initialize

Set Checking = 0.0

Set Savings = 0.0

Set Count = 0

Set User = “a”

Set Password = “a”

Set Trans = 0

Set Log = 0

Set Deposit = 0

Set Transfer = 0.0

Set Selection = 0

Database

Set Usr[0] = rbrown

Set Usr[1] = lwhite

Set Usr[2] = mblack

Set Usr[3] = rhowe

Set Pwd[0] = blue123

Set Pwd[1] = red456

Set Pwd[2] = green789

Set Pwd[3] = C04747277

Set Acct[0, 0] = 2500.00

Set Acct[0, 1] = 35.00

Set Acct[1, 0] = 500.00

Set Acct[1, 1] = 1250.00

Set Acct[2, 0] = 750.00

Set Acct[2, 1] = 200.00

Set Acct[3, 0] = 600.00

Set Acct[3, 1] = 900.00

**Process/Output Module**:

This segment of code is used to verify the validity of the login process and limit the customer to three attempts total.

Declare Index as integer

Declare Found as integer

While Log < 3

Write “Thank you for banking with us. Please enter your username.”

Input User

Set Log = Log + 1

Set Index = 0

Set Found = 0

While (Found == 0) AND (Index < 4)

If Usr[Index] == User Then

Set Found = 1

End If

End While

If Found == 0 Then

Write “Invalid User Name.”

Else

Write “” + UserName + “, please enter your password.”

End If

Input Password

Set Log = Log + 1

Set Index = 0

Set Found = 0

While (Found == 0) AND (Index < 4)

If Pwd[Index] == Pass Then

Set Found = 1

End If

End While

If Found == 0 Then

Write “Invalid Password”

Else

Write “Login Successful!”

End If

End While

If Log >= 3 Then

End Main Program

End If

The next snip shows, upon successful login, how I was able to populate only the customer’s account information into the variables checking and savings.

If Index = 0 Then

Set Savings = 2500.00

Set Checking = 35.00

If Index = 1 Then

Set Savings = 500.00

Set Checking = 1250.00

If Index = 2 Then

Set Savings = 750.00

Set Checking = 200.00

If Index = 3 Then

Set Savings = 600.00

Set Checking = 900.00

End If

End If

End If

End If

End Login Module

This snip below shows the main menu of the program. Depending on which selection the user makes, a second menu will appear asking them to choose between three options. This menu will offer the choices of LogOut, Checking, and Savings. It will also loop for three iterations if the user chooses to do so, offering multiple chances in case the user needs a second or third go at that particular option.

Write “Please choose an option from the list below.”

Write “Logout \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* 9”

Write “Deposit \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* 1”

Write “Withdrawal \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* 2”

Write “Balance Transfer \* \* \* \* \* \* \* \* \* \* \* \* 3”

Write “Check Account Balance \* \* \* \* \* \* \* \* \* 4”

The menu asking the user which account to apply the function to will always look similar to this screen.

Write “For which account?”

Write “Checking. . . . . . . . 1”

Write “Savings . . . . . . .. . .2”

Write “Logout . . . . . . . . . . 9”

The final menu will look like this.

Write “Would you like another transaction?”

Write “Yes \*\*\*\*\*\*\* 1”

Write “No \*\*\*\*\*\*\*\* 2”

In conclusion, there isn’t a lot to this program, but getting it all to work together takes forever and implementing the menus takes up a lot of room in the lines of code. This program has all of the processing and output on the same screen, so an output module is unnecessary.

**Executive Overview**:

This proof of concept pseudocode ATM shows that not all scripting has to be fancy. While other companies have pushed the limits of what their machines can do graphics wise and through implementing complicated code, we have shown that a straightforward approach can be just as effective while utilizing fewer physical resources. This means our code can run on less capable machines, meaning less expensive machines.

**The program completed in only 316 lines of code.**