APS 105S – Computer Fundamentals

Assignment #3: Modules

Fall 1996

(To be completed in your lab period Oct. 3/4)

Objective: Study and develop programs that make use of procedures and functions.

1 Procedures and Functions

Procedures and functions are used to improve the organization of your programs by separating individual tasks into "sub-programs". To illustrate the new concepts associated with procedures and functions, the quadratic equation solver program of Lab 1 has been rewritten and error checking added. The modified program is depicted in Figure 1. Please study it so that you fully understand it.

The `getvalue` procedure is used to prompt the user for input. The same procedure is used for all three coefficients.

The parameters of this procedure are declared as `"name:string; var coefficient:real"`. What is the difference between having `var` written before a parameter, and not having it? How would the procedure change if `var` was not written before `coefficient`?

What is the basic difference between a procedure and a function? In the function `discriminant` what does the `:real` at the end of the first line of the function declaration signify?

What are the global variables of the program? Identify some local variables. Could the `solvequad` procedure be rewritten without parameters? How?

2 A Bank Account Management Program

You are to construct a program that is to be used to keep track of the balances of two bank accounts: a savings account and a chequing account. The program will allow for deposits, withdrawals and interest payments to be made on either account.

To begin with, the program will initialize the balance of each account to $0. The program will then continuously prompt the user to select an action. The choices will be to make a deposit, to make a withdrawal, to calculate and deposit interest, or to quit the program. If either of the first three options are selected, the user is asked to which account the action is


```
function discriminant(a,b,c:real):real
  result b**2-4*a*c
end discriminant

procedure getvalue(name:string; var coefficient:real)
  put "Enter the value of ", name, ": "..
  get coefficient
end getvalue

procedure solvequad(a,b,c:real)
  var dis:real
  if a=0 then
    if b=0 then
      put "The equation has no solution"
    else
      put "The solution of the equation is: ", -c/b
    end if
  else
    dis:= discriminant(a,b,c)
    if dis>0 then
      var root1:real:=(-b+sqrt(dis))/(2*a)
      var root2:real:=(-b-sqrt(dis))/(2*a)
      put The roots of the equation are: ", root1, " and ", root2
    elseif dis=0 then
      var root:real:=-b/(2*a)
      put "The solution of the equation is: ", root
    else
      var realroot:real:=-b/(2*a)
      var imagroot:real:=sqrt(abs(dis))/(2*a)
      put "The roots of the equation are: ", realroot, " + i*",..
      put imagroot, " and ", realroot, " - i*", imagroot
    endif
  endif

var A,B,C:real
getvalue("a",A)
getvalue("b",B)
getvalue("c",C)
solvequad(A,B,C)
```

Figure 1: The quadratic equation solver program

```
     Bank Account Management Program

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? D
(S)avings or (C)hecking? s
Amount of deposit? 1500
Savings account balance: $1200.00

Interaction: (W)ithdrawal, (D)eposity, (I)nterest or (Q)uit? I
(S)avings or (C)hecking? c
Chequing account balance: $0.00

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? d
(S)avings or (C)hecking? c
Amount of deposit? 2000
Chequing account balance: $2000.00

Interaction:(W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? I
(S)avings or (C)hecking? s
1.5% interest applied to $0.00 = $0.00
Savings account balance: $1200.00

Interaction: (W)ithdrawal, (D)eposit, (I)nterest or (Q)uit? q
Savings account balance: $1200.00
Chequing account balance: $2000.00
End of Program
```

Figure 2: Bank Account Management Program

to apply. While deposits, withdrawals and interest calculations could be done for either of the two accounts, only one module will be written for each action.

The deposit module will accept a deposit amount, verify that it is a positive value, then update the appropriate account balance. The withdrawal module will accept a positive withdrawal value, but will not permit the withdrawal if the amount is greater than the balance of the account. It will update the account balance appropriately.

The interest module is to calculate the interest earned on the account and deposit it in the account. Separate interest rates for the two accounts will be hard-coded in the program as constants. To calculate the interest, apply the interest rate to the lowest account balance since the last time interest was applied. The user may ask for interest to be credited as often as desired.

An example of the use of the program is shown in Figure 2. The program should accept either upper or lower case letters to specify termination, e.g., either Q or q for (Q)uit.

3 Optional Section – Exception Handling

I assume the input procedure of your banking program checked the user input for errors. Well-engineered software will always anticipate problems that might come up during the execution of the program, and will always be prepared to handle them. In your program, you may have used if-then statements to detect improper input for the withdrawal and deposit procedures. However, some problems that might come up during the execution of the program may not be resolved using such program control statements. For example, if your program prompts the user to enter a real value, but s/he enters a non-numeric character, this will result in the generation of an exception, and if you have not specified in your program how to deal with an exception, the program will terminate.

A formal method of dealing with exceptions is provided in OOT by way of the exceptionHandler mechanism. Full instructions on the use of the exceptionHandler are provided in the OOT reference manual. Extend your interest program by adding handler blocks to deal with exceptions that result from improper input. If the user provides a non-numeric input for the deposit or withdrawal amounts, the exception handler should post a warning to the user and request for input again.