BACS 287 VbProj5.DOC

3/29/16 3:21 PM

**Visual Basic Project #5**

Date Due: 4/12/16

The purpose of this project is to demonstrate your mastery of form design, basic programming statements, functions, and modules in a realistic application. For full credit you must build the project as described below and it must execute correctly. Any code you write must be properly documented as discussed in class. This project should be a group project with 2 GROUP MEMBERS. Please start early and ask questions if you need help. Good Luck and start early!

**Project Components:**

For full credit, this project requires that you work with the following:

- A form with textboxes, labels, radio buttons, command buttons, and groupbox containers

- A module

- Selection statements

- Sequence constructs

- Iteration constructs (potentially, depending on your design)

- Error checking code

- Built-in financial functions

**Deliverables:**

You will turn in the following:

- A printout of the finished form along with design document (pseudocode or flowchart)

- A printout of the code (with correct variable names, appropriate comments, correct indention and

‘dead code’ removed)

- The program project folder (which includes an executable version of your program)

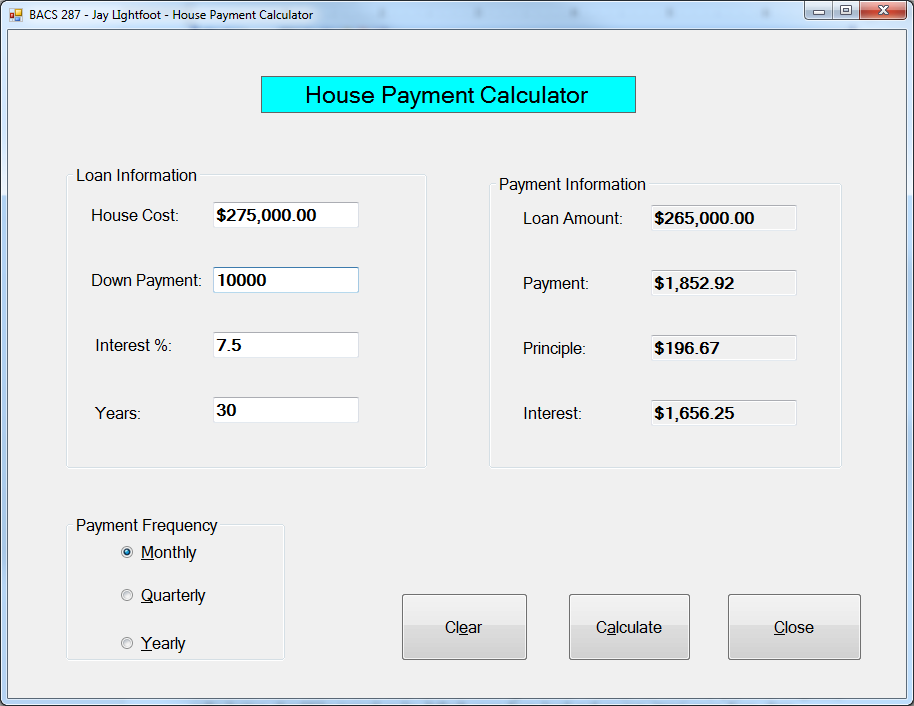
- Your program is REQUIRED to include a module and 3 user-defined functions. In addition, data

must be passed as arguments to the functions and returned as a return value for full credit.

Your name and class identification information should be on all work (including the title of the main form). If 2 people work on the project, both names must be included. The project should be stapled together in a logical order. Please use the folder that was provided to turn in your work.

**Step 1: Create the Calculator Form**

You will create a form that looks like the one shown on the following page. This form will accept house loan information as input and then calculate the payment, principal portion of the payment, and interest portion of the payment for the house loan. Your program will also use a module that contains 3 user-defined functions that you write. Each function will accept arguments from the calling program (i.e., the main form). Following that, your user-defined functions will call the built-in Visual Basic financial functions to produce the final answers. The data from your user-defined functions will be returned to the calling program and subsequently placed on the main form (formatted as currency with 2 decimals). Details of how the program operates internally are given in the text that follows. Make sure that follow the instructions carefully to ensure full credit on your work.



This form contains 4 textboxes that allow user input. These are the “house cost”, “down payment”, “interest %”, and “years” boxes shown above. All other boxes are labels. The “Loan Information” controls are all contained in a groupbox control. Likewise, the “Payment Information” controls (where the output goes) are included in a 2nd groupbox control. Finally, the 3 radio button controls on the lower left are all contained in a 3rd groupbox control labelled “Payment Frequency.” Note, the “down payment” box will hold the amount that the user plans to use to offset the loan. So, if the house costs $100,000 and they have a $10,000 down payment, the loan calculations will be for $90,000. Also note that the down payment box is initially set to 0 upon form load and clear. Finally, the “Monthly payment” frequency radio button should be turned on by default upon form load and reset to ‘true’ upon form clear. When the program starts, the focus should be in the “House Cost” textbox. Pressing the tab button should move between the controls in the following order: house cost, down payment, interest %, Years, Monthly, calculate, and clear. When clear is pressed all numeric data is cleared out, the down payment is set to 0, and the focus goes back to the “House cost” textbox.

The “Calculate” button should take the data currently found in the “Loan Information” box and error check for common errors (e.g., non-numeric values, negative values, unreasonable interest rates, down payment greater than loan amount, …). If errors are found, specific error messages should be displayed, the offending field is cleared out, and the focus is returned to the field that caused the error. Calculations should not proceed when bad data is input. In addition, the “payment frequency” selection should be used to scale the interest rate and the total number of payments input to the correct values for calculations.

The program will then call the 3 user-defined functions that you wrote. These functions must be in a module. Each must accept parameter data from the form needed to perform the calculations. (In other words, public variables are not allowed). Within these user-defined functions, you can include additional error checking as necessary. If you do, make sure that the error messages are meaningful and that calculations do not proceed with bad data. Next, each of your user-defined functions will call one of the 3 built-in VB functions to calculate the payment, principal portion of the payment, and interest portion of the payment. The syntax for the functions is shown below.

dblX = **PMT**(rate, periods, loanAmt) [[1]](#footnote-1)

dblY = **PPMT**(rate, num\_period, periods, loanAmt)

dblZ = **IPMT**(rate, num\_period, periods, loanAmt)

Where:

**rate** - interest rate per period. Thus, 8% per year with monthly payments would have a rate

value equal to .08/12 or .0066 per month.

**periods** - total number of payment periods in the loan. Thus a 4 year loan with monthly

payments would have 4\*12 or 48 payment periods. Note that the rate and periods

values must be expressed in the same unit (i.e., months, quarters, ...).

**num\_period[[2]](#footnote-2)** - payment period in the range of 1 to periods (defined above). For the first

period in the loan, the num\_period value would be 1, the second period would have a 2

and so on. This value cannot be greater than the periods value.

**loanAmt** - the total amount borrowed. This number must be negative for standard loan

results to be generated.

For more information about these built-in functions, see the VB on-line help or consult the web. One thing you should note; despite the fact that these functions involve money, they use the double datatype (not decimal) for all input and output values. The values returned from these functions should be placed in the appropriate labels in the “payment information” portion of the form. The 3 financial functions must be contained in user defined functions that you write. These must be housed in a module. Use what we covered in class lecture to make all the form, function, and module components work together.

**Step 4: Test and Print out screen shot and code**

Test the components of your project. Make sure that it produces accurate results (use the test data below). Try different ‘payment frequency’ values and make sure that the results are correct (and logical). Be sure to check that all your error messages work properly. When all components work together, make sure that you have saved the project properly and printed everything needed.

Start working on this as soon as possible and ask questions if you need help.

**Good Luck!**

**Test Data: You can use this to check your program**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Loan | Down Pmt | Int % | Years | Frequency | Loan amt | Payment | Principle | Interest |
| 125,000 | 10,000 | 5% | 30 | monthly | 115,000 | 617.34 | 138.18 | 479.17 |
| 125,000 | 10,000 | 5% | 30 | quarterly | 115,000 | 1,855.35 | 417.85 | 1,437.50 |
| 65,000 | 0 | 7% | 15 | monthly | 65,000 | 584.24 | 205.07 | 379.17 |
| 275,000 | 50,000 | 5.25 | 10 | monthly | 225,000 | 2,414.06 | 1,429.69 | 984.38 |

1. Note that the return value of the functions is also double data type. Your code needs to be written to expect this. [↑](#footnote-ref-1)
2. Note, you will merely use the value of 1 for the num\_period argument in this program since this is a static view of the payment. Were you creating an amortization table, this value would vary from 1 to the total number of periods in the loan. [↑](#footnote-ref-2)