# Lab 3

## User Defined Functions

Skills Required

* Create Functions
* Include Headers and other files
* Loops (while, for), Conditional (if, switch), Datatypes, assignment, etc.

Requirements

* You must use at least one if else
* You must use a switch statement conditional

1. Create or open Lab 3 Project in Visual Studio 2019
   1. Create a Lab03.cpp in source files.
   2. Remember to include iostream. Including string may also be helpful.
2. Assignment

This assignment will keep track of a customer’s bank account balance.

* 1. Create Functions
     1. Before the main function in the program, create a new function called deposit that has 2 parameters; account and amount. Both are float. This function will simply add the amount to the account value. This function doesn’t return any value, so the return should be void. Your function will probably resemble this function.

void deposit(float account, float amount) {

account = account + amount;

}

* + 1. In the main function call the deposit function and pass the amount and depositAmount to the function. After the function call output the new amount. Did the amount value change?
    2. The reason the amount value did not changed is that the amount we passed in was passed in by value. This means C++ copied our value into a new memory area. You may want to place a breakpoint inside the function and step through the program to see how this happens. In order to pass by reference, we simply put an & before the parameter declaration. Make the change and try your program again, it should now output the updated amount value. Your function now probably looks like the following.

void deposit(float& account, float amount) {

account = account + amount;

}

* 1. Pre-Declaring Functions
     1. Copy and paste the deposit function to below the main function. Now try to run your program. You should get a build error.



* + 1. Because the function is below main, as the parser works from top to bottom it can’t find a deposit function and fails. Add the function definition above main, this lets the compiler that there is a function in this module that these given parameters

void deposit(float& account, float amount);

* + 1. Compile and run the program. You’ll notice that it builds and runs correctly again.
  1. Separate files.

Having all your code in one file can become messy. Putting them in separate files helps re-use and organization.

Let’s add a new file to the source files. You can do this by right clicking on Source Files and select add->New Item. Call it checkingmodule.cpp, and move our deposit function into this module.

Create a header file. Right click on Header Files and select add->New Item. Change the type of file to Header and call it checkingmodule.h

Move our deposit declaration that we did previously into the checkingmodule.h file.

* + 1. Now try to run your project. You’ll get a deposit identifier not found error. This is because we haven’t included our file header in lab03.cpp
    2. Include the checkingmodule.h file

#include <iostream>

#include <string>

#include "checkingmodule.h"

using namespace std;

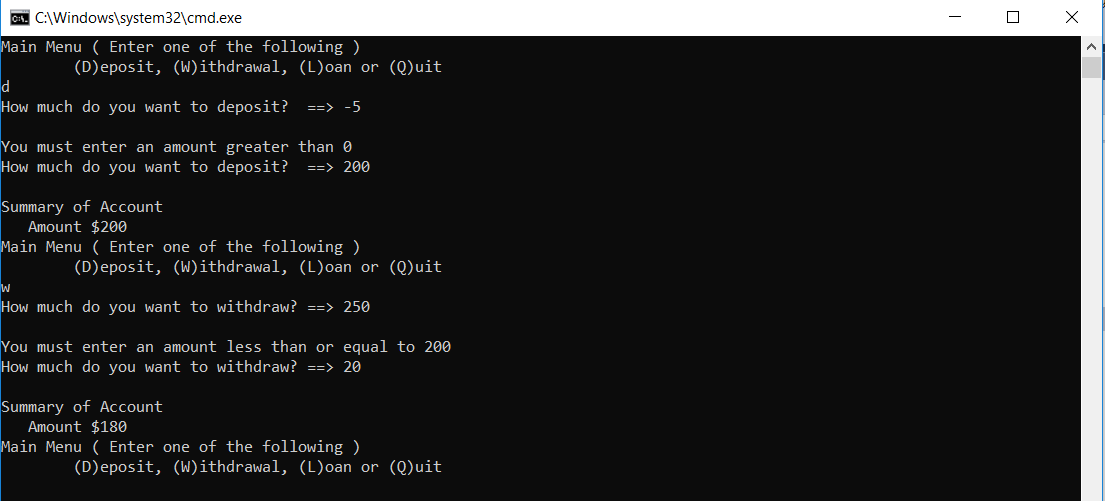
* + 1. The program should work again.
    2. Congratulations, you’ve learned how to create a function, pass by value and pass by reference, and use separate files for functions. Now let’s use these skills to build the solution.
  1. Continued development.

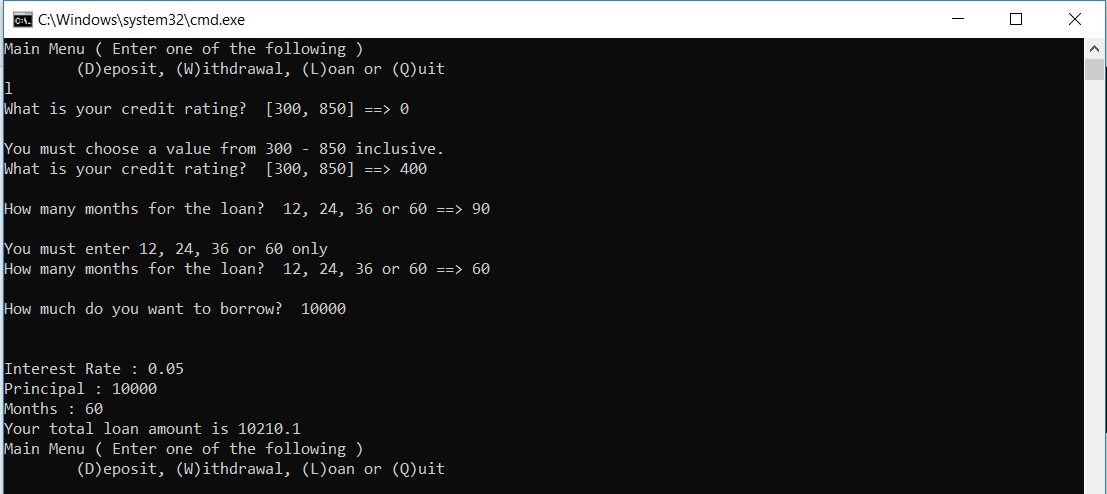
Create and test the following functions in the checkingmodule.cpp file. Remember to add the function declarations to the header file. Make sure you test each function.

* + 1. MainMenu function takes no arguments, but returns a single character either D, W, L, or Q. This function should continually output the options to the user, once the user enters D, W, L or Q then it returns the valid choice. You will need to add the include iostream, string and use namespace std at the top of checkingmodule.cpp.
    2. GetValue function takes one argument: a string that is displayed to the user to ask them for input. The function returns a float. It should continually ask the user for an amount by showing them the prompt passed in, it must be greater than 0. It should return the selected amount.
    3. GetValue overload function takes two arguments a string to display to the user to ask for input and an upper limit. It returns a float. It should ask the user for an amount to withdraw. The amount must be greater than 0 and less than the argument. It should then return the selected amount
    4. GetInterestRate function takes one argument an integer which is their credit score, and returns a float the amount of interest they will be charged. If the credit score is 500 or lower, they will be charged 5% interest, or 0.05. If the credit score is greater than 500 and less than or equal to 700 then they will get 3% interest. If their credit score is greater than 700, then the interest rate is 1%.
    5. Withdraw function that takes 2 arguments, an amount value passed by reference that is the users account balance, and amount to withdraw. It should not return any value, but should only decrement the account balance by the amount.
    6. GetLoanMonths function that takes no arguments and continually prompts the user for number of months for the loan until they give a valid #. Valid loan amount in months is 12, 24, 36 and 60.
    7. GetCreditRating function takes no arguments and returns an integer from 300 to 850 inclusive. It should prompt until the user chooses a valid value.
    8. CalculateLoanAmount function takes arguments, Principal a float, interest rate a float, and the number of months of the loan as an integer, and returns the calculated loan amount. The formula for Amount is

You can use the pow function, which takes 2 arguments. pow(2, 5) returns 2 to the 5th power.

* + 1. OutputAccount function takes one argument an amount, and outputs the amount in the account. (Get creative in making it look nicel.)
  1. Once the functions are written and you have tested to make sure they work correctly, write a program to run until the user chooses quit, and keeps track of the amount they have in their account. It should start out with the account amount at zero.





1. Save all your files; if you’re using GitHub, commit your changes (if you haven’t already) and push them to GitHub.
2. Delete the contents of the Debug directory and upload your zip file to Canvas by the deadline; OR, post the link to your GitHub repository for this project.