Project One: How Much Car Can You Afford?

Out: Sep. 23, 2023; Due: Oct. 8, 2023

Introduction

This project will give you experience in writing, compiling, and debugging a simple C++ program. You will gain experience with header files and multiple-file compilation. We expect the program itself to be straightforward for any student who has successfully completed the prerequisites for this course.

Buying a Car

Suppose that you want to buy yourself a new car. The key question, of course, is what car to buy? As any good salesperson will "helpfully" ask you, one way to answer this problem is to figure out how much you can pay each month. While you may not wish to reveal your answer to this question to the salesperson, it is a useful metric to keep in mind when purchasing a car. To assist you in making this calculation, you are going to write a simple program that tells you how much you can afford to pay based upon a monthly payment at a given interest rate for a fixed number of years. For instance, if you were to specify that you are willing to pay $500 per month for 5 years at an interest rate of 6%, you could afford to take out a loan of $25,862.78 to pay for a car.

Programming Assignment

You will implement a program that calculates the loan amount given the monthly payment, interest rate, and loan duration.

Calculating Interest

In this assignment, you will assume that loan interest is compounded monthly. Each month, the lender calculates the interest that must be paid by multiplying the outstanding balance on the loan by the interest rate. The monthly payment is first used to pay off the interest; the remainder is applied to the principal (i.e., it is used to reduce the outstanding balance on the loan). Therefore, given an outstanding balance at the beginning of the month (B), a monthly payment

(P), and a yearly interest rate (r), one can calculate the outstanding balance at the beginning of the next month (B') as:

$$B' = B - (P - B \cdot r/12)$$

or

$$B' = B(1 + r/12) - P$$

Input/Output

Rather than ask you to write the input and output routines for this assignment, we have provided a library that does this I/O for you. The definitions are given in the header file `io.h` seen below:

```
**************************************************************************
#ifndef __IO_H__
#define __IO_H__

double GetParam(string prompt, double min, double max);
// EFFECTS: prints the prompt, and reads a double from cin.
// If the value is between min and max, inclusive, returns
// it. Otherwise, repeats.

void PrintHeader (void);
// EFFECTS: prints out a nice header for the payment info table.
// MODIFIES: standard output stream

void PrintMonthlyData (int month, double principal,
                       double interest, double loaned);
// EFFECTS: prints out a row in the payment info table.
// MODIFIES: standard output stream

#endif
**************************************************************************
```

Use the `GetParam` function to obtain the parameters. There are three: the monthly payment (between 1 and 100,000, inclusive), interest rate (between 0 and 1 inclusive), and years of the loan (between 1 and 100, inclusive; must be an integer). To obtain each parameter, your program must ask the user to type it in. Use the `prompt` argument for this purpose. The three prompts are:

```
"Please enter the monthly payment: "
"Please enter the interest rate: "
"Please enter the duration of the loan, in years: "
```

You must use exactly these prompts. Don't forget the trailing single spaces!

The duration of the loan should be an integral value. You can assume that the user input is always an integer, but it may be outside the range. (See "Notes and Hints" below.)

Note that when using `GetParam` function to obtain the parameters, it will repeat asking for user input until the input is inside the range specified by `min` and `max`.

Your program should display output in a nice table format. Your program should first call the `PrintHeader()` function to print out the table header. It should then call `PrintMonthlyData()` once for each month of the loan, starting with the **last** month of the loan and proceeding to the **first** month of the loan (month 1). This is a little odd, but it will make your programming task easier.

The table will have four columns. First is the month of the loan -- the first month of the loan is defined to be month 1 (but note that this prints out last!). Second is the amount of the loan principal that is paid as part of the monthly payment. Third is the amount of interest that is paid as part of the monthly payment. The last column is the outstanding balance on the loan at the beginning of this month. Note that the bottom value in this column tells you the amount of the loan (i.e., how much car you can afford).

For example, if one were to specify a loan with a rate of 0.05 (5%) with a monthly payment of $100 and a one year duration, your program should print out:

```
Month    Principal          Interest             Balance
-----    ---------         ---------           ---------
12           99.59              0.41               99.59
11           99.17              0.83              198.76
10           98.76              1.24              297.52
9            98.35              1.65              395.87
8            97.94              2.06              493.81
7            97.54              2.46              591.35
6            97.13              2.87              688.48
5            96.73              3.27              785.21
4            96.33              3.67              881.53
3            95.93              4.07              977.46
```

```
2           95.53           4.47            1072.99
1           95.13           4.87            1168.12
```

Your program should not take any input or produce any output other than the I/O specified in this section. In particular, ensure that your program does not print out any debugging messages or extra output before you hand it in.

Files

There are several files located in the Project-One-Related-Files.zip in our Sakai Resources:

`io.h`	The header file for the I/O functions described above. This should be included (via #include "`io.h`") in your solution.
`io.cpp`	The implementation of the I/O functions. Should **not** be #included by your program.
`test1`	A sample input file for your program.
`output1`	The output that should be generated by input `test1`.

You should copy the above files into your working directory. **DO NOT modify io.h or io.cpp!**

You should put **all** of the functions you write in a single file, called `p1.cpp`. You may use only the C++ standard libraries, and no others.

Compiling and Testing

You are required to compile your program under the Linux environment. When you compile your program, you should use the following command line:

```
g++ -Wall p1.cpp io.cpp -o p1
```

We have given you one test case to see if your program is working correctly. You may run this test by copying `test1` and `output1` to your working directory and executing the following commands in Linux:

```
./p1 < test1 > test.out
diff test.out output1
```

This runs your program, taking input from the file "test1" instead of the keyboard, and placing output into the file "test.out" instead of the screen. Then, the "diff" program compares your test

output (test.out) with the correct output (output1). If they are identical, your program passes that test case. If diff reports any differences at all, you have a bug somewhere.

We will test your code using this test case, as well as several others. You should therefore definitely pass this test case. However, you should also create a family of test cases that exercises your program with different inputs, since the test case we have given you is not sufficient to catch all bugs.

Submitting and Due Date

You should submit your source code via the online judgment system. Your submission should only include your source code file `p1.cpp`. The due date is 11:59 pm on Oct. 8th, 2023.

Grading

Your program will be graded along two criteria:

1) Functional Correctness
2) General Style

An example of Functional Correctness is whether or not you produce the correct output. General Style speaks to the cleanliness and readability of your code. We don't need you to follow any particular style, as long as your style is consistent and clear. Some typical style requirements include: 1) appropriate use of indenting and white space, 2) program appropriately split into subroutines, 3) variable and function names that reflect their use, and 4) informative comments at the head of each function.

Notes and Hints

You do not need to handle any erroneous input -- for example, a user who types "foo" instead of "0.05". You may assume that we will always supply numbers in the expected form: for the monthly payment and interest, the inputs are always any valid real values; for the duration of loan in years, the input is always an integral value. However, these numbers might be outside the allowable range. You can assume that the entered values, although might be outside the allowable range, are always within the range from -2,000,000 to 2,000,000.

Given the equation for calculating interest and a little algebra, it is quite easy to calculate B as a function of B', r, and P. Also, remember that the outstanding balance after the final payment MUST be zero!