Computations

Workshop 4

In this workshop, you will code and execute a C-language program that accepts numerical values from the user, stores the values in variables of appropriate data type, performs calculations on the stored variables and casts from one data type to another.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities:

- to declare variables of integral and floating point types
- to code a simple calculation using C operators and expressions
- to accept a numerical value from the user using scanf
- to cast a value from one data type to another

IN-LAB: (30%)

INSTRUCTIONS

Loan calculation

Write a program that calculates the remaining balance on a loan after the first, second and third monthly payments. You should call your program loan.c.

Execution and Output Example:

```
Enter amount of loan: 20000.00
Enter interest rate: 6.0
Enter monthly payment: 386.66
Balance remaining after first payment: 19713.34
Balance remaining after second payment: 19425.25
Balance remaining after third payment: 19135.71
```

Display each balance with two digits after the decimal point. *Hint*: Each month, the balance is decreased by the amount of the payment, but increased by the balance times the monthly interest rate. To find the monthly interest rate, convert the annual interest rate entered by the user to a ratio (divide by 100) and then divide it by 12.

SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above. Part 1 is due by 23:39 on the day after your lab. Failure to use the submitter will result in a 40% penalty.

If not on matrix already, upload your **loan.c** to your matrix account. Compile and run your code and make sure everything works properly.

AtThePrompt> gcc -Wall loan.c –o ws4<ENTER>

-Wall activates the display of warnings in GCC compiler.

-o sets the executable name (ws4)

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

~profname.proflastname/submit btm200/w4_lab <ENTER>
and follow the instructions.

AT-HOME: (30%)

INSTRUCTIONS

Modify the code you developed in the lab to display the information on the loan for 12 months as a table. The table should look like the following:

```
Enter amount of loan: 20000.00
Enter interest rate: 6.0
Enter monthly payment: 386.66
```

Month	Balance	Interest	Payment	Remaining
01	20000.00	100.00	386.66	19713.34
02	19713.34	98.57	386.66	19425.25
03	19425.25	97.13	386.66	19135.71

…

The easiest way to space out the columns is to use the field width in the format specifier to control the width of the numbers. This can be used to format both the titles and the numbers.

AT-HOME REFLECTION (40%)

In 3 or 4 sentences answer the following questions and write your answer in a file called **`reflect.txt.`**

1. Describe how you spaced the table columns and lined them up with the numbers beneath them.
2. Why do you add the interest to the loan before you subtract the monthly payment?
3. Interest rates are quoted as a yearly rate. How do you convert them to be used to calculate the monthly interest?
4. Your program might have had fractional cents in the results. How did you get the proper number of cents displayed?

SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above.
If not on matrix already, upload your **loan.c and reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.
AtThePrompt> gcc -Wall loan.c –o ws4<ENTER>
-Wall activates the display of warnings in GCC compiler.
-o sets the executable name (ws)
Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

~profname.proflastname/submit btm200/w4_home<ENTER>

and follow the instructions.

SUBMISSION POLICY

The workshop is **due within 4 days following the in-lab assigned date by 23:59. Failure to use the submitter will result in a 40% penalty.**

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible for regularly backing up your work.

Please Note

- A successful submission does not guarantee full credit for this workshop.
- If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.