Karina Reyes Santiago

A01700666

FINAL PROJECT

Programming Languages

Contents

[Context of the problem 2](#_Toc7650218)

[Solution 3](#_Toc7650219)

[Programming paradigm 4](#_Toc7650220)

[Results 5](#_Toc7650221)

[Conclusions 6](#_Toc7650222)

[Setup instructions 7](#_Toc7650223)

[References 7](#_Toc7650224)

# Context of the problem

It is common that many of us grow up without having a financial background or making assumptions about how it works only by what we hear or read every day.

Normally, we arrive at the university and we graduate without knowing how the finances work or how to have a good management of our money.

One of the many issues whose terms we usually do not know are the loans and their interests.

For example, in Mexico different surveys are carried out by public entities, which allow to know the figures regarding the financial education that is lived in the country.

Within these results, it is stated that within the percentage of the population that has a departmental or self-service credit card 6% does not know if they charge interest, of those who have a bank credit card, 3.4% do not know if they charge interest, personal credit 1.2% do not know if they charge interest, mortgage credit 5.9% do not know if they charge interest and a group loan 2.8% do not know if they charge interest (Gómez, C., 2018).

The way in which a loan works will depend on the institution to which it goes or whether it is personal loans, for home or for car. Reducing the situation to its simplest form, a loan will allow you to access a certain amount of money that you need at that moment and then you pay back that money in installments.

The process of spreading out the loan into a series of fixed payments over time is called amortization.

Usually, these payments are made monthly and each payment stays the same each month. However, the payment is made up of two parts that change over time:

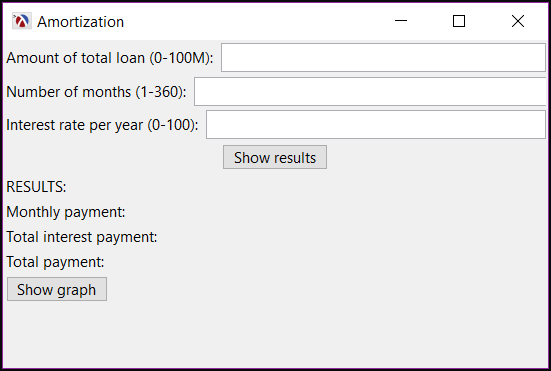
1. Interests. That is, what the borrower gets for making the loan.

2. Principal. That is, the amount that is paid from the principal loan.

However, this is where certain confusions may arise. If you have no knowledge of how to calculate these monthly payments and what percentage of them goes to interest and which to the principal payment, you could end up paying up to twice the principal loan.

# Solution

A proposed solution for the previous problem it is a simple program that ask the user three inputs: the amount of the loan, the number of months to pay the loan and the interest rate per year.



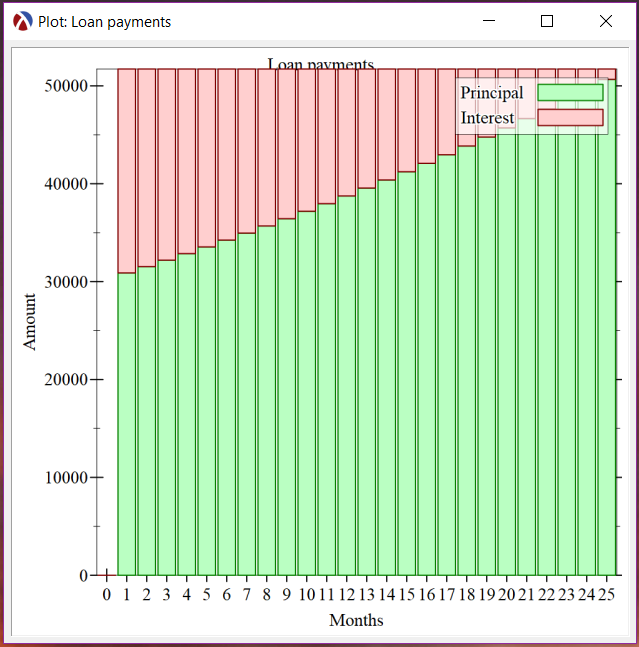
*Picture 1. Screenshot of the program user interface*

Based on that data, the program calculates the amount of the monthly payment, the total amount of interests and the total payment for the loan (including interests).

To calculate the monthly payments the program uses a formula that there is already defined in the financial area (Massey University, 2018):

Also, as it is known that graphs help to have a better understanding, the program includes a feature to show a stacked histogram. A stacked graph is useful for looking at changes in, and it can be used when the sum of the values is as important as the individual items (Salm, 2014).

In this case, the stacked graph plotted by the program shows how the monthly payments change over time, making easy to see the percentage of interests that are paid at the total amount of the payment, including the interest and the principal.



*Picture 2. Example of a stacked graph*

## Programming paradigm

For the development of the solution it was used Racket as the programming language for the source code, this language is based on Scheme, which is a dialect of Lisp. Racket is multi-paradigm, including the functional paradigm.

The functional paradigm was chosen because it treats computation as calculation of stateless mathematical functions, which minimize side effects (Progopedia, n.d.). This means that the result depends only on the inputs and as the results for the amortizations calculus is just defined by the input that the user gives, functional paradigm it is perfect to achieve the solution.

It is known that less efficiency is a possible disadvantage of functional programming (University of Central Florida, n.d.), but as the calculus required for amortization is not that complex and the program is just developed for loans of maximum 30 years (360 months), efficiency is not a problem for the purpose of this problem’s solution.

# Results

To test that the calculated results were correct on the program, many tests were made using examples already calculated on the Internet.

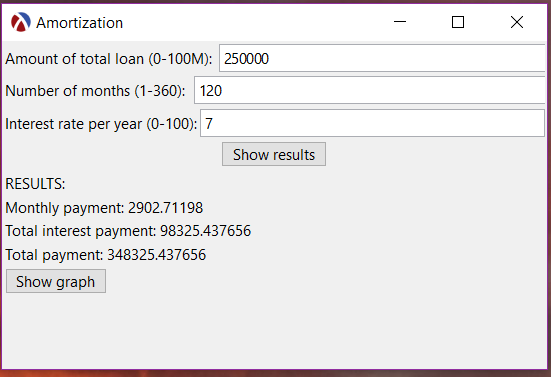
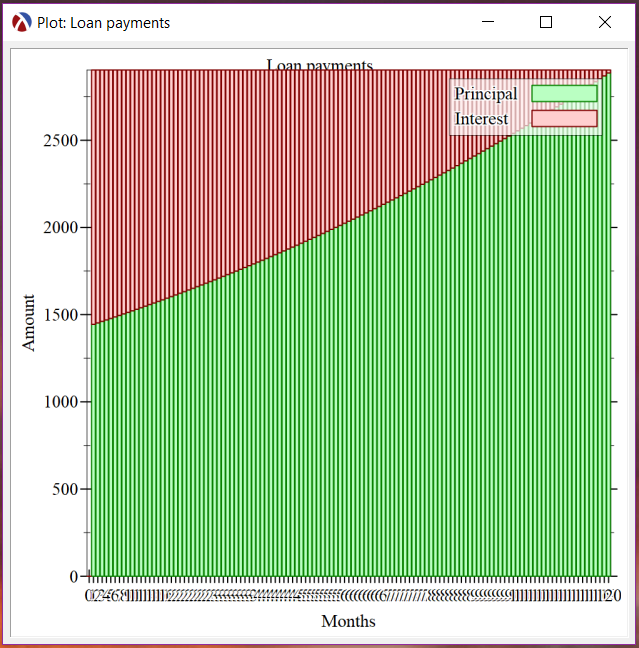
As evidence, there is an example explained on Fundera website (2019) that says as follows:

“If you pay 7% interest on a $250,000 10-year loan, you will pay a total of $98,325 in interest over the life of the loan. In contrast, if you pay 40% interest on a $250,000 9-month loan, you would only pay $43,486 in interest.”

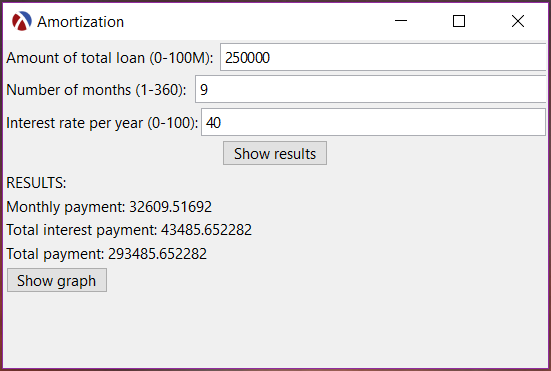
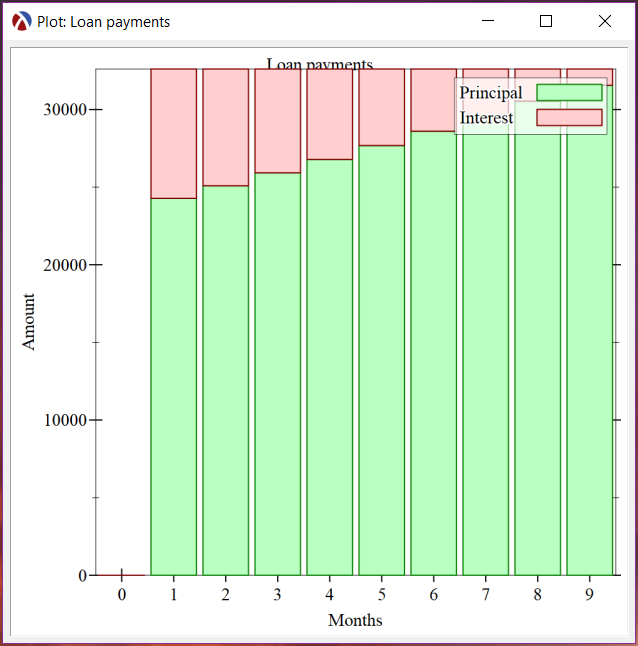
This example involves a problem with two parts. Both with the same amount of money for the loan but with different repayment period and different interest rate.

The first part is the example for a long-term loan, which means that it has a repayment period that could be of 1 to 25 years. On the other hand, the second part of the example is for a short-term loan, which repayment period could be of 3 to 18 months.

To prove if the results showed in the written example are correct, here is the problem solved with the program:

*Picture 3. Results for long-term loan Picture 4. Stacked graph for long-term loan*

*Picture 5. Results for short-term loan Picture 6. Stacked graph for short-term loan*

# Conclusions

During the development of this project, tests were also performed using excel sheets with complete amortization tables for the verification of the results of the monthly payments and the interest earned in the program. One of these files is included as evidence in the GitHub folder of the project.

In this way, it may be easier to have a clearer idea of whether a loan is convenient or not.

# Setup instructions

1. Clone the repository from GitHub with the following command:

git clone https://github.com/karyrs15/programming\_lang.git

or download it from: <https://github.com/karyrs15/programming_lang>

1. Extract files from the amortization\_exe.zip file.
2. Open the amortization folder.
3. Open the executable amortization.exe to execute the program.

# References

Fundera. (2019). *Where to Find the Best Long-Term Small Business Loans*. From: <https://www.fundera.com/business-loans/guides/long-term-loans>

Gómez, C. (2018). *La educación financiera en México*. Instituto Belisario Domínguez. From: <http://bibliodigitalibd.senado.gob.mx/bitstream/handle/123456789/4190/CI_53.pdf?sequence=1&isAllowed=y>

Hawlk, K. (2016). *How Do Personal Loans Work?* From Student Loan Hero website: <https://studentloanhero.com/featured/how-do-personal-loans-work/>

Massey University. (2018). *Finance formulas*. From: <http://owll.massey.ac.nz/maths-and-statistics/finance-formulas.php>

Miller, M. (2019). *Learn Racket by Example: GUI Programming.* From: <https://dev.to/goober99/learn-racket-by-example-gui-programming-3epm>

Pritchard, J. (2019). *How Amortization Works.* From the balance website: <https://www.thebalance.com/how-amortization-works-315522>

Progopedia. (n.d.). *Paradigm: Functional*. From: <http://progopedia.com/paradigm/functional/>

*Racket Documentation.* From Racket website: <https://docs.racket-lang.org>

Salm, M. (2014). *Stacked Graph*. From BetterEvaluation website: <https://www.betterevaluation.org/en/evaluation-options/stacked_graph>

University of Central Florida. (n.d.). *Major Programming Paradigms*. From: <http://www.eecs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html#functional>