Functional Programming.

Ortiz Vega Angelo
Course Code: CE3104
Name: Languages, Compilers and interpreters.
Module: Languages.
Academic Area of Computer Engineering.
Cartago, Costa Rica.

-Keywords: Programming Languages, High-Level Programs, Programming Paradigm, Functional Programming, Lisp, Scheme, DrRacket.

-Content:

Functional programming is a programming paradigm — a style of building the structure and elements of computer programs — that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data

What is the potential of functional programming?

a. Type Inference: Type inference automatically assigns a data type to a function without the need for the programmer to write it
    i. Strict
    ii. Non Strict
b. Pattern matching: allows to define functions by cases.
c. Components
    i. Primitive Operations.
    ii. Function creator through functions.
    iii. Apply functions to your arguments and produce a value.
    iv. Set of primitive objects.
    v. Mechanism to link a name with a function.

List Processor = LISP -> SCHEME -> DRRACKET.

LISP:

Lisp is a family of computer programming languages with a long history and a distinctive, fully parenthesized prefix notation. Originally specified in 1958, Lisp is the second-oldest high-level programming language in widespread use today. Only Fortran is older, by one year. Lisp has changed since its early days, and many dialects have existed over its history. Today, the best-known general-purpose Lisp dialects are Clojure, Common Lisp, and Scheme.

Scheme:

Scheme is a programming language that supports multiple paradigms, including functional and imperative programming. It is one of the three main dialects of Lisp, alongside Common Lisp and Clojure. Unlike Common Lisp, Scheme follows a minimalist design philosophy, specifying a small standard

core with powerful tools for language extension.

Racket:

Racket is a general-purpose, multi-paradigm programming language based on the Scheme dialect of Lisp. It is designed to be a platform for programming language design and implementation. In addition to the core Racket language, *Racket* is also used to refer to the family of Racket programming languages and the set of tools supporting development on and with Racket. Racket is also used for scripting, computer science education, and research.

The idea of functions as first-class entities is that functions are also treated as values and used as data.

Functions as first-class entities can:

- refer to it from constants and variables.
- pass it as a parameter to other functions.
- return it as result from other functions.