LECTURE 1 – INTRODUCTION TO STRUCTURED PROGRAMMING

Table of Contents

1.0 What **is programming?**

Literally, is the action or process of writing computer programs. To me, is *problem solving using algorithms.*

Terms related to programming:

- **Programming paradigm**: a pattern that serves as a school of thought for programming of computers
- **Programming technique**: related to an algorithmic idea for solving a particular class of problems. For example, ”divide and conquer” or ”program development by stepwise refinement”
- **Programming style**: the way we express ourselves in a computer program. Relates to elegance (or lack of elegance)
- **Programming culture**: The totality of programming behavior, which often is tightly related to a family of programming languages

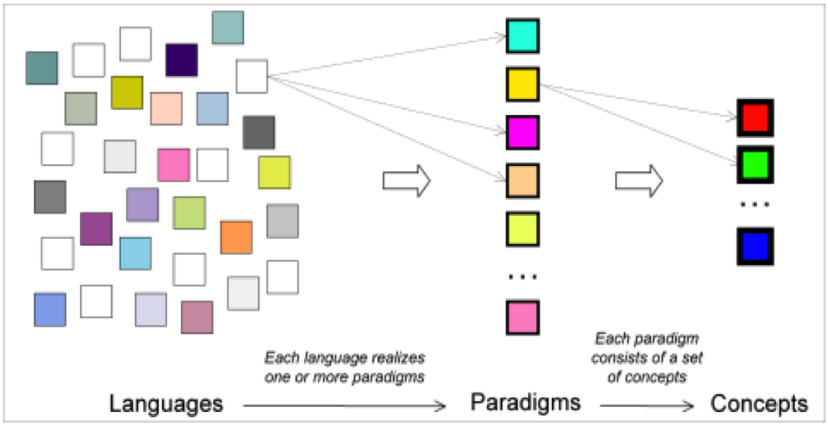

Figure 1.1: Languages, paradigms and concepts .

1.1 Programming paradigms

Solving a programming problem requires choosing the right concepts. All but the smallest toy problems require different sets of concepts for different parts of the program.

A programming paradigm, or programming model, is an approach to programming a computer based on a mathematical theory or a coherent set of principles. It is a way of conceptualizing what it means to perform computation and how tasks to be carried out on the computer should be structured and organized.

Programming languages realize programming paradigms. There are many fewer programming paradigms than programming languages.

Examples of programming paradigms:

1. Imperative
2. Functional
3. Logical
4. Object-oriented.

Most popular languages are imperative and use structured programming techniques.

Structured programming techniques involve giving the code you write

structures; these often involve writing code in blocks. Each program is logically built up from three basic logic principles:

- **Sequence** – the program performs instructions in sequence, one after the other (code executed line by line).
- **Selection** – the program selects one of several operations depending on the prerequisites. The program thus makes a selection based on some condition. (branching statements such as *if..then..else*, or *case*)
- **Iteration/ repetition** – the program repeats a series of instructions a certain number of times. (iterative statements such as *for*, *while*, *repeat*, *loop*).

1.1 Imperative paradigm

Is based on the ideas of a Von Neummann architecture. A command has a measurable effect on the program and the order of commands is important. ***First do this and next do that*** . Its main characteristics are incremental change of the program state (variables) as a function of time; execution of commands in an order governed by control structures; and the use of procedures, abstractions of one or more actions, which can be called as a single command. Languages representatives of Imperative paradigm: *Fortran*, *Algol*, *Basic*, *C*, *Pascal*.

Program 1 Example of Pascal code

```
1  Program
   Example1 ;
   Var
3  Num1, Num2, Sum :
   In te ger ;Begin
5  Write ( ' Input
   number 1: ' ) ;
   Readln (Num1) ;
7  Write ( ' Input
```

```
    number 2: ' ) ;
    Readln (Num2) ;
 9  Sum := Num1
    + Num2 ;
    Writeln
    (Sum) ;
11


    R
    e
    a
    d
    l
    n
    ;
    E
    n
    d
```

1.2 Functional paradigm

Based on mathematics and theory of functions. The values produced are non-mutable and time plays a minor role compared to imperative program. All computations are done by applying functions with no side effects.Functions are firsts class citizens. ***Evaluate an expression and use the resulting value for something***. Representatives: *Haskell*, *Clojure*. Example in Haskell:

Program 2 Example of Haskel code

```
1   Compute the sum of integers
    from 1 to n.sumtorial :: Integer
      > Integer
3   sumtorial 0 = 0
    sumtorial n = n + sumtorial (n 1)
5   main = print (sumtorial 10)
```

1.3 Logical paradigm

The logic paradigm fits well when applied in problem domains that deal with the extraction of knowledge from basic facts and relations. Is based on axioms, inference rules, and queries. Program execution becomes a systematic search in a set of facts, making use of a set of inference rules. ***Answer a question via* search *for a solution***.

Program 3 Example of PROLOG code

```
    mortal(X) :- human(X).
2   human(socrates).
    human(pitagoras).
4   ?- mortal(socrates).yes
6   ?- mortal(chico).
    no
```

1.4 Object-oriented paradigm

Data as well as operations are encapsulated in objects. Information hiding is used to pro- tect internal properties of an object. Objects interact by means of message passing. In most object-oriented languages objects are grouped in classes and classes are organized in inheritance hierarchies. ***Send messages between objects to simulate the temporal evolution of aset of real world phenomena***. Representatives: *C++*, *Java*.

Program 4 Example of C++ code

```
class Employee
{

public:
string name;
int age;
double salary;
};


Employee e1;
e1.name = "socrates";e1.age = 26;
e1.salary = 1000;

Employee e2;e2.name = "chico";
e2.age = 30;
e2.salary = 5000;


double cost = e1.salary+e2.salary;
```

There are other paradigms, such as Visual paradigm, Constraint based paradigm, Aspect-oriented paradigm and Event-oriented paradigm.

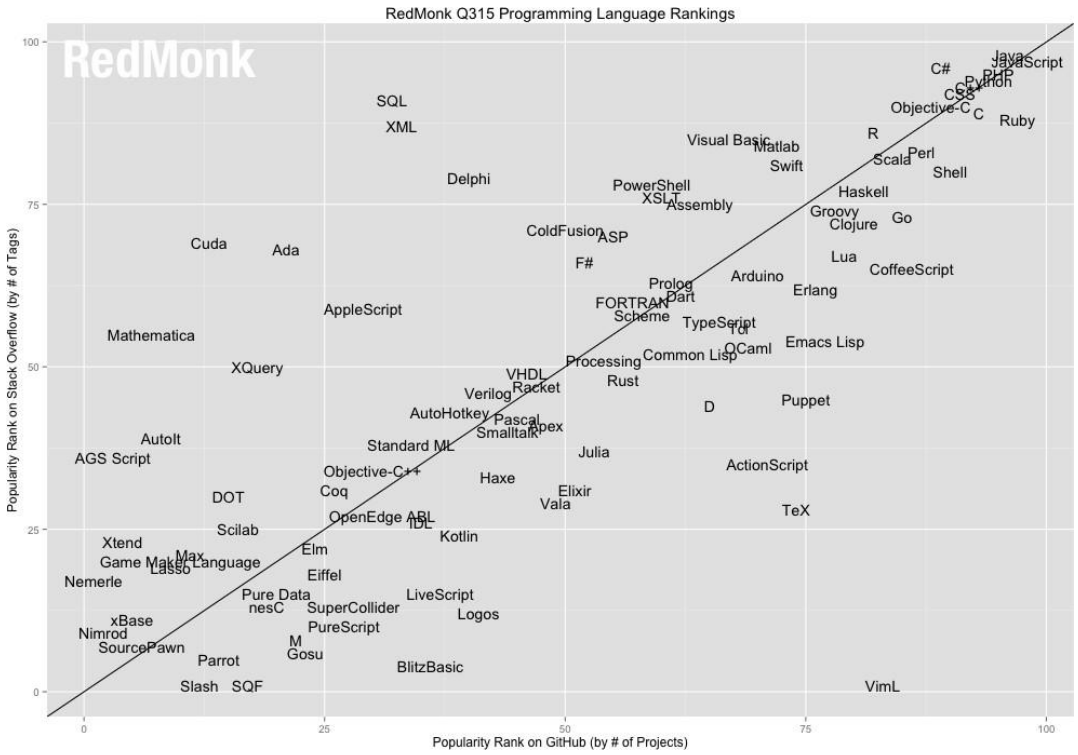

Figure 1.2: Ranking of languages. Source: http://redmonk.com/sogrady/2015/07/01/language- rankings-6-15/

1.5 Programming languages

What we have to learn to study a programming language? Syntax, semantics and how to best use the languages features to implement the programming paradigm more adequate to solve your problem. How many languages are out there? Which languages should I know?

Any ranking is influenced by communities of the development, investments from third parties and presence of projects and statistics. But this course is not about advocating the use of this or that language. **This course will teach C++.**