Programs, languages and compilers

Initial concepts

1. Software

If we look for the definition of *software* on the Internet, we can find some of them, although they are basically the same. We call **software** the set of programs, documentation and data closely related in order to make up a computer application.

Each software product is (or can be) different from the rest, because it is developed for a different customer, or to fulfill a different purpose. So, developing this product implies some previous steps: understand what we have to do, make a previous design and implement it, as we will see later. Therefore, we can't compare software development with industrial production (such as keyboard manufacturing, for instance), where everything is much more automated. Software development requires the creation of a software project, and a group of people working in a coordinated way. Besides, software does not break along time, although its performance can be reduced because of its own updates and improvements.

1.1. Software components

From previous definition of software, we can figure out that it is composed of three elements:

- **Programs**: sets of instructions that provide the desired functionality. They are written in a specific programming language.
- **Data**: programs need data to work with. These data can be retrieved and/or stored from/in databases or files.
- **Documentation**: software documents explain how to use it, which elements are part of it and how they are interconnected, so that it can be updated or fixed in the future.

1.2. Software types

There are two main types of software:

- **Application software**: they bring some kind of service to the customer, such as text processors, spreadsheets, drawing, etc. In this type of software we can find some subcategories, such as management software (payroll, accounting), engineering or scientific software (CAD, simulators...), network software (web browsers, FTP clients or servers...), etc.
- **System software**: they manage a computer system. They are basically programs that support or help other programs. In this category we can talk about operating systems or compilers, for instance.

2. Languages and compilers

We have seen that programs are sets of instructions that are provided to a computer to complete a task. These instructions are written in a programming **language** of our choice, and this way we create some text files called **source code**, written in the chosen language.

2.1. Language types

When we want to choose a specific programming language, we distinguish between **high level** languages (close to human language, and so, easier to understand by the programmers), and **low level** languages (close to machine language, and so, harder to understand by the programmers, but more efficient).

- We have a wide variety of **high level** languages to choose, depending on the type of application that we want to implement. We can talk about C, C++, C#, Java, Javascript, PHP, Python, etc.
- Among **low level** languages, maybe the most popular one is the assembly language, a very concrete set of instructions that are easily translated into machine language.

Here we can see a simple program written in Java that just prints "Hello" on the screen:

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Hello");
    }
}
```

The same program written in C could look more or less like this:

```
#include <stdio.h>

int main()
{
    printf("Hello");
    return 0;
}
```

2.2. Language processors

Computers can't understand any of the programming languages that humans use to create their programs. In order to make them work, their instructions need to be translated into a language that computers can understand. This language is called *machine code*, and is composed of bits (zeros and ones).

If we want to translate a given programming language into machine code, we use a tool called **compiler**, although this assertion is not completely true. There are some different language processors that can be used, depending on the language itself:

- **Compilers**: they translate the code written in a specific language into machine code, and they generate an executable file with the result. For instance, if we compile a program written in C under Windows, we will get an *EXE* file that we will be able to run.
- **Interpreters**: they translate from the specified programming language to machine code "on the fly". In other words, they don't generate any executable file. So, every time we need to run the program, we need to have the source file available. This kind of language processor is very usual in languages like PHP or Python. This way, the response time increases a little, but the program can be run in multiple platforms.
- **Virtual machines**: an intermediate solution between compilers and interpreters is the one used by languages like Java. These programs are not compiled into a native machine code (there is no *EXE* file in Java, for instance), and they are not interpreted either. Java compiles the source code and translates it into its own intermediate machine code. Then, it runs its virtual machine (JVM, *Java Virtual Machine*), that is in charge of interpreting and running this code every time we need. This way, we don't need to have the source code available before we run the program, and we don't depend on a given platform either (Windows, Linux, etc.). We only need to have a JVM installed in our system in order to run our Java programs. The same thing happens with C# and its virtual platform *.NET*.

3. Some popular languages

There are some studies and analysis that try to classify the programming languages according to its popularity or current usage. Some of the most popular ones are:

- TIOBE index
- PYPL
- IEEE Spectrum
- RedMonk ranking

Regarding this last one, it is based on crossing data between the main source code repository (GitHub) and the main programming help page (StackOverflow). Take a look at both rankings, check the results and see if they differ in some important language(s). Do you miss any other language in the top 10?

> **Exercise 1**:
>
> Look for an online code editor that lets us work with many different languages. Here you can get one: Ideone. Find out how to write a program to say "Hello" in some different languages, such as Java, C, Python, JavaScript, or PHP.