# 2020-02-21

## Giuseppe

Missed the lesson because the metro got delayed.

## Abbadi

…

## Grande Omega

Structure preserving transformation is to change the content of the data structure (a fundamental pillar of modern computing) without destroying its structure.

Data structure allows the programmer to think in term of what exists in a program at any given time. It also guarantees that nothing else exists outside the given data types and give structure to the data by shaping the allowed shapes of data.

A container can be simple, although these container are not adaptable or flexible which is the goal of this course.

Generic data structures means that the structure can the way the container behaves depending on the given parameters. This gives the programmer a generic container that can be used in multiple ways.

Transforming containers would be as example: is\_even:Fun<number,boolean>

d

# 2020-02-14

## Giuseppe

Typescript (TS) will be used during this minor. TS is according to the teacher the most powerful language in the world.

Person John = new Person();

Above is C# written code, and it is clear that Person is repeated twice. Repeating something twice is against human nature. Repeating something twice is against human nature.

A programmer translates human logic to computer logic.

Types aren’t meant to be objects, but are meant represent concepts that define the constrains of a type.

When you don’t have constrains – “”typing stuff in a stupid way” then it is unknown how the code should be used.

Part of code (i.e. threads) should take the ownership of an object – meaning other parts of the codes shouldn’t be able to change the value that part is altering.

Functions are essential. “Lazy evaluation” – on demand computing code while compiling (this is the default way in Haskell). When you have return function A that returns function B that takes a parameter I. Function B will not be compiled/compute if no parameter I is given. Even if function A is called.

Types and lambdas can both have the same name (i.e. Fun).

Zero’s (o) are needed when no value is given.

## Abbadi

End of the period there will bet wo exams:

* Practicum one – implementing something in visual studio
* Theory – multiple questions (Grande Omega)

There are four projects with their own goal and criteria. Remember:

* The code needs to be unbreakable (this will be tested);
* Needs to be done before end of project 4;
* An oral exam will be given (so, you can work together but you need to understand).
* Only one has to be chosen;
* Needs to be done in typescript;
* Start right away with the project, but may have to rewrite it later on.

Create new NPM project:

Run: Npm init

Run: Npm install typescript

Make a typescript file

Run: .\node\_modules\.bin\tsc .\main.ts -w

Run another NPM: node .\main.js

## Grande Omega

A programmer should be able to translate real-world phenomenon and translate it into code. Programming is often done with an imperative mindset and without thinking much about types. Imperative programming is focussing on how things are computed and take into account more than just the problem.

Shared states often will result into a problems. A ‘data race’ occurs which method alters the variable first.

Types are annotations that define much more than which data and functions we can expect to find in a variable or symbol: types allow us to model all sorts of provable properties of data, and can be used to model even dynamic abstract information such as effects or exceptions.

The underpinning principle that we will try to carry around during the whole course is that of composition. The basic idea of composition is that, given two entities of the same sort, we can compose them into a new one.

A function is *non-deterministic* when the output does not depend only on its input, but also how long it has being running.

Referentially transparent means that a value is not depend on the state of the program. In other words, if the variable X has the value 5 then the value will always be 5.

In this course *functors* and *monads* will be discussed further down the road.