> It is not too misleading, at least historically, to say that categories are what one must define in order to define functors, and that functors are what one must define in order to define natural transformations.(Freyd)

A **natural transformation** is a morphism of functors

Definition 0.1. Given categories $\mathcal{C}, \mathcal{D}$ and functors $F, G : \mathcal{C} \to \mathcal{D}$, a **natural transformation** $\alpha : F \Rightarrow G$

$$\begin{tikzcd} \mathcal{C} \arrow[r, bend left=50, "F"{name=U}] \arrow[r, bend right=50, "G"'{name=D}] & \mathcal{D} \arrow[Rightarrow, from=U, to=D, "\alpha"] \end{tikzcd}$$

is made of a morphism $\alpha_c : Fc \to Gc$ in $\mathcal{D}$ for every object $c \in \mathcal{C}$[1] (the **components**) so that for any $f : c \to c'$ in $\mathcal{C}$ the diagram

$$\begin{tikzcd} Fc \arrow[r, "\alpha_c"] \arrow[d, "Ff"'] & Gc \arrow[d, "Gf"] \\ Fc' \arrow[r, "\alpha_{c'}"'] & Gc' \end{tikzcd}$$

commutes ($Gf \circ \alpha_c = \alpha_{c'} \circ Ff$, this is the **naturality condition**).

See [1] Ch.1.4, [2] Sec.1.4, [3] Ch.10.

A natural transformation is a way to compare two functors: if there exists a natural transformation between two functors then they are somehow related.

A **natural isomorphism** is a natural transformation $\alpha : F \Rightarrow G$ where the components are isomorphisms, in this case the functors F and G are isomorphic $F \cong G$ that is they do the same thing.

A natural transformation takes a morphism in the category $\mathcal{C}$ and takes it to a commuting diagram in the category $\mathcal{D}$. This allows us

[1] that is for every object $c \in \mathcal{C}$ we *pick* a specific morphism α_c in $\mathcal{D}(Fc, Gc)$

to define an higher level language in a category, a natural transformation introduces a bunch of commuting diagrams without the need to introduce them one by one (lower level language).

If we look at functors as container then natural transformation re-package containers, in this sense natural transformation are not allowed to modify or add elements to a container because there is no way to do it in a polymorphic way (they can delete content though).

Natural transformation in programming are ubiquitous: *most* polymorphic functions between algebraic data types (including identity and constant functors) can be natural transformations.

See also

- Category Theory 9.1: Natural transformations - Bartosz Milewski - YouTube
- Natural Transformations (examples)

---

[1] E. Riehl, *Category Theory in Context* (Dover, 2015).
[2] P. Perrone, *Notes on Category Theory with Examples from Basic Mathematics*, (2019).
[3] B. Milewski, *Category Theory for Programmers* (2019).