III. Conception de programmes

Types abstraits et types concrets

Lorsque l'on conçoit un programme pour résoudre un problème, il est nécessaire de déterminer de quelles fonctions et algorithmes on aura besoin et comment seront organisés/ stockées les données.

Une conception se fera donc en plusieurs étapes :

1. Déterminer les fonctions nécessaires
2. En déduire les types de données qui seront utiles en regroupant les fonctionnalités par concept abordé
   → *type abstrait*
3. Pour chaque type, décrire les données qu'il représente et coder les algorithmes des fonctions du type
   → *type concret*

III.1 Notion de Type Abstrait

Une fois les fonctionnalités nécessaires déterminées, émergent des entités définies par des propriétés et des fonctions.

Les propriétés de ces entités représentent différentes valeurs, chacune pouvant avoir un type différent. On pourra par exemple de représenter une `Personne` par son nom (`text`), son prénom (`text`), et son âge (`Int`).

Utiliser plusieurs variables pour chaque `Personne` représentée s'avère très vite limité et on a besoin de pouvoir utiliser une seule variable pour représenter l'entité `Personne`, et donc déterminer un type regroupant à la fois ces propriétés et les fonctions associées.

Ce regroupement de propriétés et de fonctions s'appelle un *type abstrait*.

Un *type abstrait* représente donc une entité proposant un ensemble d'opérations et de propriétés qui la représente.

La manière dont est codée le type abstrait n'a pas à être connue par les utilisateurs de ce type.

Même si les types de base sont aussi des types abstraits, on aura besoin de définir des types complexes et ceux-ci devront également être abstraits, c'est à dire que l'algorithme qui les utilise n'a pas à connaître son codage et ne doit travailler qu'avec des valeurs et les opérations définies pour ce type.

Conclusion sur la notion de type abstrait

- Un *type abstrait* représente une famille d'objets apparentés par les propriétés et opérations définies sur ses objets. Les mêmes opérations peuvent s'appliquer à chacun des objets.
- Les opérations sont choisies *selon les besoins* d'une application.
- D'une application à l'autre, les opérations peuvent être très différentes, même si les types portent le même nom : *le nom ne définit pas un type* !

Conclusion sur la notion de type abstrait

- Un *type abstrait* représente une famille d'objets apparentés par les propriétés et opérations définies sur ses objets. Les mêmes opérations peuvent s'appliquer à chacun des objets.
- Les opérations sont choisies *selon les besoins* d'une application.
- D'une application à l'autre, les opérations peuvent être très différentes, même si les types portent le même nom : *le nom ne définit pas un type* !

Définition : *Types abstraits*

On appelle *type abstrait* une *spécification* d'un ensemble de données, de leurs *propriétés* et de *l'ensemble des opérations* que l'on peut effectuer sur ces données.