Lista 5

Algebraiczne typy danych (ADT) – typy iloczynowe i sumaryczne

Typy iloczynowe (ang. *Product types*)

Dla każdego z poniższych zadań należy przedstawić implementację w języku OCaml jak lub Scala. Do implementacji typów iloczynowych z anonimowymi elementami należy wykorzystać krotki. W przypadku typów z nazwanymi polami, dla języka OCaml należy użyć typów rekordowych, a w dla Scali klas.

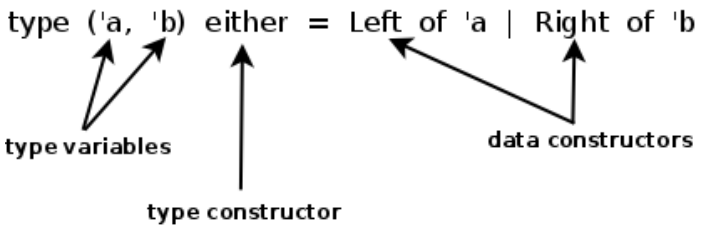
- https://dev.realworldocaml.org/guided-tour.html#tuples
- https://dev.realworldocaml.org/records.html
- https://docs.scala-lang.org/tour/tuples.html
- https://docs.scala-lang.org/tour/classes.html

1. Z wykorzystaniem krotek, skonstruuj typ iloczynowy **Point2D** reprezentujący punkt w dwuwymiarowej przestrzeni**.** Niech punkt będzie reprezentowany przez parę współrzędnych kartezjańskich typu zmiennoprzecinkowego pojedynczej precyzji. Następnie:
    a. zdefiniuj funkcję **distance** przyjmującą parę punktów oraz zwracającą odległość euklidesową między typami punktami.
    b. zastanów się, jak można zmodyfikować program, żeby reprezentować punkty w trójwymiarowej oraz n-wymiarowej przestrzeni.
2. Skonstruuj typ iloczynowy **Person** reprezentujący osobę, który komponuje informacje o imieniu, nazwisku, wieku, płci, rozmiarze buta. Dobierz odpowiednie typy składowe. Następnie skonstruuj typ iloczynowy **Partnership**, który komponuje dokładnie dwa wystąpienia typu **Person**.
   Przedstaw po dwie implementacje, jedną wykorzystującą krotki oraz drugą opartą o typy rekordowe z nazwanymi polami. Utwórz przykładowe wystąpienia typu **Partnership** a następnie zbadaj mutowalność struktur. Zaproponuj i zaimplementuj dowolną funkcję odwzorowującą **Partnership** w **Person** (np.

zwracającą osobę młodszą, o mniejszym rozmiarze buta lub dowolną inną funkcję).

Typy sumaryczne (ang. *Sum types*)

Dla każdego z poniższych zadań należy przedstawić implementację zarówno w języku OCaml, jak i Scala. Do realizacji typów sumarycznych należy wykorzystać wbudowane środki abstrakcji w języki. W przypadku OCaml należy wykorzystać słowo kluczowe **type** w sposób przedstawiony poniżej:



W przypadku Scala należy skorzystać z **wyliczeń (https://docs.scala-lang.org/scala3/reference/enums/enums.html).**

3. Skonstruuj własny typ sumaryczny **WeekDay** reprezentujący dzień tygodnia. Niech wartości tego typu reprezentują poszczególne dni. Wykorzystując mechanizm dopasowywania do wzorca. Następnie:
    a. napisz prostą funkcję **weekDayToString**, która odwzorowuje **WeekDay w** ciąg znaków reprezentujący dany dzień tygodnia w języku polskim.
    b. Napisz funkcję **nextDay**, która przyjmuje dzień tygodnia oraz zwraca dzień występujący po nim.
4. Skonstruuj własną implementację typu dla reprezentacji obecności lub nieobecności wartości o nazwie **Maybe**. Typ powinien obejmować dwa przypadki:
    a. **Just** – reprezentujący wynik obliczeń, przechowujący wartość,
    b. **Nothing** – reprezentujący brak wyniku obliczeń.

Wykorzystaj polimorfizm parametryczny, aby wsparcie dla wartości opcjonalnych różnych typów. Następnie:

- Skonstruuj funkcję **safeHead**, która na wejściu przyjmuje listę i zwraca "głowę" listy opakowaną w typ **Maybe.** Jeśli lista jest pusta, funkcja zwraca **Nothing.** W przeciwnym wypadku – **Just.**

Łączenie typów sumarycznych i iloczynowych: Typy hybrydowe

5. Stosując odpowiednie połączenie typów iloczynowe i sumaryczne, zaproponuj typ **SolidFigure** mający możliwość reprezentacji różnych brył: prostopadłościan, stożek, kulę, walec. Dla każdej z brył określ, jakie dane są niezbędne do obliczenia objętości bryły i wykorzystaj je do określenia struktur danych. Zdefiniuj funkcję **volume** obliczającą objętość dowolnej bryły reprezentowanej przez opracowany typ.