LABORATORIUM 8. ZASTOSOWANIE KROTEK I WYLICZEŃ.

Cel laboratorium:

Poznanie krotek i wyliczeń oraz posługiwanie się nimi w języku Swift.

Zakres tematyczny zajęć:

- tworzenie krotek,
- tablice składające się z krotek,
- tworzenie typów wyliczeniowych,
- zarządzanie typami wyliczeniowymi.

Pytania kontrolne:

1. Co to jest krotka i do czego służy?
2. Jak definiuje się krotkę?
3. Jak odwołać się do elementów krotki?
4. Jak utworzyć tablicę składającą się z krotek?
5. Co to jest typ wyliczeniowy? Do czego służy?
6. W jaki sposób stosuje się typ wyliczeniowy?
7. W jaki sposób należy zdefiniować typ wyliczeniowy, aby można było wyświetlić wszystkie jego elementy oraz je zliczyć?

Krotka (ang. Tuple) jest charakterystycznym typem dla języka Swift. Krotki umożliwiają tworzenie i przekazywanie grup wartości. Ten typ można zastosować, aby zwrócić wiele wartości z funkcji jako pojedynczą wartość złożoną. Krotka może zawierać dane różnych typów. Dane przechowywane przez jedną krotkę nie muszą być tego samego typu.

Przykłady krotek zostały przedstawione na Listingu 8.1. Krotkę można zdefiniować podając kolejne wartości. Wtedy dostęp do poszczególnych jej elementów można uzyskać poprzez indeks, zaczynając od zera. Przy definiowaniu krotki można zdefiniować poszczególne nazwy dla poszczególnych składowych. Wtedy dostęp do elementów uzyskuje się przez te nazwy. Przykład krotki zawierającej trzy wymiary także został przedstawiony na Listingu 8.1.

Listing 8.1.Tworzenie krotki

```
let osoba = ("Ania", 16)
print("\(osoba.0) ma \(osoba.1) lat”)
//Ania ma 16 lat

//drugi sposób
let osoba = (imie:"Ania", wiek: 36)
print("\(osoba.imie) ma \(osoba.wiek) lat”)
```

```
//krotka trzyelementowa
let dim3 = (x: 1, y: 0, z: 8)
print("x=\(dim3.x); y=\(dim3.y); z=\(dim3.z) ")
```

Modyfikowanie krotki polega na przypisaniu nowych wartości, co zostało przedstawione na Listingu 8.2. Jeśli krotka została zdefiniowana bez nazw, wystarczy podać nowe dane zgodnie z typem. Jeśli krotka zawiera nazwy, muszą się one zgadzać przy modyfikacji.

Listing 8.2.Modyfikowanie krotki

```
var osoba = (imie:"Anna", wiek: 16)
print("\(osoba.imie) ma \(osoba.wiek) lat ")
osoba = ("Monika", 17)
print("\(osoba.imie) ma \(osoba.wiek) lat ")
osoba = (imiona: "Monika, wiek: 18) //błąd
```

Tablica składająca się z krotek została przedstawiona na Listingu 8.3. Tablica składa się z dwóch elementów: nazwiska osoby oraz numeru karty bibliotecznej. Dodanie nowego elementu tablicy może nastąpić przy pomocy metody *append()* lub za pomocą operatora +=. Wyświetlenie danych następuje poprzez zdefiniowane nazwy.

Listing 8.3.Tablica składająca się z krotek

```
var tab: [(nazwisko: String, nrKarty: Int)] = []
tab += [(nazwisko: "Kowalski", nrKarty: 123)]
tab.append((nazwisko: "Kowal", nrKarty: 531))
for i in 0..<tab.count{
    print("\(tab[i].nazwisko) - nr karty: \(tab[i].nrKarty)")
}
```

Wyliczenia (ang. *Enumerations*) definiują typ wspólny dla grupy powiązanych wartości. Umożliwiają pracę z tymi wartościami w sposób bezpieczny. Wyliczenia w Swift są elastyczne, nie trzeba podawać wartości dla każdego przypadku wyliczenia. Podana wartość wyliczenia może być ciągiem, znakiem lub wartością dowolnego typu całkowitego lub zmiennoprzecinkowego.

Wyliczenia są typami. Mają dostęp do metod i właściwości. Wyliczenia mogą również definiować inicjatory, aby zapewnić początkową wartość. Mogą być rozszerzane w celu dodania funkcjonalności poza ich początkową implementację.

Wyliczenia należy poprzedzić słowem kluczowym *enum*, po której umieszczana jest jego nazwa. Wewnątrz definiowane są przypadki wyliczenia. Przykład wyliczenia dostępnych walut został przedstawiony na Listingu 8.4. Ponieważ definiowany jest nowy typ, zaleca się, aby jego nazwa rozpoczynała się z wielkiej litery. Nazwa powinna być w liczbie pojedynczej. Definicje przypadków mogą także zostać wymienione po przecinku (Listing 8.5).

Listing 8.4.Definiowanie typu wyliczeniowego

```
enum Waluta{
    case pln
    case eur
    case usd
    case chf
    case gbp
}
```

Listing 8.5.Definiowanie typu wyliczeniowego

```
enum Mebel{
    case krzeslo, stol, szafa
}
```

Definiowanie zmiennych z użyciem typu wyliczeniowego zostało przedstawione na Listingu 8.6. Przy modyfikacji ich wartości nie trzeba stosować nazwy typu wyliczeniowego, ale można zastosować samą kropkę i przypadek wyliczeniowy.

Listing 8.6. Definiowanie zmiennej przy użyciu typu wyliczeniowego

```
var walutaPln = Waluta.pln
var walutaEur = Waluta.eur
```

Dopasowanie wartości do przypadków wyliczeń można wykonać za pomocą instrukcji *switch*, co przedstawiono na Listingu 8.7. Należy pamiętać, aby instrukcja *switch* zawierała wszystkie elementy wyliczeniowe.

Listing 8.7. Dopasowanie wartości

```
var waluta = Waluta.pln

switch waluta {
case .chf:
    print("Frank szwajcarski")
case .eur:
    print("Euro")
case .gbp:
    print("Funt brytyjski")
case .pln:
    print("Polski złoty")
case .usd:
    print("Dolar amerykański")
}
```

W przypadku, gdy wymagany jest dostęp do zliczenia wszystkich zdefiniowanych elementów, należy zastosować *: CaseIterable* po nazwie typu wyliczeniowego. Na Listingu 8.8 przedstawiono typ wyliczeniowy *Mebel*, który umożliwia zliczenie wszystkich elementów za pomocą właściwości *allCases.count*.

Listing 8.8. Zliczenie zdefiniowanych elementów typu wyliczeniowego

```
enum Mebel: CaseIterable{
    case krzeslo, stol, szafa, kanapa
}


let liczbaMebli = Mebel.allCases.count
print("Liczba mebli: \(liczbaMebli)")
//"Liczba mebli: 4"
```

Właściwość allCases można użyć do wyświetlenia wszystkich elementów typu wyliczeniowego, co zostało przedstawione na Listingu 8.9.

Listing 8.9. Wyświetlenie przypadków typu wyliczeniowego

```
for mebel in Mebel.allCases {
    print(mebel)
}
```

Typ wyliczeniowy może zawierać elementy różnego typu. Na Listingu 8.10 przedstawiono definicję typu wyliczeniowego, który może przyjmować dwa lub trzy wymiary.

Listing 8.10. Wyświetlenie przypadków typu wyliczeniowego

```
enum Wymiar {
    case dwuwymiarowy(Double, Double)
    case trojwymiarowy(Double, Double, Double)
}

var w = Wymiar.dwuwymiarowy(30.5, 12.0)
w = .trojwymiarowy(23.8, 0.0, 10.0)
```

Zadanie 8.1.

Polecenie 1. Napisz program konsolowy, który zdefiniuje krotkę *osoba* składającą się z elementów takich jak: imię, nazwisko, rok urodzenia.

Polecenie 2. Utwórz dwie krotki. Wyświetl, która osoba jest starsza, a która młodsza, lub obie, jeśli są w takim samym wieku.

Zadanie 8.2.

Polecenie 1. Napisz program konsolowy, który zdefiniuje krotkę *student* składającą się z elementów takich jak: nazwisko oraz trzech ocen jako typu wyliczeniowego.

Polecenie 2. Wczytaj dane trzech studentów. Wyświetl dane studentów w kolejności malejącej według uzyskanej średniej. Należy utworzyć typ wyliczeniowy oceny (2, 3.0, 3.5, 4.0, 4.5, 5.0) oraz zapewnić poprawne wprowadzenie ocen.

Zadanie 8.3.

Polecenie 1. Wprowadź od użytkownika liczbę mieszkań, a następnie wczytaj dane mieszkań do tablicy. Każde mieszkanie składa się z elementów takich jak: lokalizacja, powierzchnia oraz cena za metr.

Polecenie 2. Wyświetl mieszkanie najdroższe oraz najtańsze.

Polecenie 3. Wyświetl mieszkanie o podanej lokalizacji. Lokalizację podaje użytkownik.

Zadanie 8.4.

Polecenie 1. Zdefiniuj typ wyliczeniowy *Miesiac*, który zawiera 12 miesięcy.

Polecenie 2. Dopasuj porę roku dla danego miesiąca za pomocą instrukcji *switch*. Załóż, że dany miesiąc jest przyporządkowany do jednej pory roku.

Zadanie 8.5.

Polecenie 1. Zdefiniuj typ wyliczeniowy *Standard* dla mieszkania (wysoki, średni, niski).

Polecenie 2. Zmodyfikuj zad. 8.3 uzupełniając dane o standard mieszkania.

Polecenie 3. Wyświetl posortowane mieszkania według standardów w kolejności niski, średni oraz wysoki.