LABORATORIUM 6. ZASTOSOWANIE TABLIC.

Cel laboratorium:

Poznanie tablic i posługiwanie się nimi w języku Swift.

Zakres tematyczny zajęć:

- tworzenie tablic,
- dostęp do elementów tablicy i zarządzanie tablicą,
- tablica i instrukcje iteracyjne.
- wybrane metody stosowane z tablicami.

Pytania kontrolne:

1. Co to jest tablica i do czego się ją stosuje w języku Swift?
2. Jak utworzyć pustą tablicę?
3. Jaką metodą można wypełnić tabelę domyślnymi wartościami?
4. W jaki sposób można dodawać elementy do tablicy?
5. W jaki sposób można modyfikować zawartość tabeli?
6. Jak można sprawdzić, czy tabela nie zawiera elementów?
7. W jaki sposób można uzyskać dostęp do kolejnych elementów tabeli?
8. Jakie metody można zastosować z tablicami?

Tablica (ang. Array) jest jednym z podstawowych typów kolekcji, obok zbiorów i słowników. Elementy w tablicy są uporządkowane, co oznacza, że każdy element ma przypisany odpowiedni indeks. Numeracja elementów tablicy rozpoczyna się od 0. Służą one do przechowanie danych tego samego typu. Po utworzeniu tablicy i przypisaniu jej do zmiennej, może ona być edytowalna. Można dodawać nowe elementy, usuwać lub modyfikować istniejące. Jeśli tablica zostanie utworzona i przypisana do stałej, nie jest ona modyfikowalna.

Tworzenie pustej tablicy zostało przedstawione na Listingu 6.1. Przy deklaracji można zdefiniować typ (np. *Int*). Do wyświetlenia liczby jej elementów zastosowano funkcję *count*. W przedstawionym przykładzie liczba elementów jest równa zero.

Listing 6.1. Tworzenie pustej tablicy

```
var emptyArr: [Int] = []

print("Liczba elementów tablicy = \(emptyArr.count)")
```

Dodanie elementu do tablicy można zrealizować za pomocą metody *append()*, której parametrem jest wartość tego elementu. Na Listingu 6.2 do utworzonej pustej tablicy dodano 2 elementy (o wartościach odpowiednio 12 i 32). Następnie usunięto te elementy poprzez przypisanie *[]*.

Listing 6.2.Dodawanie elementów do tablicy

```
var tab: [Int] = []
tab.append(12)
tab.append(32)

print("Liczba elementów tablicy = \(tab.count)")

tab = []
print("Liczba elementów tablicy = \(tab.count)")
```

Podczas tworzenia tablicy można podstawić do niej domyślne wartości przy pomocy metody *repeating:*, która przyjmuje dwa parametry: wartość domyślna oraz liczba elementów. Na Listingu 6.3 przedstawiono zdefiniowanie tablicy liczb całkowitych zawierającej 5 elementów, każdy o wartości 11 oraz tablicy liczb zmiennoprzecinkowych o domyślnej wartości 1.0 dla 5 elementów.

Listing 6.3.Tworzenie tablicy z domyślnymi wartościami

```
var tabInt: [Int] = Array(repeating: 11, count: 5)
print("Liczba elementów tablicy = \(tabInt.count)”)

var tabDouble: [Double] = Array(repeating: 1.0, count: 5)
```

Tworzenie nowej tablicy składającej się elementów z dwóch innych tablic przedstawiono na Listingu 6.4. Nowa tablica zawiera elementy z *tab1*, a następnie z *tab2*.

Listing 6.4.Tworzenie nowej tablicy na podstawie istniejących

```
var tab1: [Int] = Array(repeating: 11, count: 5)
var tab2: [Int] = Array(repeating: 3, count: 2)
var tab = tab1 + tab2
```

Dodanie elementów jednej tablicy do drugiej można także zrealizować za pomocą metody *append()*, co przedstawiono na Listingu 6.5. Do tabeli *tab* zostaną dołączone elementy tabeli *tab2*.

Listing 6.5.Dodanie elementów do tablicy

```
 var tab: [Int] = [1, 2, 3, 4, 5]
 var tab2: [Int] = [11, 12, 13, 14, 15]
 tab.append(contentsOf: tab2)
```

Tworząc tablicę można przypisać im początkowe wartości. Można wskazać, jakiego typu będą te elementy albo informację tę pominąć, co przedstawiono na Listingu 6.6.

Listing 6.6.Tworzenie tablicy ze zdefiniowanymi początkowymi wartościami

```
var zwierzeta: [String] = ["pies", "kot", "chomik"]
var zwierzeta = ["pies", "kot", "chomik"]
```

W celu sprawdzenia, czy tabela jest pusta, można zastosować właściwość *isEmpty* lub zweryfikować, czy liczba elementów tablicy jest równa zero (Listing 6.7).

Listing 6.7. Sprawdzenie czy tabela jest pusta

```
if tab.isEmpty {
    print("Tablica nie zawiera elementów")
}
else{
    print("Tabela zawiera \(tab.count) elementów")
}

if tab.count == 0 {
    print("Tablica nie zawiera elementów")
}
else{
    print("Tabela zawiera \(tab.count) elementów")
}
```

Dodać elementy można za pomocą metody *append()* lub operatora +=, co przedstawiono na Listingu 6.8. Nowe elementy zostaną dołączone na jej koniec. Przy operatorze += należy pamiętać, aby nowe elementy umieścić w *[]*.

Wstawienie elementu na konkretnym indeksie tabeli można wykonać przy pomocy metody *insert(_:at:)*. Jeśli na tym indeksie istniał element, zostanie on przesunięty na kolejne miejsce w tabeli (Listing 6.8). Należy pamiętać, aby nie wykroczyć poza zakres tablicy.

Listing 6.8. Dodawanie elementów do tabeli

```
var tab: [Int] = []
tab.append(25) //25
tab += [42] //25, 42
tab += [12,3] //25, 42, 12, 3
tab.append(4) //25, 42, 12, 3, 4
tab.insert(60,1) //25, 60, 4, 42, 12, 3
```

Usunięcie elementu tablicy na konkretnym indeksie realizowane jest za pomocą metody *remove(at:)*. Usunięcie spowoduje zmniejszenie liczby elementów tabeli o jeden (Listing 6.9). Metoda *removeFirst()* spowoduje usunięcie pierwszego elementu tablicy, a *removeLast()* – ostatniego.

Listing 6.9. Usuwanie elementów z tabeli

```
var tab: [Int] = [25, 4, 42, 12, 3]
//25, 4, 42, 12, 3
tab.insert(60, at: 1)
//25, 60, 4, 42, 12, 3
tab.remove(at: 3)
//25, 60, 4, 12, 3
tab.remove(at: 0)
//60, 4, 12, 3
tab.removeFirst()
//4, 12, 3
tab.removeLast()
//4, 12
```

Modyfikacja elementu tablicy polega na nadpisaniu nowej wartości na konkretnym indeksie. Można zmodyfikować kilka wartości podając zakres, co przedstawiono na Listingu 6.10. Przypisanie mniejszej liczby elementów niż w podanym zakresie spowoduje zmniejszenie liczby elementów tabeli.

Listing 6.10. Modyfikowanie elementów tabeli

```
var tab: [Int] = [25, 4, 42, 12, 3]
tab[0] = 30
//30, 4, 42, 12, 3
tab[2...3] = [120, 80]
//30, 4, 120, 80, 3
tab[1...3] = [800, 750]
//30, 800, 750, 3
```

Dostęp do elementów tabeli realizowany jest poprzez instrukcje iteracyjne. Na Listingu 6.11 przedstawiono dostęp do kolejnych elementów przy pomocy instrukcji *for-in*.

Listing 6.11. Wyświetlenie kolejnych elementów tabeli

```
var tab: [Int] = [25, 4, 42, 12, 3]
for i in tab{
    print(i)
}
```

W celu uzyskania dostępu do indeksu elementu oraz jego wartości można zastosować metodę *enumerated()*. Dla każdego elementu w tablicy metoda ta zwraca krotkę złożoną z liczby całkowitej oraz wartości elementu, co przedstawiono na Listingu 6.12.

Listing 6.12. Wyświetlenie kolejnych elementów tabeli z metodą enumerated()

```
var tab: [Int] = [25, 4, 42, 12, 3]
for (i, v) in tab.enumerated(){
    print("Element na indeksie \(i) ma wartość \(v)")
}
```

Na tabeli można wykonywać operacje takie jak:

- *sort()* – sortuje elementy tablicy (Listing 6.13);
- *shuffle()* – zmienia kolejność elementów tablicy (Listing 6.14);
- *forEach()* – dostęp do kolejnych elementów tablicy (Listing 6.15);
- *contains()* – sprawdza, czy występuje element o podanej wartości (Listing 6.16);
- *swapAt()* – zamienia elementy na podanych pozycjach (Listing 6.17);
- *reverse()* – odwraca kolejność elementów tablicy (Listing 6.18).

Listing 6.13. Sortowanie elementów tablicy

```
var tab: [Int] = [25, 4, 42, 12, 3]
tab.sort()//domyślne sortowanie rosnące
for i in tab{
    print(i)
}
//3, 4, 12, 25, 43
tab.sort(by:>) //sortowanie malejące
for i in tab{
    print(i)
}
//42, 25, 12, 4, 3
```

Listing 6.14.Zmiana kolejności elementów tablicy

```
var tab: [Int] = [25, 4, 42, 12, 3]
tab.shuffle()
```

Listing 6.15.Dostęp do kolejnych elementów tablicy

```
var tab: [Int] = [25, 4, 42, 12, 3]
tab.forEach {
    print($0)
}
```

Listing 6.16. Sprawdzenie czy element istnieje w tablicy

```
var tab: [Int] = [25, 4, 42, 12, 3]
let num = Int.random(in: 1..<50)
if tab.contains(num){
    print("element \(num) występuje w tabeli")
}
else {
    print("element \(num) nie występuje w tabeli")
}
```

Listing 6.17. Zamiana elementów tablicy

```
var tab: [Int] = [25, 4, 42, 12, 3]
tab.swapAt(1, 3)
//25, 12, 42, 4, 3
```

Listing 6.18. Zamiana elementów tablicy

```
var tab: [Int] = [25, 4, 42, 12, 3]
tab.reverse()
//3, 12, 42, 4, 25
```

Zadanie 6.1.

Polecenie 1. Przeanalizuj kod z Listingu 6.19, znajdź błąd i go popraw.

Listing 6.19. Kod do analizy

```
var tab: [Int] = Array(repeating: 1.0, count: 12)
for i in tab {
    print(i)
}
```

Zadanie 6.2.

Polecenie 1. Napisz program konsolowy, który wygeneruje tablicę elementów całkowitych o wartościach domyślnych z zakresu 1-100. Rozmiar tej tablicy należy ustawić na 10.

Polecenie 2. Dodaj wczytaną liczbę całkowitą na zerowy element tablicy.

Polecenie 3. Dodaj wczytaną liczbę całkowitą na indeksie tabeli, który zostanie wygenerowany losowo.

Polecenie 4. Wyświetl elementy otrzymanych tablic.

Zadanie 6.3.

Polecenie 1. Napisz program konsolowy, który wczyta od użytkownika liczbę elementów tablicy oraz jej elementy całkowite, a następnie sprawdzi i wyświetli, czy wczytana liczba całkowita:

- jest pierwszym elementem tablicy,
- jest ostatnim elementem tablicy.

Należy sprawdzić, czy wczytane dane są liczbami całkowitymi oraz czy wczytana liczba elementów tablicy jest dodatnia.

Zadanie 6.4.

Polecenie 1. Napisz program konsolowy, który wczyta od użytkownika liczbę elementów tablicy oraz wygeneruje elementy całkowite do dwóch tablic z zakresu 1-20. Program powinien sprawdzić, czy zawierają one identyczne elementy, nie koniecznie na odpowiadających pozycjach. Należy wyświetlić komunikat o obie tablice.
Należy sprawdzić, czy wczytana liczba elementów tablicy jest dodatnia.

Zadanie 6.5.

Polecenie 1. Napisz program konsolowy, który zdefiniuje tablicę dwuwymiarową, przypisze jej wartości, a następnie wyświetli ją w postaci macierzy. Przykład kodu został przedstawiony na Listingu 6.20.

Listing 6.20. Tablica dwuwymiarowa

```
let mac: [[Int]] = [[10, 20, 30], [40, 50, 60]]
for i in 0 ..< mac.count {
    for j in 0 ..< mac[i].count{
        print("\(mac[i][j])", terminator:"  ")
    }
    print()
}
```

Zadanie 6.6.

Polecenie 1. Napisz program konsolowy, który wczyta od użytkownika liczbę wierszy i kolumn macierzy, a następnie wygeneruje jej elementy zmiennoprzecinkowe z zakresu -100-100. Program ma znaleźć największą i najmniejszą liczbę macierzy, a także indeks, na którym występuje. Należy wyświetlić wygenerowaną macierz (Listing 6.20).

Zadanie 6.7.

Polecenie 1. Napisz program konsolowy, który wczyta od użytkownika liczbę elementów tablicy i jej elementy całkowite, a następnie wyświetli najdłuższy podciąg rosnący. Jeśli taki podciąg nie istnieje, należy wyświetlić odpowiednią informację.